Task-16

GO_STP_5247

Naive Bayes classifiers are built on Bayesian classification methods. These rely on Bayes's theorem, which is an equation describing the relationship of conditional probabilities of statistical quantities.

Create a Model using Naive Bayes classifiers to predict whether a passenger on the titanic would have been survived or not.

```
[1] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sb
    %matplotlib inline
```

Load the dataset

```
[2] titanic = pd.read_csv("/content/titanic_train.csv")
    titanic.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S

✓ 0s completed at 3:45 PM

```
[4] print("Columns present in dataset:\n", titanic.columns) # columns present in dataset
```

```
Columns present in dataset:
 Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
[5] titanic.isnull().sum() # check total null values inside data
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
[6] # fill values of age column

    titanic.fillna(titanic.mean(), inplace = True)
    titanic.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
```

```
[7]  # fill values of Embarked column

     titanic["Embarked"].fillna("S", inplace = True)
     titanic.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         0
dtype: int64
```

```
[8]  # drop Cabin column because it has lot of null values. 687/891

     drop_cabin = titanic.isnull().sum()[titanic.isnull().sum() > (50/100 * titanic.shape[0])]
     drop_cabin
```

```
Cabin    687
dtype: int64
```

```
[9]  drop_cabin.index
```

```
Index(['Cabin'], dtype='object')
```

```
[10] titanic.drop(drop_cabin.index, axis = 1, inplace = True)
     titanic.isnull().sum()
```

[11]
```
titanic.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

[12]
```
titanic.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.033207	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.069809	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.331339	0.083081	0.018443	-0.549500
Age	0.033207	-0.069809	-0.331339	1.000000	-0.232625	-0.179191	0.091566
SibSp	-0.057527	-0.035322	0.083081	-0.232625	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.179191	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.091566	0.159651	0.216225	1.000000

[13]
```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

[14]
```
# create a new column Family size by adding SibSp and Parch

titanic["FamilySize"] = titanic["SibSp"] + titanic["Parch"]
titanic.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	FamilySize
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S	1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S	0

```
[15] titanic.drop(["SibSp", "Parch"], axis = 1, inplace = True)
     titanic.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Embarked	FamilySize
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	A/5 21171	7.2500	S	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	PC 17599	71.2833	C	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	STON/O2. 3101282	7.9250	S	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	113803	53.1000	S	1
4	5	0	3	Allen, Mr. William Henry	male	35.0	373450	8.0500	S	0

```
[16] titanic.corr()
```

	PassengerId	Survived	Pclass	Age	Fare	FamilySize
PassengerId	1.000000	-0.005007	-0.035144	0.033207	0.012658	-0.040143
Survived	-0.005007	1.000000	-0.338481	-0.069809	0.257307	0.016639
Pclass	-0.035144	-0.338481	1.000000	-0.331339	-0.549500	0.065997
Age	0.033207	-0.069809	-0.331339	1.000000	0.091566	-0.248512
Fare	0.012658	0.257307	-0.549500	0.091566	1.000000	0.217138
FamilySize	-0.040143	0.016639	0.065997	-0.248512	0.217138	1.000000

```
[17] # filtered alone persons/passengers

     titanic["Alone"] = [0 if titanic["FamilySize"][i] > 0 else 1 for i in titanic.index]
     titanic.head()
```

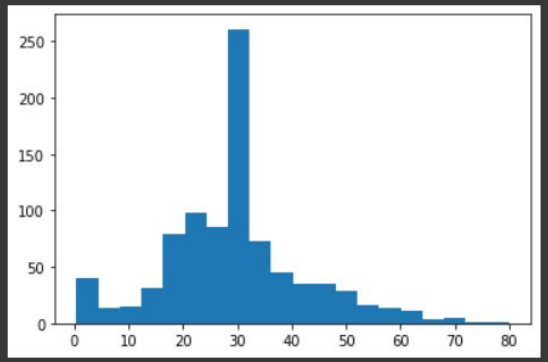
```
[18] titanic.corr()
```

	PassengerId	Survived	Pclass	Age	Fare	FamilySize	Alone
PassengerId	1.000000	-0.005007	-0.035144	0.033207	0.012658	-0.040143	0.057462
Survived	-0.005007	1.000000	-0.338481	-0.069809	0.257307	0.016639	-0.203367
Pclass	-0.035144	-0.338481	1.000000	-0.331339	-0.549500	0.065997	0.135207
Age	0.033207	-0.069809	-0.331339	1.000000	0.091566	-0.248512	0.179775
Fare	0.012658	0.257307	-0.549500	0.091566	1.000000	0.217138	-0.271832
FamilySize	-0.040143	0.016639	0.065997	-0.248512	0.217138	1.000000	-0.690922
Alone	0.057462	-0.203367	0.135207	0.179775	-0.271832	-0.690922	1.000000

