Titanic Passanger Survival Analysis

1. Defining the problem statement¶

Complete the analysis of what sorts of people were likely to survive.
In particular, we ask you to apply the tools of machine learning to predict which passengers survived the Titanic tragedy.

In [1]:
```
from IPython.display import Image
Image(url= "https://static1.squarespace.com/static/5006453fe4b09ef2252ba068/5095eab
```

Out[1]:

In [2]:
```
import os
import pandas as pd
import numpy as np
train = pd.read_csv('train.xls')
test = pd.read_csv('test.xls')
```

In [3]:
```
train.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

In [4]:

```
train.isnull().sum()
print("Train Shape:",train.shape)
test.isnull().sum()
print("Test Shape:",test.shape)
```

```
Train Shape: (891, 12)
Test Shape: (418, 11)
```

In [5]:

```
train.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 [6]:

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

2.Data Dictionary

- Survived: 0 = No, 1 = Yes
- pclass: Ticket class 1 = 1st, 2 = 2nd, 3 = 3rd
- sibsp: # of siblings / spouses aboard the Titanic
- parch: # of parents / children aboard the Titanic
- ticket: Ticket number
- cabin: Cabin number
- embarked: Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton

Total rows and columns

We can see that there are 891 rows and 12 columns in our training dataset.

In [7]:
```
train.head(10)
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN

In [8]:
```
train.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

In [9]:

```
test.describe()
```

Out[9]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	3.000000	76.000000	8.000000	9.000000	512.329200

In [10]:

```
train.isnull().sum()
```

Out[10]:

```
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
```

In [11]:

```
test.isnull().sum()
```

Out[11]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

3.Data Visualization using Matplotlib and Seaborn packages.

In [12]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
```

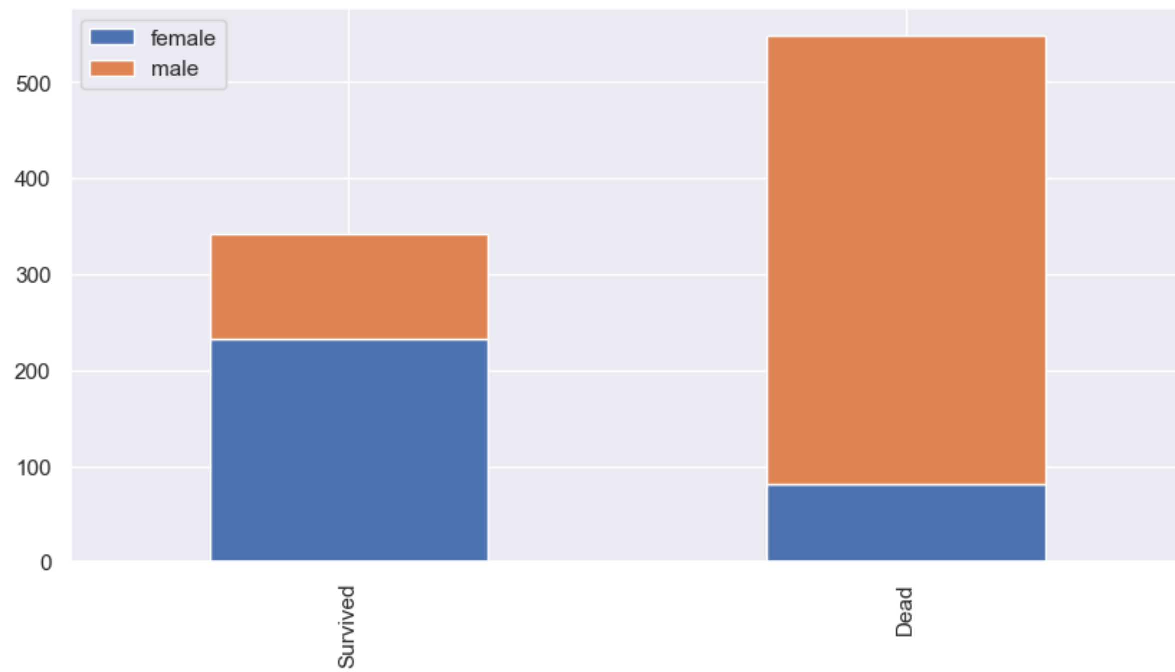
Bar Chart for Categorical Features

- Pclass
- Sex
- SibSp (# of siblings and spouse)
- Parch (# of parents and children)
- Embarked
- Cabin

In [13]:
```
def bar_chart(feature):
    survived = train[train['Survived']==1][feature].value_counts()
    dead = train[train['Survived']==0][feature].value_counts()
    df = pd.DataFrame([survived,dead])
    df.index = ['Survived','Dead']
    df.plot(kind='bar',stacked=True, figsize=(10,5))
```

In [14]:
```
bar_chart('Sex')
print("Survived :\n",train[train['Survived']==1]['Sex'].value_counts())
print("Dead:\n",train[train['Survived']==0]['Sex'].value_counts())
```

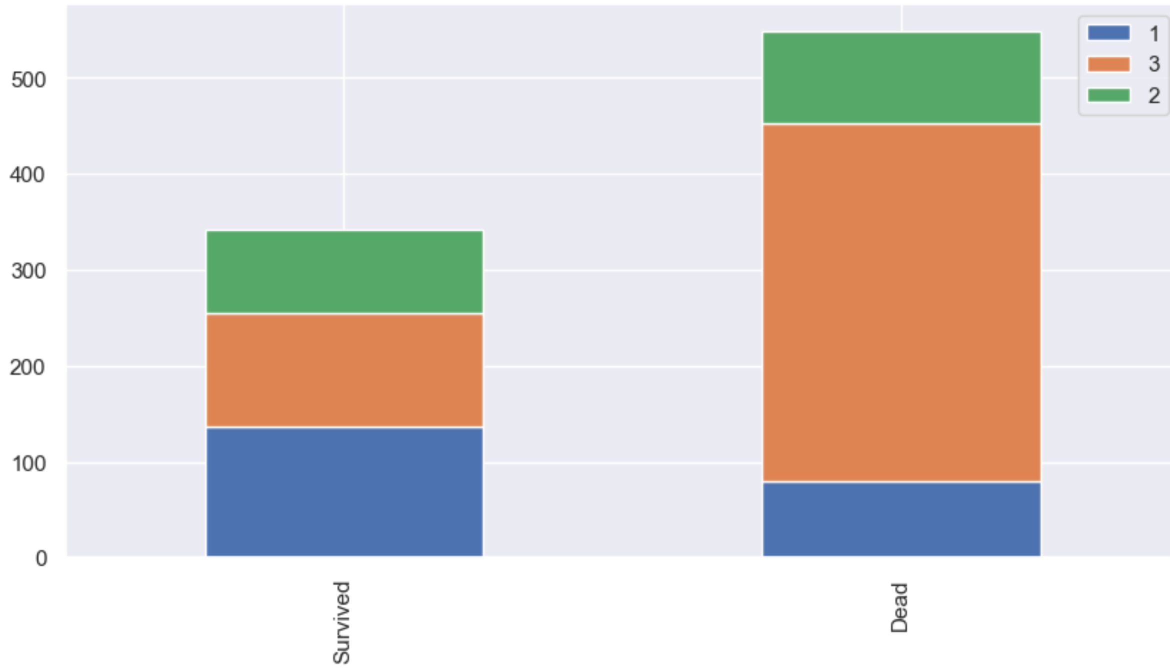
```
Survived :
 female    233
male      109
Name: Sex, dtype: int64
Dead:
 male      468
female     81
Name: Sex, dtype: int64
```



The Chart confirms **Women more likely survivied than Men**.

In [15]:
```
bar_chart('Pclass')
print("Survived :\n",train[train['Survived']==1]['Pclass'].value_counts())
print("Dead:\n",train[train['Survived']==0]['Pclass'].value_counts())
```

