SmartBridge Externship

Artificial Intelligence

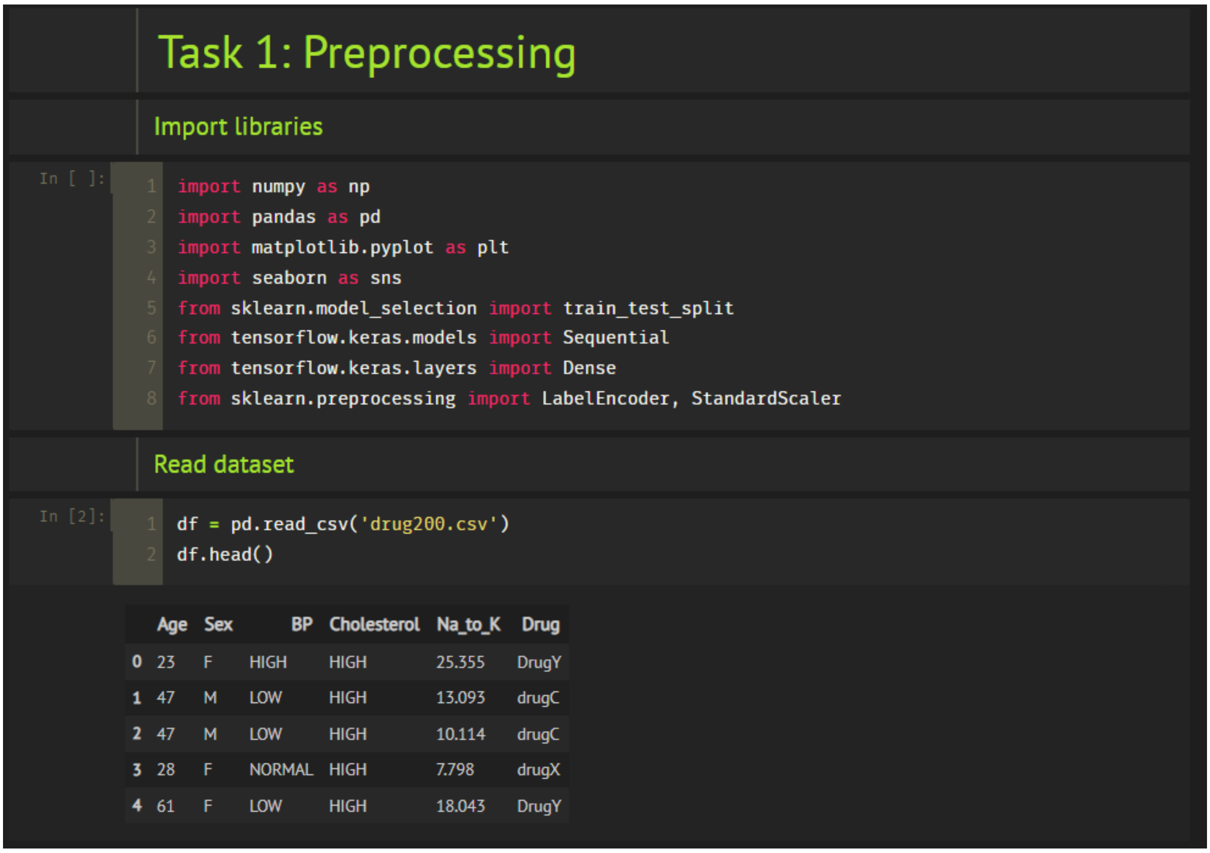
Assignment-2

Build an ANN model for Drug classification.

This project aims to analyze the relationship between various medical parameters and drug effectiveness. The dataset consists of patient information, including age, sex, blood pressure levels (BP), cholesterol levels, sodium-to-potassium ratio (Na_to_K), drug type, and corresponding labels. The goal is to develop a model that can accurately predict the class or category of a given drug based on its features.

Dataset Link: https://www.kaggle.com/datasets/prathamtripathi/drug-classification

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



Task 1: Preprocessing

Import libraries

In [ ]:

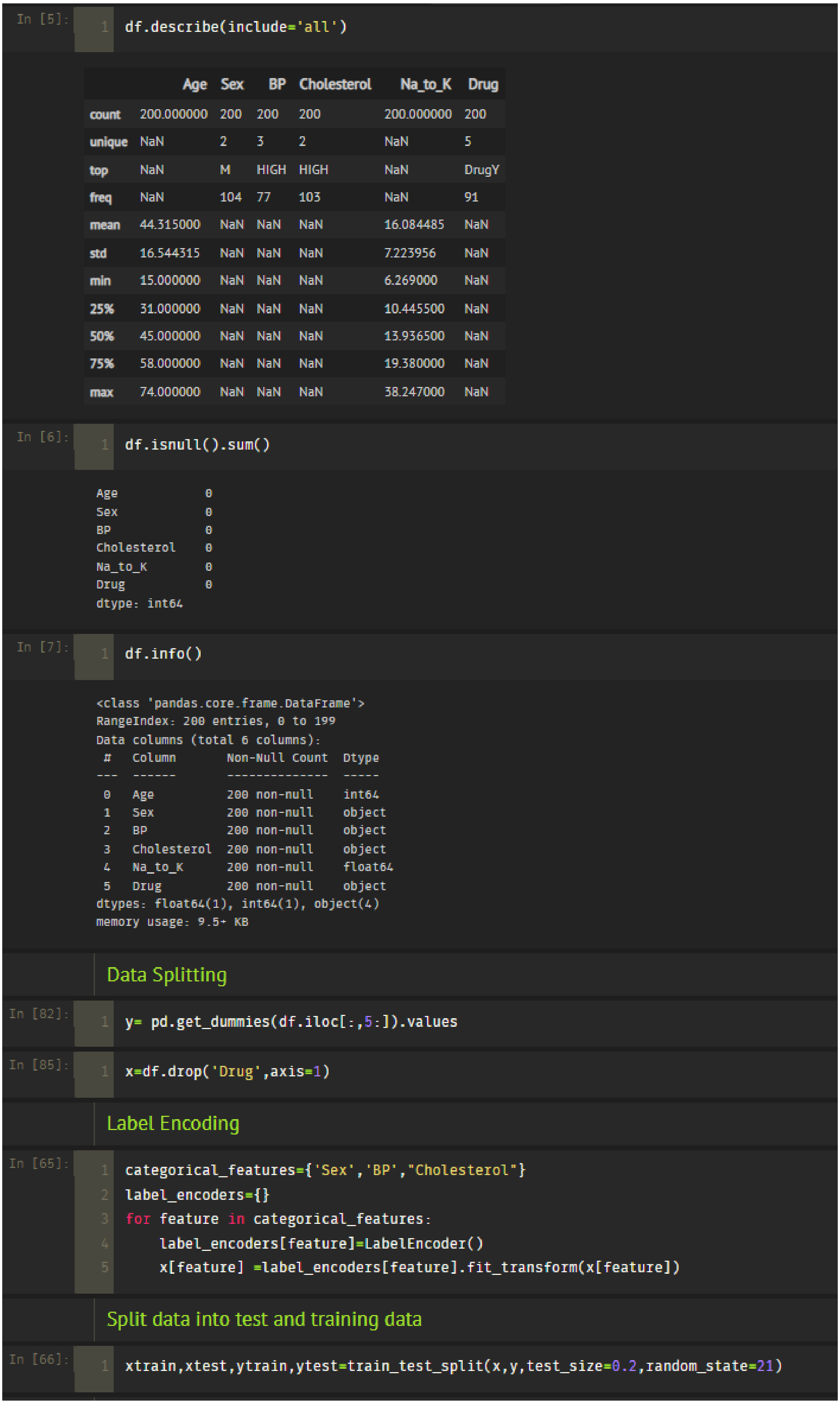
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

Read dataset

In [2]:

```
df = pd.read_csv('drug200.csv')
df.head()
```

	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



In [5]:

```
df.describe(include='all')
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
count	200.000000	200	200	200	200.000000	200
unique	NaN	2	3	2	NaN	5
top	NaN	M	HIGH	HIGH	NaN	DrugY
freq	NaN	104	77	103	NaN	91
mean	44.315000	NaN	NaN	NaN	16.084485	NaN
std	16.544315	NaN	NaN	NaN	7.223956	NaN
min	15.000000	NaN	NaN	NaN	6.269000	NaN
25%	31.000000	NaN	NaN	NaN	10.445500	NaN
50%	45.000000	NaN	NaN	NaN	13.936500	NaN
75%	58.000000	NaN	NaN	NaN	19.380000	NaN
max	74.000000	NaN	NaN	NaN	38.247000	NaN

In [6]:

```
df.isnull().sum()
```

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

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          200 non-null    int64  
 1   Sex          200 non-null    object 
 2   BP           200 non-null    object 
 3   Cholesterol  200 non-null    object 
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object 
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
```

Data Splitting

In [82]:

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

In [85]:

```
x=df.drop('Drug',axis=1)
```

Label Encoding

In [65]:

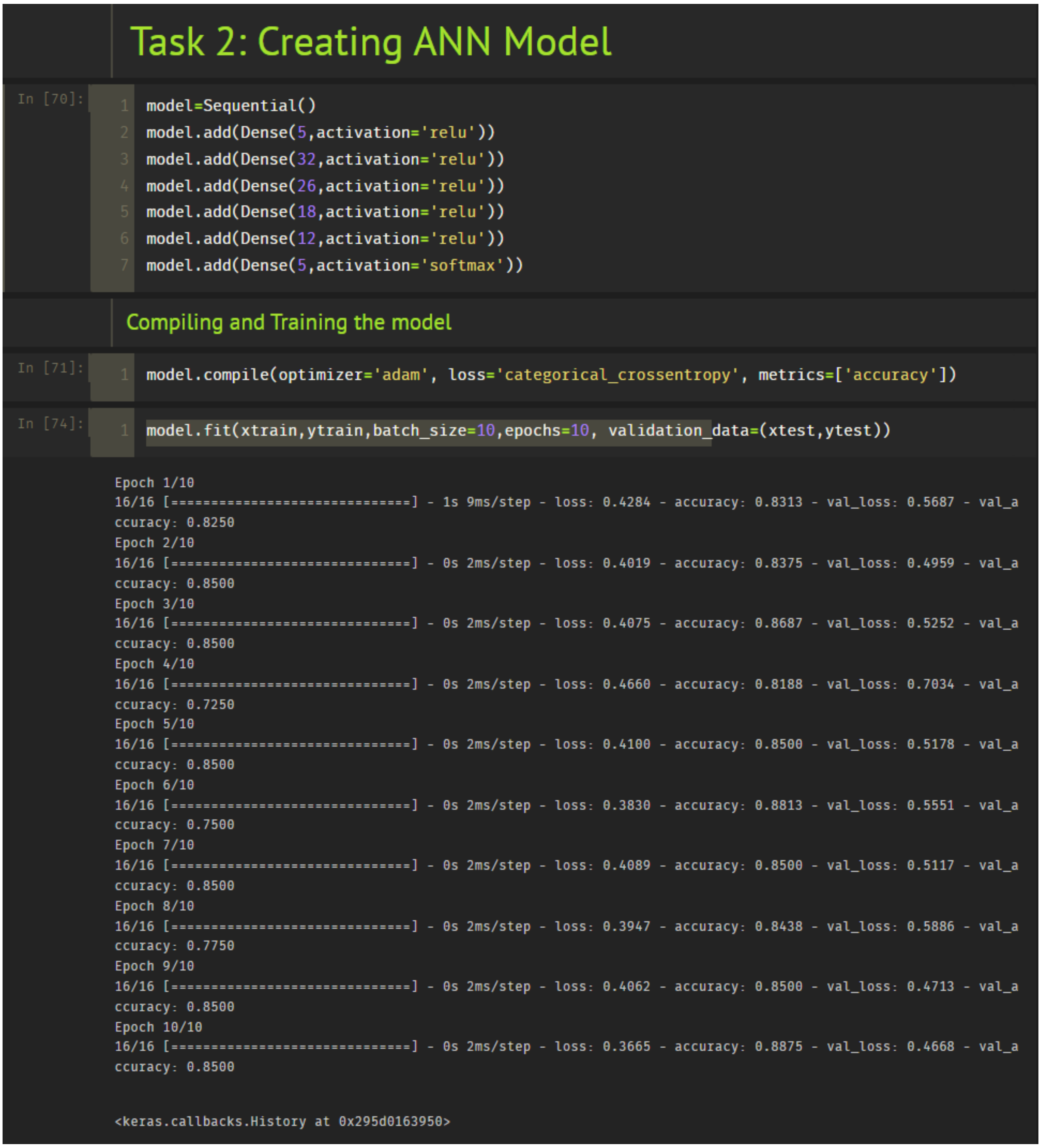
```
categorical_features={'Sex','BP',"Cholesterol"}
label_encoders={}
for feature in categorical_features:
    label_encoders[feature]=LabelEncoder()
    x[feature] =label_encoders[feature].fit_transform(x[feature])
```

Split data into test and training data

In [66]:

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=21)
```

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



Task 2: Creating ANN Model

In [70]:

```
model=Sequential()
model.add(Dense(5,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(26,activation='relu'))
model.add(Dense(18,activation='relu'))
model.add(Dense(12,activation='relu'))
model.add(Dense(5,activation='softmax'))
```

Compiling and Training the model

In [71]:

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

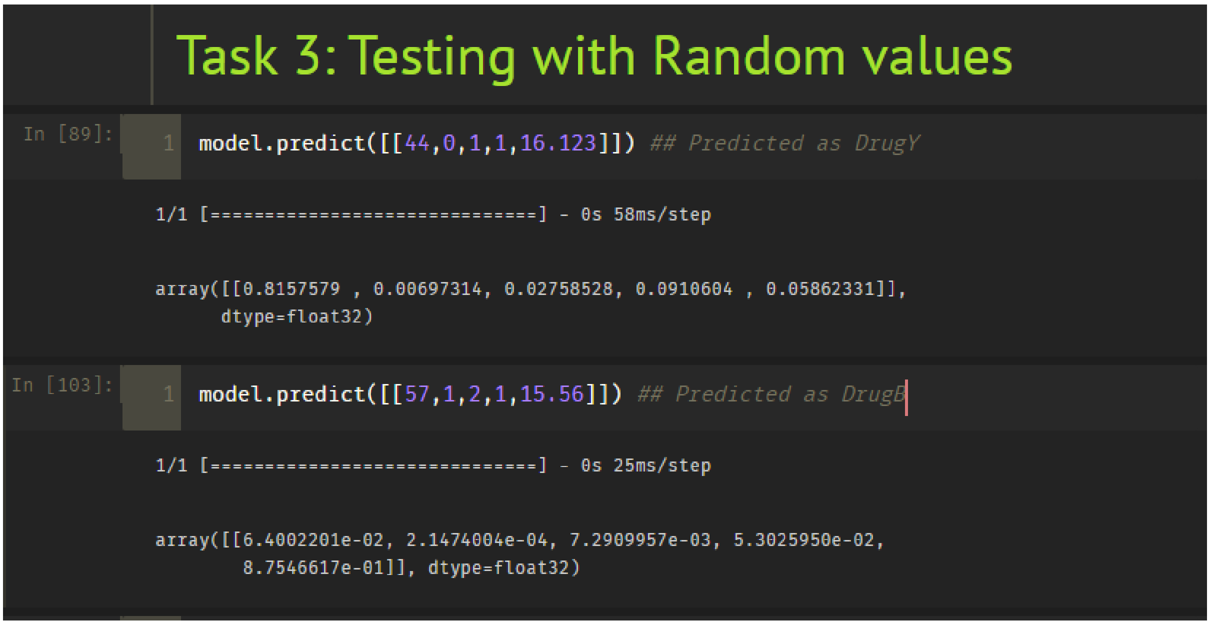
In [74]:

```
model.fit(xtrain,ytrain,batch_size=10,epochs=10, validation_data=(xtest,ytest))
```

```
Epoch 1/10
16/16 [==============================] - 1s 9ms/step - loss: 0.4284 - accuracy: 0.8313 - val_loss: 0.5687 - val_a
ccuracy: 0.8250
Epoch 2/10
16/16 [==============================] - 0s 2ms/step - loss: 0.4019 - accuracy: 0.8375 - val_loss: 0.4959 - val_a
ccuracy: 0.8500
Epoch 3/10
16/16 [==============================] - 0s 2ms/step - loss: 0.4075 - accuracy: 0.8687 - val_loss: 0.5252 - val_a
ccuracy: 0.8500
Epoch 4/10
16/16 [==============================] - 0s 2ms/step - loss: 0.4660 - accuracy: 0.8188 - val_loss: 0.7034 - val_a
ccuracy: 0.7250
Epoch 5/10
16/16 [==============================] - 0s 2ms/step - loss: 0.4100 - accuracy: 0.8500 - val_loss: 0.5178 - val_a
ccuracy: 0.8500
Epoch 6/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3830 - accuracy: 0.8813 - val_loss: 0.5551 - val_a
ccuracy: 0.7500
Epoch 7/10
16/16 [==============================] - 0s 2ms/step - loss: 0.4089 - accuracy: 0.8500 - val_loss: 0.5117 - val_a
ccuracy: 0.8500
Epoch 8/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3947 - accuracy: 0.8438 - val_loss: 0.5886 - val_a
ccuracy: 0.7750
Epoch 9/10
16/16 [==============================] - 0s 2ms/step - loss: 0.4062 - accuracy: 0.8500 - val_loss: 0.4713 - val_a
ccuracy: 0.8500
Epoch 10/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3665 - accuracy: 0.8875 - val_loss: 0.4668 - val_a
ccuracy: 0.8500


<keras.callbacks.History at 0x295d0163950>
```

Task - 3 Test the model with random data.



Task 3: Testing with Random values

```
In [89]:  model.predict([[44,0,1,1,16.123]]) ## Predicted as DrugY
```

```
1/1 [==============================] - 0s 58ms/step

array([[0.8157579 , 0.00697314, 0.02758528, 0.0910604 , 0.05862331]],
      dtype=float32)
```

```
In [103]:  model.predict([[57,1,2,1,15.56]]) ## Predicted as DrugB
```

```
1/1 [==============================] - 0s 25ms/step

array([[6.4002201e-02, 2.1474004e-04, 7.2909957e-03, 5.3025950e-02,
        8.7546617e-01]], dtype=float32)
```