Filtered out survived ratio according to conditions and visualize them

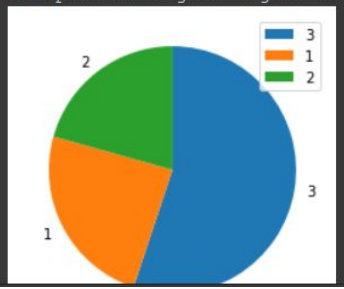
```
[19] # sex ratio of passengers

     sb.countplot(x = "Sex", data = titanic);
```

```
# age distribution

plt.hist(x = titanic["Age"], bins = 20);
```



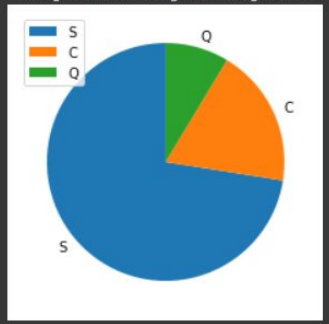

```
[21] # passenger class
     x = titanic["Pclass"].value_counts()
     plt.pie(x, labels = x.index, startangle = 90, counterclock = False);
     plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f9fb9f98ed0>
```

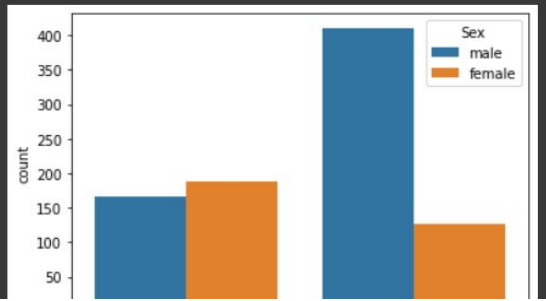


```
#Embarked
y = titanic["Embarked"].value_counts()
plt.pie(y, labels = y.index, startangle = 90, counterclock = True);
plt.legend()
```

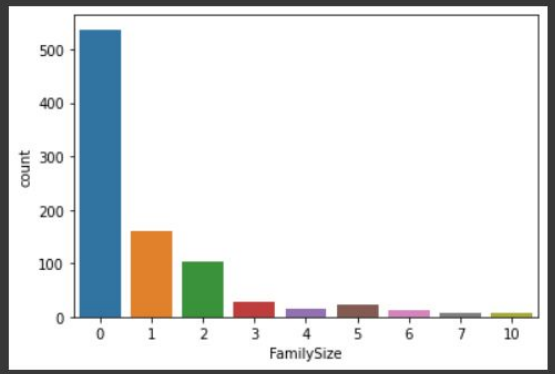
```
<matplotlib.legend.Legend at 0x7f9fb8661e50>
```



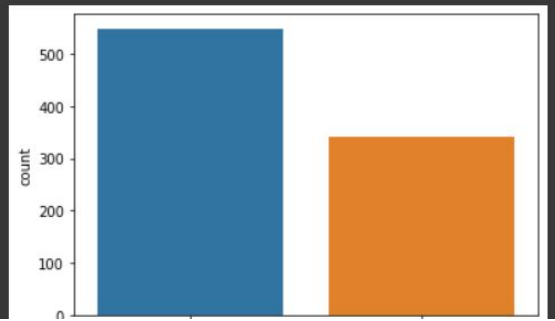

```
[23] # survive rate of alone person according to their sex

     sb.countplot(x = "Alone", hue = "Sex", data = titanic);
```



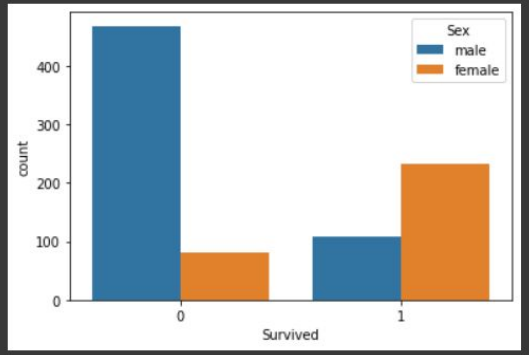
```
# survive rate of family

sb.countplot(x = "FamilySize", data = titanic);
```



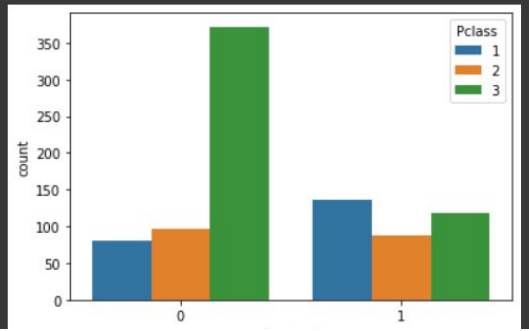

```
[25] # total survived passengers

sb.countplot(x = "Survived", data = titanic);
```



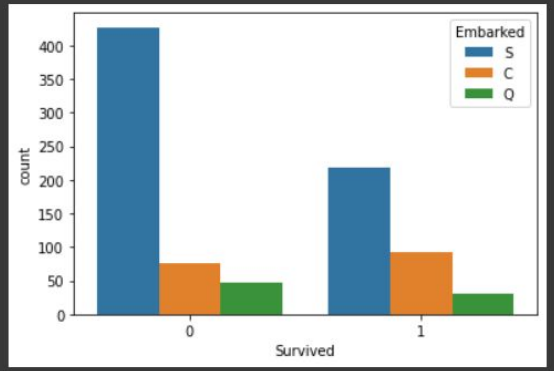
[26]
```
# survived ratio according to sex
sb.countplot(x = "Survived", hue = "Sex", data = titanic);
```



[27]
```
# accoring to pclass
sb.countplot(x = "Survived", hue = "Pclass", data = titanic);
```

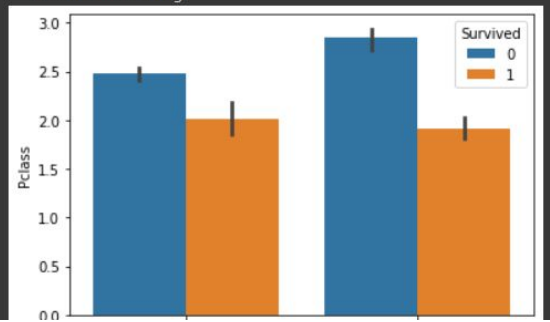


```
[28] # according to embarked
     sb.countplot(x = "Survived", hue = "Embarked", data = titanic);
```




```
[29] # accroding to sex and passenger class
     sb.barplot("Sex", "Pclass", hue = "Survived", data = titanic);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only
  FutureWarning
```



Label Encoding for Sex and Embarked

```
[30] from sklearn.preprocessing import LabelEncoder

     le = LabelEncoder()
     titanic["Sex"] = le.fit_transform(titanic["Sex"])
     titanic["Embarked"] = le.fit_transform(titanic["Embarked"])
     print("Encoded values for Sex:", titanic["Sex"].unique())
     print("Encoded values for Embarked:", titanic["Embarked"].unique())
```

```
Encoded values for Sex: [1 0]
Encoded values for Embarked: [2 0 1]
```

```
[31] titanic.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Embarked	FamilySize	Alone
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	A/5 21171	7.2500	2	1	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	PC 17599	71.2833	0	1	0
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	STON/O2. 3101282	7.9250	2	0	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	113803	53.1000	2	1	0
4	5	0	3	Allen, Mr. William Henry	1	35.0	373450	8.0500	2	0	1

Features and Target

```
[32] features = titanic[["Pclass", "Sex", "Age", "Fare", "Embarked", "FamilySize", "Alone"]]
     target = titanic["Survived"]
```

Divide data for training and testing

```
[33] from sklearn.model_selection import train_test_split

     xtrain, xtest, ytrain, ytest = train_test_split(features, target, test_size = 0.3, random_state = 45)
     print("Shape of xtrain:", xtrain.shape)
     print("Shape of ytrain:", ytrain.shape)
     print("Shape of xtest:", xtest.shape)
     print("Shape of ytest:", ytest.shape)
```

```
Shape of xtrain: (623, 7)
Shape of ytrain: (623,)
Shape of xtest: (268, 7)
Shape of ytest: (268,)
```

Create a model and train the data

```
[34] from sklearn.naive_bayes import GaussianNB

     gnb = GaussianNB()
     gnb.fit(xtrain, ytrain)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

Test the testing data and make prediction

```
[35] ypred = gnb.predict(xtest)
     print("Prediction made by model:\n", ypred)
```

```
Prediction made by model:
 [0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
```

```
[35] ypred = gnb.predict(xtest)
     print("Prediction made by model:\n", ypred)
```

```
Prediction made by model:
 [0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
 1 0 1 0 0 0 0 0 0 1 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0
 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0 0 0 1 1 0 0
 0 1 1 0 1 0 0 0 0 1 0 1 0 1 1 1 1 0 0 1 1 1 1 0 0 1 0 1 1 0 1 0 1 0 1 1 1
 0 0 1 1 1 0 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1
 0 1 0 1 0 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0
 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 1 0 0 0 1 1
 0 0 0 1 0 0 0 1 0]
```

Confusion Matrix and Accuracy

```
[36] from sklearn.metrics import confusion_matrix, accuracy_score

     matrix = confusion_matrix(ytest, ypred)
     print("Confusion Matrix of a model:\n", matrix)
```

```
Confusion Matrix of a model:
 [[154  24]
 [ 20  70]]
```

```
accuracy = accuracy_score(ytest, ypred)
print("Accuracy of mdoel: {}%".format(accuracy*100))
```

```
Accuracy of mdoel: 83.5820895522388%
```