```
Survived :
 1    136
3    119
2     87
Name: Pclass, dtype: int64
Dead:
 3    372
2     97
1     80
Name: Pclass, dtype: int64
```

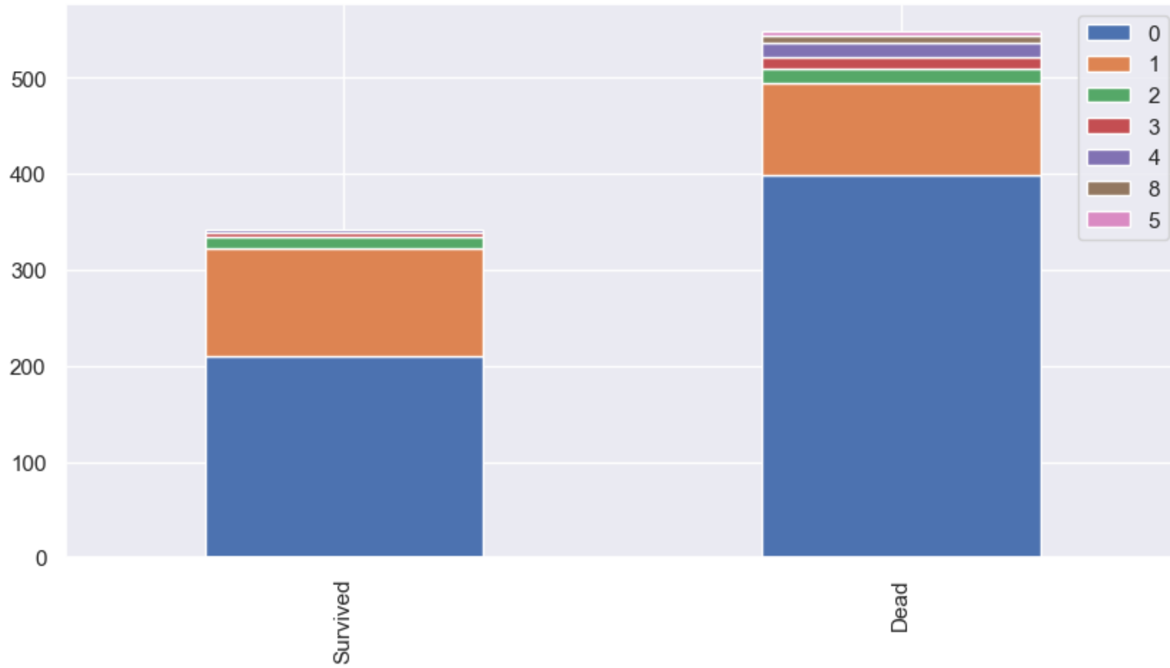


The Chart confirms **1st class** more likely survivied than **other classes**.

The Chart confirms **3rd class** more likely dead than **other classes**

In [16]:
```
bar_chart('SibSp')
print("Survived :\n",train[train['Survived']==1]['SibSp'].value_counts())
print("Dead:\n",train[train['Survived']==0]['SibSp'].value_counts())
```

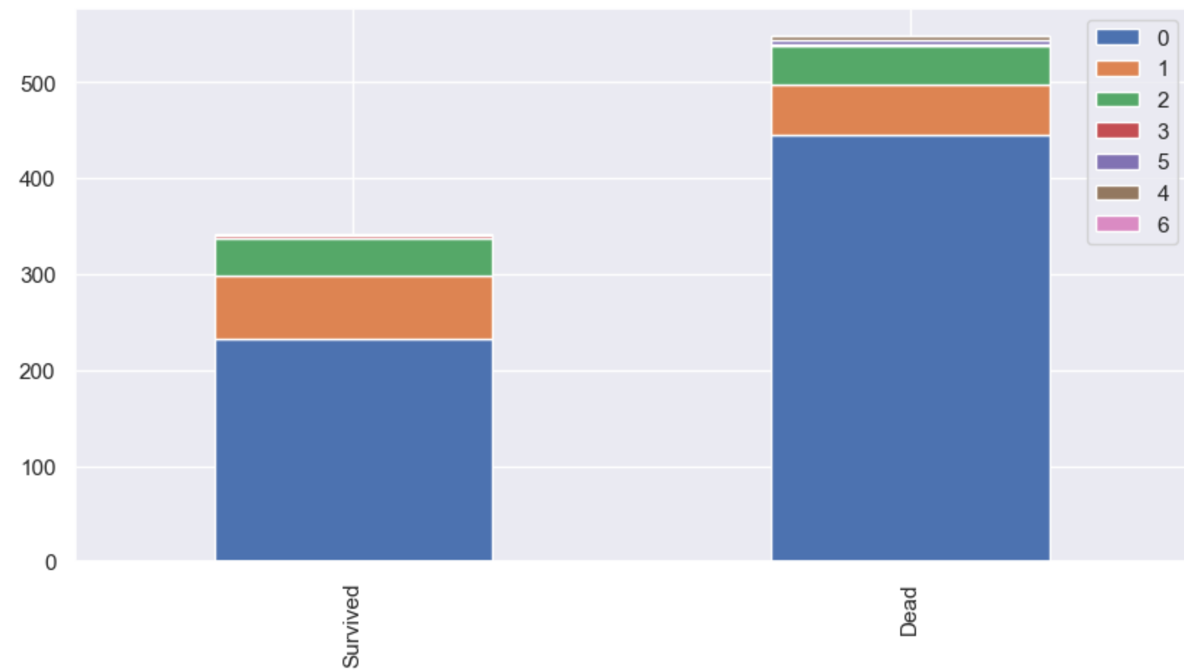
```
Survived :
 0    210
1    112
2     13
3      4
4      3
Name: SibSp, dtype: int64
Dead:
 0    398
1     97
4     15
2     15
3     12
8      7
5      5
Name: SibSp, dtype: int64
```



The Chart confirms a **person aboarded with more than 2 siblings or spouse** more likely survived.
The Chart confirms a **person aboarded without siblings or spouse** more likely dead

In [17]:
```
bar_chart('Parch')
print("Survived :\n",train[train['Survived']==1]['Parch'].value_counts())
print("Dead:\n",train[train['Survived']==0]['Parch'].value_counts())
```

