```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense,Activation
```

Task 1

Reading the dataset and preprosessing

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

	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
...	...	...	...	...	...	...
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

200 rows × 6 columns

```
#Data Preprosessing
df.isnull().any()
```

```
Age            False
Sex            False
BP             False
Cholesterol    False
Na_to_K        False
Drug           False
dtype: bool
```

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

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

```
print(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
```

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

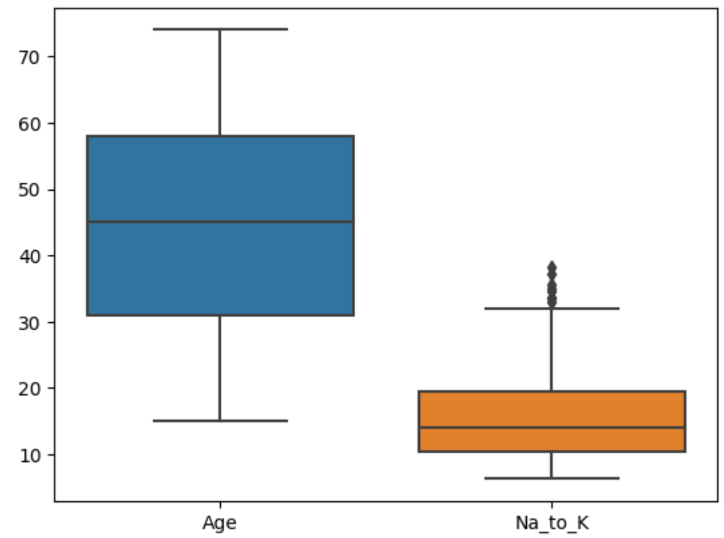
	Age	Sex	BP	Cholesterol	Na_to_K	Drug
count	200.000000	200	200	200	200.000000	200
unique	NaN	2	3	2	NaN	5
top	NaN	M	HIGH	HIGH	NaN	drugY
freq	NaN	104	77	103	NaN	91
mean	44.315000	NaN	NaN	NaN	16.084485	NaN
std	16.544315	NaN	NaN	NaN	7.223956	NaN
min	15.000000	NaN	NaN	NaN	6.269000	NaN
25%	31.000000	NaN	NaN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

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

```
drugY    91
drugX    54
drugA    23
drugC    16
drugB    16
Name: Drug, dtype: int64
```

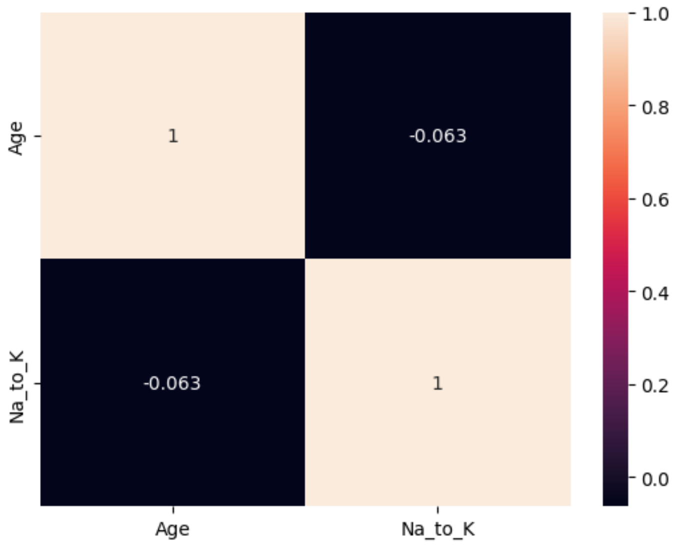
```
#box plots
sns.boxplot(df)
```

```
<Axes: >
```



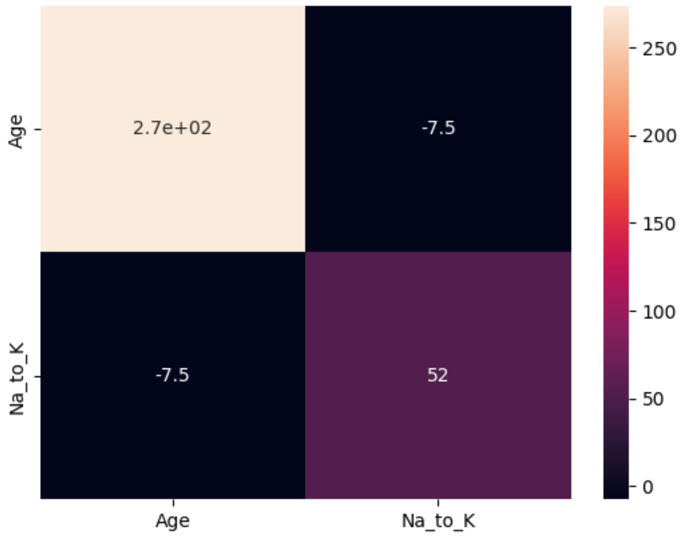
```
sns.heatmap(df.corr(),annot = True)
```

```
<Axes: >
```



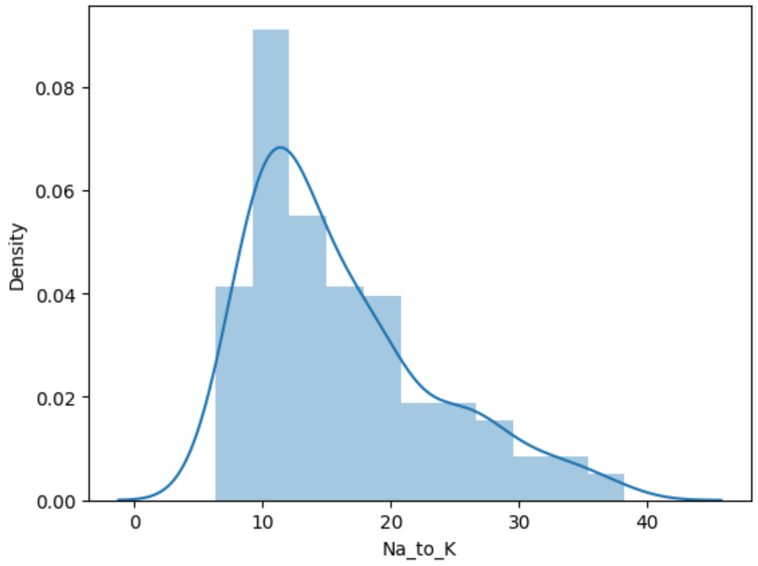
```
sns.heatmap(df.cov(),annot=True)
```

```
<Axes: >
```



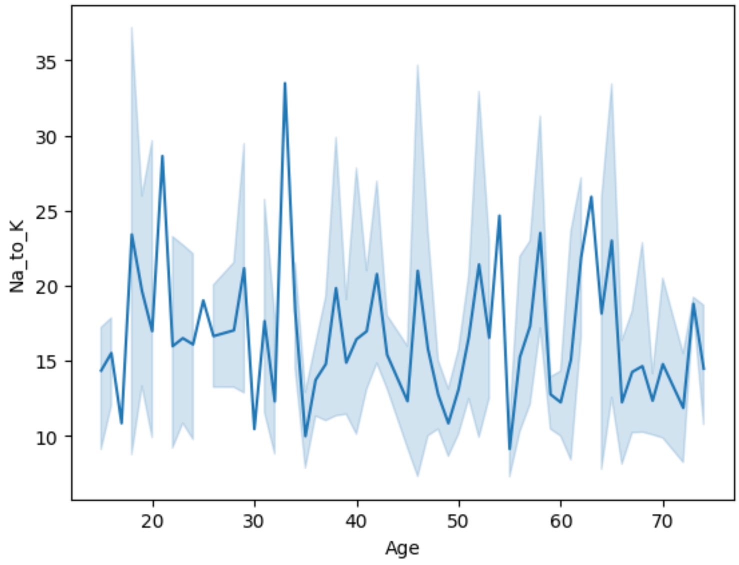
```
sns.distplot(df['Na_to_K'])
```

```
<Axes: xlabel='Na_to_K', ylabel='Density'>
```



```
sns.lineplot(x = df['Age'],y=df['Na_to_K'])
```

```
<Axes: xlabel='Age', ylabel='Na_to_K'>
```



```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['Sex'] = label_encoder.fit_transform(df['Sex'])
df['BP'] = label_encoder.fit_transform(df['BP'])
df['Cholesterol'] = label_encoder.fit_transform(df['Cholesterol'])

# Normalize numerical variables
scaler = preprocessing.StandardScaler()
df[['Age', 'Na_to_K']] = scaler.fit_transform(df[['Age', 'Na_to_K']])

# Split the data into features and labels
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
y = pd.get_dummies(df.iloc[:,5:]).values
print(df['Drug'].unique())
```

```
['drugY' 'drugC' 'drugX' 'drugA' 'drugB']
```

```
#split the data set into training and testing sets
xtrain,xtest,ytrain,ytest = train_test_split(x,y, test_size = 0.2, random_state=15)
print(x)
print(y)
```

```
          Age  Sex  BP  Cholesterol   Na_to_K
0   -1.291591    0   0            0  1.286522
1    0.162699    1   1            0 -0.415145
2    0.162699    1   1            0 -0.828558
3   -0.988614    0   2            0 -1.149963
4    1.011034    0   1            0  0.271794
..        ...  ...  ..          ...       ...
195  0.708057    0   1            0 -0.626917
196 -1.715759    1   1            0 -0.565995
197  0.465676    1   2            0 -0.859089
198 -1.291591    1   2            1 -0.286500
199 -0.261469    0   1            1 -0.657170

[200 rows x 5 columns]
[[0 0 0 0 1]
 [0 0 1 0 0]
 [0 0 1 0 0]
 [0 0 0 1 0]
 [0 0 0 0 1]
 [0 0 0 1 0]
 [0 0 0 0 1]
 [0 0 1 0 0]
 [0 0 0 0 1]
 [0 0 0 0 1]
 [0 0 1 0 0]
 [0 0 0 0 1]
 [0 0 0 0 1]
 [0 0 0 0 1]
 [0 0 0 1 0]
 [0 0 0 0 1]
 [0 0 0 1 0]
 [1 0 0 0 0]
 [0 0 1 0 0]
 [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 1 0]
 [0 0 0 0 1]
 [0 0 0 0 1]
 [0 0 0 1 0]
 [0 1 0 0 0]
 [0 0 0 1 0]
 [0 0 0 0 1]
 [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 0]
 [0 0 0 0 1]
 [0 1 0 0 0]
 [0 0 0 0 1]
 [0 0 0 1 0]
```

```
x.shape,y.shape
```

```
((200, 5), (200, 5))
```

```
xtrain
```

	Age	Sex	BP	Cholesterol	Na_to_K
47	1.435202	1	1	0	-0.803995
142	0.950439	1	0	1	-1.035750
149	-1.352186	1	1	0	-1.100975
152	0.647462	1	2	1	-1.224485
182	-1.473377	0	1	1	-0.610403
...	...	...	...	...	...
156	-0.806828	1	0	1	-0.674101
128	0.162699	1	1	1	2.422679
119	1.011034	0	0	0	1.303175
133	-1.230996	1	2	0	1.346334
140	0.283889	1	0	1	-1.362151

160 rows × 5 columns

Task 2

Build the ANN model

```
model = Sequential()
model.add(Dense(32, activation='relu',input_dim=x.shape[1]))#input layer
model.add(Dense(16, activation='relu')) #hidden layer
model.add(Dense(8, activation='relu')) #hidden layer
model.add(Dense(4, activation='relu')) #hidden layer
model.add(Dense(5, activation='softmax')) #Output layer
```

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

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                192

 dense_1 (Dense)             (None, 16)                528

 dense_2 (Dense)             (None, 8)                 136

 dense_3 (Dense)             (None, 4)                 36

 dense_4 (Dense)             (None, 5)                 25

=================================================================
Total params: 917
Trainable params: 917
Non-trainable params: 0
_________________________________________________________________
```

```
#Training the model
model.fit(xtrain,ytrain,epochs=10,batch_size=5,validation_data=(xtest,ytest))
```

```
Epoch 1/10
32/32 [==============================] - 1s 8ms/step - loss: 1.5833 - accuracy: 0.4500 - val_loss: 1.5270 - val_accuracy: 0.4750
Epoch 2/10
32/32 [==============================] - 0s 2ms/step - loss: 1.4501 - accuracy: 0.4500 - val_loss: 1.4008 - val_accuracy: 0.4750
Epoch 3/10
32/32 [==============================] - 0s 2ms/step - loss: 1.3154 - accuracy: 0.4500 - val_loss: 1.2479 - val_accuracy: 0.4750
Epoch 4/10
32/32 [==============================] - 0s 2ms/step - loss: 1.1855 - accuracy: 0.4500 - val_loss: 1.1322 - val_accuracy: 0.4750
Epoch 5/10
32/32 [==============================] - 0s 2ms/step - loss: 1.0947 - accuracy: 0.4812 - val_loss: 1.0402 - val_accuracy: 0.5000
Epoch 6/10
32/32 [==============================] - 0s 2ms/step - loss: 1.0170 - accuracy: 0.5750 - val_loss: 0.9702 - val_accuracy: 0.5000
Epoch 7/10
32/32 [==============================] - 0s 3ms/step - loss: 0.9555 - accuracy: 0.6000 - val_loss: 0.9188 - val_accuracy: 0.5250
Epoch 8/10
32/32 [==============================] - 0s 3ms/step - loss: 0.9111 - accuracy: 0.6000 - val_loss: 0.8762 - val_accuracy: 0.5250
```

```
Epoch 9/10
32/32 [==============================] - 0s 2ms/step - loss: 0.8675 - accuracy: 0.6062 - val_loss: 0.8407 - val_accuracy: 0.5250
Epoch 10/10
32/32 [==============================] - 0s 3ms/step - loss: 0.8336 - accuracy: 0.5875 - val_loss: 0.8031 - val_accuracy: 0.5250
<keras.callbacks.History at 0x7f84441af760>
```

```
ypred = model.predict(xtest)
```

```
2/2 [==============================] - 0s 8ms/step
```

```
ypred.shape
```

```
(40, 5)
```

Task 3

Testing the data with the random values

```
ypred1 = model.predict([[1.432,0,0,1,-1.094]])
ypred1
```

```
1/1 [==============================] - 0s 79ms/step
array([[0.19395398, 0.15538718, 0.20652266, 0.25762147, 0.18651469]],
      dtype=float32)
```