به نام خدا

سید پوریا احمدی 9723002

گزارش پروژه دوم

سوال شماره یک :

ابتدا کتابخانه های لازم را import شده است.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
from matplotlib import pyplot as plt
```

سپس داده ها را خوانده و missing value هایی که وجود دارند را (? هارا) تبدیل به np.Nan کرده و سپس مد آن ستون را به جای آنها قرار می دهیم.

```
df=pd.read_csv("Dataset1.csv")
# filling missing data
column_names=df.columns
for a in column_names:
    df[a]=df[a].replace('?', np.NaN)
df= df.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

حال باید داده های عددی را جدا کنیم برای categorize کردن آنها و طبق کد زیر ستون هایی که از int هستند مشخص می شوند.

```
num_data = df.select_dtypes(include=['int64'])
cat_data = df.select_dtypes(include=['object'])

num_columns = num_data.columns
cat_columns = cat_data.columns

print("Numerical Vaiables: ", num_columns)
print("Categorical Vaiables: ", cat_columns)
```

```
Numerical Vaiables:  Index(['age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss',
       'hours-per-week'],
      dtype='object')
Categorical Vaiables:  Index(['workclass', 'education', 'marital-status', 'occupation',
       'relationship', 'race', 'sex', 'native-country', 'income'],
      dtype='object')
```

حال باید numerical variable ها را categorize کرد، برای اینکار یک تابع برای هر ستون عددی در نظر گرفته یک سری تبدیلات را انجام میدهیم:

قبل از آن باید ببینیم هر ستون چه مقادیری دارد و از دو حالت می شود استفاده کرد:

1- Df.describe ()

2- کد زیر:

```
# for c in df.columns:
#     print ("---- %s ---" % c)
#     print (df[c].value_counts())
```

حال برای هر ستون یک تابع نوشته شده که بتوان ستون را categorizeکرد:

توجه شود چون education_num را در ستون دیگری به نام education داشتیم از دیتاست حذف کردیم.

```
def age_group(x):
    x = int(x)
    if( 0 < x < 20 ):
        return "<20"
    elif( 20 < x < 31 ):
        return "19-30"
    elif( 30 < x < 41 ):
        return "31-40"
    elif( 40 < x < 51 ):
        return "41-50"
    elif( 50 < x < 61 ):
        return "51-60"
    elif( 60 < x < 71 ):
        return "61-70"
    else:
        return ">70"
```

```
def fnlwgt(x):
    x = int(x)
    if(  x < 100001 ):
        return "low"
    elif( 100001 < x < 300001 ):
        return "low_med"
    elif( 300001 < x < 500000 ):
        return "high_med"
    else:
        return "high"
```

```
def capital_gain(x):
    x = int(x)
    if(  x > 0 ):
        return "yes"
    else:
        return "no"
```

```
def capital_loss(x):
    x = int(x)
    if(  x > 0 ):
        return "yes"
    else:
        return "no"
```

```
def hours_per_week(x):
    x = int(x)
    if( x > 60 ):
        return "too much"
    elif ( x > 40 ):
        return "much"
    elif ( x > 20 ):
        return "normal"
    else:
        return "low"
```

حال یک تابع نوشته شده است که این تابع ستون های جدیدی با مقادیر برگردانده شده از توابع بالا در دیتاست insert میکند و ستون های عددی ما حذف می شوند و همچنین یک group- به نام آن ستون ها اضافه می شود

سپس برای ستون martial-status یک سری کتگوری راحت تر در نظر میگیریم که در تابع pre_proccesing اعمال شده:

```
def pre_processing (df):
    df['age-group'] = df['age'].apply(age_group)
    df['fnlwgt-group'] = df['fnlwgt'].apply(fnlwgt)
    df['capital-gain-group'] = df['capital-gain'].apply(capital_gain)
    df['capital-loss-group'] = df['capital-loss'].apply(capital_loss)
    df['hours-per-week-group'] = df['hours-per-week'].apply(hours_per_week)

    df = df.drop(['age'], axis = 1)
    df = df.drop(['fnlwgt'], axis = 1)
    df = df.drop(['hours-per-week'], axis = 1)
    df = df.drop(['capital-gain'], axis = 1)
    df = df.drop(['capital-loss'], axis = 1)
    df = df.drop(['education-num'], axis = 1)
    df =df.replace(['Never-married', 'Married-civ-spouse', 'Divorced',
       'Married-spouse-absent', 'Separated', 'Married-AF-spouse',
       'Widowed'], ['not married', 'married', 'divorced',
       'married', 'not married', 'married',
       'not married'])
    return df
```

سپس تابع mapping را داریم که به ازای هر category در هر ستون یک عدد به آن map میکند (به جای one hot encoding) استفاده می شود.

Flag برای زمانی است که اگر income را داشته باشیم آنرا فعال کرده و آن ها categorize می شود.

برای native-country با استفاده از کد مورد دوم می توان دید که 41 کشور داریم اما United-state با اختلاف از همه بیشتر است پس دو حالت در نظر میگیریم یا افراد آمریکایی هستن یا خیر. برای همین همه ستون ها به جز US را تبدیل به non_us می کنیم و بعد یک عدد به آنها map میکنیم.

```python
def mapping(df , flag):
    df['sex'] = df['sex'].map({'Male': 0, 'Female': 1}).astype(int)
    df['race'] = df['race'].map({'Black': 0, 'Asian-Pac-Islander': 1,'Other': 2, 'White': 3, 'Amer-Indian-Eskimo': 4}).astype(int
    df['marital-status'] = df['marital-status'].map({'married': 0, 'not married': 1, 'divorced': 2}).astype(int)
    df['workclass']= df['workclass'].map({'Private': 0, 'Self-emp-not-inc': 1,'Local-gov': 2, 'State-gov': 3, 'Self-emp-inc': 4,
                                          'Federal-gov': 5, 'Without-pay': 6, 'Never-worked':7}).astype(int)
    df['relationship'] = df['relationship'].map({'Not-in-family': 0, 'Wife': 1, 'Other-relative': 2, 'Unmarried': 3,
                                                 'Husband': 4,'Own-child': 5}).astype(int)
    df['education']= df['education'].map({'Some-college': 0, 'Preschool': 1, '5th-6th': 2, 'HS-grad': 3, 'Masters': 4,
                                          '12th': 5, '7th-8th': 6, 'Prof-school': 7,'1st-4th': 8, 'Assoc-acdm': 9,
                                          'Doctorate': 10, '11th': 11,'Bachelors': 12, '10th': 13, 'Assoc-voc': 14,
                                          '9th': 15}).astype(int)
    df['occupation'] = df['occupation'].map({ 'Farming-fishing': 0, 'Tech-support': 1, 'Adm-clerical': 2, 'Handlers-cleaners': 3,
                                             'Prof-specialty': 4,'Machine-op-inspct': 5, 'Exec-managerial': 6,
                                            'Priv-house-serv': 7,'Craft-repair': 8,'Sales': 9, 'Transport-moving': 10,
                                            'Armed-Forces': 11, 'Other-service': 12,'Protective-serv':13}).astype(int)
    if flag == 0:
        df['income'] = df['income'] .map({'<=50K': 0, '>50K': 1}).astype(int)
    df['fnlwgt-group'] = df['fnlwgt-group'].map({'low_med': 0, 'low': 1,'high_med': 2, 'high': 3}).astype(int)

    df['capital-gain-group'] = df['capital-gain-group'] .map({'yes': 0, 'no': 1}).astype(int)
    df['capital-loss-group'] = df['capital-loss-group'] .map({'yes': 0, 'no': 1}).astype(int)
    df['hours-per-week-group'] = df['hours-per-week-group'].map({'low': 0, 'normal': 1,'much': 2, 'too much': 3}).astype(int)
    df['age-group']= df['age-group'].map({'31-40': 0, '19-30': 1, '41-50': 2, '51-60': 3, '61-70': 4,
                                          '<20': 5, '>70': 6}).astype(int)
    m =df['native-country'].unique()
    for i in range(len(m)-1):
        df = df.replace([m[i+1]],'non_us')
    df['native-country'] = df['native-country'] .map({'United-States': 0, 'non_us': 1}).astype(int)
    return df
```

حال این دو تابع صدا زده شده و آماده سازی داده ها تمام می شود.

حال باید داده ها train و test تقسیم شوند که از کتابخانه sklearn استفاده شده و تقسیم بندی انجام می شود.

همچنین X , y را نیز مشخص می کنیم:

```
: df = pre_processing(df)
  df = mapping(df , 0)
```

```
: X = df.drop('income', axis=1)
  y = df['income']
  # Splitting to training and testing
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

حال داده ها باید آموزش ببینند :

همانطور که در اولین عکس مشخص است از sklearn درخت تصمیم را import شده و به آموزش داده ها و سپس تست کردن آنها میپردازیم:

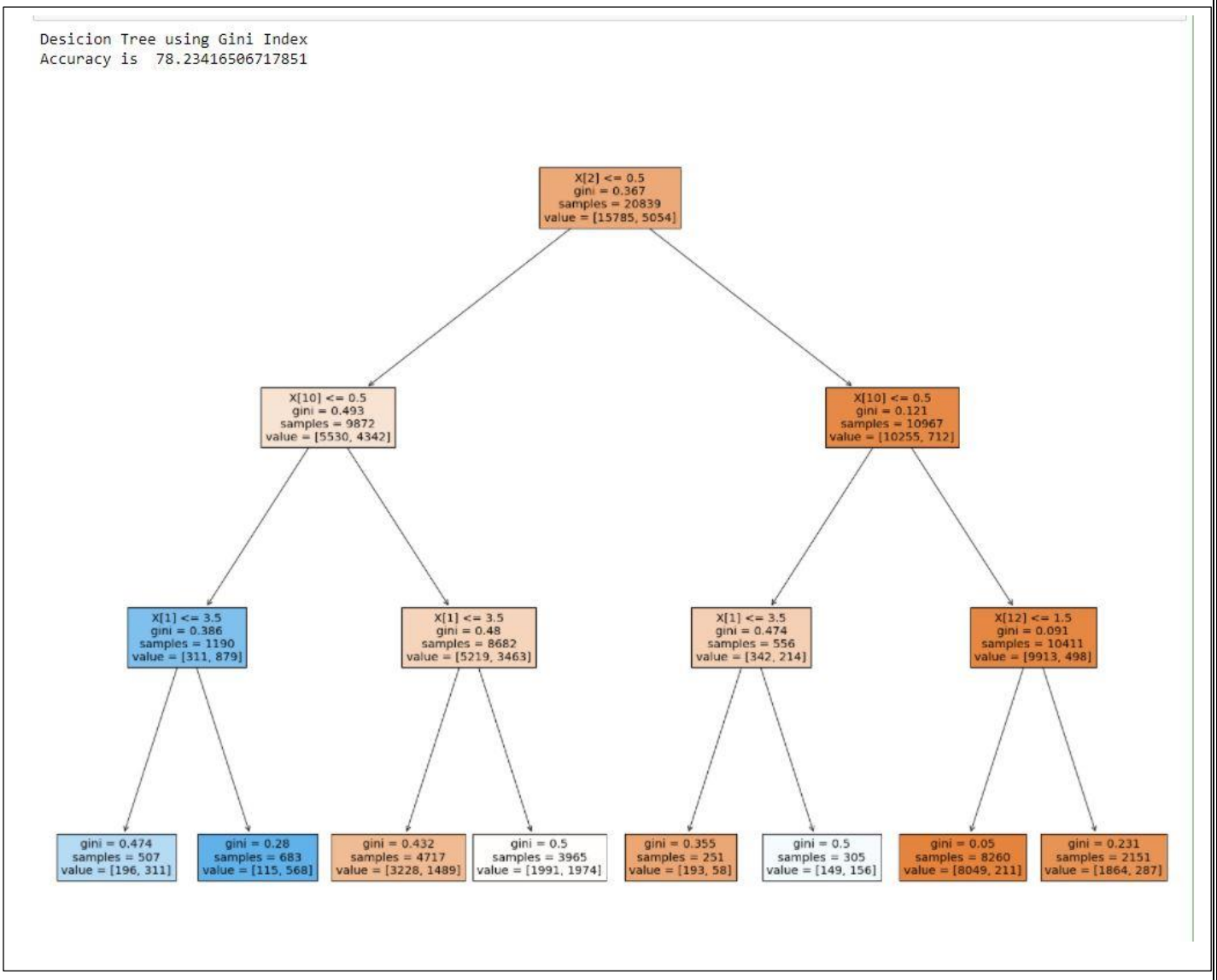
ایتدا با معیار gini این کار ا انجام میدهیم و درخت مربوطه را چاپ می کنیم.

توجه: برای اینکه درخت واضح باشد max_depth کم شده در فایل ارسالی سعی شده بهینه آن استفاده شود و ممکن است عکس کوچک تر و نا واضح تر باشد

در فایل ارسالی max_depth و max_leaf_node سعی شده بهینه باشند.

```
data_clf_gini=DecisionTreeClassifier(criterion='gini', max_depth=3)
data_clf_gini.fit(X_train,y_train)
y_pred=data_clf_gini.predict(X_test)
print("Desicion Tree using Gini Index\nAccuracy is ", accuracy_score(y_test,y_pred)*100)
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(data_clf_gini,filled=True)
```

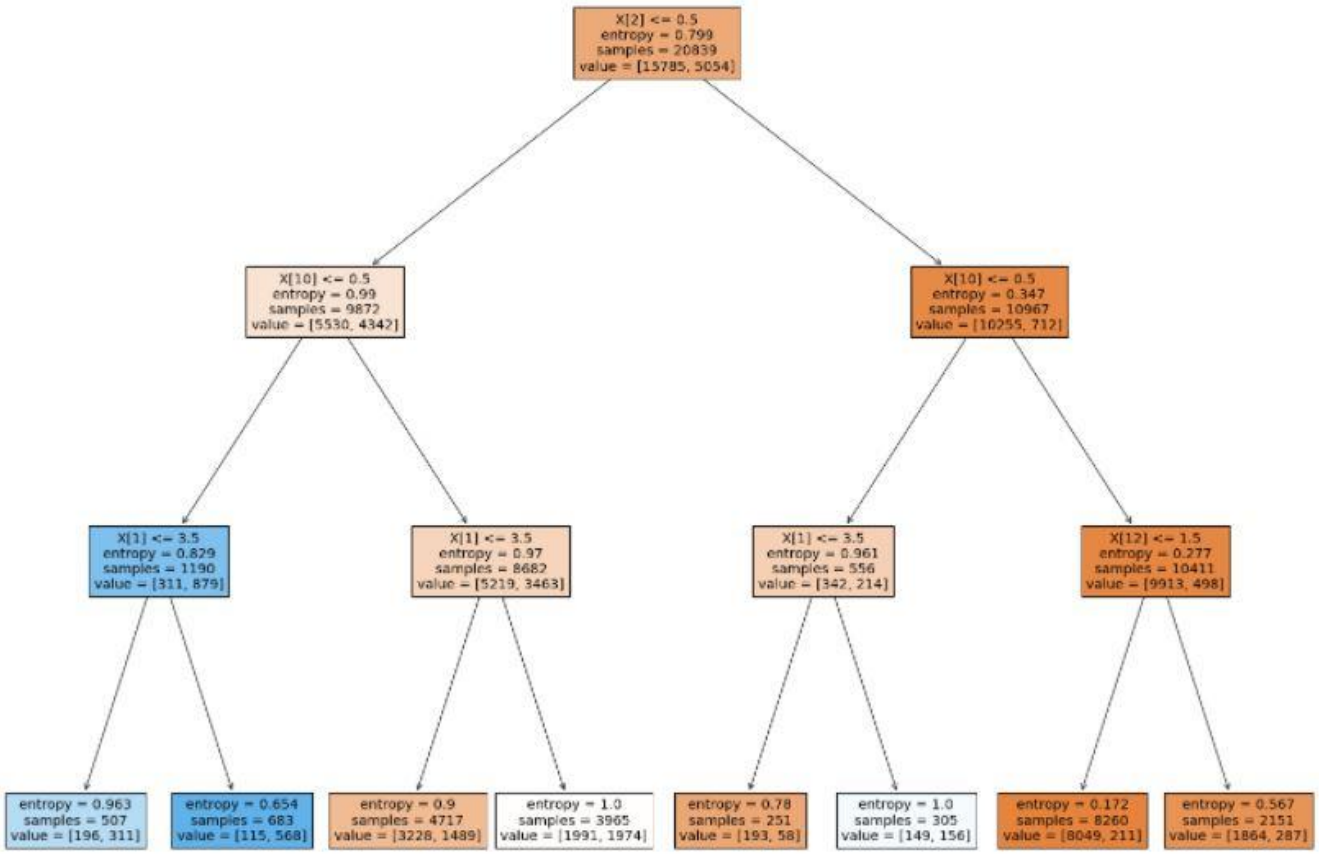
```
Desicion Tree using Gini Index
Accuracy is  78.23416506717851
```



حال با entropy :

```
data_clf_entropy=DecisionTreeClassifier(criterion='entropy', max_depth=3)
data_clf_entropy.fit(X_train,y_train)
y_pred=data_clf_entropy.predict(X_test)
print("Desicion Tree using entropy Index\nAccuracy is ", accuracy_score(y_test,y_pred)*100)
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(data_clf_entropy,filled=True)
```

```
Desicion Tree using entropy Index
Accuracy is  78.23416506717851
```



همچنان توجه شود که عمق سه عمق optimal نیست و برای وضوح درخت انتخاب شده.

در نهایت داده هایی که قرار است پیش بینی شوند را خوانده و pre_proccesing و mapping انجام می شود و با استفاده از دو مدلی که داریم پیش بینی را انجام می دهیم:

توجه شود اینجا flag غیر صفر است چون ستون income نداریم:

```
df_test=pd.read_csv("Dataset1_Unknown.csv")
# filling missing data
column_names=df_test.columns
for a in column_names:
    df_test[a]=df_test[a].replace('?', np.NaN)

df_test= df_test.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

```
df_test = pre_processing(df_test)
df_test = mapping(df_test ,1 )
```

```
y_pred_test_gini=data_clf_gini.predict(df_test)
y_pred_test_entropy=data_clf_entropy.predict(df_test)
```

سوال 2 :

در این سوال داده عددی نداریم پس تابع pre_proccesingقسمت قبل را نداریم و تا تابع mapping همه چیز مانند سوال قبل است:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
```

```
df=pd.read_csv("Dataset2.csv")
# filling missing data
column_names=df.columns
for a in column_names:
    df[a]=df[a].replace('?', np.NaN)
df= df.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

```
num_data = df.select_dtypes(include=['int64'])
cat_data = df.select_dtypes(include=['object'])

num_columns = num_data.columns
cat_columns = cat_data.columns

print("Numerical Vaiables: ", num_columns)
print("Categorical Vaiables: ", cat_columns)
```

```
Numerical Vaiables:  Index([], dtype='object')
Categorical Vaiables:  Index(['poisonous', 'cap-shape', 'cap-surface', 'cap-color', 'bruises', 'odor',
       'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color',
       'stalk-shape', 'stalk-root', 'stalk-surface-above-ring',
       'stalk-surface-below-ring', 'stalk-color-above-ring',
       'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number',
       'ring-type', 'spore-print-color', 'population', 'habitat'],
      dtype='object')
```

همچنین نام ستون ها را داریم و شروع به mapکردن اعداد به مقادیر مختلف هر ستون میکنم (با تابع mapping):

دوباره برای poisonous از flag استفاده شده.

```
for c in df.columns:
    print ("---- %s ---" % c)
    print (df[c].value_counts())
```

```
k     671
b     362
s      26
c       4
Name: cap-shape, dtype: int64
---- cap-surface ---
y    2605
s    2026
f    1865
g       3
Name: cap-surface, dtype: int64
---- cap-color ---
n    1845
g    1487
e    1175
y     853
w     827
b     129
p     121
c      37
```

```
def mapping(df , flag):
    if flag == 0:
        df['poisonous']= df['poisonous'].map({'p': 0, 'e': 1}).astype(int)
    df['cap-shape']= df['cap-shape'].map({'x': 0, 'f': 1, 'k': 2, 'b': 3,'s': 4, 'c': 5}).astype(int)
    df['cap-surface']= df['cap-surface'].map({'y': 0, 's': 1, 'f': 2, 'g': 3}).astype(int)
    df['cap-color']= df['cap-color'].map({'n': 0, 'g': 1, 'e': 2, 'y': 3,'w': 4, 'b': 5, 'p': 6, 'c': 7, 'u': 8,'r': 9}).astype(i
    df['bruises']= df['bruises'].map({'f': 0, 't': 1}).astype(int)
    df['odor']= df['odor'].map({'n': 0, 'f': 1, 's': 2, 'y': 3,'l': 4, 'a': 5, 'p': 6,'c': 7, 'm': 8}).astype(int)
    df['gill-attachment']= df['gill-attachment'].map({'f': 0, 'a': 1}).astype(int)
    df['gill-spacing']= df['gill-spacing'].map({'c': 0, 'w': 1}).astype(int)
    df['gill-size']= df['gill-size'].map({'b': 0, 'n': 1}).astype(int)
    df['gill-color']= df['gill-color'].map({'b': 0, 'p': 1, 'w': 2, 'n': 3,'h': 4, 'g': 5, 'u': 6, 'k': 7, 'y': 8,'e': 9, 'o': 10
    df['stalk-shape']= df['stalk-shape'].map({'t': 0, 'e': 1}).astype(int)
    df['stalk-root']= df['stalk-root'].map({'b': 0, 'e': 1, 'c': 2, 'r': 3}).astype(int)
    df['stalk-surface-above-ring']= df['stalk-surface-above-ring'].map({'s': 0, 'k': 1, 'f': 2, 'y': 3}).astype(int)
    df['stalk-surface-below-ring']= df['stalk-surface-below-ring'].map({'s': 0, 'k': 1, 'f': 2, 'y': 3}).astype(int)
    df['stalk-color-above-ring']= df['stalk-color-above-ring'].map({'w': 0, 'p': 1, 'g': 2, 'n': 3,'b': 4, 'o': 5, 'y': 6, 'c': 7
    df['stalk-color-below-ring']= df['stalk-color-below-ring'].map({'w': 0, 'p': 1, 'g': 2, 'n': 3,'b': 4, 'o': 5, 'y': 6, 'c': 7
    df['veil-type']= df['veil-type'].map({'p': 0}).astype(int)
    df['veil-color']= df['veil-color'].map({'w': 0, 'n': 1, 'o': 2, 'y': 3}).astype(int)
    df['ring-number']= df['ring-number'].map({'o': 0, 'n': 1, 't': 2}).astype(int)
    df['ring-type']= df['ring-type'].map({'p': 0, 'e': 1, 'l': 2, 'f': 3, 'n': 4}).astype(int)
    df['spore-print-color']= df['spore-print-color'].map({'w': 0, 'n': 1, 'k': 2, 'h': 3,'r': 4, 'b': 5, 'u': 6, 'y': 7, 'o': 8})
    df['population']= df['population'].map({'v': 0, 'y': 1, 's': 2, 'a': 3, 'n': 4, 'c': 5}).astype(int)
    df['habitat']= df['habitat'].map({'d': 0, 'p': 1, 'g': 2, 'l': 3,'u': 4, 'm': 5, 'w': 6}).astype(int)
    return df
```

دوباره داده ها را split میکنیم :

```
df = mapping(df , 0)
```

```
X = df.drop('poisonous', axis=1)
y = df['poisonous']
# Splitting to training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

حال با 3 تا k متفاوت داده ها را آموزش می دهیم:

```
neigh_3 = KNeighborsClassifier(n_neighbors=3)
neigh_3.fit(X_train, y_train)
y_pred_3 = neigh_3.predict(X_test)
print("accurcy for 3 is ", accuracy_score(y_test,y_pred_3)*100)
```

```
accurcy for 3 is  99.92307692307692
```

```
neigh_5 = KNeighborsClassifier(n_neighbors=5)
neigh_5.fit(X_train, y_train)
y_pred_5 = neigh_5.predict(X_test)
print("accurcy for 5 is ", accuracy_score(y_test,y_pred_5)*100)
```

```
accurcy for 5 is  99.6923076923077
```

```
neigh_10 = KNeighborsClassifier(n_neighbors=10)
neigh_10.fit(X_train, y_train)
y_pred_10 = neigh_10.predict(X_test)
print("accurcy for 10 is ", accuracy_score(y_test,y_pred_10)*100)
```

```
accurcy for 10 is  99.6923076923077
```

و دوباره برای تست دیتا ست را خوانده و عین قبل پیش بینی را انجام می دهیم:

```
df_test=pd.read_csv("Dataset2_Unknown.csv")
# filling missing data
column_names=df_test.columns
for a in column_names:
    df_test[a]=df_test[a].replace('?', np.NaN)

df_test= df_test.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

```
df_test = mapping(df_test ,1 )
```

```
y_pred_test_10 = neigh_10.predict(df_test)
y_pred_test_5 = neigh_5.predict(df_test)
y_pred_test_3 = neigh_3.predict(df_test)
```

```
y_pred_test_10
```

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

سوال 3 :

مانند قبل کار های لازم انجام می شود:

این دفعه همه ستون ها عددی می باشند .

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
from sklearn.naive_bayes import GaussianNB
```

```
df=pd.read_csv("Dataset3.csv")
# filling missing data
column_names=df.columns
for a in column_names:
    df[a]=df[a].replace('?', np.NaN)
df= df.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

```
num_data = df.select_dtypes(include=['int64'])
cat_data = df.select_dtypes(include=['object'])

num_columns = num_data.columns
cat_columns = cat_data.columns

print("Numerical Vaiables: ", num_columns)
print("Categorical Vaiables: ", cat_columns)
```

```
Numerical Vaiables:  Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'slope', 'ca', 'thal', 'disease'],
      dtype='object')
Categorical Vaiables:  Index([], dtype='object')
```

```
# for c in df.columns:
#     print ("---- %s ---" % c)
#     print (df[c].value_counts())
```

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	242.000000	242.000000	242.000000	242.000000	242.000000	242.000000	242.00000	242.000000	242.000000	242.000000	242.000000	242.000000	242.00(
mean	54.797521	0.681818	1.016529	131.921488	247.334711	0.136364	0.53719	150.628099	0.322314	0.997934	1.421488	0.739669	2.32:
std	8.938489	0.466736	1.026483	17.557199	54.119351	0.343886	0.53183	22.195709	0.468331	1.178629	0.621230	1.027827	0.58(
min	34.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.00000	88.000000	0.000000	0.000000	0.000000	0.000000	0.00(
25%	48.000000	0.000000	0.000000	120.000000	211.000000	0.000000	0.00000	139.250000	0.000000	0.000000	1.000000	0.000000	2.00(
50%	56.000000	1.000000	1.000000	130.000000	243.000000	0.000000	1.00000	154.500000	0.000000	0.600000	1.000000	0.000000	2.00(
75%	61.000000	1.000000	2.000000	140.000000	275.750000	0.000000	1.00000	167.750000	1.000000	1.600000	2.000000	1.000000	3.00(
max	77.000000	1.000000	3.000000	192.000000	564.000000	1.000000	2.00000	195.000000	1.000000	6.200000	2.000000	4.000000	3.00(