```
Survived :
 0    233
1     65
2     40
3      3
5      1
Name: Parch, dtype: int64
Dead:
 0    445
1     53
2     40
5      4
4      4
3      2
6      1
Name: Parch, dtype: int64
```

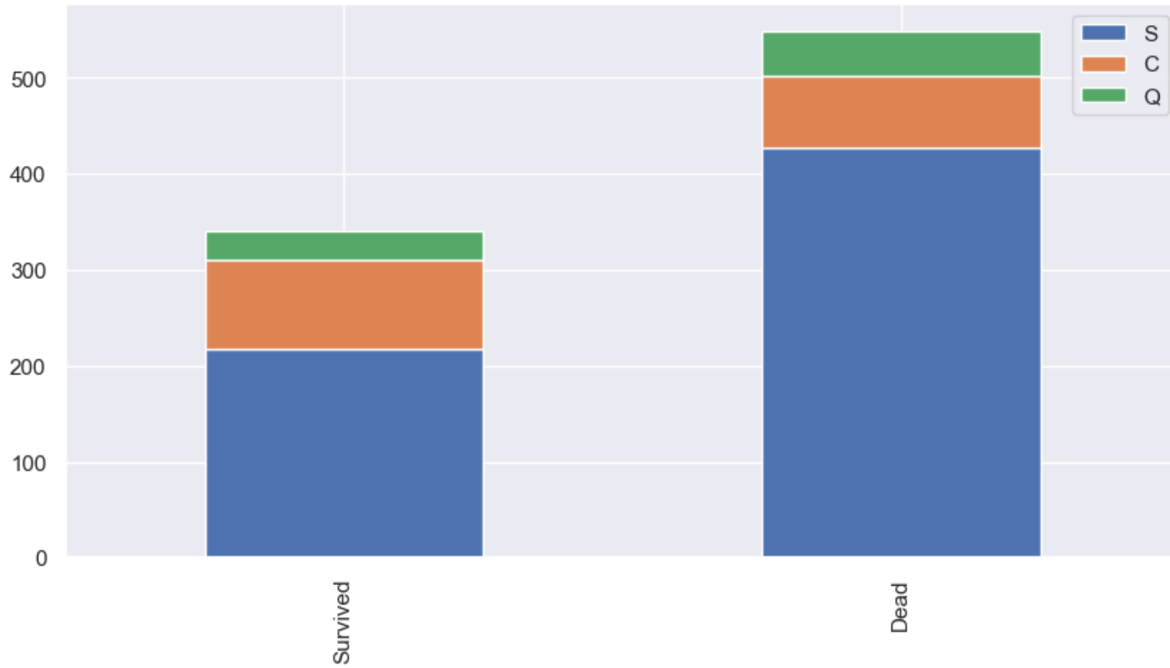


The Chart confirms a **person aboarded with more than 2 parents or children more likely survived.**

The Chart confirms a **person aboarded alone more likely dead**

In [18]:
```
bar_chart('Embarked')
print("Survived :\n",train[train['Survived']==1]['Embarked'].value_counts())
print("Dead:\n",train[train['Survived']==0]['Embarked'].value_counts())
```

```
Survived :
 S    217
C     93
Q     30
Name: Embarked, dtype: int64
Dead:
 S    427
C     75
Q     47
Name: Embarked, dtype: int64
```



The Chart confirms a **person aboarded from C** slightly more likely survived.
The Chart confirms a **person aboarded from Q** more likely dead.
The Chart confirms a **person aboarded from S** more likely dead.

4. Feature engineering

Feature engineering is the process of using domain knowledge of the data to create features (**feature vectors**) that make machine learning algorithms work.

feature vector is an n-dimensional vector of numerical features that represent some object. Many algorithms in machine learning require a numerical representation of objects, since such representations facilitate processing and statistical analysis.

In [19]:
```
train.head()
```

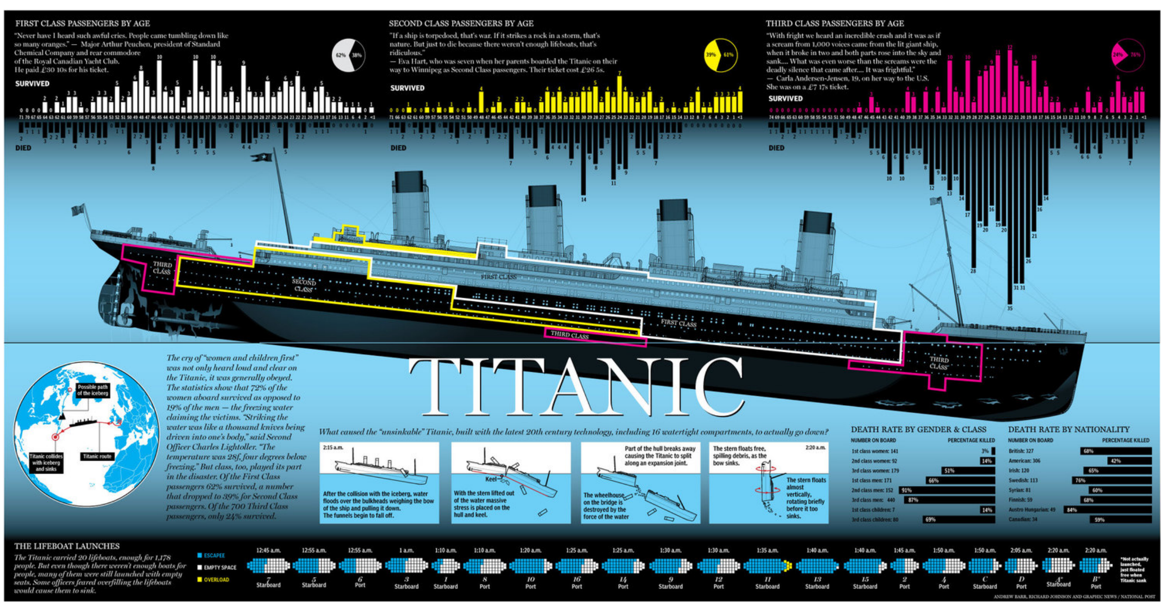
Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

4.1 how titanic sank?

In [20]:
```
Image(url= "https://static1.squarespace.com/static/5006453fe4b09ef2252ba068/t/5090b
```

Out[20]:



In [21]:
```
train.head(10)
```

Out[21]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN

1.Title

In [22]:
```
train_test_data = [train,test]

for dataset in train_test_data:
    dataset['Title'] = dataset['Name'].str.extract(' ([A-Za-z]+)\.', expand=False)
```

In [23]:
```
train['Title'].value_counts()
```

Out[23]:
```
Mr          517
Miss        182
Mrs         125
Master       40
Dr            7
Rev           6
Mlle          2
Major         2
Col           2
Countess      1
Capt          1
Ms            1
Sir           1
Lady          1
Mme           1
Don           1
Jonkheer      1
Name: Title, dtype: int64
```

In [24]:
```
test['Title'].value_counts()
```

Out[24]:
```
Mr        240
Miss       78
Mrs        72
Master     21
Col         2
Rev         2
Ms          1
Dr          1
Dona        1
Name: Title, dtype: int64
```

Title Map

Mr : 0

Miss : 1

Mrs: 2

Others: 3

In [25]:
```
title_mapping = {"Mr": 0, "Miss": 1, "Mrs": 2,
                 "Master": 3, "Dr": 3, "Rev": 3, "Col": 3, "Major": 3, "Mlle": 3,"C
                 "Ms": 3, "Lady": 3, "Jonkheer": 3, "Don": 3, "Dona" : 3, "Mme": 3,

for dataset in train_test_data:
    dataset['Title'] = dataset["Title"].map(title_mapping)
```

In [26]:
```
dataset.head()
```

Out[26]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [27]:

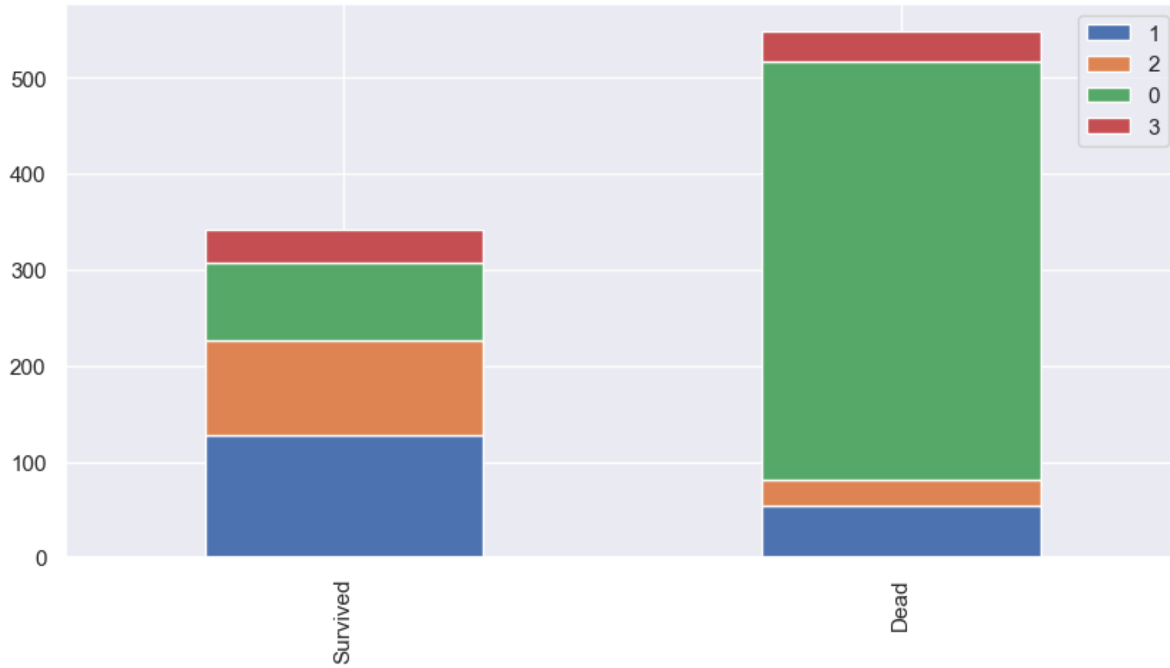
```
test.head()
```

Out[27]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [28]:

```
bar_chart('Title')
```



In [29]:
```
train.drop('Name', axis=1, inplace=True)
test.drop('Name', axis=1, inplace=True)
```

In [30]:
```
train.head()
```

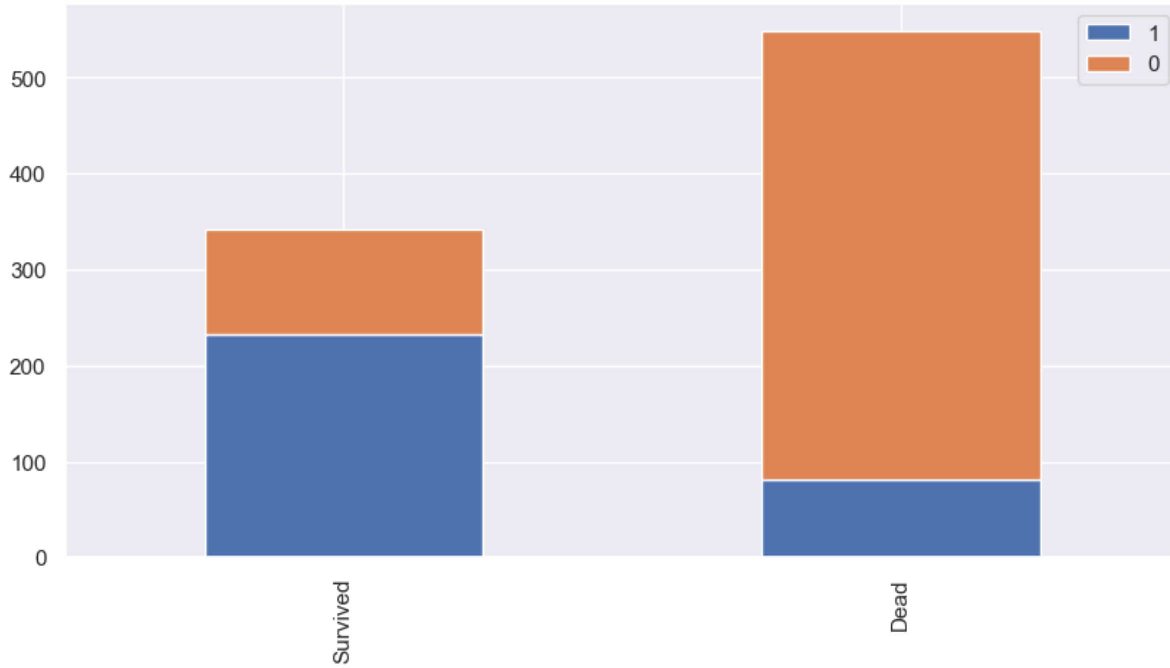
Out[30]:

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

2.Sex

In [31]:
```
sex_mapping = {"male": 0, "female": 1}
for dataset in train_test_data:
    dataset['Sex'] = dataset['Sex'].map(sex_mapping)
```

In [32]:
```
bar_chart('Sex')
```



In [33]:
```
test.head()
```

Out[33]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	892	3	0	34.5	0	0	330911	7.8292	NaN	Q	0
1	893	3	1	47.0	1	0	363272	7.0000	NaN	S	2
2	894	2	0	62.0	0	0	240276	9.6875	NaN	Q	0
3	895	3	0	27.0	0	0	315154	8.6625	NaN	S	0
4	896	3	1	22.0	1	1	3101298	12.2875	NaN	S	2

3.Age

In [34]:
```
train["Age"].fillna(train.groupby("Title")["Age"].transform("median"), inplace= Tru
test["Age"].fillna(test.groupby('Title')['Age'].transform("median"), inplace= True)
```

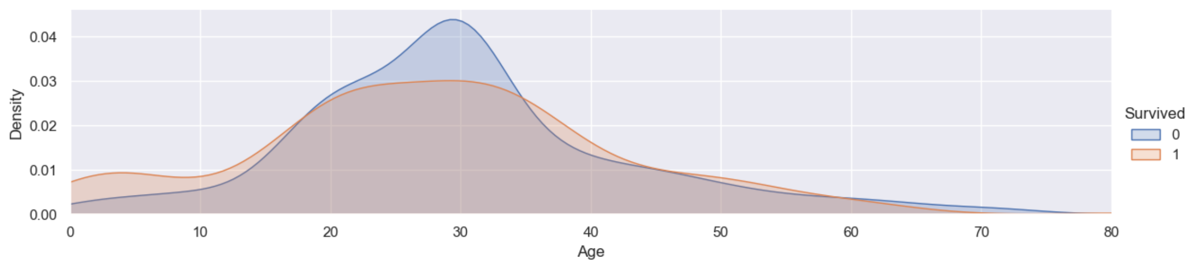
In [35]:
```
train.head()
```

Out[35]:

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

In [36]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Age',shade= True)
facet.set(xlim=(0, train['Age'].max()))
facet.add_legend()
plt.show()
```

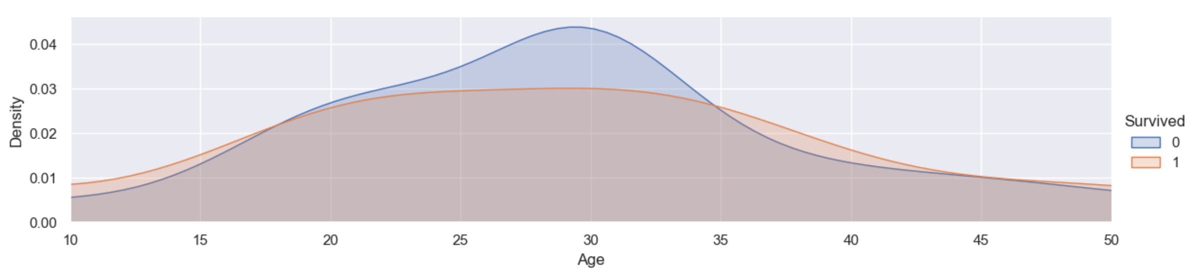
```
C:\Users\Saketh\anaconda3\lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
C:\Users\Saketh\anaconda3\lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
```



In [37]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Age',shade= True)
facet.set(xlim=(0, train['Age'].max()))
facet.add_legend()
plt.xlim(10,50)
```

```
C:\Users\Saketh\anaconda3\lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
C:\Users\Saketh\anaconda3\lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
```

Out[37]:
```
(10.0, 50.0)
```



Those who were **20 to 30 years old** were **more dead and more survived.**

In [38]:
```
train.info()
test.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   Sex          891 non-null    int64  
 4   Age          891 non-null    float64
 5   SibSp        891 non-null    int64  
 6   Parch        891 non-null    int64  
 7   Ticket       891 non-null    object 
 8   Fare         891 non-null    float64
 9   Cabin        204 non-null    object 
 10  Embarked     889 non-null    object 
 11  Title        891 non-null    int64  
dtypes: float64(2), int64(7), object(3)
memory usage: 83.7+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Sex          418 non-null    int64  
 3   Age          418 non-null    float64
 4   SibSp        418 non-null    int64  
 5   Parch        418 non-null    int64  
 6   Ticket       418 non-null    object 
 7   Fare         417 non-null    float64
 8   Cabin        91 non-null     object 
 9   Embarked     418 non-null    object 
 10  Title        418 non-null    int64  
dtypes: float64(2), int64(6), object(3)
memory usage: 36.0+ KB
```

4.2 Binning

Binning/Converting Numerical Age to Categorical Variable

feature vector map:

- child: 0
- young: 1
- adult: 2
- mid-age: 3
- senior: 4

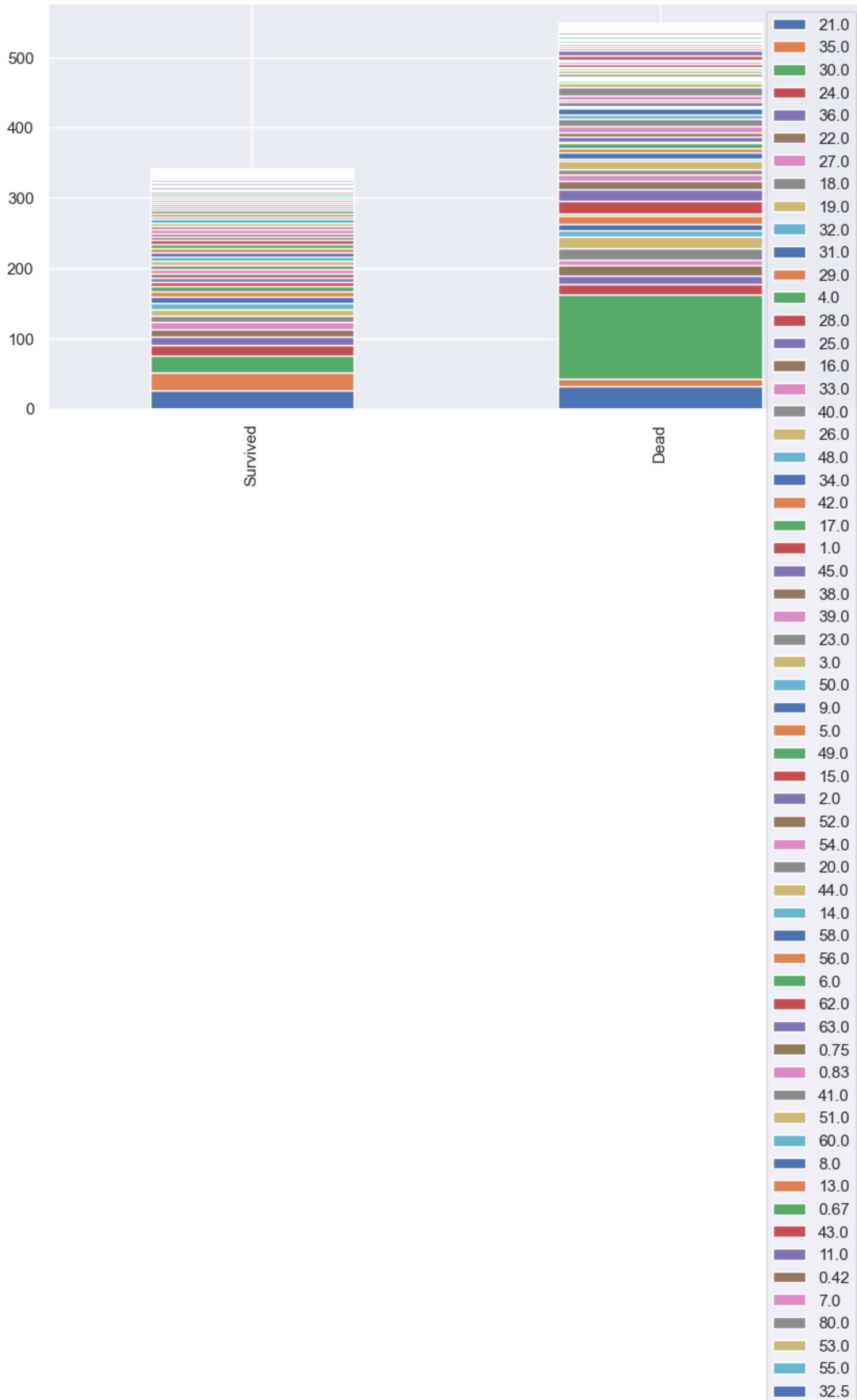
In [39]:
```
train.head()
```

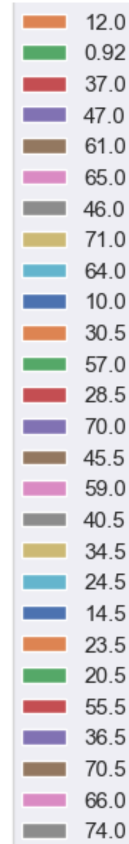
Out[39]:

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

> # Map the value of Age

In [40]:
```
train.head()
bar_chart('Age')
```

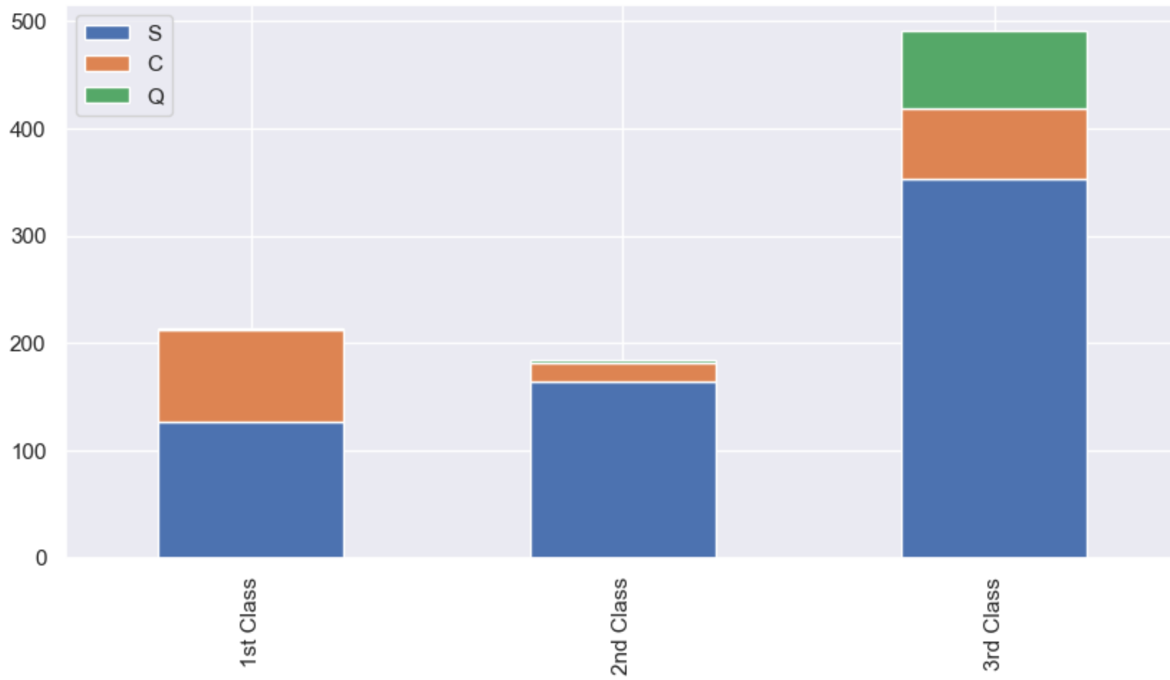

500
400
300
200
100
0
Survived
Dead
21.0
35.0
30.0
24.0
36.0
22.0
27.0
18.0
19.0
32.0
31.0
29.0
4.0
28.0
25.0
16.0
33.0
40.0
26.0
48.0
34.0
42.0
17.0
1.0
45.0
38.0
39.0
23.0
3.0
50.0
9.0
5.0
49.0
15.0
2.0
52.0
54.0
20.0
44.0
14.0
58.0
56.0
6.0
62.0
63.0
0.75
0.83
41.0
51.0
60.0
8.0
13.0
0.67
43.0
11.0
0.42
7.0
80.0
53.0
55.0
32.5



Map the value of Pclass

In [41]:

```
Pclass1 = train[train['Pclass'] == 1]['Embarked'].value_counts()
Pclass2 = train[train['Pclass'] == 2]['Embarked'].value_counts()
Pclass3 = train[train['Pclass'] == 3]['Embarked'].value_counts()
df = pd.DataFrame([Pclass1,Pclass2,Pclass3])
df.index = ['1st Class','2nd Class','3rd Class']
df.plot(kind = 'bar', stacked =  True, figsize=(10,5))
plt.show()
print("Pclass1:\n",Pclass1)
print("Pclass2:\n",Pclass2)
print("Pclass3:\n",Pclass3)
```




```
Pclass1:
 S    127
C     85
Q      2
Name: Embarked, dtype: int64
Pclass2:
 S    164
C     17
Q      3
Name: Embarked, dtype: int64
Pclass3:
 S    353
Q     72
C     66
Name: Embarked, dtype: int64
```

more than 50 % of 1st class are from S embark.

more than 50 % of 2st class are from S embark.

more than 50 % of 3st class are from S embark.

fill out missing embark with S embark

Map the value of Embarked

In [42]:
```
for dataset in train_test_data:
    dataset['Embarked'] =  dataset['Embarked'].fillna('S')
train.head()
```

Out[42]:

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

In [43]:
```
embarked_mapping = {'S':0,'C':1,'Q':2}
for dataset in train_test_data:
    dataset['Embarked'] = dataset['Embarked'].map(embarked_mapping)
```

Map the value of Fare

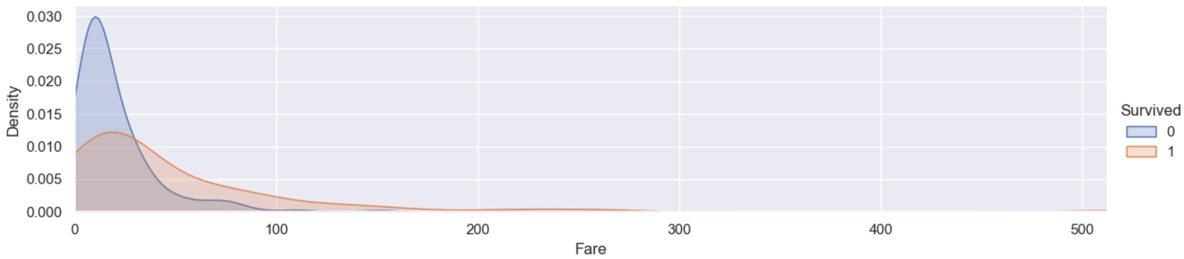
In [44]:
```
train["Fare"].fillna(train.groupby("Pclass")["Fare"].transform("median"), inplace=T
test["Fare"].fillna(test.groupby("Pclass")["Fare"].transform("median"), inplace=Tru
train.head(10)
```

Out[44]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	T
0	1	0	3	0	22.0	1	0	A/5 21171	7.2500	NaN	0	
1	2	1	1	1	38.0	1	0	PC 17599	71.2833	C85	1	
2	3	1	3	1	26.0	0	0	STON/O2. 3101282	7.9250	NaN	0	
3	4	1	1	1	35.0	1	0	113803	53.1000	C123	0	
4	5	0	3	0	35.0	0	0	373450	8.0500	NaN	0	
5	6	0	3	0	30.0	0	0	330877	8.4583	NaN	2	
6	7	0	1	0	54.0	0	0	17463	51.8625	E46	0	
7	8	0	3	0	2.0	3	1	349909	21.0750	NaN	0	
8	9	1	3	1	27.0	0	2	347742	11.1333	NaN	0	
9	10	1	2	1	14.0	1	0	237736	30.0708	NaN	1	

In [45]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4 )
facet.map(sns.kdeplot, 'Fare', shade = True)
facet.set(xlim = (0, train['Fare'].max()))
facet.add_legend()
plt.show()
```

```
C:\Users\Saketh\anaconda3\lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
C:\Users\Saketh\anaconda3\lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
```



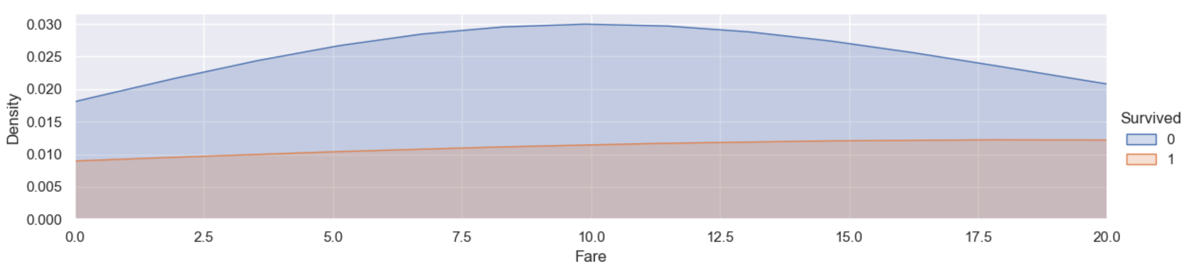
In [46]:

```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Fare',shade= True)
facet.set(xlim=(0, train['Fare'].max()))
facet.add_legend()
plt.xlim(0, 20)
```

```
C:\Users\Saketh\anaconda3\lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
C:\Users\Saketh\anaconda3\lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
```

Out[46]:

```
(0.0, 20.0)
```



In [47]:

```
train.head()
```

Out[47]:

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

In [48]:

```
train.Cabin.value_counts().head()
```

Out[48]:

```
B96 B98        4
G6             4
C23 C25 C27    4
C22 C26        3
F33            3
Name: Cabin, dtype: int64
```

Map the value of Cabin

In [49]:

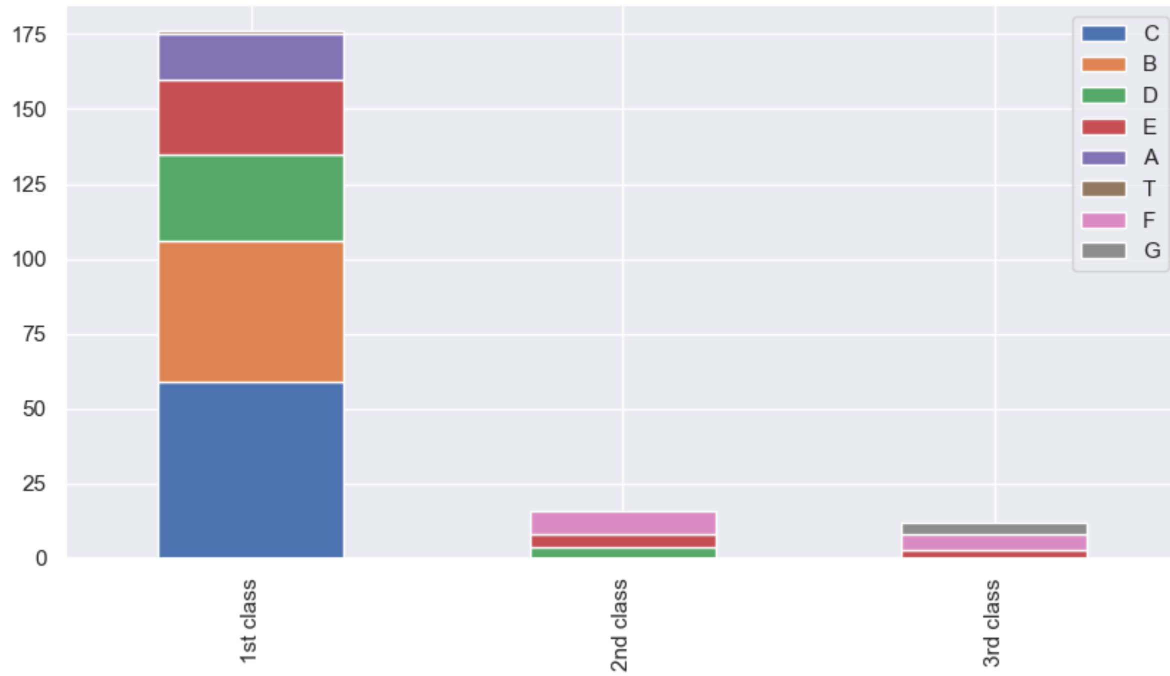
```
for dataset in train_test_data:
    dataset['Cabin'] =  dataset['Cabin'].str[:1]
```

In [50]:

```
Pclass1 = train[train['Pclass']==1]['Cabin'].value_counts()
Pclass2 = train[train['Pclass']==2]['Cabin'].value_counts()
Pclass3 = train[train['Pclass']==3]['Cabin'].value_counts()
df = pd.DataFrame([Pclass1, Pclass2, Pclass3])
df.index = ['1st class','2nd class', '3rd class']
df.plot(kind='bar',stacked=True, figsize=(10,5))
```

Out[50]:

```
<Axes: >
```



In [51]:
```
cabin_mapping = {"A": 0, "B": 0.4, "C": 0.8, "D": 1.2, "E": 1.6, "F": 2, "G": 2.4,
for dataset in train_test_data:
    dataset['Cabin'] = dataset['Cabin'].map(cabin_mapping)
```

In [52]:
```
# fill missing Fare with median fare for each Pclass
train["Cabin"].fillna(train.groupby("Pclass")["Cabin"].transform("median"), inplace
test["Cabin"].fillna(test.groupby("Pclass")["Cabin"].transform("median"), inplace=T
```

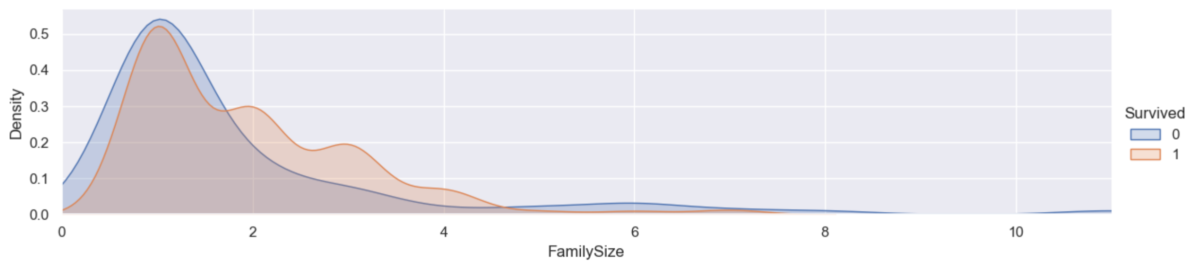
Map the value of Family Size

In [53]:
```
train["FamilySize"] = train["SibSp"] + train["Parch"] + 1
test["FamilySize"] = test["SibSp"] + test["Parch"] + 1
```

In [54]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'FamilySize',shade= True)
facet.set(xlim=(0, train['FamilySize'].max()))
facet.add_legend()
plt.xlim(0)
```

```
C:\Users\Saketh\anaconda3\lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
C:\Users\Saketh\anaconda3\lib\site-packages\seaborn\axisgrid.py:848: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
```

Out[54]:
```
(0.0, 11.0)
```



In [55]:
```
family_mapping = {1: 0, 2: 0.4, 3: 0.8, 4: 1.2, 5: 1.6, 6: 2, 7: 2.4, 8: 2.8, 9: 3.
for dataset in train_test_data:
    dataset['FamilySize'] = dataset['FamilySize'].map(family_mapping)
```

In [56]:
```
train.head()
```

Out[56]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	T
0	1	0	3	0	22.0	1	0	A/5 21171	7.2500	2.0	0	
1	2	1	1	1	38.0	1	0	PC 17599	71.2833	0.8	1	
2	3	1	3	1	26.0	0	0	STON/O2. 3101282	7.9250	2.0	0	
3	4	1	1	1	35.0	1	0	113803	53.1000	0.8	0	
4	5	0	3	0	35.0	0	0	373450	8.0500	2.0	0	

In [57]:
```
features_drop = ['Ticket','SibSp','Parch']
train = train.drop(features_drop, axis = 1)
test = test.drop(features_drop,axis=1)
train = train.drop(['PassengerId'], axis=1)
```

In [58]:
```
train_data = train.drop('Survived', axis = 1)
target = train['Survived']
train_data.shape, target.shape
```

Out[58]:
```
((891, 8), (891,))
```

In [59]:
```
train_data.head(10)
```

Out[59]:

	Pclass	Sex	Age	Fare	Cabin	Embarked	Title	FamilySize
0	3	0	22.0	7.2500	2.0	0	0	0.4
1	1	1	38.0	71.2833	0.8	1	2	0.4
2	3	1	26.0	7.9250	2.0	0	1	0.0
3	1	1	35.0	53.1000	0.8	0	2	0.4
4	3	0	35.0	8.0500	2.0	0	0	0.0
5	3	0	30.0	8.4583	2.0	2	0	0.0
6	1	0	54.0	51.8625	1.6	0	0	0.0
7	3	0	2.0	21.0750	2.0	0	3	1.6
8	3	1	27.0	11.1333	2.0	0	2	0.8
9	2	1	14.0	30.0708	1.8	1	2	0.4

5. Modelling

In [60]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier,ExtraTreeClassifier
from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier,BaggingCla
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

import numpy as np
```

In [61]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Survived    891 non-null    int64  
 1   Pclass      891 non-null    int64  
 2   Sex         891 non-null    int64  
 3   Age         891 non-null    float64
 4   Fare        891 non-null    float64
 5   Cabin       891 non-null    float64
 6   Embarked    891 non-null    int64  
 7   Title       891 non-null    int64  
 8   FamilySize  891 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 62.8 KB
```

6.Cross Validation(k-fold)

In [62]:
```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
k_fold = KFold(n_splits=10, shuffle=True, random_state=0)
```

In [63]:
```
clf = [KNeighborsClassifier(n_neighbors = 13),DecisionTreeClassifier(),
       RandomForestClassifier(n_estimators=13),GaussianNB(),SVC(),ExtraTreeClassifi
      GradientBoostingClassifier(n_estimators=10, learning_rate=1,max_features=3, m
def model_fit():
    scoring = 'accuracy'
    for i in range(len(clf)):
        score = cross_val_score(clf[i], train_data, target, cv=k_fold, n_jobs=1, sc
        print("Score of Model",i,":",round(np.mean(score)*100,2))

model_fit()
```

```
Score of Model 0 : 72.39
Score of Model 1 : 77.22
Score of Model 2 : 81.14
Score of Model 3 : 79.23
Score of Model 4 : 67.34
Score of Model 5 : 76.33
Score of Model 6 : 81.15
Score of Model 7 : 81.37
Score of Model 8 : 80.13
```

Score of **Model 4** : **83.5** Which is **Support Vector Machine**

7.Testing

In [64]:
```
clf1 = SVC()
clf1.fit(train_data, target)
test
test_data = test.drop(['PassengerId'], axis=1)
test_data
prediction = clf1.predict(test_data)
```

In [65]:
```
test_data['Survived'] = prediction
test_data.head()
```

Out[65]:

	Pclass	Sex	Age	Fare	Cabin	Embarked	Title	FamilySize	Survived
0	3	0	34.5	7.8292	2.0	2	0	0.0	0
1	3	1	47.0	7.0000	2.0	0	2	0.4	0
2	2	0	62.0	9.6875	2.0	2	0	0.0	0
3	3	0	27.0	8.6625	2.0	0	0	0.0	0
4	3	1	22.0	12.2875	2.0	0	2	0.8	0

You can see that **Survived** Column which was predicted using this Support **Vector Machine Algorithm.**

In [66]:
```
test_data_sample = pd.read_csv('test.xls')
test_data_sample["Survived"] = prediction
```

In [67]:
```
test_data_sample[['PassengerId','Survived']].to_csv("submission.csv", index = False
```

In [ ]:
```
```