In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
titanic = pd.read_csv(r"E:\fsds_course\17th - ML\TITANIC PROJECT\DATASET\ti
titanic.tail()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	

Performing Data Cleaning and Analysis

1. Understanding meaning of each column: Data Dictionary: Variable Description

Survived - Survived (1) or died (0) Pclass - Passenger’s class (1 = 1st, 2 = 2nd, 3 = 3rd) Name - Passenger’s name Sex - Passenger’s sex Age - Passenger’s age SibSp - Number of siblings/spouses aboard Parch - Number of parents/children aboard (Some children travelled only with a nanny, therefore parch=0 for them.) Ticket - Ticket number Fare - Fare Cabin - Cabin Embarked - Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)

2. Analysing which columns are completely useless in predicting the survival and deleting them Note - Don't just delete the columns because you are not finding it useful. Or focus is not on deleting the columns. Our focus is on analysing how each column is affecting the result or the prediction and in accordance with that deciding whether to keep the column or to delete the column or fill the null values of the column by some values and if yes, then what values.

In [3]:
```
titanic.describe()
```

Out[3]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	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	14.526497	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	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [4]:
```
#Name column can never decide survival of a person, hence we can safely del

del titanic["Name"]
titanic.head()
```

Out[4]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Er
0	1	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	male	35.0	0	0	373450	8.0500	NaN	

In [5]:
```
del titanic["Ticket"]
titanic.head()
```

Out[5]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	1	0	3	male	22.0	1	0	7.2500	NaN	S
1	2	1	1	female	38.0	1	0	71.2833	C85	C
2	3	1	3	female	26.0	0	0	7.9250	NaN	S
3	4	1	1	female	35.0	1	0	53.1000	C123	S
4	5	0	3	male	35.0	0	0	8.0500	NaN	S

In [6]:
```
del titanic["Fare"]
titanic.head()
```

Out[6]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Cabin	Embarked
0	1	0	3	male	22.0	1	0	NaN	S
1	2	1	1	female	38.0	1	0	C85	C
2	3	1	3	female	26.0	0	0	NaN	S
3	4	1	1	female	35.0	1	0	C123	S
4	5	0	3	male	35.0	0	0	NaN	S

In [7]:
```
del titanic["Cabin"]
titanic.head()
```

Out[7]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Embarked
0	1	0	3	male	22.0	1	0	S
1	2	1	1	female	38.0	1	0	C
2	3	1	3	female	26.0	0	0	S
3	4	1	1	female	35.0	1	0	S
4	5	0	3	male	35.0	0	0	S

In [8]:
```
# Changing Value for "Male, Female" string values to numeric values , male=

def getNumber(str):
    if str=="male":
        return 1
    else:
        return 2
titanic["Gender"]=titanic["Sex"].apply(getNumber)

#We have created a new column called "Gender" and
#filling it with values 1,2 based on the values of sex column

titanic.head()
```

Out[8]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Embarked	Gender
0	1	0	3	male	22.0	1	0	S	1
1	2	1	1	female	38.0	1	0	C	2
2	3	1	3	female	26.0	0	0	S	2
3	4	1	1	female	35.0	1	0	S	2
4	5	0	3	male	35.0	0	0	S	1

In [9]:
```
#Deleting Sex column, since no use of it now
del titanic["Sex"]
titanic.head()
```

Out[9]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Embarked	Gender
0	1	0	3	22.0	1	0	S	1
1	2	1	1	38.0	1	0	C	2
2	3	1	3	26.0	0	0	S	2
3	4	1	1	35.0	1	0	S	2
4	5	0	3	35.0	0	0	S	1

In [10]:
```
titanic.isnull().sum()
```

Out[10]:
```
PassengerId      0
Survived         0
Pclass           0
Age            177
SibSp            0
Parch            0
Embarked         2
Gender           0
dtype: int64
```

Fill the null values of the Age column. Fill mean Survived age(mean age of the survived people) in the column where the person has survived and mean not Survived age (mean age of the people who have not survived) in the column where person has not survived

In [11]:
```
meanS = titanic[titanic.Survived==1].Age.mean()
meanS
```

Out[11]:
```
28.343689655172415
```

Creating a new "Age" column , filling values in it with a condition if goes True then given values (here meanS) is put in place of last values else nothing happens, simply the values are copied from the "Age" column of the dataset

In [12]:
```
titanic["age"]=np.where(pd.isnull(titanic.Age) & titanic["Survived"]==1, me
titanic.head()
```

Out[12]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Embarked	Gender	age
0	1	0	3	22.0	1	0	S	1	22.0
1	2	1	1	38.0	1	0	C	2	38.0
2	3	1	3	26.0	0	0	S	2	26.0
3	4	1	1	35.0	1	0	S	2	35.0
4	5	0	3	35.0	0	0	S	1	35.0

In [13]:
```
titanic.isnull().sum()
```

Out[13]:
```
PassengerId      0
Survived         0
Pclass           0
Age            177
SibSp            0
Parch            0
Embarked         2
Gender           0
age            125
dtype: int64
```

In [14]:
```
# Finding the mean age of "Not Survived" people

meanNS=titanic[titanic.Survived==0].Age.mean()
meanNS
```

Out[14]:
```
30.62617924528302
```

In [15]:
```
titanic.age.fillna(meanNS,inplace=True)
titanic.head()
```

Out[15]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Embarked	Gender	age
0	1	0	3	22.0	1	0	S	1	22.0
1	2	1	1	38.0	1	0	C	2	38.0
2	3	1	3	26.0	0	0	S	2	26.0
3	4	1	1	35.0	1	0	S	2	35.0
4	5	0	3	35.0	0	0	S	1	35.0

In [16]:
```
titanic.isnull().sum()
```

Out[16]:
```
PassengerId      0
Survived         0
Pclass           0
Age            177
SibSp            0
Parch            0
Embarked         2
Gender           0
age              0
dtype: int64
```

In [17]:
```
del titanic["Age"]
titanic.head()
```

Out[17]:

	PassengerId	Survived	Pclass	SibSp	Parch	Embarked	Gender	age
0	1	0	3	1	0	S	1	22.0
1	2	1	1	1	0	C	2	38.0
2	3	1	3	0	0	S	2	26.0
3	4	1	1	1	0	S	2	35.0
4	5	0	3	0	0	S	1	35.0

We want to check if "Embarked" column is is important for analysis or not, that is whether survival of the person depends on the Embarked column value or not

In [18]:
```
# Finding the number of people who have survived
# given that they have embarked or boarded from a particular port

survivedQ = titanic[titanic.Embarked == 'Q'][titanic.Survived==1].shape[0]
survivedC = titanic[titanic.Embarked == 'C'][titanic.Survived==1].shape[0]
survivedS = titanic[titanic.Embarked == 'S'][titanic.Survived==1].shape[0]
print(survivedQ)
print(survivedC)
print(survivedS)
```

```
30
93
217

C:\Users\rutik\AppData\Local\Temp\ipykernel_14952\1289359436.py:4: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  survivedQ = titanic[titanic.Embarked == 'Q'][titanic.Survived==1].shape
[0]
C:\Users\rutik\AppData\Local\Temp\ipykernel_14952\1289359436.py:5: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  survivedC = titanic[titanic.Embarked == 'C'][titanic.Survived==1].shape
[0]
C:\Users\rutik\AppData\Local\Temp\ipykernel_14952\1289359436.py:6: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  survivedS = titanic[titanic.Embarked == 'S'][titanic.Survived==1].shape
[0]
```

In [19]:
```
survivedQ = titanic[titanic.Embarked == 'Q'][titanic.Survived ==0].shape[0]
survivedC = titanic[titanic.Embarked == 'C'][titanic.Survived ==0].shape[0]
survivedS = titanic[titanic.Embarked == 'S'][titanic.Survived ==0].shape[0]
print(survivedQ)
print(survivedC)
print(survivedS)
```

```
C:\Users\rutik\AppData\Local\Temp\ipykernel_14952\3782576952.py:1: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  survivedQ = titanic[titanic.Embarked == 'Q'][titanic.Survived ==0].shape
[0]
C:\Users\rutik\AppData\Local\Temp\ipykernel_14952\3782576952.py:2: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  survivedC = titanic[titanic.Embarked == 'C'][titanic.Survived ==0].shape
[0]

47
75
427

C:\Users\rutik\AppData\Local\Temp\ipykernel_14952\3782576952.py:3: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  survivedS = titanic[titanic.Embarked == 'S'][titanic.Survived ==0].shape
[0]
```

As there are significant changes in the survival rate based on which port the passengers aboard the ship. We cannot delete the whole embarked column(It is useful). Now the Embarked column has some null values in it and hence we can safely say that deleting some rows from total rows will not affect the result. So rather than trying to fill those null values with some vales. We can simply remove them.

In [20]:
```
titanic.dropna(inplace=True)
titanic.head()
```

Out[20]:

	PassengerId	Survived	Pclass	SibSp	Parch	Embarked	Gender	age
0	1	0	3	1	0	S	1	22.0
1	2	1	1	1	0	C	2	38.0
2	3	1	3	0	0	S	2	26.0
3	4	1	1	1	0	S	2	35.0
4	5	0	3	0	0	S	1	35.0

In [21]:
```
titanic.isnull().sum()
```

Out[21]:
```
PassengerId    0
Survived       0
Pclass         0
SibSp          0
Parch          0
Embarked       0
Gender         0
age            0
dtype: int64
```

In [22]:
```
#Renaming "age" and "gender" columns
titanic.rename(columns={'age':'Age'},inplace=True)
titanic.head()
```

Out[22]:

	PassengerId	Survived	Pclass	SibSp	Parch	Embarked	Gender	Age
0	1	0	3	1	0	S	1	22.0
1	2	1	1	1	0	C	2	38.0
2	3	1	3	0	0	S	2	26.0
3	4	1	1	1	0	S	2	35.0
4	5	0	3	0	0	S	1	35.0

In [23]:
```
titanic.rename(columns={'Gender':'Sex'}, inplace=True)
titanic.head()
```

Out[23]:

	PassengerId	Survived	Pclass	SibSp	Parch	Embarked	Sex	Age
0	1	0	3	1	0	S	1	22.0
1	2	1	1	1	0	C	2	38.0
2	3	1	3	0	0	S	2	26.0
3	4	1	1	1	0	S	2	35.0
4	5	0	3	0	0	S	1	35.0

In [24]:
```
def getEmb(str):
    if str=="S":
        return 1
    elif str=='Q':
        return 2
    else:
        return 3

titanic["Embark"]=titanic["Embarked"].apply(getEmb)
titanic.head()
```

Out[24]:

	PassengerId	Survived	Pclass	SibSp	Parch	Embarked	Sex	Age	Embark
0	1	0	3	1	0	S	1	22.0	1
1	2	1	1	1	0	C	2	38.0	3
2	3	1	3	0	0	S	2	26.0	1
3	4	1	1	1	0	S	2	35.0	1
4	5	0	3	0	0	S	1	35.0	1

In [25]:
```
del titanic['Embarked']
titanic.rename(columns={'Embark': 'Embarked'},inplace=True)
titanic.head()
```

Out[25]:

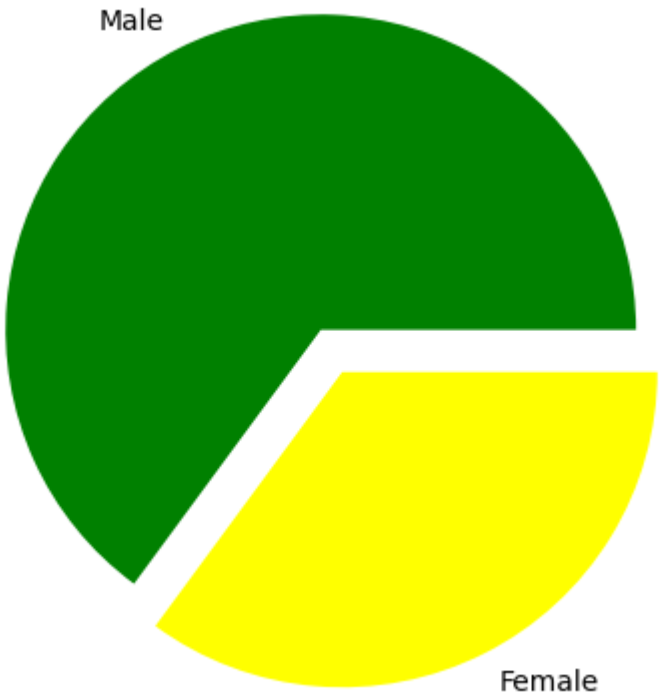
	PassengerId	Survived	Pclass	SibSp	Parch	Sex	Age	Embarked
0	1	0	3	1	0	1	22.0	1
1	2	1	1	1	0	2	38.0	3
2	3	1	3	0	0	2	26.0	1
3	4	1	1	1	0	2	35.0	1
4	5	0	3	0	0	1	35.0	1

In [26]:
```
#Drawing a pie chart for number of males and females aboard
import matplotlib.pyplot as plt
from matplotlib import style

males = (titanic['Sex']== 1).sum()
#Summing up all the values of column gender with a
#condition for male and similary for females

females = (titanic['Sex']== 2).sum()
print(males)
print(females)
p=[males ,females]
plt.pie(p,  #giving array
       labels=['Male','Female'], #Corres# Corresponding colors
        colors=['green', 'yellow'],# Corresponding colors
        explode=(0.15,0),#How much the gap should me there between the pies
        startangle=0 )  #what start angle should be given
plt.axis('equal')
plt.show()
```

```
577
312
```



In [27]:
```
# More Precise Pie Chart
Males = titanic[titanic.Sex==1][titanic.Survived==1].shape[0]
print(Males)

MaleN = titanic[titanic.Sex==1][titanic.Survived==0].shape[0]
print(MaleN)

Females = titanic[titanic.Sex==2][titanic.Survived==1].shape[0]
print(Females)

FemaleN = titanic[titanic.Sex==2][titanic.Survived==1].shape[0]
print(FemaleN)
```

```
109
468
231
231

C:\Users\rutik\AppData\Local\Temp\ipykernel_14952\1050023975.py:2: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  Males = titanic[titanic.Sex==1][titanic.Survived==1].shape[0]
C:\Users\rutik\AppData\Local\Temp\ipykernel_14952\1050023975.py:5: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  MaleN = titanic[titanic.Sex==1][titanic.Survived==0].shape[0]
C:\Users\rutik\AppData\Local\Temp\ipykernel_14952\1050023975.py:8: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  Females = titanic[titanic.Sex==2][titanic.Survived==1].shape[0]
C:\Users\rutik\AppData\Local\Temp\ipykernel_14952\1050023975.py:11: UserWa
rning: Boolean Series key will be reindexed to match DataFrame index.
  FemaleN = titanic[titanic.Sex==2][titanic.Survived==1].shape[0]
```

In [28]:
```
chart=[Males,MaleN,Females,FemaleN]
colors=['lightskyblue','yellowgreen','yellow','orange']
lables=["Survived Male","Not Survived Male","Survived Female","Not Survived
explode=[0,0.05,0,0.1]
plt.pie(chart,labels=lables,colors=colors,explode=explode,startangle=100,co
```

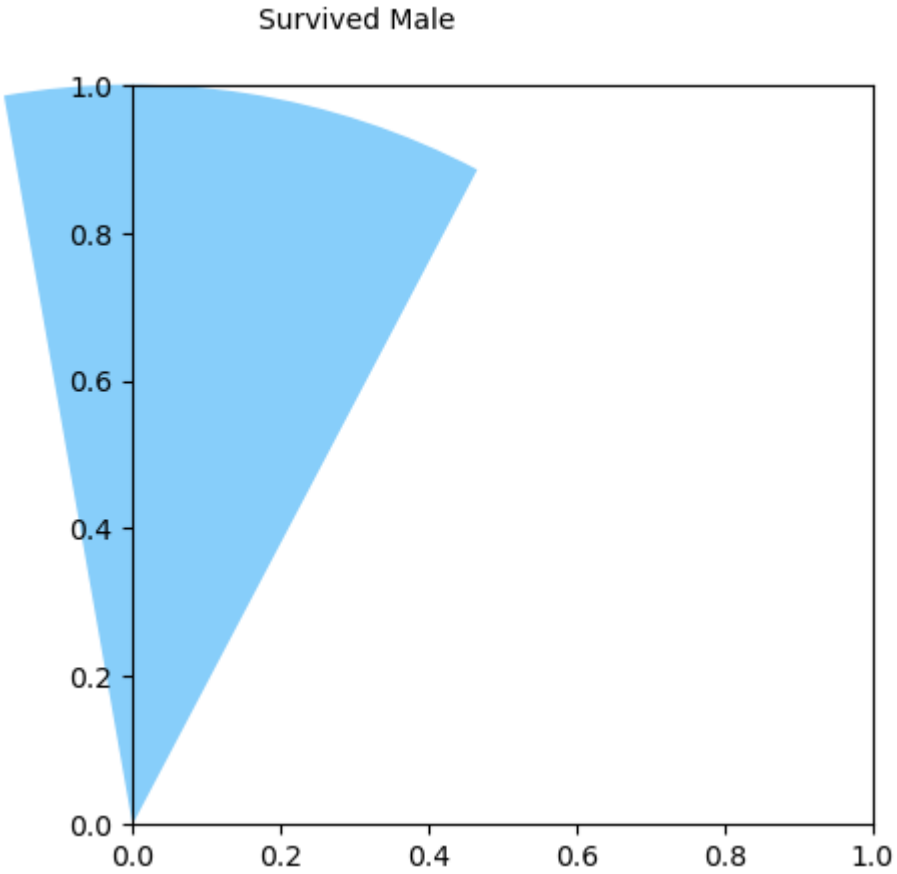
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[28], line 5
      3 lables=["Survived Male","Not Survived Male","Survived Female","Not Survived Female"]
      4 explode=[0,0.05,0,0.1]
----> 5 plt.pie(chart,labels=lables,colors=colors,explode=explode,startangle=100,counterclock=False,autopct="%")

File C:\ProgramData\anaconda3\lib\site-packages\matplotlib\pyplot.py:2772, in pie(x, explode, labels, colors, autopct, pctdistance, shadow, labeldistance, startangle, radius, counterclock, wedgeprops, textprops, center, frame, rotatelabels, normalize, hatch, data)
   2765 @_copy_docstring_and_deprecators(Axes.pie)
   2766 def pie(
   2767         x, explode=None, labels=None, colors=None, autopct=None,
   (...)
   2770         textprops=None, center=(0, 0), frame=False,
   2771         rotatelabels=False, *, normalize=True, hatch=None, data=None):
-> 2772     return gca().pie(
   2773         x, explode=explode, labels=labels, colors=colors,
   2774         autopct=autopct, pctdistance=pctdistance, shadow=shadow,
   2775         labeldistance=labeldistance, startangle=startangle,
   2776         radius=radius, counterclock=counterclock,
   2777         wedgeprops=wedgeprops, textprops=textprops, center=center,
   2778         frame=frame, rotatelabels=rotatelabels, normalize=normalize,
   2779         hatch=hatch, **({"data": data} if data is not None else {}))

File C:\ProgramData\anaconda3\lib\site-packages\matplotlib\__init__.py:1442, in _preprocess_data.<locals>.inner(ax, data, *args, **kwargs)
   1439 @functools.wraps(func)
   1440 def inner(ax, *args, data=None, **kwargs):
   1441     if data is None:
-> 1442         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1444     bound = new_sig.bind(ax, *args, **kwargs)
   1445     auto_label = (bound.arguments.get(label_namer)
   1446                   or bound.kwargs.get(label_namer))

File C:\ProgramData\anaconda3\lib\site-packages\matplotlib\axes\_axes.py:3284, in Axes.pie(self, x, explode, labels, colors, autopct, pctdistance, shadow, labeldistance, startangle, radius, counterclock, wedgeprops, textprops, center, frame, rotatelabels, normalize, hatch)
   3282 yt = y + pctdistance * radius * math.sin(thetam)
   3283 if isinstance(autopct, str):
-> 3284     s = autopct % (100. * frac)
   3285 elif callable(autopct):
   3286     s = autopct(100. * frac)

ValueError: incomplete format
```



In [ ]:

In [ ]: