Practical No.2

Data Science and Visualization (Honors Course)

Name:Preeti Shantilal Chaudhari

PRN: 72017986G

Class: TE ENTC 'B'

In this practical we will predict the probability os survival on the basis of age,gender and passenger class in titanic dataset.

Firstly,we will import certain libraries.

In [3]:

```
import numpy as np #This library provide support to the arrays
import seaborn as sns #Library for data visualization
import pandas as pd #for data manipulation
```

In [4]:

```
ds = sns.load_dataset('titanic') #The titanic dataset is already built in seaborn library
```

In [5]:

```
ds.head(10) #Displaying the first 10 rows
```

Out[5]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_mal
0	0	3	male	22.0	1	0	7.2500	S	Third	man	Tru
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fals
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fals
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fals
4	0	3	male	35.0	0	0	8.0500	S	Third	man	Tru
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	Tru
6	0	1	male	54.0	0	0	51.8625	S	First	man	Tru
7	0	3	male	2.0	3	1	21.0750	S	Third	child	Fals
8	1	3	female	27.0	0	2	11.1333	S	Third	woman	Fals
9	1	2	female	14.0	1	0	30.0708	C	Second	child	Fals

In [6]:

```
len(ds) #To find the number of entries
```

Out[6]:

```
891
```

Data Cleaning

In [8]:

```
ds['age'] = ds['age'].fillna(ds['age'].median())
#The Null or Not Available values in the Age Column will be replaced by the median values
```

In [10]:

```
x = ds['age'].values  #x is Input
y = ds['survived']    #y is the output
# We will predicting the number of survived persons(y) on the basis of their age(x).
```

In [11]:

```
x.shape #checking the shape
```

Out[11]:

```
(891,)
```

In [13]:

```
x=x.reshape(-1,1) #Reshaping the column.
# -1 indicates to keep 891 as it is and 1 indicates to add 1 to the shaoe of column.
```

In [14]:

```
x.shape
```

Out[14]:

```
(891, 1)
```

Data Cleaning part has been completed.

Splitting the data for Testing and Training

In [15]:

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

In [16]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=0,test_size=0.25)
#test_size indicates how many samples should be present for test dataset.
#Therefore,25% of the entries will got to test dataset and the rest 75% will go to train da
```

In [17]:

```
len(x_train)
```

Out[17]:

```
668
```

In [18]:

```
len(y_train)
```

Out[18]:

```
668
```

In [19]:

```
len(x_test)
```

Out[19]:

```
223
```

In [21]:

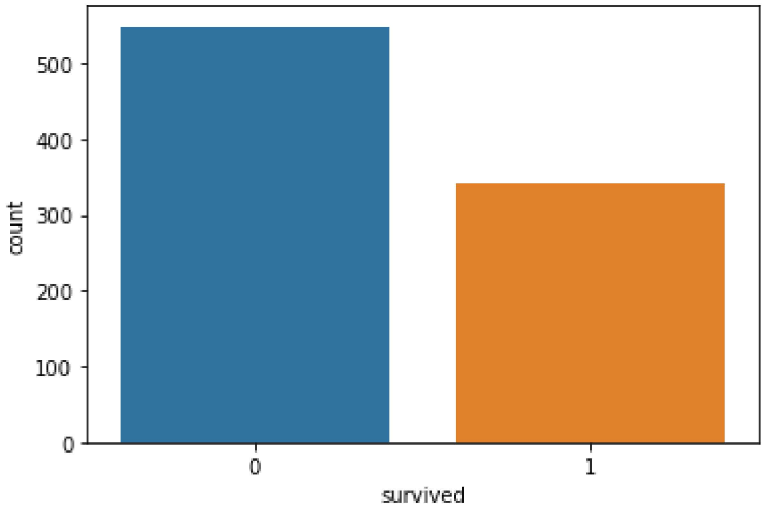
```
from collections import Counter
```

In [22]:

```
Counter(y)
sns.countplot(x=y)
```

Out[22]:

```
<AxesSubplot:xlabel='survived', ylabel='count'>
```



The counter will count the number of individuals who have survived and those who have not survived.

Making the prediction using GaussianNB

In [24]:

```
from sklearn.naive_bayes import GaussianNB
```

In [25]:

```
model = GaussianNB()
```

In [26]:

```
model.fit(x_train,y_train) #Fitting the model
```

Out[26]:

```
GaussianNB()
```

In [28]:

```
y_pred = model.predict_proba(x_test)  #Storing the predicted values in y_pred
```

In [29]:

```
y_pred
```

Out[29]:

```
array([[0.62876821, 0.37123179],
       [0.62876821, 0.37123179],
       [0.51657653, 0.48342347],
       [0.62876821, 0.37123179],
       [0.63148867, 0.36851133],
       [0.62876821, 0.37123179],
       [0.64689984, 0.35310016],
       [0.63625703, 0.36374297],
       [0.6192395 , 0.3807605 ],
       [0.62876821, 0.37123179],
       [0.62264556, 0.37735444],
       [0.64689984, 0.35310016],
       [0.62876821, 0.37123179],
       [0.51657653, 0.48342347],
       [0.61560095, 0.38439905],
       [0.56600603, 0.43399397],
       [0.61172787, 0.38827213],
       [0.59385049, 0.40614951],
```

The probability values are displayed.We need to convert them into binary values of 0 and 1.

In [31]:

```
y_pred=model.predict(x_test)
```

In [32]:

```
y_pred
```

Out[32]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1], dtype=int64)
```

In [33]:

```
y_test
```

Out[33]:

```
495    0
648    0
278    0
31     1
255    1
      ..
167    0
306    1
379    0
742    1
10     1
Name: survived, Length: 223, dtype: int64
```

Determing the accuracy

Accuracy is given by (True Positive+True Negative)/Total no. of entries

In [34]:

```
from sklearn.metrics import accuracy_score
```

In [35]:

```
accuracy_score(y_test, y_pred)
```

Out[35]:

```
0.6547085201793722
```

The accuracy of the model is 65.47%.

Hence we have predicted the survival on the basis of age.

Predicting the survival on the basis of pclass

In [36]:

```
x=ds['pclass'].values
y=ds['survived']
```

In [37]:

```
x=x.reshape(-1,1)
```

In [38]:

```
x.shape
```

Out[38]:

```
(891, 1)
```

In [40]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=0,test_size=0.25)
#Updating all the values once again
```

In [41]:

```
model = GaussianNB()
```

In [42]:

```
model.fit(x_train,y_train)
```

Out[42]:

```
GaussianNB()
```

In [43]:

```
model.predict_proba(x_test)
```

Out[43]:

```
array([[0.7559945 , 0.2440055 ],
       [0.7559945 , 0.2440055 ],
       [0.7559945 , 0.2440055 ],
       [0.29957107, 0.70042893],
       [0.7559945 , 0.2440055 ],
       [0.29957107, 0.70042893],
       [0.29957107, 0.70042893],
       [0.29957107, 0.70042893],
       [0.29957107, 0.70042893],
       [0.7559945 , 0.2440055 ],
       [0.7559945 , 0.2440055 ],
       [0.58864036, 0.41135964],
       [0.7559945 , 0.2440055 ],
       [0.58864036, 0.41135964],
       [0.29957107, 0.70042893],
       [0.7559945 , 0.2440055 ],
       [0.7559945 , 0.2440055 ],
       [0.58864036, 0.41135964],
```

In [44]:

```
y_pred=model.predict(x_test)
```

Determining the Accuracy

In [45]:

```
accuracy_score(y_test, y_pred)
```

Out[45]:

```
0.7085201793721974
```

The accuracy score is 70.85% compared to previous attribute.Hence,we can say that pclass attribute gives more accuarcy.

We will now use two columns and determine the accuracy

In [46]:

```
x = ds[['sex','pclass']]
y = ds['survived']
```

In [47]:

```
x['sex'].head()
```

Out[47]:

```
0      male
1    female
2    female
3    female
4      male
Name: sex, dtype: object
```

We will encode the entries in a certain format.

In [48]:

```
from sklearn.preprocessing import LabelEncoder
```

In [49]:

```
enc=LabelEncoder()
```

In [50]:

```
x['sex']=enc.fit_transform(x['sex']) #Converting the strings into number
```

```
<ipython-input-50-3ffee5c86835>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  x['sex']=enc.fit_transform(x['sex'])
```

In [52]:

```
x['sex'].head() #male-1 female-0
```

Out[52]:

```
0    1
1    0
2    0
3    0
4    1
Name: sex, dtype: int32
```

In [53]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=0,test_size=0.25)
```

In [54]:

```
model=GaussianNB()
```

In [55]:

```
model.fit(x_train,y_train)
```

Out[55]:

```
GaussianNB()
```

In [56]:

```
model.predict_proba(x_test)
```

Out[56]:

```
array([[0.91599965, 0.08400035],
       [0.91599965, 0.08400035],
       [0.91599965, 0.08400035],
       [0.03226176, 0.96773824],
       [0.19452137, 0.80547863],
       [0.60085059, 0.39914941],
       [0.03226176, 0.96773824],
       [0.03226176, 0.96773824],
       [0.60085059, 0.39914941],
       [0.19452137, 0.80547863],
       [0.91599965, 0.08400035],
       [0.1003456 , 0.8996544 ],
       [0.91599965, 0.08400035],
       [0.1003456 , 0.8996544 ],
       [0.03226176, 0.96773824],
       [0.19452137, 0.80547863],
       [0.91599965, 0.08400035],
       [0.83433941, 0.16566059],
```

By using two variables/attributes we can predict the result more accurately;therefore always use subset of two variables which provide most accurate result.

In [57]:

```
y_pred=model.predict(x_test)
```

Determining the accuracy

In [58]:

```
accuracy_score(y_test, y_pred)
```

Out[58]:

```
0.7802690582959642
```

Therefore by using the relevant columns we can predict the result with more accuracy.

In [ ]: