PYTHON NOTEBOOK

ON

TITANIC DATA ANALYSIS

UNIVERSITY OF GLOUCESTERSHIRE

DATA SCIENCE

PRESENTED BY:

PRITIBEN SORATHIYA

ABHINAV DUGGYALA

RAKIB AHMED NOEL

Table of Contents

Introduction 3
Data Description 3
Question and problem definition 4
Acquire data 4
Analyse by describing data 5
Data cleansing 6
Data Analysis 8

Introduction

The notebook has been designed with a workflow to solve data science competitions at a learning stage. The main focus of this notebook is step-by-step workflow, explaining each step and the analysis of the data.

The data (and explanation of the data) can be obtained from: https://www.kaggle.com/datasets/shitaljagtap/titanic-dataset

Firstly, raw excel (.xls) data will be loaded into a Python series.

Secondly, data will be cleaned for any missing data and find out the usable data which can be analysed.

Thirdly, investigation will be carried out to find the pattern of death and survival. This will be done along with visualisations. Visualisation of the data makes generating a hypothesis.

Finally, a prediction model using algorithms used to predict how accurate the models work on the titanic dataset.

Comments have been done to understand the code and how the code works are above the code with a leading hashtag(#).

Data Description

(from https://www.kaggle.com/c/titanic)

- **survival:** Survival (0 = No; 1 = Yes)
- **pclass:** Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
- **name:** Name
- **sex:** Sex
- **age:** Age
- **sibsp:** Number of Siblings/Spouses Aboard
- **parch:** Number of Parents/Children Aboard
- **ticket:** Ticket Number
- **fare:** Passenger Fare
- **cabin:** Cabin
- **embarked:** Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

Question and problem definition

If we want to analyse the data, we need to understand the pattern of the data. We have analysed based on some questions which we based as our analytics. These question answers are based on survival and death rate pattern. Below is the questions that arises to do the analysis:

1. Was there any age which has survived most?
2. Was there any class identification to survive?
3. Which gender survived most?
4. Was there and gender and age combination on survival rate?
5. Was there any class and gender combination on survival rate?

Acquire data

As a first step we need to load data as part of our notebook. The packages and libraries has been included at the beginning of the notebook. Data visualisation libraries are also included at the beginning of the notebook. We have included warning simplifier to ignore common errors. Here we have imported data and view the first 10 rows of the data to check if the data was correctly loaded.

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#used to ignore warning
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
get_ipython().run_line_magic('matplotlib', 'inline')
#import dataset
titanic_data = pd.read_excel("titanic_data.xls")
#view he first 10 rows of the dataset
titanic_data.head(10)
```

Out[1]:

	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
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

Analyse by describing data

Before the analysis we check the current working directory to take the backup of the script for future use.

```
In [2]: import os
        os.getcwd()
Out[2]: '/home/noel/python_project'
```

```
In [3]: len(titanic_data)
Out[3]: 891
```

```
In [4]: titanic_data.index
Out[4]: RangeIndex(start=0, stop=891, step=1)
```

```
In [5]: titanic_data.columns
Out[5]: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
               'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
              dtype='object')
```

```
In [6]: titanic_data.info()
        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 891 entries, 0 to 890
        Data columns (total 12 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          714 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  Cabin        204 non-null    object 
         11  Embarked     889 non-null    object 
        dtypes: float64(2), int64(5), object(5)
        memory usage: 83.7+ KB
```

```
In [7]: titanic_data.dtypes
Out[7]: PassengerId      int64
        Survived         int64
        Pclass           int64
        Name            object
        Sex             object
        Age            float64
        SibSp            int64
        Parch            int64
        Ticket          object
        Fare           float64
        Cabin           object
        Embarked        object
        dtype: object
```

Then we check the length of the data, index, data information, column values, data information and data types. For any kind of cleansing or data manipulation we need to have a clear view of data. So before doing any operation we have analysed this type of information.

Described the dataset that we have imported. The description and pattern of the dataset is described by using this description.

In [8]:
```
titanic_data.describe()
```

Out[8]:

	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

Data cleansing

Before going for any analysis, we first checked the duplicate data and then clean the data for any kind of inconsistency and irrelevant data.

In [9]:
```
# Identify and remove duplicate entries
titanic_data_duplicates = titanic_data.duplicated()
print('Number of duplicate entries is/are {}'.format(titanic_data_duplicates.sum()))
```

```
Number of duplicate entries is/are 0
```

Then we need to check if there is any missing values. There may be garbage values or null values. Before doing any cleansing we need to check if there is any null values.

In [10]:
```
titanic_data.isna()
```

Out[10]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

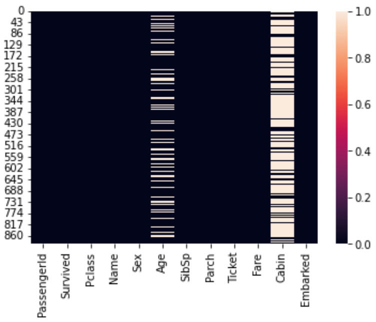
891 rows × 12 columns

In [11]:
```
titanic_data.isna().sum()
```

Out[11]:
```
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
```

We can see from the value there is some NULL values in age, cabin and embarked. This can be visualised using heatmap.

```
In [12]: sns.heatmap(titanic_data.isna())
Out[12]: <AxesSubplot:>
```



Above we are getting values in the form of true or false. False means no null values true means null value is present. But we don't know how many exact values are null.

```
In [13]: (titanic_data['Age'].isna().sum()/len(titanic_data['Age']))*100
Out[13]: 19.865319865319865

In [14]: titanic_data['Age'].fillna(titanic_data['Age'].mean(),inplace=True)

In [15]: titanic_data['Age'].isna().sum()
Out[15]: 0

In [16]: (titanic_data['Cabin'].isna().sum()/len(titanic_data['Cabin']))*100
Out[16]: 77.10437710437711
```

Either to discard any null values we check percentage of null value if percentage is more than 30% there will be problem in imputing those null values.

```
In [17]: titanic_data.drop(['PassengerId', 'Ticket', 'Cabin'], axis=1,inplace=True)
         titanic_data.head()
Out[17]:
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	S
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	S
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	S

```
In [18]: titanic_data.isnull().sum()
Out[18]: Survived    0
         Pclass      0
         Name        0
         Sex         0
         Age         0
         SibSp       0
         Parch       0
         Fare        0
         Embarked    2
         dtype: int64
```

Here we deleted Passenger ID, ticket and cabin column as this will not be part of our analysis. Also the cabin column contains null values.

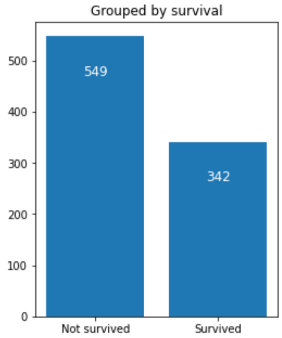
Data Analysis

Then we check the total number of people survived and total number of people died.

In [19]:
```
survived_count = titanic_data.groupby('Survived')['Survived'].count()
survived_count
```

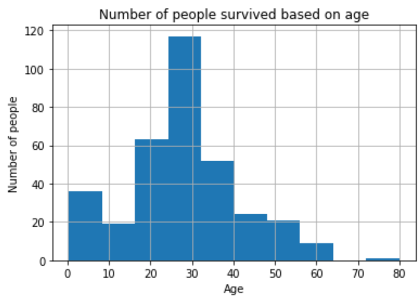
Out[19]:
```
Survived
0    549
1    342
Name: Survived, dtype: int64
```

In [20]:
```
plt.figure(figsize=(4,5))
plt.bar(survived_count.index, survived_count.values)
plt.title('Grouped by survival')
plt.xticks([0,1],['Not survived', 'Survived'])
for i, value in enumerate(survived_count.values):
    plt.text(i, value-70, str(value), fontsize=12, color='white',
             horizontalalignment='center', verticalalignment='center')
plt.show()
```

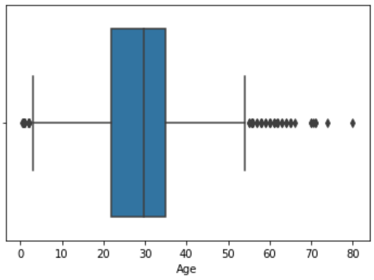


As per data description and data analysis we can see that 342 people were survived on that journey and 549 people died. If we go for more segregation as part of out analysis, we can find that young age survival rate was high which range of age is from 0 to 30 and above 6o years of age survival rate is very low.

In [22]:
```
titanic_data[titanic_data.Survived==1]["Age"].hist()
plt.title("Number of people survived based on age")
plt.xlabel("Age")
plt.ylabel("Number of people")
plt.show()
```



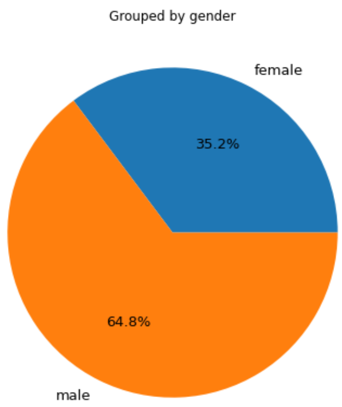
```
In [23]: sns.boxplot(titanic_data['Age'])
Out[23]: <AxesSubplot:xlabel='Age'>
```



Below is the number of male and female on Titanic.

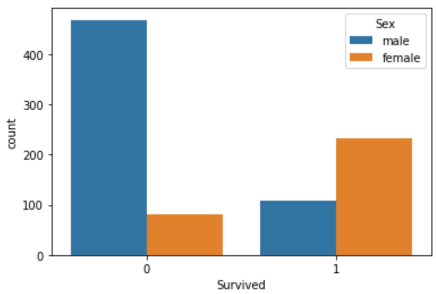
```
In [25]: sex_count = titanic_data.groupby('Sex')['Sex'].count()
         sex_count
Out[25]: Sex
         female    314
         male      577
         Name: Sex, dtype: int64
```

```
In [26]: plt.figure(figsize=(7,7))
         plt.title('Grouped by gender')
         plt.pie(sex_count.values, labels=['female', 'male'],
          autopct='%1.1f%%', textprops={'fontsize':13})
         plt.show()
```



Next step of analysis was done on how many people survived based on class. Identify if there was any class factor or the class factor was not accountable.

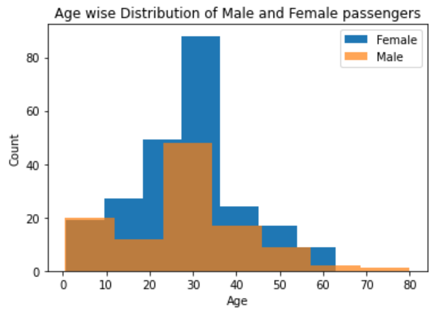
```
In [29]: sns.countplot(x = 'Survived',hue = 'Sex', data = titanic_data)
Out[29]: <AxesSubplot:xlabel='Survived', ylabel='count'>
```



As part of our further analysis, we can check the age factor with sex for the survival rate. We find that youngest female survival rate was higher than any other survival factor rate.

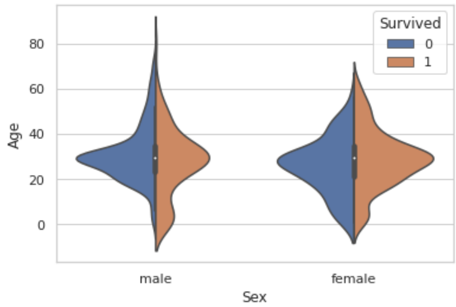
In [34]:
```
plt.hist(titanic_data[titanic_data.Survived==1]['Age'][(titanic_data['Sex'] == 'female')].dropna(), bins=7, label='Fe
plt.hist(titanic_data[titanic_data.Survived==1]['Age'][(titanic_data['Sex'] == 'male')].dropna(), bins=7, label='Male
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age wise Distribution of Male and Female passengers')
plt.legend()
```

Out[34]:
```
<matplotlib.legend.Legend at 0x7f75d7a0e3a0>
```



In [110]:
```
sns.violinplot(x ="Sex", y ="Age", hue ="Survived",
data = titanic_data, split = True)
```

Out[110]:
```
<AxesSubplot:xlabel='Sex', ylabel='Age'>
```

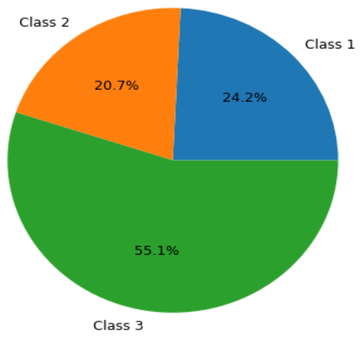


Next step we are going to check the class factor. We will identify if there was any class factor on survival rate. Below we find that there was 491 people in Class 3 which was highest.

In [111]:
```
pclass = titanic_data.groupby('Pclass')['Pclass'].count()
pclass
```

Out[111]:
```
Pclass
1    216
2    184
3    491
Name: Pclass, dtype: int64
```

```
In [37]: plt.figure(figsize=(7,7))
         plt.pie(pclass.values, labels=['Class 1', 'Class 2', 'Class 3'],
          autopct='%1.1f%%', textprops={'fontsize':13})
         plt.show()
```

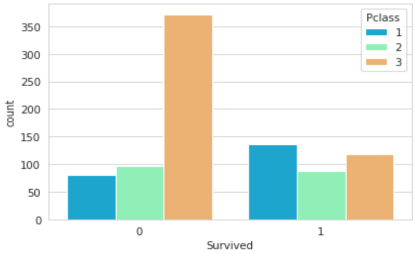


Now we will analyse from which class the survival rate was highest. We find the class 1 was highest survival rate although the class 3 people was highest.

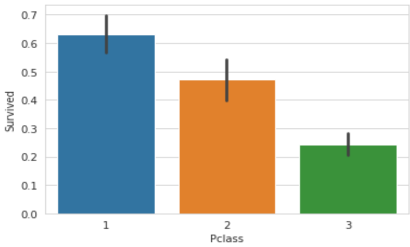
```
In [38]: # Get the actual numbers grouped by suvival and class where survival: Survival (0 = No; 1 = Yes)
         groupedby_class_survived_size = titanic_data.groupby(['Survived','Pclass']).size()
         print (groupedby_class_survived_size)

         Survived  Pclass
         0         1          80
                   2          97
                   3         372
         1         1         136
                   2          87
                   3         119
         dtype: int64
```

```
In [40]: sns.set_style('whitegrid')
         sns.countplot(x='Survived',hue='Pclass',data=titanic_data,palette='rainbow')
Out[40]: <AxesSubplot:xlabel='Survived', ylabel='count'>
```




```
In [41]: sns.barplot(x='Pclass',y='Survived',data=titanic_data)
Out[41]: <AxesSubplot:xlabel='Pclass', ylabel='Survived'>
```



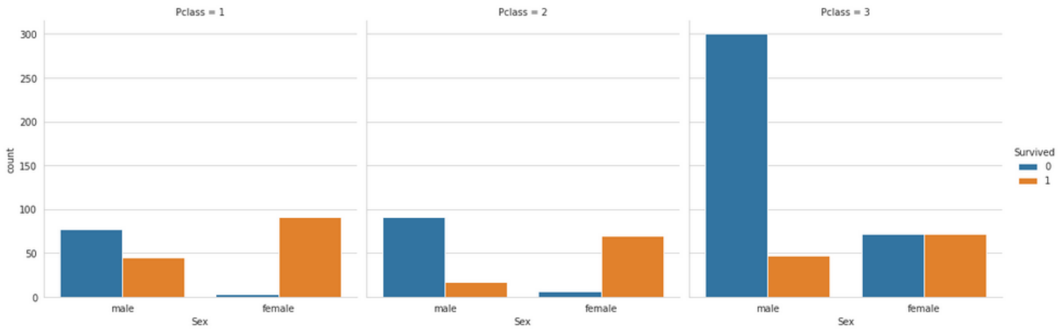
As part of our more deep analysis, we can check if the class factor along with sex for the survival rate.

```
In [42]: # Get the actual numbers grouped by suvival and class where survival: Survival (0 = No; 1 = Yes)
groupedby_class_survived_size = titanic_data.groupby(['Survived','Pclass','Sex']).size()
print (groupedby_class_survived_size)

Survived  Pclass  Sex
0         1       female      3
                  male       77
          2       female      6
                  male       91
          3       female     72
                  male      300
1         1       female     91
                  male       45
          2       female     70
                  male       17
          3       female     72
                  male       47
dtype: int64
```

```
In [43]: sns.catplot(x ='Sex', hue ='Survived',
kind ='count', col ='Pclass', data = titanic_data)

Out[43]: <seaborn.axisgrid.FacetGrid at 0x7f75d46e4130>
```



Here we can find the female class 1 survival rate was highest.

Below is the summary picture of total number where it shows the survival rate on total people, number different class people, number of different sex and number of embarked on different types.

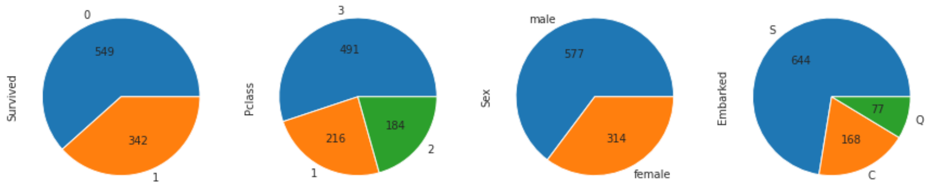
```
In [45]: plt.figure(figsize=(15,6))
plt.subplot(141)
values=titanic_data['Survived'].value_counts(dropna=True)
titanic_data.Survived.value_counts().plot(kind='pie',autopct= lambda x: '{:.0f}'.format(x*values.sum()/100))

plt.subplot(142)
values=titanic_data['Pclass'].value_counts(dropna=True)
titanic_data['Pclass'].value_counts().plot(kind='pie',autopct= lambda x: '{:.0f}'.format(x*values.sum()/100))

plt.subplot(143)
values=titanic_data['Sex'].value_counts(dropna=True)
titanic_data['Sex'].value_counts().plot(kind='pie',autopct= lambda x: '{:.0f}'.format(x*values.sum()/100))

plt.subplot(144)
values=titanic_data['Embarked'].value_counts(dropna=True)
titanic_data['Embarked'].value_counts().plot(kind='pie',autopct= lambda x: '{:.0f}'.format(x*values.sum()/100))

Out[45]: <AxesSubplot:ylabel='Embarked'>
```

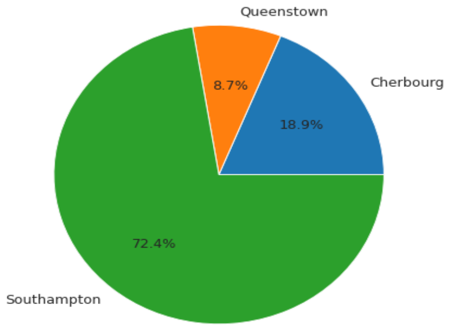


Next information we have found about the embankment of the total people. This is only informational.

In [46]:
```
embark = titanic_data.groupby('Embarked')['Embarked'].count()
embark
```

Out[46]:
```
Embarked
C    168
Q     77
S    644
Name: Embarked, dtype: int64
```

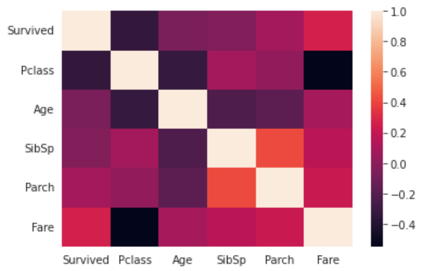
In [47]:
```
plt.figure(figsize=(7,7))
plt.pie(embark.values, labels=['Cherbourg', 'Queenstown', 'Southampton'],
        autopct='%1.1f%%', textprops={'fontsize':13})
plt.show()
```



On next step we have done the correlation. Correlation ranges from -1 to +1. Values closer to zero means there is no linear trend between the two variables. The close to 1 the correlation is the more positively correlated they are; that is as one increases so does the other and the closer to 1 the stronger this relationship is. The diagonals are all light pink because those squares are correlating each variable to itself (so it's a perfect correlation). For the rest the larger the number and darker the colour the higher the correlation between the two variables. The plot is also symmetrical about the diagonal since the same two variables are being paired together in those squares.

In [48]:
```
#correlation
sns.heatmap(titanic_data.corr())
```

Out[48]: <AxesSubplot:>



```
In [49]: titanic_data.head(10)
```

Out[49]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	7.2500	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	71.2833	C
2	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	7.9250	S
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	53.1000	S
4	0	3	Allen, Mr. William Henry	male	35.000000	0	0	8.0500	S
5	0	3	Moran, Mr. James	male	29.699118	0	0	8.4583	Q
6	0	1	McCarthy, Mr. Timothy J	male	54.000000	0	0	51.8625	S
7	0	3	Palsson, Master. Gosta Leonard	male	2.000000	3	1	21.0750	S
8	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.000000	0	2	11.1333	S
9	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.000000	1	0	30.0708	C

Next step we are going to do the regression analysis. Here name, sex, ticket, embarked are non-numerical column these are not so useful for prediction hence we can drop it. So we convert column into dummy numerical values. Below we convert sex into dummy numerical values.

```
In [50]: gender=pd.get_dummies(titanic_data['Sex'],drop_first=True)

In [51]: titanic_data['Gender']=gender

In [53]: titanic_data.drop(['Name','Sex','Embarked'],axis=1,inplace=True)

In [54]: titanic_data.head()
```

Out[54]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Gender
0	0	3	22.0	1	0	7.2500	1
1	1	1	38.0	1	0	71.2833	0
2	1	3	26.0	0	0	7.9250	0
3	1	1	35.0	1	0	53.1000	0
4	0	3	35.0	0	0	8.0500	1

```
In [55]: x=titanic_data[['Pclass','Age','SibSp','Parch','Fare','Gender']]
         y=titanic_data['Survived']

In [56]: y

Out[56]: 0      0
         1      1
         2      1
         3      1
         4      0
                ..
         886    0
         887    1
         888    0
         889    1
         890    0
         Name: Survived, Length: 891, dtype: int64

In [57]: from sklearn.model_selection import train_test_split

In [58]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)

In [59]: from sklearn.linear_model import LogisticRegression

In [60]: lr = LogisticRegression(solver='lbfgs', max_iter=1000)
```

```
In [62]: lr.fit(x_train,y_train)
Out[62]: LogisticRegression
         LogisticRegression(max_iter=1000)
```

```
In [63]: predict=lr.predict(x_test)
```

```
In [64]: from sklearn.metrics import confusion_matrix
```

```
In [65]: pd.DataFrame(confusion_matrix(y_test,predict),columns=['Predicted No','Predicted Yes'],index=['Actual No','Actual Yes
```

Out[65]:

	Predicted No	Predicted Yes
Actual No	156	19
Actual Yes	34	86

As per above predicted no means people not survived it's the predicted by our model and as per actual no is also same 156 and as per predicted by model 19 is survived but actual they are not survived and 34 passengers model predicted not survived but actual yes they are survived and 86 actually survived both predicted and actual matches.

For Checking accuracy we check classification report. Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.
Recall: Recall is the ratio of correctly predicted positive observations to the all observations in actual class F1 score- F1 score is the weighted average of precision and recall.

```
In [66]: from sklearn.metrics import classification_report
```

```
In [67]: print(classification_report(y_test,predict))
              precision    recall  f1-score   support

           0       0.82      0.89      0.85       175
           1       0.82      0.72      0.76       120

    accuracy                           0.82       295
   macro avg       0.82      0.80      0.81       295
weighted avg       0.82      0.82      0.82       295
```