Kshtiij Gupta PR-**6**

```
import pandas as pd
import io
from google.colab import files
uploaded=files.upload()
```

Choose files drug200.csv

- **drug200.csv**(text/csv) - 6027 bytes, last modified: 18/10/2024 - 100% done

```
Saving drug200.csv to drug200.csv
```

```
df=pd.read_csv("drug200.csv")
print("The first 5 rows of the dataframe")
df.head(10)
```

```
The first 5 rows of the dataframe
```

	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
5	22	F	NORMAL	HIGH	8.607	drugX
6	49	F	NORMAL	HIGH	16.275	drugY
7	41	M	LOW	HIGH	11.037	drugC
8	60	M	NORMAL	HIGH	15.171	drugY
9	43	M	LOW	NORMAL	19.368	drugY

Next steps: Generate code with df | View recommended plots | New interactive sheet

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

df.dtypes
```

	0
Age	int64
Sex	object
BP	object
Cholesterol	object
Na_to_K	float64
Drug	object

dtype: object

```
# Importing LabelEncoder from Sklearn
# library from preprocessing Module.
from sklearn.preprocessing import LabelEncoder
```

```
# Creating a instance of label Encoder.
le = LabelEncoder()
```

```
# Using .fit_transform function to fit label
# encoder and return encoded label
label = le.fit_transform(df['Drug'])
```

```


# printing label
label
```

```
array([4, 2, 2, 3, 4, 3, 4, 2, 4, 4, 2, 4, 4, 4, 3, 4, 3, 0, 2, 4, 4, 4,
       4, 4, 4, 4, 4, 3, 4, 4, 3, 1, 3, 4, 3, 3, 0, 3, 3, 3, 4, 1, 4, 3,
       3, 3, 0, 2, 4, 4, 4, 3, 4, 4, 1, 2, 1, 4, 3, 4, 4, 0, 4, 3, 1, 4,
       0, 3, 4, 4, 1, 4, 3, 4, 4, 4, 0, 4, 0, 3, 1, 3, 2, 0, 2, 1, 3, 4,
       4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 0, 0, 2, 3, 4, 3, 3, 4, 1, 4,
       0, 3, 3, 3, 3, 4, 3, 3, 0, 4, 4, 4, 4, 4, 1, 4, 4, 3, 4, 3, 4, 4,
       3, 4, 4, 3, 1, 0, 1, 3, 0, 4, 1, 4, 0, 3, 3, 0, 3, 2, 0, 1, 3, 3,
       4, 2, 0, 4, 2, 3, 3, 1, 3, 4, 4, 4, 4, 3, 4, 0, 3, 3, 4, 4, 0, 4,
       0, 4, 4, 4, 4, 3, 3, 4, 4, 4, 1, 0, 4, 4, 4, 0, 4, 2, 4, 2, 2, 3,
       3, 3])
```

```
# removing the column  from df
# as it is of no use now.
df.drop("Drug", axis=1, inplace=True)
df["Drug"] = label
df
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	4
1	47	M	LOW	HIGH	13.093	2
2	47	M	LOW	HIGH	10.114	2
3	28	F	NORMAL	HIGH	7.798	3
4	61	F	LOW	HIGH	18.043	4
...	...	...	...	...	...	...
195	56	F	LOW	HIGH	11.567	2
196	16	M	LOW	HIGH	12.006	2
197	52	M	NORMAL	HIGH	9.894	3
198	23	M	NORMAL	NORMAL	14.020	3
199	40	F	LOW	NORMAL	11.349	3

200 rows × 6 columns

Next steps: Generate code with df View recommended plots New interactive sheet

```
# Importing LabelEncoder from Sklearn
# library from preprocessing Module.
from sklearn.preprocessing import LabelEncoder
```

```
# Creating a instance of label Encoder.
le = LabelEncoder()
```

```
# Using .fit_transform function to fit label
# encoder and return encoded label
label = le.fit_transform(df['Cholesterol'])
```

```
# printing label
label
```

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

```
label
```

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

```
# removing the column  from df
# as it is of no use now.
df.drop("Cholesterol", axis=1, inplace=True)
df["Cholesterol"] = label
df
```

	Age	Sex	BP	Na_to_K	Drug	Cholesterol
0	23	F	HIGH	25.355	4	0
1	47	M	LOW	13.093	2	0
2	47	M	LOW	10.114	2	0
3	28	F	NORMAL	7.798	3	0
4	61	F	LOW	18.043	4	0
...	...	...	...	...	...	...
195	56	F	LOW	11.567	2	0
196	16	M	LOW	12.006	2	0
197	52	M	NORMAL	9.894	3	0
198	23	M	NORMAL	14.020	3	1
199	40	F	LOW	11.349	3	1

200 rows × 6 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Importing LabelEncoder from Sklearn
# library from preprocessing Module.
from sklearn.preprocessing import LabelEncoder
```

```
# Creating a instance of label Encoder.
le = LabelEncoder()
```

```
# Using .fit_transform function to fit label
# encoder and return encoded label
label = le.fit_transform(df['BP'])
```

```
# printing label
label
```

```
array([0, 1, 1, 2, 1, 2, 2, 1, 2, 1, 1, 0, 1, 1, 2, 0, 1, 0, 1, 0, 1, 2,
       1, 1, 1, 0, 0, 2, 1, 1, 2, 0, 1, 0, 2, 2, 0, 1, 2, 2, 2, 0, 2, 2,
       2, 2, 0, 1, 2, 1, 0, 2, 1, 0, 0, 1, 0, 0, 2, 0, 1, 0, 1, 1, 0, 2,
       0, 2, 2, 0, 0, 2, 2, 2, 0, 1, 0, 0, 0, 1, 0, 2, 1, 0, 1, 0, 2, 1,
       0, 2, 2, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 2, 0, 1, 2, 1, 0, 2,
       0, 2, 1, 1, 2, 0, 2, 2, 0, 0, 2, 0, 2, 2, 0, 0, 0, 2, 1, 2, 2, 1,
       1, 2, 0, 1, 0, 0, 0, 2, 0, 1, 0, 0, 0, 2, 1, 0, 1, 1, 0, 0, 2, 1,
       1, 1, 0, 1, 1, 1, 2, 0, 2, 0, 0, 1, 1, 2, 1, 0, 2, 1, 2, 1, 0, 0,
       0, 2, 2, 2, 0, 2, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 2,
       2, 1])
```

```
# removing the column  from df
# as it is of no use now.
df.drop("BP", axis=1, inplace=True)
df["BP"] = label
df
```

	Age	Sex	Na_to_K	Drug	Cholesterol	BP
0	23	F	25.355	4	0	0
1	47	M	13.093	2	0	1
2	47	M	10.114	2	0	1
3	28	F	7.798	3	0	2
4	61	F	18.043	4	0	1
...	...	...	...	...	...	...
195	56	F	11.567	2	0	1
196	16	M	12.006	2	0	1
197	52	M	9.894	3	0	2
198	23	M	14.020	3	1	2
199	40	F	11.349	3	1	1

200 rows × 6 columns

Next steps: Generate code with df View recommended plots New interactive sheet

```
# Importing LabelEncoder from Sklearn
# library from preprocessing Module.
from sklearn.preprocessing import LabelEncoder
```

```
# Creating a instance of label Encoder.
le = LabelEncoder()
```

```
# Using .fit_transform function to fit label
# encoder and return encoded label
label = le.fit_transform(df['Sex'])
```

```
# printing label
label
```

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

```
# removing the column  from df
# as it is of no use now.
df.drop("Sex", axis=1, inplace=True)
df["Sex"] = label
df
```

	Age	Na_to_K	Drug	Cholesterol	BP	Sex
0	23	25.355	4	0	0	0
1	47	13.093	2	0	1	1
2	47	10.114	2	0	1	1
3	28	7.798	3	0	2	0
4	61	18.043	4	0	1	0
...	...	...	...	...	...	...
195	56	11.567	2	0	1	0
196	16	12.006	2	0	1	1
197	52	9.894	3	0	2	1
198	23	14.020	3	1	2	1
199	40	11.349	3	1	1	0

200 rows × 6 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.corr()["Drug"]
```

	Drug
Age	-0.004828
Na_to_K	0.589120
Drug	1.000000
Cholesterol	0.055629
BP	0.372868
Sex	-0.098573

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

```
#1.divide dataset in test data and train data
#divide x_data and y_data
y_data2 = df['Drug']
x_data2=df.drop('Drug',axis=1)
```

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

```
x_train, x_test, y_train, y_test = train_test_split(x_data2, y_data2, test_size=0.2, random_state=1)
#random_state=1 gives you different results everytime
```

```
print("number of test samples :", x_test.shape[0])
print("number of training samples:",x_train.shape[0])
```

```
number of test samples : 40
number of training samples: 160
```

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

```


#LOGISTIC REGRESSION

from sklearn.linear_model import LogisticRegression
LR = LogisticRegression().fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
yhat1 = LR.predict(x_train)
yhat1
```

```
array([1, 4, 4, 3, 1, 4, 3, 3, 4, 1, 4, 4, 4, 4, 4, 4, 3, 3, 3, 0, 3, 3,
       4, 3, 0, 3, 3, 3, 4, 0, 1, 2, 4, 2, 4, 3, 4, 3, 4, 4, 4, 4, 3, 4,
       0, 4, 2, 3, 0, 3, 4, 0, 1, 1, 3, 3, 2, 3, 3, 4, 4, 4, 4, 3, 2, 1,
       0, 1, 4, 3, 4, 2, 4, 4, 4, 3, 4, 4, 0, 4, 4, 4, 0, 3, 1, 4, 4, 3,
       4, 2, 3, 0, 3, 0, 4, 4, 4, 4, 4, 3, 4, 1, 3, 3, 4, 1, 4, 1, 4, 4, 0,
       4, 4, 0, 3, 4, 3, 0, 4, 2, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 0,
       3, 2, 4, 4, 3, 4, 4, 4, 1, 4, 0, 3, 3, 0, 4, 4, 4, 4, 4, 3, 0, 4,
       3, 4, 0, 3, 1, 4])
```

```
yhat = LR.predict(x_test)
yhat
```

```
array([3, 4, 3, 4, 4, 3, 3, 4, 4, 4, 3, 3, 4, 3, 1, 0, 3, 3, 1, 4, 3, 3,
       3, 4, 1, 3, 3, 4, 1, 3, 2, 4, 4, 4, 0, 4, 0, 4, 4, 3])
```

```
from sklearn.metrics import jaccard_score
```

```
jaccard_score(y_test, yhat, average='macro')
```

```
0.6291666666666667
```

```
#Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, yhat, labels=[1,0])
```

```
array([[2, 0],
       [1, 3]])
```

```
from sklearn.metrics import precision_recall_fscore_support
precision_recall_fscore_support(y_test, yhat, average='macro')
```

```
(0.85, 0.7764705882352941, 0.7458904314076727, None)
```

```
#KNN
from sklearn.neighbors import KNeighborsClassifier
k = 5
#Train Model and Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(x_train,y_train)
neigh
```

```
▾ KNeighborsClassifier ⓘ ?
KNeighborsClassifier()
```

```
yhat = neigh.predict(x_test)
yhat[0:5]
```

```
array([1, 4, 1, 0, 4])
```

```
#ACCURACY
#Jaccard
from sklearn.metrics import jaccard_score
jaccard_score(y_test, yhat,average='macro')
```

```
0.3095238095238095
```

```
#Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, yhat, labels=[1,0])
```

```
array([[1, 0],
       [0, 3]])
```

```
from sklearn.metrics import precision_recall_fscore_support
precision_recall_fscore_support(y_test, yhat, average='macro')
```

```
(0.4382352941176471, 0.48438914027149327, 0.3988235294117647, None)
```

```


##decision tree
from sklearn.tree import DecisionTreeClassifier
drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
drugTree.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier ⓘ ?
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

```
predTree = drugTree.predict(x_test)
```

```
#ACCURACY
#Jaccard
from sklearn.metrics import jaccard_score
jaccard_score(y_test, predTree,average='macro')
```

```
1.0
```

```
#Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,predTree, labels=[1,0])
```

```
array([[2, 0],
       [0, 4]])
```