توابعی که categorize انجام می دهند:

```
def oldpeak_group(x):
    x = int(x)
    if( x == 0 ):
        return "0"
    elif(  x ==1 ):
        return "1"
    elif( x==2 ):
        return "2"
    elif(x ==3):
        return "3"
    if( x == 4 ):
        return "4"
    elif(  x ==5 ):
        return "5"
    else:
        return "6"
```

```
def thalach_group(x):
    x = int(x)
    if( 70 < x < 101 ):
        return "71-100"
    elif( 100 < x < 126 ):
        return "101-125"
    elif( 125< x < 151 ):
        return "126-150"
    elif( 150 < x < 176):
        return "151-175"
    else:
        return ">175"
```

```
def trestbps_group(x):
    x = int(x)
    if( 90 < x < 111 ):
        return "91-110"
    elif( 110 < x < 131 ):
        return "111-130"
    elif( 130< x < 151 ):
        return "131-150"
    elif( 150 < x < 171):
        return "151-170"
    else:
        return ">170"
```

```
def age_group(x):
    x = int(x)
    if( 30 < x < 41 ):
        return "31-40"
    elif( 40 < x < 51 ):
        return "41-50"
    elif( 50 < x < 56 ):
        return "51-55"
    elif( 55 < x < 61 ):
        return "56-60"
    elif( 60 < x < 71 ):
        return "61-70"
    else:
        return ">70"
```

```
def chol_group(x):
    x = int(x)
    if( 100 < x < 201 ):
        return "101-200"
    elif( 200 < x < 301 ):
        return "201-300"
    elif( 300< x < 401 ):
        return "301-400"
    elif( 400 < x < 501):
        return "401-500"
    else:
        return ">500"
```

حال توابع mapping, pre_proccesing:

در بعضی از ستون ها با توجه به اینکه تعداد بعضی از اعدا کم بودن انگار خود به خود categorize بودند و نیاز به کاری نبود:

```python
def pre_processing (df):
    df['chol-group'] = df['chol'].apply(chol_group)
    df['age-group'] = df['age'].apply(age_group)
    df['trestbps-group'] = df['trestbps'].apply(trestbps_group)
    df['thalach-group'] = df['thalach'].apply(thalach_group)
    df['oldpeak-group'] = df['oldpeak'].apply(oldpeak_group)

    df = df.drop(['chol'], axis = 1)
    df = df.drop(['age'], axis = 1)
    df = df.drop(['trestbps'], axis = 1)
    df = df.drop(['thalach'], axis = 1)
    df = df.drop(['oldpeak'], axis = 1)

    return df
```

```python
def mapping (df, flag):
    df['sex'] = df['sex'].astype(int)
    df['cp'] = df['cp'].astype(int)
    df['fbs'] = df['fbs'].astype(int)
    df['restecg'] = df['restecg'].astype(int)
    df['exang'] = df['exang'].astype(int)
    df['slope'] = df['slope'].astype(int)
    df['ca'] = df['ca'].astype(int)
    df['thal'] = df['thal'].astype(int)
    if flag == 0:
        df['disease'] = df['disease'].astype(int)
    df['oldpeak-group'] = df['oldpeak-group'].astype(int)
    df['age-group'] = df['age-group'].map({'31-40': 0, '41-50': 1, '51-55': 2, '56-60': 3, '61-70': 4, '>70': 5}).astype(int)
    df['chol-group'] = df['chol-group'].map({'101-200': 0, '201-300': 1,'301-400': 2, '401-500': 3, '>500': 4}).astype(int)
    df['trestbps-group'] = df['trestbps-group'].map({'91-110': 0, '111-130': 1,'131-150': 2, '151-170': 3, '>170': 4}).astype(int
    df['thalach-group'] = df['thalach-group'].map({'71-100': 0, '101-125': 1,'126-150': 2, '151-175': 3, '>175': 4}).astype(int)

    return df
```

```python
df = pre_processing(df)
df = mapping (df , 0)
```

Split کردن دیتا و ساخت مدل :

```
:  X = df.drop('disease', axis=1)
   y = df['disease']
   # Splitting to training and testing
   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
:  data_clf = GaussianNB()
   data_clf.fit(X_train,y_train)
   y_pred = data_clf.predict(X_test)
   print("Accuracy is ", accuracy_score(y_test,y_pred)*100)

   Accuracy is  79.59183673469387
```

و در نهایت پیشبینی :

```
df_test=pd.read_csv("Dataset3_Unknown.csv")
# filling missing data
column_names=df_test.columns
for a in column_names:
    df_test[a]=df_test[a].replace('?', np.NaN)

df_test= df_test.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

```
df_test = pre_processing(df_test)
df_test = mapping(df_test ,1 )
```

```
y_pred_test=data_clf.predict(df_test)
```