titanic-solution

February 4, 2018

1 Titanic: Machine Learning from Disaster

1.0.1 Predict survival on the Titanic

1.1 Defining the problem statement

https://www.kaggle.com/c/titanic/data

1.2 Collecting the data

1.2.1 load train, test dataset using Pandas

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

        train = pd.read_csv('input/train.csv')
        test = pd.read_csv('input/test.csv')
```

1.3 data analysis

Printing first 5 rows of the dataset - dataset.head()

```
In [2]: train.head(80)

Out[2]:     PassengerId  Survived  Pclass  \
        0             1         0       3
        1             2         1       1
        2             3         1       3
        3             4         1       1
        4             5         0       3
        5             6         0       3
        6             7         0       1
        7             8         0       3
        8             9         1       3
        9            10         1       2
        10           11         1       3
        11           12         1       1
        12           13         0       3
        13           14         0       3
        14           15         0       3
```

15	16	1	2
16	17	0	3
17	18	1	2
18	19	0	3
19	20	1	3
20	21	0	2
21	22	1	2
22	23	1	3
23	24	1	1
24	25	0	3
25	26	1	3
26	27	0	3
27	28	0	1
28	29	1	3
29	30	0	3
..	...	...	...
50	51	0	3
51	52	0	3
52	53	1	1
53	54	1	2
54	55	0	1
55	56	1	1
56	57	1	2
57	58	0	3
58	59	1	2
59	60	0	3
60	61	0	3
61	62	1	1
62	63	0	1
63	64	0	3
64	65	0	1
65	66	1	3
66	67	1	2
67	68	0	3
68	69	1	3
69	70	0	3
70	71	0	2
71	72	0	3
72	73	0	2
73	74	0	3
74	75	1	3
75	76	0	3
76	77	0	3
77	78	0	3
78	79	1	2
79	80	1	3

```
                                                   Name    Sex   Age  SibSp  \
```

0	Braund, Mr. Owen Harris	male	22.00	1
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.00	1
2	Heikkinen, Miss. Laina	female	26.00	0
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.00	1
4	Allen, Mr. William Henry	male	35.00	0
5	Moran, Mr. James	male	NaN	0
6	McCarthy, Mr. Timothy J	male	54.00	0
7	Palsson, Master. Gosta Leonard	male	2.00	3
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.00	0
9	Nasser, Mrs. Nicholas (Adele Achem)	female	14.00	1
10	Sandstrom, Miss. Marguerite Rut	female	4.00	1
11	Bonnell, Miss. Elizabeth	female	58.00	0
12	Saundercock, Mr. William Henry	male	20.00	0
13	Andersson, Mr. Anders Johan	male	39.00	1
14	Vestrom, Miss. Hulda Amanda Adolfina	female	14.00	0
15	Hewlett, Mrs. (Mary D Kingcome)	female	55.00	0
16	Rice, Master. Eugene	male	2.00	4
17	Williams, Mr. Charles Eugene	male	NaN	0
18	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.00	1
19	Masselmani, Mrs. Fatima	female	NaN	0
20	Fynney, Mr. Joseph J	male	35.00	0
21	Beesley, Mr. Lawrence	male	34.00	0
22	McGowan, Miss. Anna "Annie"	female	15.00	0
23	Sloper, Mr. William Thompson	male	28.00	0
24	Palsson, Miss. Torborg Danira	female	8.00	3
25	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	female	38.00	1
26	Emir, Mr. Farred Chehab	male	NaN	0
27	Fortune, Mr. Charles Alexander	male	19.00	3
28	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0
29	Todoroff, Mr. Lalio	male	NaN	0
..	...	...	...	...
50	Panula, Master. Juha Niilo	male	7.00	4
51	Nosworthy, Mr. Richard Cater	male	21.00	0
52	Harper, Mrs. Henry Sleeper (Myna Haxtun)	female	49.00	1
53	Faunthorpe, Mrs. Lizzie (Elizabeth Anne Wilkin...	female	29.00	1
54	Ostby, Mr. Engelhart Cornelius	male	65.00	0
55	Woolner, Mr. Hugh	male	NaN	0
56	Rugg, Miss. Emily	female	21.00	0
57	Novel, Mr. Mansouer	male	28.50	0
58	West, Miss. Constance Mirium	female	5.00	1
59	Goodwin, Master. William Frederick	male	11.00	5
60	Sirayanian, Mr. Orsen	male	22.00	0
61	Icard, Miss. Amelie	female	38.00	0
62	Harris, Mr. Henry Birkhardt	male	45.00	1
63	Skoog, Master. Harald	male	4.00	3
64	Stewart, Mr. Albert A	male	NaN	0
65	Moubarek, Master. Gerios	male	NaN	1
66	Nye, Mrs. (Elizabeth Ramell)	female	29.00	0

67	Crease, Mr. Ernest James	male	19.00	0
68	Andersson, Miss. Erna Alexandra	female	17.00	4
69	Kink, Mr. Vincenz	male	26.00	2
70	Jenkin, Mr. Stephen Curnow	male	32.00	0
71	Goodwin, Miss. Lillian Amy	female	16.00	5
72	Hood, Mr. Ambrose Jr	male	21.00	0
73	Chronopoulos, Mr. Apostolos	male	26.00	1
74	Bing, Mr. Lee	male	32.00	0
75	Moen, Mr. Sigurd Hansen	male	25.00	0
76	Staneff, Mr. Ivan	male	NaN	0
77	Moutal, Mr. Rahamin Haim	male	NaN	0
78	Caldwell, Master. Alden Gates	male	0.83	0
79	Dowdell, Miss. Elizabeth	female	30.00	0

	Parch	Ticket	Fare	Cabin	Embarked
0	0	A/5 21171	7.2500	NaN	S
1	0	PC 17599	71.2833	C85	C
2	0	STON/O2. 3101282	7.9250	NaN	S
3	0	113803	53.1000	C123	S
4	0	373450	8.0500	NaN	S
5	0	330877	8.4583	NaN	Q
6	0	17463	51.8625	E46	S
7	1	349909	21.0750	NaN	S
8	2	347742	11.1333	NaN	S
9	0	237736	30.0708	NaN	C
10	1	PP 9549	16.7000	G6	S
11	0	113783	26.5500	C103	S
12	0	A/5. 2151	8.0500	NaN	S
13	5	347082	31.2750	NaN	S
14	0	350406	7.8542	NaN	S
15	0	248706	16.0000	NaN	S
16	1	382652	29.1250	NaN	Q
17	0	244373	13.0000	NaN	S
18	0	345763	18.0000	NaN	S
19	0	2649	7.2250	NaN	C
20	0	239865	26.0000	NaN	S
21	0	248698	13.0000	D56	S
22	0	330923	8.0292	NaN	Q
23	0	113788	35.5000	A6	S
24	1	349909	21.0750	NaN	S
25	5	347077	31.3875	NaN	S
26	0	2631	7.2250	NaN	C
27	2	19950	263.0000	C23 C25 C27	S
28	0	330959	7.8792	NaN	Q
29	0	349216	7.8958	NaN	S
..	...	...	...	...	...
50	1	3101295	39.6875	NaN	S
51	0	A/4. 39886	7.8000	NaN	S

```
52      0       PC 17572  76.7292        D33      C
53      0           2926  26.0000        NaN      S
54      1         113509  61.9792        B30      C
55      0          19947  35.5000        C52      S
56      0     C.A. 31026  10.5000        NaN      S
57      0           2697   7.2292        NaN      C
58      2     C.A. 34651  27.7500        NaN      S
59      2        CA 2144  46.9000        NaN      S
60      0           2669   7.2292        NaN      C
61      0         113572  80.0000        B28    NaN
62      0          36973  83.4750        C83      S
63      2         347088  27.9000        NaN      S
64      0       PC 17605  27.7208        NaN      C
65      1           2661  15.2458        NaN      C
66      0     C.A. 29395  10.5000        F33      S
67      0      S.P. 3464   8.1583        NaN      S
68      2        3101281   7.9250        NaN      S
69      0         315151   8.6625        NaN      S
70      0     C.A. 33111  10.5000        NaN      S
71      2        CA 2144  46.9000        NaN      S
72      0   S.O.C. 14879  73.5000        NaN      S
73      0           2680  14.4542        NaN      C
74      0           1601  56.4958        NaN      S
75      0         348123   7.6500      F G73      S
76      0         349208   7.8958        NaN      S
77      0         374746   8.0500        NaN      S
78      2         248738  29.0000        NaN      S
79      0         364516  12.4750        NaN      S

[80 rows x 12 columns]
```

1.3.1 Data Dictionary

https://www.kaggle.com/c/titanic/data - 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

```
In [3]: ##test.head()
```

```
In [4]: train.shape ### shape of ds (row, column)
```

```
Out[4]: (891, 12)
```

```
In [5]: test.shape
```

Out[5]: (418, 11)

```
In [6]: train.info() ### sum of missing vals
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

```
In [7]: test.info() ### info of vals
```

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

```
In [8]: train.isnull().sum() ### sum of missing vals
```

```
Out[8]: 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
```

age missing 177, cabin missin 687, embarked missing 2

In [9]:
```
test.isnull().sum()
```

Out[9]:
```
PassengerId       0
Pclass            0
Name              0
Sex               0
Age              86
SibSp             0
Parch             0
Ticket            0
Fare              1
Cabin           327
Embarked          0
dtype: int64
```

age missing 86 fare missing 1 cabin missing 327

1.3.2 import python lib for visualization

In [10]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set() # setting seaborn default for plots
```

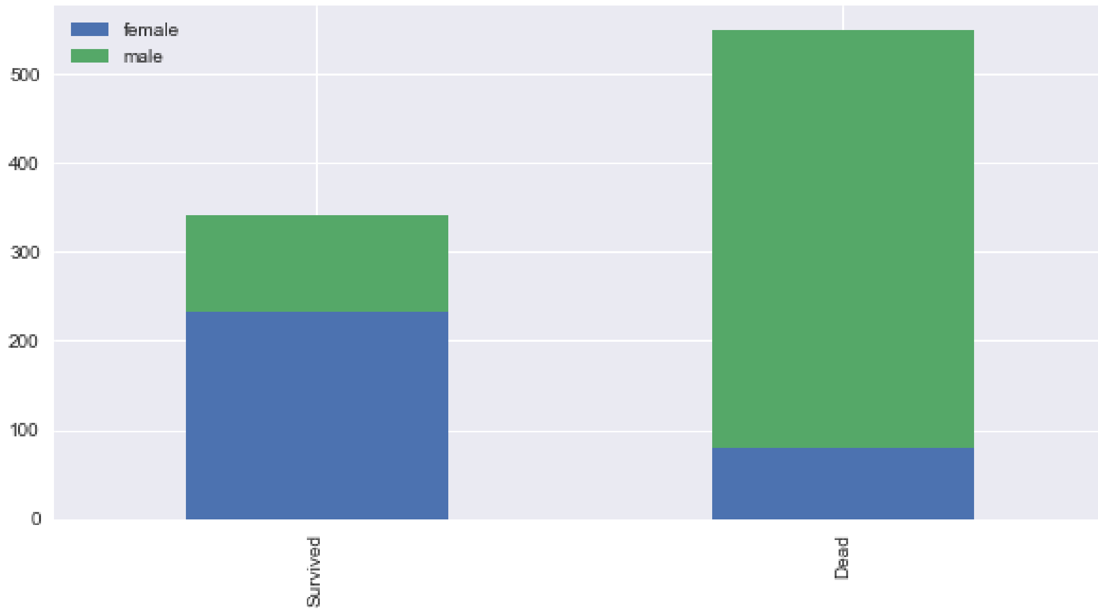
1.3.3 Bar Chart for Categorical Features

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

In [11]:
```
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])
```

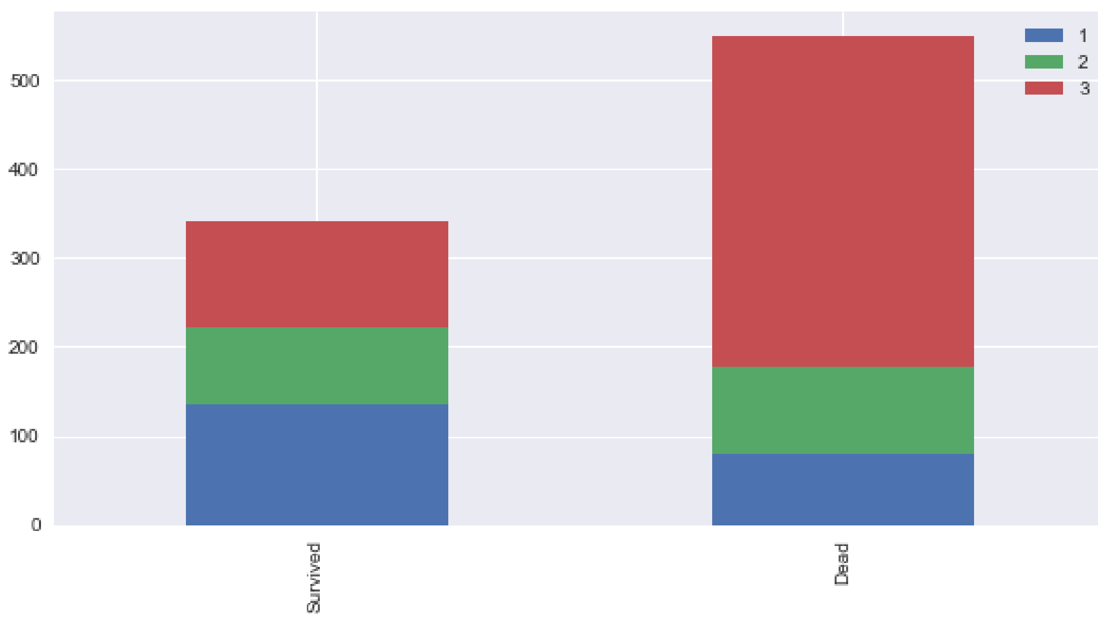
```
    df.index = ['Survived','Dead']
    df.plot(kind='bar',stacked=True, figsize=(10,5))
```

```
In [12]: bar_chart('Sex')
```



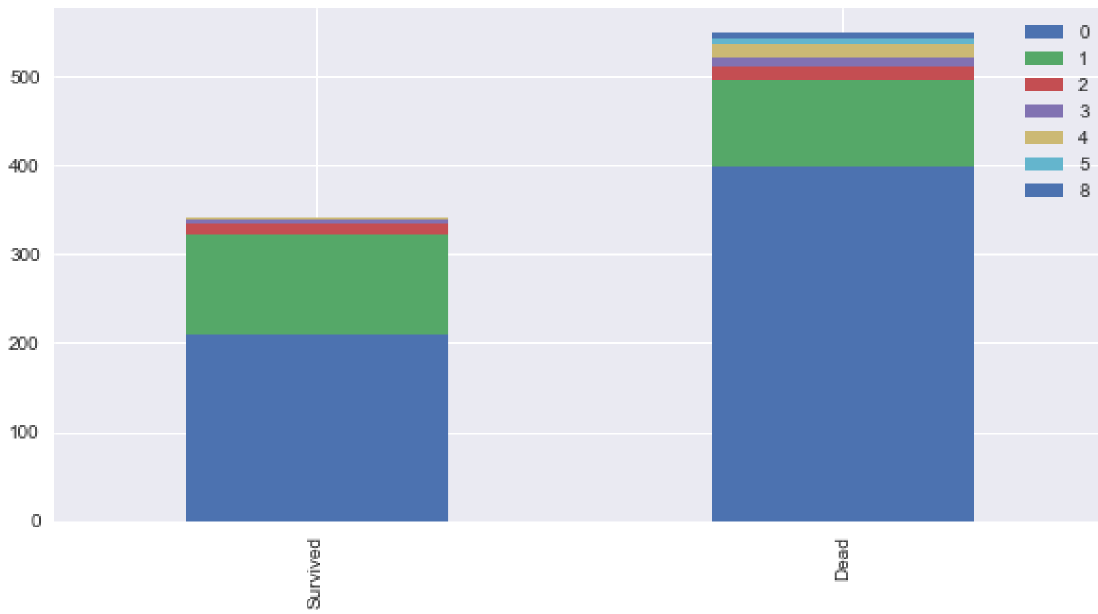
Women more likely survivied than **Men**

```
In [13]: bar_chart('Pclass')
```



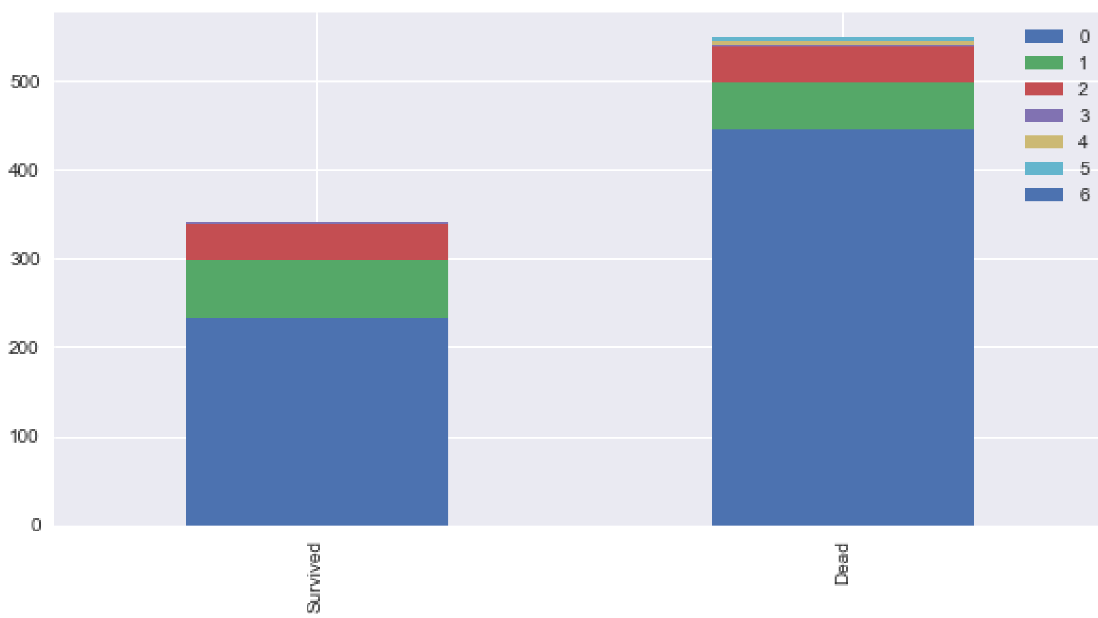
1st class more likely survivied than **other classes**
3rd class more likely dead than **other classes**

```
In [14]: bar_chart('SibSp')
```



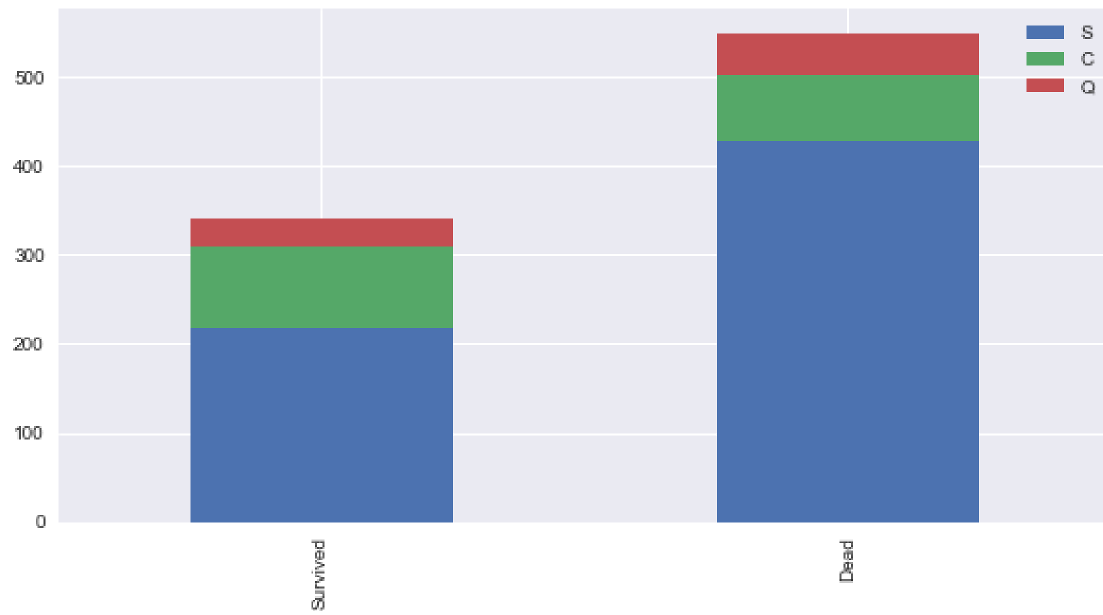
a person aboarded with more than 2 siblings or spouse more likely survived
** a person aboarded without siblings or spouse** more likely dead

```
In [15]: bar_chart('Parch')
```



a person aboarded with more than 2 parents or children more likely survived
** a person aboarded alone** more likely dead

```
In [16]: bar_chart('Embarked')
```



a person aboarded from C slightly more likely survived
a person aboarded from Q more likely dead
a person aboarded from S more likely dead

1.4 Feature engineering

```
In [17]: #train.head()
```

Pclass is key feature for classifier

```
In [18]: train.head(10)
```

```
Out[18]:    PassengerId  Survived  Pclass  \
         0            1         0       3
         1            2         1       1
         2            3         1       3
         3            4         1       1
         4            5         0       3
         5            6         0       3
         6            7         0       1
         7            8         0       3
```

```
8            9          1       3
9           10          1       2

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0
5                                   Moran, Mr. James    male   NaN      0
6                            McCarthy, Mr. Timothy J    male  54.0      0
7                     Palsson, Master. Gosta Leonard    male   2.0      3
8  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
9                Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
5      0            330877   8.4583   NaN        Q
6      0             17463  51.8625   E46        S
7      1            349909  21.0750   NaN        S
8      2            347742  11.1333   NaN        S
9      0            237736  30.0708   NaN        C
```

In [19]:
```
train_test_data = [train, test] # combining train and test dataset

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

In [20]:
```
# delete unnecessary feature from dataset : name
train.drop('Name', axis=1, inplace=True)
test.drop('Name', axis=1, inplace=True)
```

In [21]:
```
train['Title'].value_counts()
```

Out[21]:
```
Mr          517
Miss        182
Mrs         125
Master       40
Dr            7
Rev           6
Mlle          2
Major         2
Col           2
Countess      1
Capt          1
```

```
Lady          1
Mme           1
Don           1
Jonkheer      1
Ms            1
Sir           1
Name: Title, dtype: int64
```

In [22]:
```
test['Title'].value_counts()
```

Out[22]:
```
Mr        240
Miss       78
Mrs        72
Master     21
Col         2
Rev         2
Dr          1
Dona        1
Ms          1
Name: Title, dtype: int64
```

Title map Mr : 0
Miss : 1
Mrs: 2
Others: 3

In [23]:
```
title_mapping = {"Mr": 0, "Miss": 1, "Mrs": 2,
                 "Master": 3, "Dr": 3, "Rev": 3, "Col": 3, "Major": 3, "Mlle": 3,"Cou
                 "Ms": 3, "Lady": 3, "Jonkheer": 3, "Don": 3, "Dona" : 3, "Mme": 3,"Ca
for dataset in train_test_data:
    dataset['Title'] = dataset['Title'].map(title_mapping)
```

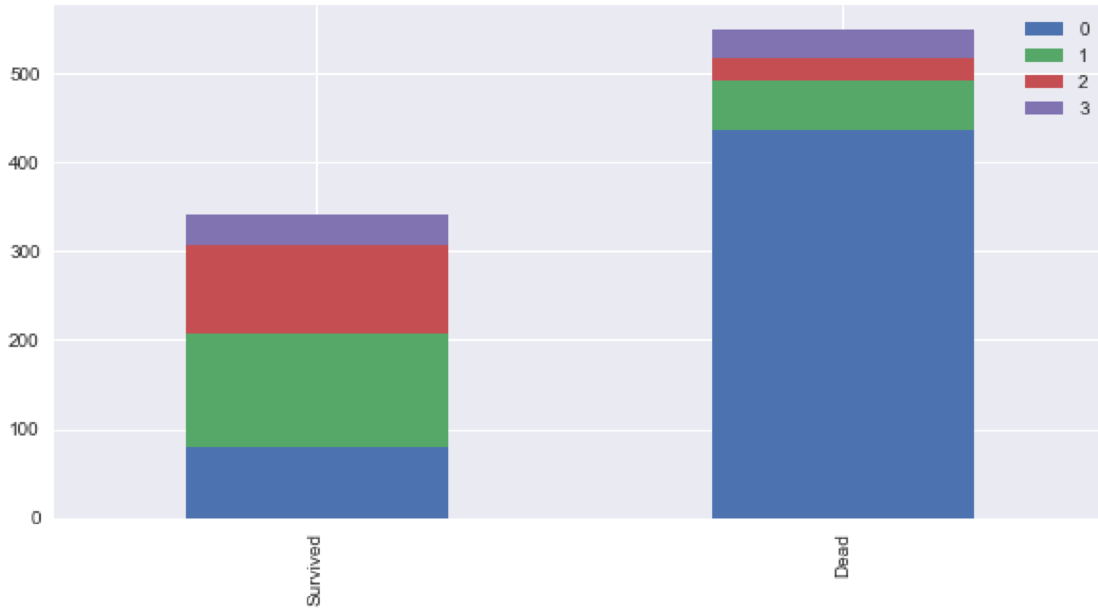
In [24]:
```
#train.head()
```

In [25]:
```
test.head()
```

Out[25]:
```
   PassengerId  Pclass     Sex   Age  SibSp  Parch   Ticket     Fare Cabin  \
0          892       3    male  34.5      0      0   330911   7.8292   NaN
1          893       3  female  47.0      1      0   363272   7.0000   NaN
2          894       2    male  62.0      0      0   240276   9.6875   NaN
3          895       3    male  27.0      0      0   315154   8.6625   NaN
4          896       3  female  22.0      1      1  3101298  12.2875   NaN

  Embarked  Title
0        Q      0
1        S      2
2        Q      0
3        S      0
4        S      2
```

In [26]: bar_chart('Title')



In [27]: train.head()

Out[27]:

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

	Ticket	Fare	Cabin	Embarked	Title
0	A/5 21171	7.2500	NaN	S	0
1	PC 17599	71.2833	C85	C	2
2	STON/O2. 3101282	7.9250	NaN	S	1
3	113803	53.1000	C123	S	2
4	373450	8.0500	NaN	S	0

In [28]: test.head()

Out[28]:

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

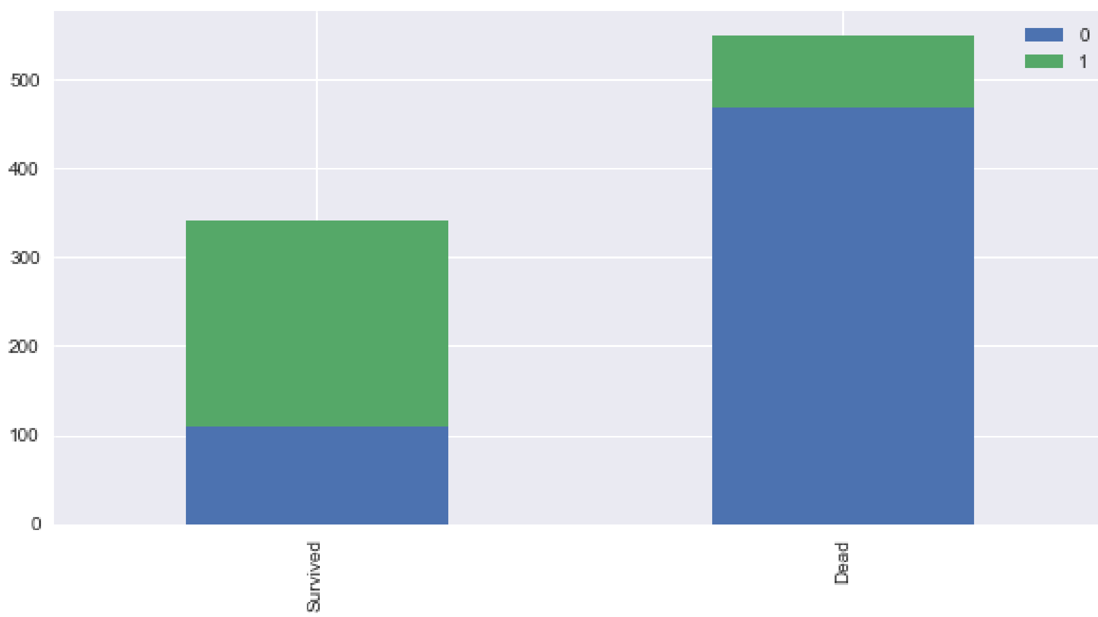
```
  Embarked  Title
0        Q      0
1        S      2
2        Q      0
3        S      0
4        S      2
```

1.4.1 Sex

male: 0 female: 1

In [29]:
```python
sex_mapping = {"male": 0, "female": 1}
for dataset in train_test_data:
    dataset['Sex'] = dataset['Sex'].map(sex_mapping)
```

In [30]:
```python
bar_chart('Sex')
```



1.4.2 Age

some age is missing Let's use Title's median age for missing Age

In [31]:
```python
train.head(100)
```

Out[31]:
```
   PassengerId  Survived  Pclass  Sex    Age  SibSp  Parch            Ticket  \
0            1         0       3    0  22.00      1      0         A/5 21171
1            2         1       1    1  38.00      1      0          PC 17599
2            3         1       3    1  26.00      0      0  STON/O2. 3101282
3            4         1       1    1  35.00      1      0            113803
```

4	5	0	3	0	35.00	0	0	373450
5	6	0	3	0	NaN	0	0	330877
6	7	0	1	0	54.00	0	0	17463
7	8	0	3	0	2.00	3	1	349909
8	9	1	3	1	27.00	0	2	347742
9	10	1	2	1	14.00	1	0	237736
10	11	1	3	1	4.00	1	1	PP 9549
11	12	1	1	1	58.00	0	0	113783
12	13	0	3	0	20.00	0	0	A/5. 2151
13	14	0	3	0	39.00	1	5	347082
14	15	0	3	1	14.00	0	0	350406
15	16	1	2	1	55.00	0	0	248706
16	17	0	3	0	2.00	4	1	382652
17	18	1	2	0	NaN	0	0	244373
18	19	0	3	1	31.00	1	0	345763
19	20	1	3	1	NaN	0	0	2649
20	21	0	2	0	35.00	0	0	239865
21	22	1	2	0	34.00	0	0	248698
22	23	1	3	1	15.00	0	0	330923
23	24	1	1	0	28.00	0	0	113788
24	25	0	3	1	8.00	3	1	349909
25	26	1	3	1	38.00	1	5	347077
26	27	0	3	0	NaN	0	0	2631
27	28	0	1	0	19.00	3	2	19950
28	29	1	3	1	NaN	0	0	330959
29	30	0	3	0	NaN	0	0	349216
..	...	...	...	...	...	...	...	...
70	71	0	2	0	32.00	0	0	C.A. 33111
71	72	0	3	1	16.00	5	2	CA 2144
72	73	0	2	0	21.00	0	0	S.O.C. 14879
73	74	0	3	0	26.00	1	0	2680
74	75	1	3	0	32.00	0	0	1601
75	76	0	3	0	25.00	0	0	348123
76	77	0	3	0	NaN	0	0	349208
77	78	0	3	0	NaN	0	0	374746
78	79	1	2	0	0.83	0	2	248738
79	80	1	3	1	30.00	0	0	364516
80	81	0	3	0	22.00	0	0	345767
81	82	1	3	0	29.00	0	0	345779
82	83	1	3	1	NaN	0	0	330932
83	84	0	1	0	28.00	0	0	113059
84	85	1	2	1	17.00	0	0	SO/C 14885
85	86	1	3	1	33.00	3	0	3101278
86	87	0	3	0	16.00	1	3	W./C. 6608
87	88	0	3	0	NaN	0	0	SOTON/OQ 392086
88	89	1	1	1	23.00	3	2	19950
89	90	0	3	0	24.00	0	0	343275
90	91	0	3	0	29.00	0	0	343276

91	92	0	3	0	20.00	0	0	347466
92	93	0	1	0	46.00	1	0	W.E.P. 5734
93	94	0	3	0	26.00	1	2	C.A. 2315
94	95	0	3	0	59.00	0	0	364500
95	96	0	3	0	NaN	0	0	374910
96	97	0	1	0	71.00	0	0	PC 17754
97	98	1	1	0	23.00	0	1	PC 17759
98	99	1	2	1	34.00	0	1	231919
99	100	0	2	0	34.00	1	0	244367

	Fare	Cabin	Embarked	Title
0	7.2500	NaN	S	0
1	71.2833	C85	C	2
2	7.9250	NaN	S	1
3	53.1000	C123	S	2
4	8.0500	NaN	S	0
5	8.4583	NaN	Q	0
6	51.8625	E46	S	0
7	21.0750	NaN	S	3
8	11.1333	NaN	S	2
9	30.0708	NaN	C	2
10	16.7000	G6	S	1
11	26.5500	C103	S	1
12	8.0500	NaN	S	0
13	31.2750	NaN	S	0
14	7.8542	NaN	S	1
15	16.0000	NaN	S	2
16	29.1250	NaN	Q	3
17	13.0000	NaN	S	0
18	18.0000	NaN	S	2
19	7.2250	NaN	C	2
20	26.0000	NaN	S	0
21	13.0000	D56	S	0
22	8.0292	NaN	Q	1
23	35.5000	A6	S	0
24	21.0750	NaN	S	1
25	31.3875	NaN	S	2
26	7.2250	NaN	C	0
27	263.0000	C23 C25 C27	S	0
28	7.8792	NaN	Q	1
29	7.8958	NaN	S	0
..	...	...	...	...
70	10.5000	NaN	S	0
71	46.9000	NaN	S	1
72	73.5000	NaN	S	0
73	14.4542	NaN	C	0
74	56.4958	NaN	S	0
75	7.6500	F G73	S	0

```
76    7.8958          NaN        S       0
77    8.0500          NaN        S       0
78   29.0000          NaN        S       3
79   12.4750          NaN        S       1
80    9.0000          NaN        S       0
81    9.5000          NaN        S       0
82    7.7875          NaN        Q       1
83   47.1000          NaN        S       0
84   10.5000          NaN        S       1
85   15.8500          NaN        S       2
86   34.3750          NaN        S       0
87    8.0500          NaN        S       0
88  263.0000  C23 C25 C27        S       1
89    8.0500          NaN        S       0
90    8.0500          NaN        S       0
91    7.8542          NaN        S       0
92   61.1750          E31        S       0
93   20.5750          NaN        S       0
94    7.2500          NaN        S       0
95    8.0500          NaN        S       0
96   34.6542           A5        C       0
97   63.3583      D10 D12        C       0
98   23.0000          NaN        S       2
99   26.0000          NaN        S       0

[100 rows x 12 columns]
```

In [32]:
```
# fill missing age with median age for each title (Mr, Mrs, Miss, Others)
train["Age"].fillna(train.groupby("Title")["Age"].transform("median"), inplace=True)
test["Age"].fillna(test.groupby("Title")["Age"].transform("median"), inplace=True)
```

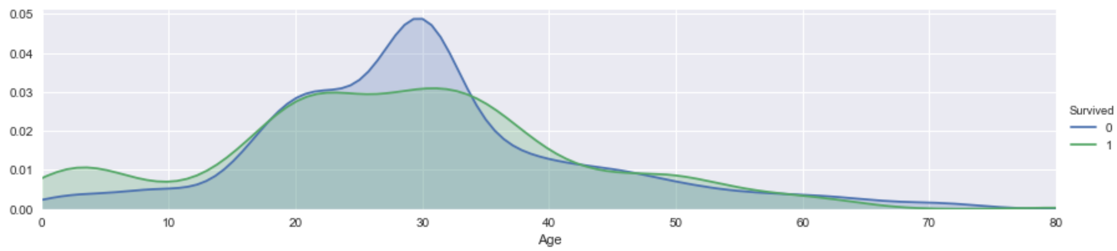
In [33]:
```
train.head(30)
train.groupby("Title")["Age"].transform("median")
```

Out[33]:
```
0       30.0
1       35.0
2       21.0
3       35.0
4       30.0
5       30.0
6       30.0
7        9.0
8       35.0
9       35.0
10      21.0
11      21.0
12      30.0
13      30.0
```

```
14     21.0
15     35.0
16      9.0
17     30.0
18     35.0
19     35.0
20     30.0
21     30.0
22     21.0
23     30.0
24     21.0
25     35.0
26     30.0
27     30.0
28     21.0
29     30.0
       ...
861    30.0
862    35.0
863    21.0
864    30.0
865    35.0
866    21.0
867    30.0
868    30.0
869     9.0
870    30.0
871    35.0
872    30.0
873    30.0
874    35.0
875    21.0
876    30.0
877    30.0
878    30.0
879    35.0
880    35.0
881    30.0
882    21.0
883    30.0
884    30.0
885    35.0
886     9.0
887    21.0
888    21.0
889    30.0
890    30.0
Name: Age, Length: 891, dtype: float64
```

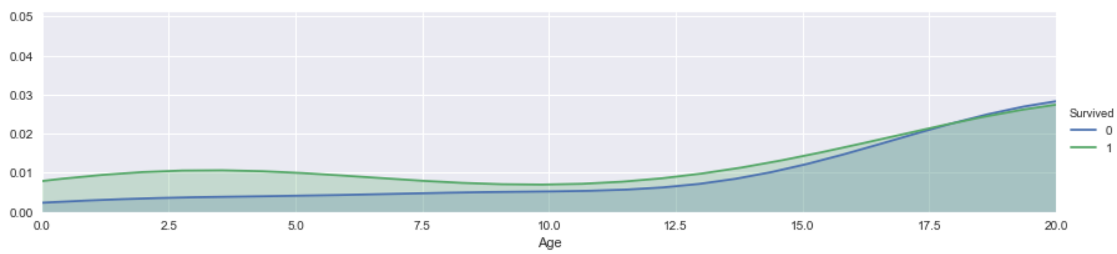
In [34]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Age',shade= True)
facet.set(xlim=(0, train['Age'].max()))
facet.add_legend()

plt.show()
```



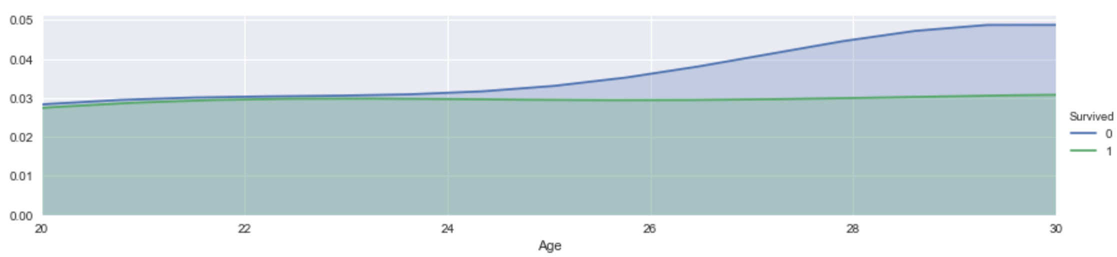
In [35]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Age',shade= True)
facet.set(xlim=(0, train['Age'].max()))
facet.add_legend()
plt.xlim(0, 20)
```

Out[35]: (0, 20)



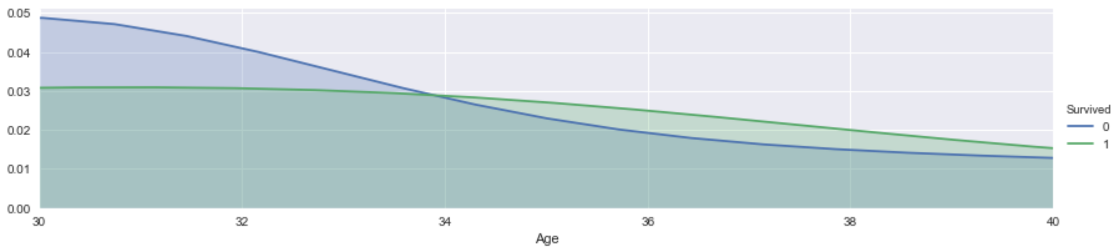
In [36]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Age',shade= True)
facet.set(xlim=(0, train['Age'].max()))
facet.add_legend()
plt.xlim(20, 30)
```

Out[36]: (20, 30)



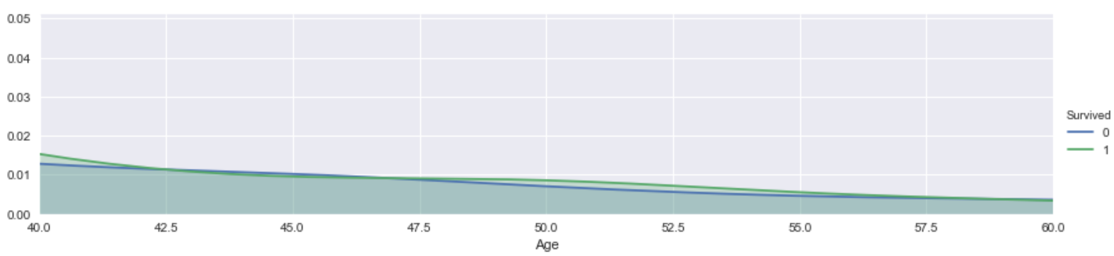
In [37]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Age',shade= True)
facet.set(xlim=(0, train['Age'].max()))
facet.add_legend()
plt.xlim(30, 40)
```

Out[37]: (30, 40)



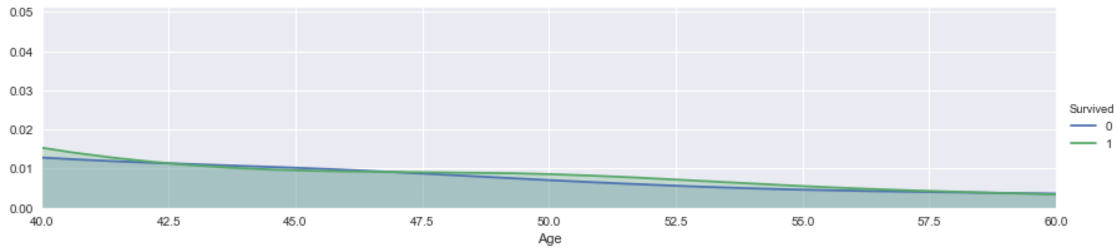
In [38]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Age',shade= True)
facet.set(xlim=(0, train['Age'].max()))
facet.add_legend()
plt.xlim(40, 60)
```

Out[38]: (40, 60)



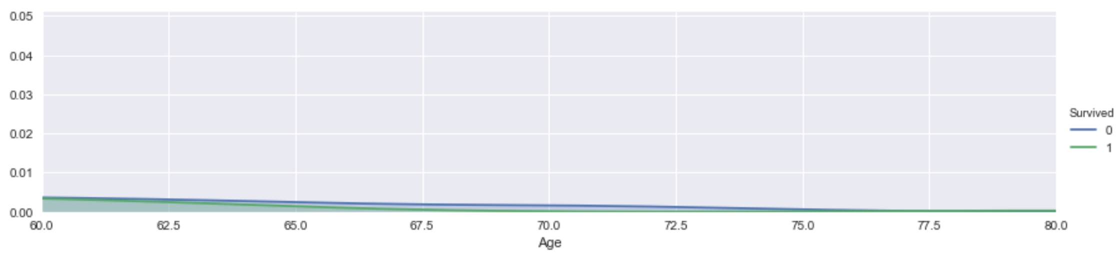
In [39]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Age',shade= True)
facet.set(xlim=(0, train['Age'].max()))
facet.add_legend()
plt.xlim(40, 60)
```

Out[39]: (40, 60)




```
In [40]: 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(60)
```

```
Out[40]: (60, 80.0)
```




```
In [41]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Sex            891 non-null int64
Age            891 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
Title          891 non-null int64
dtypes: float64(2), int64(7), object(3)
memory usage: 83.6+ KB
```

In [42]: test.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Sex            418 non-null int64
Age            418 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
Title          418 non-null int64
dtypes: float64(2), int64(6), object(3)
memory usage: 36.0+ KB
```

1.5

Binning/Converting Numerical Age to Categorical Variable
feature vector map:
child: 0
young: 1
adult: 2
mid-age: 3
senior: 4

In [43]:
```
for dataset in train_test_data:
    dataset.loc[ dataset['Age'] <= 16, 'Age'] = 0,
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 26), 'Age'] = 1,
    dataset.loc[(dataset['Age'] > 26) & (dataset['Age'] <= 36), 'Age'] = 2,
    dataset.loc[(dataset['Age'] > 36) & (dataset['Age'] <= 62), 'Age'] = 3,
    dataset.loc[ dataset['Age'] > 62, 'Age'] = 4
```

In [44]: train.head()

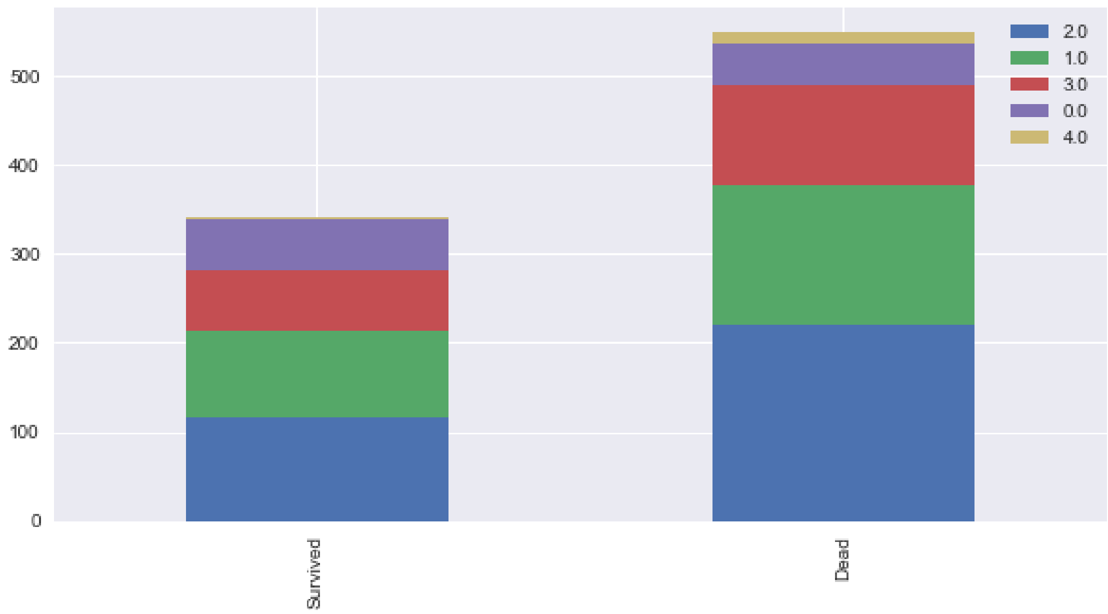
Out[44]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	\
0	1	0	3	0	1.0	1	0	A/5 21171	
1	2	1	1	1	3.0	1	0	PC 17599	
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	
3	4	1	1	1	2.0	1	0	113803	
4	5	0	3	0	2.0	0	0	373450	

	Fare	Cabin	Embarked	Title
0	7.2500	NaN	S	0
1	71.2833	C85	C	2

```
2   7.9250    NaN        S       1
3  53.1000   C123        S       2
4   8.0500    NaN        S       0
```

In [45]:
```
bar_chart('Age')
```

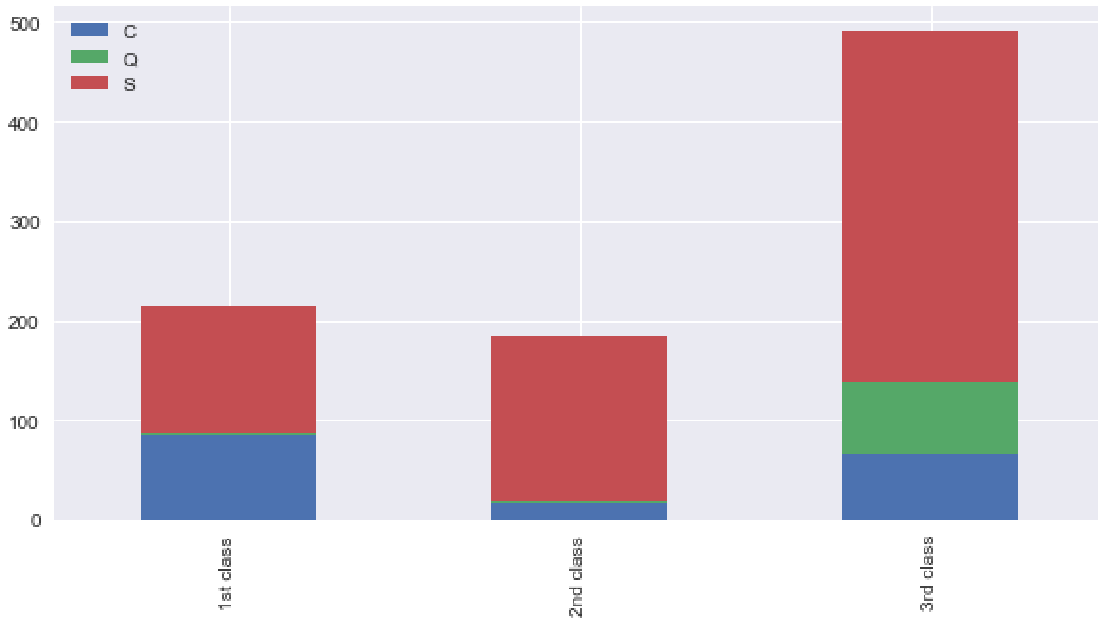


1.5.1 Embarked

filling missing values

In [46]:
```
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))
```

Out[46]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26830466390>
```



more than 50% of 1st class are from S embark
more than 50% of 2nd class are from S embark
more than 50% of 3rd class are from S embark
fill out missing embark with S embark

In [47]:
```
for dataset in train_test_data:
    dataset['Embarked'] = dataset['Embarked'].fillna('S')
```

In [48]:
```
train.head()
```

Out[48]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	\
0	1	0	3	0	1.0	1	0	A/5 21171	
1	2	1	1	1	3.0	1	0	PC 17599	
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	
3	4	1	1	1	2.0	1	0	113803	
4	5	0	3	0	2.0	0	0	373450	

	Fare	Cabin	Embarked	Title
0	7.2500	NaN	S	0
1	71.2833	C85	C	2
2	7.9250	NaN	S	1
3	53.1000	C123	S	2
4	8.0500	NaN	S	0

In [49]:
```
embarked_mapping = {"S": 0, "C": 1, "Q": 2}
for dataset in train_test_data:
    dataset['Embarked'] = dataset['Embarked'].map(embarked_mapping)
```

1.5.2 Fare

In [50]:
```
# fill missing Fare with median fare for each Pclass
train["Fare"].fillna(train.groupby("Pclass")["Fare"].transform("median"), inplace=True
test["Fare"].fillna(test.groupby("Pclass")["Fare"].transform("median"), inplace=True)
train.head(50)
```

Out[50]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	\
0	1	0	3	0	1.0	1	0	A/5 21171	
1	2	1	1	1	3.0	1	0	PC 17599	
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	
3	4	1	1	1	2.0	1	0	113803	
4	5	0	3	0	2.0	0	0	373450	
5	6	0	3	0	2.0	0	0	330877	
6	7	0	1	0	3.0	0	0	17463	
7	8	0	3	0	0.0	3	1	349909	
8	9	1	3	1	2.0	0	2	347742	
9	10	1	2	1	0.0	1	0	237736	
10	11	1	3	1	0.0	1	1	PP 9549	
11	12	1	1	1	3.0	0	0	113783	
12	13	0	3	0	1.0	0	0	A/5. 2151	
13	14	0	3	0	3.0	1	5	347082	
14	15	0	3	1	0.0	0	0	350406	
15	16	1	2	1	3.0	0	0	248706	
16	17	0	3	0	0.0	4	1	382652	
17	18	1	2	0	2.0	0	0	244373	
18	19	0	3	1	2.0	1	0	345763	
19	20	1	3	1	2.0	0	0	2649	
20	21	0	2	0	2.0	0	0	239865	
21	22	1	2	0	2.0	0	0	248698	
22	23	1	3	1	0.0	0	0	330923	
23	24	1	1	0	2.0	0	0	113788	
24	25	0	3	1	0.0	3	1	349909	
25	26	1	3	1	3.0	1	5	347077	
26	27	0	3	0	2.0	0	0	2631	
27	28	0	1	0	1.0	3	2	19950	
28	29	1	3	1	1.0	0	0	330959	
29	30	0	3	0	2.0	0	0	349216	
30	31	0	1	0	3.0	0	0	PC 17601	
31	32	1	1	1	2.0	1	0	PC 17569	
32	33	1	3	1	1.0	0	0	335677	
33	34	0	2	0	4.0	0	0	C.A. 24579	
34	35	0	1	0	2.0	1	0	PC 17604	
35	36	0	1	0	3.0	1	0	113789	
36	37	1	3	0	2.0	0	0	2677	
37	38	0	3	0	1.0	0	0	A./5. 2152	
38	39	0	3	1	1.0	2	0	345764	
39	40	1	3	1	0.0	1	0	2651	
40	41	0	3	1	3.0	1	0	7546	

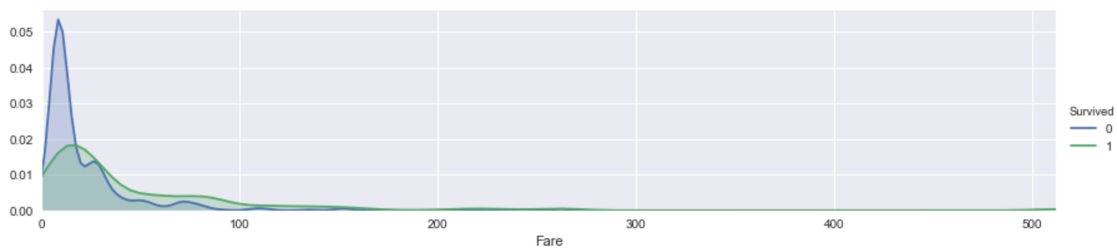
41	42	0	2	1	2.0	1	0	11668
42	43	0	3	0	2.0	0	0	349253
43	44	1	2	1	0.0	1	2	SC/Paris 2123
44	45	1	3	1	1.0	0	0	330958
45	46	0	3	0	2.0	0	0	S.C./A.4. 23567
46	47	0	3	0	2.0	1	0	370371
47	48	1	3	1	1.0	0	0	14311
48	49	0	3	0	2.0	2	0	2662
49	50	0	3	1	1.0	1	0	349237

	Fare	Cabin	Embarked	Title
0	7.2500	NaN	0	0
1	71.2833	C85	1	2
2	7.9250	NaN	0	1
3	53.1000	C123	0	2
4	8.0500	NaN	0	0
5	8.4583	NaN	2	0
6	51.8625	E46	0	0
7	21.0750	NaN	0	3
8	11.1333	NaN	0	2
9	30.0708	NaN	1	2
10	16.7000	G6	0	1
11	26.5500	C103	0	1
12	8.0500	NaN	0	0
13	31.2750	NaN	0	0
14	7.8542	NaN	0	1
15	16.0000	NaN	0	2
16	29.1250	NaN	2	3
17	13.0000	NaN	0	0
18	18.0000	NaN	0	2
19	7.2250	NaN	1	2
20	26.0000	NaN	0	0
21	13.0000	D56	0	0
22	8.0292	NaN	2	1
23	35.5000	A6	0	0
24	21.0750	NaN	0	1
25	31.3875	NaN	0	2
26	7.2250	NaN	1	0
27	263.0000	C23 C25 C27	0	0
28	7.8792	NaN	2	1
29	7.8958	NaN	0	0
30	27.7208	NaN	1	3
31	146.5208	B78	1	2
32	7.7500	NaN	2	1
33	10.5000	NaN	0	0
34	82.1708	NaN	1	0
35	52.0000	NaN	0	0
36	7.2292	NaN	1	0

37	8.0500	NaN	0	0
38	18.0000	NaN	0	1
39	11.2417	NaN	1	1
40	9.4750	NaN	0	2
41	21.0000	NaN	0	2
42	7.8958	NaN	1	0
43	41.5792	NaN	1	1
44	7.8792	NaN	2	1
45	8.0500	NaN	0	0
46	15.5000	NaN	2	0
47	7.7500	NaN	2	1
48	21.6792	NaN	1	0
49	17.8000	NaN	0	2

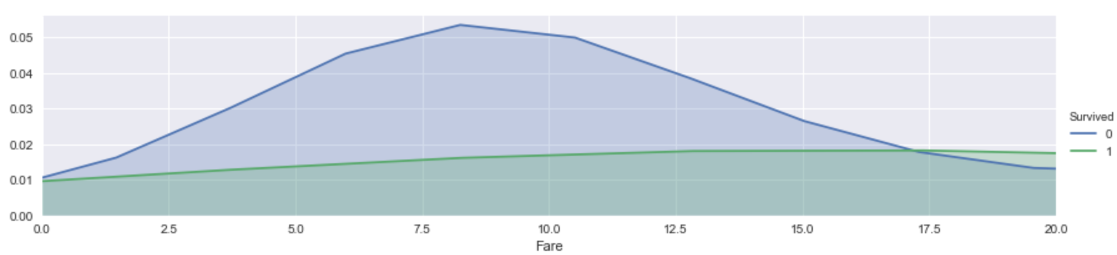
In [51]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Fare',shade= True)
facet.set(xlim=(0, train['Fare'].max()))
facet.add_legend()

plt.show()
```



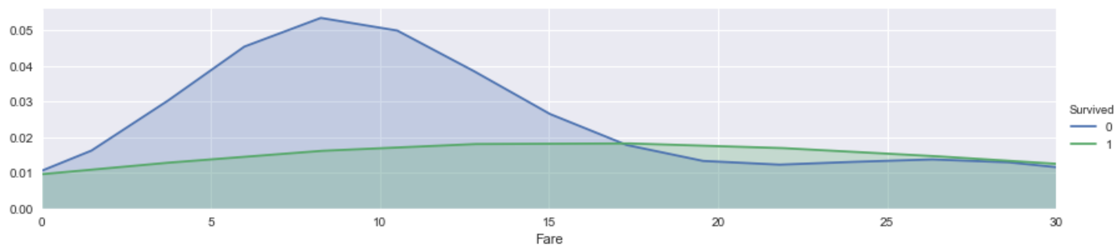
In [52]:
```
facet = sns.FacetGrid(train, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Fare',shade= True)
facet.set(xlim=(0, train['Fare'].max()))
facet.add_legend()
plt.xlim(0, 20)
```

Out[52]: (0, 20)



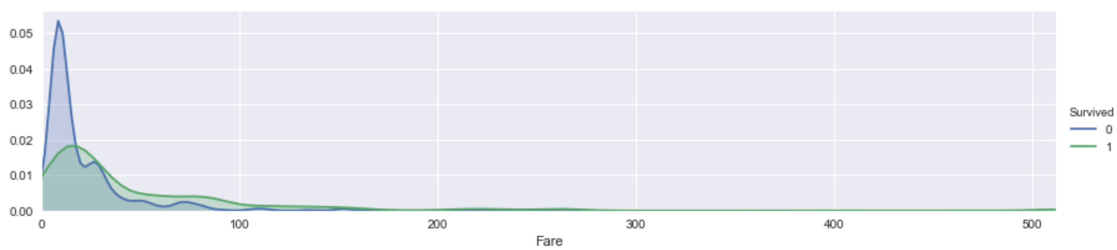
```
In [53]: facet = sns.FacetGrid(train, hue="Survived",aspect=4)
         facet.map(sns.kdeplot,'Fare',shade= True)
         facet.set(xlim=(0, train['Fare'].max()))
         facet.add_legend()
         plt.xlim(0, 30)
```

```
Out[53]: (0, 30)
```




```
In [54]: 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)
```

```
Out[54]: (0, 512.32920000000001)
```




```
In [55]: for dataset in train_test_data:
             dataset.loc[ dataset['Fare'] <= 17, 'Fare'] = 0,
             dataset.loc[(dataset['Fare'] > 17) & (dataset['Fare'] <= 30), 'Fare'] = 1,
             dataset.loc[(dataset['Fare'] > 30) & (dataset['Fare'] <= 100), 'Fare'] = 2,
             dataset.loc[ dataset['Fare'] > 100, 'Fare'] = 3
```

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

```
Out[56]:    PassengerId  Survived  Pclass  Sex  Age  SibSp  Parch            Ticket  \
         0            1         0       3    0  1.0      1      0         A/5 21171
         1            2         1       1    1  3.0      1      0          PC 17599
```

```
2            3         1       3    1  1.0       0       0  STON/O2. 3101282
3            4         1       1    1  2.0       1       0            113803
4            5         0       3    0  2.0       0       0            373450

   Fare Cabin  Embarked  Title
0   0.0   NaN         0      0
1   2.0   C85         1      2
2   0.0   NaN         0      1
3   2.0  C123         0      2
4   0.0   NaN         0      0
```

1.5.3 Cabin

In [57]:
```
train.Cabin.value_counts()
```

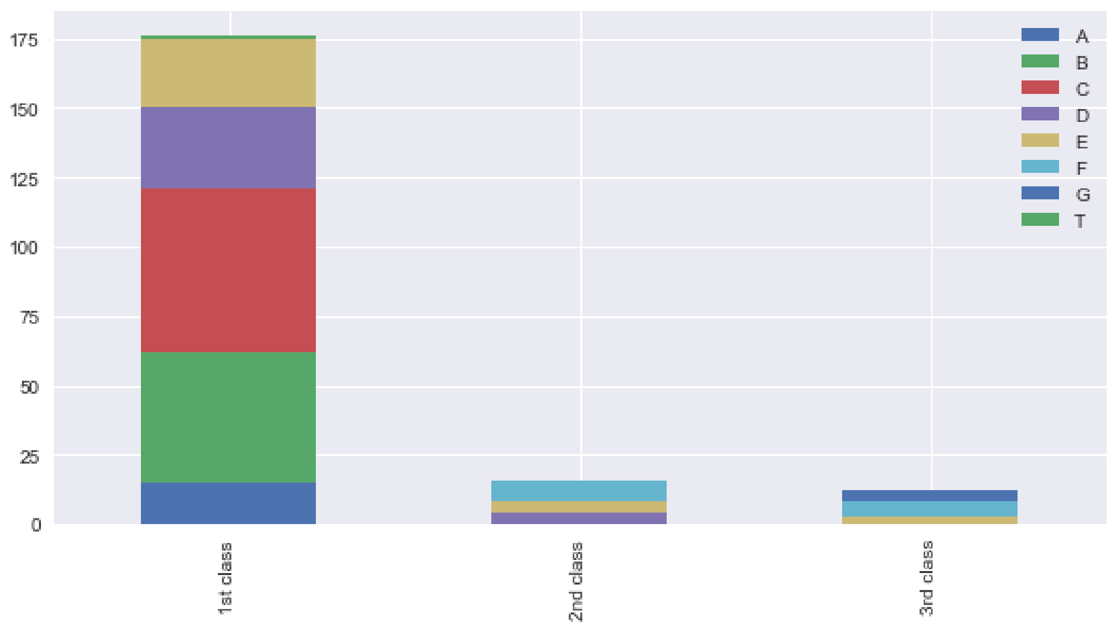
Out[57]:
```
C23 C25 C27        4
G6                 4
B96 B98            4
F33                3
F2                 3
E101               3
C22 C26            3
D                  3
D36                2
D20                2
D33                2
C65                2
E44                2
C83                2
B5                 2
B57 B59 B63 B66    2
E24                2
E33                2
E25                2
C2                 2
B18                2
C78                2
C68                2
B35                2
E8                 2
F4                 2
D35                2
C123               2
C93                2
B22                2
                  ..
C30                1
E50                1
```

```
B30                  1
B41                  1
C104                 1
D9                   1
A7                   1
B101                 1
C87                  1
C128                 1
F E69                1
B94                  1
C49                  1
E12                  1
D28                  1
C118                 1
B79                  1
B86                  1
C103                 1
F38                  1
A16                  1
E36                  1
C110                 1
C47                  1
B82 B84              1
B37                  1
E31                  1
A32                  1
E10                  1
D48                  1
Name: Cabin, Length: 147, dtype: int64
```

In [58]:
```
for dataset in train_test_data:
    dataset['Cabin'] = dataset['Cabin'].str[:1]
```

In [59]:
```
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[59]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2683155d0f0>
```




```
In [60]: cabin_mapping = {"A": 0, "B": 0.4, "C": 0.8, "D": 1.2, "E": 1.6, "F": 2, "G": 2.4, "T"
         for dataset in train_test_data:
             dataset['Cabin'] = dataset['Cabin'].map(cabin_mapping)
```

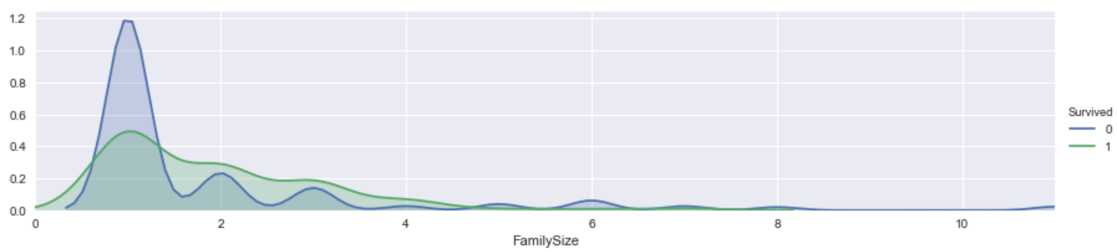
```
In [61]: # fill missing Fare with median fare for each Pclass
         train["Cabin"].fillna(train.groupby("Pclass")["Cabin"].transform("median"), inplace=Tr
         test["Cabin"].fillna(test.groupby("Pclass")["Cabin"].transform("median"), inplace=True
```

1.5.4 FamilySize

```
In [62]: train["FamilySize"] = train["SibSp"] + train["Parch"] + 1
         test["FamilySize"] = test["SibSp"] + test["Parch"] + 1
```

```
In [63]: 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)
```

```
Out[63]: (0, 11.0)
```



In [64]:
```
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: 3.2,
for dataset in train_test_data:
    dataset['FamilySize'] = dataset['FamilySize'].map(family_mapping)
```

In [65]:
```
train.head()
```

Out[65]:
```
   PassengerId  Survived  Pclass  Sex  Age  SibSp  Parch            Ticket  \
0            1         0       3    0  1.0      1      0         A/5 21171
1            2         1       1    1  3.0      1      0          PC 17599
2            3         1       3    1  1.0      0      0  STON/O2. 3101282
3            4         1       1    1  2.0      1      0            113803
4            5         0       3    0  2.0      0      0            373450

   Fare  Cabin  Embarked  Title  FamilySize
0   0.0    2.0         0      0         0.4
1   2.0    0.8         1      2         0.4
2   0.0    2.0         0      1         0.0
3   2.0    0.8         0      2         0.4
4   0.0    2.0         0      0         0.0
```

In [66]:
```
train.head()
```

Out[66]:
```
   PassengerId  Survived  Pclass  Sex  Age  SibSp  Parch            Ticket  \
0            1         0       3    0  1.0      1      0         A/5 21171
1            2         1       1    1  3.0      1      0          PC 17599
2            3         1       3    1  1.0      0      0  STON/O2. 3101282
3            4         1       1    1  2.0      1      0            113803
4            5         0       3    0  2.0      0      0            373450

   Fare  Cabin  Embarked  Title  FamilySize
0   0.0    2.0         0      0         0.4
1   2.0    0.8         1      2         0.4
2   0.0    2.0         0      1         0.0
3   2.0    0.8         0      2         0.4
4   0.0    2.0         0      0         0.0
```

In [67]:
```
### delete features
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 [68]:
```
train_data = train.drop('Survived', axis=1)
target = train['Survived']

train_data.shape, target.shape
```

Out[68]: ((891, 8), (891,))

In [69]:
```
train_data.head(10)
```

Out[69]:

	Pclass	Sex	Age	Fare	Cabin	Embarked	Title	FamilySize
0	3	0	1.0	0.0	2.0	0	0	0.4
1	1	1	3.0	2.0	0.8	1	2	0.4
2	3	1	1.0	0.0	2.0	0	1	0.0
3	1	1	2.0	2.0	0.8	0	2	0.4
4	3	0	2.0	0.0	2.0	0	0	0.0
5	3	0	2.0	0.0	2.0	2	0	0.0
6	1	0	3.0	2.0	1.6	0	0	0.0
7	3	0	0.0	1.0	2.0	0	3	1.6
8	3	1	2.0	0.0	2.0	0	2	0.8
9	2	1	0.0	2.0	1.8	1	2	0.4

1.6 Modelling

In [70]:
```
# Importing Classifier Modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

import numpy as np
```

In [71]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
Survived      891 non-null int64
Pclass        891 non-null int64
Sex           891 non-null int64
Age           891 non-null float64
Fare          891 non-null float64
Cabin         891 non-null float64
Embarked      891 non-null int64
Title         891 non-null int64
FamilySize    891 non-null float64
dtypes: float64(4), int64(5)
memory usage: 62.7 KB
```

1.6.1 Cross Validation (K-fold)

In [72]:
```
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)
```

1.6.2 kNN

```
In [73]: clf = KNeighborsClassifier(n_neighbors = 13)
         scoring = 'accuracy'
         score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
         print(score)
```

```
[ 0.82222222  0.76404494  0.80898876  0.83146067  0.87640449  0.82022472
  0.85393258  0.79775281  0.84269663  0.84269663]
```

```
In [74]: # kNN Score
         round(np.mean(score)*100, 2)
```

```
Out[74]: 82.599999999999994
```

1.6.3 Decision Tree

```
In [75]: clf = DecisionTreeClassifier()
         scoring = 'accuracy'
         score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
         print(score)
```

```
[ 0.76666667  0.83146067  0.7752809   0.7752809   0.88764045  0.7752809
  0.83146067  0.82022472  0.74157303  0.79775281]
```

```
In [76]: # decision tree Score
         round(np.mean(score)*100, 2)
```

```
Out[76]: 80.030000000000001
```

1.6.4 Ramdom Forest

```
In [77]: clf = RandomForestClassifier(n_estimators=13)
         scoring = 'accuracy'
         score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
         print(score)
```

```
[ 0.8         0.84269663  0.79775281  0.78651685  0.8988764   0.79775281
  0.83146067  0.79775281  0.75280899  0.80898876]
```

```
In [78]: # Random Forest Score
         round(np.mean(score)*100, 2)
```

```
Out[78]: 81.150000000000006
```

1.6.5 Naive Bayes

```
In [79]: clf = GaussianNB()
         scoring = 'accuracy'
         score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
         print(score)
```

```
[ 0.85555556  0.73033708  0.75280899  0.75280899  0.70786517  0.80898876
  0.76404494  0.80898876  0.86516854  0.83146067]
```

```
In [80]: # Naive Bayes Score
         round(np.mean(score)*100, 2)
```

```
Out[80]: 78.780000000000001
```

1.6.6 SVM

```
In [81]: clf = SVC()
         scoring = 'accuracy'
         score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
         print(score)
```

```
[ 0.83333333  0.80898876  0.83146067  0.82022472  0.84269663  0.82022472
  0.84269663  0.85393258  0.83146067  0.86516854]
```

```
In [82]: round(np.mean(score)*100,2)
```

```
Out[82]: 83.5
```

1.7 Testing

```
In [83]: clf = SVC()
         clf.fit(train_data, target)

         test_data = test.drop("PassengerId", axis=1).copy()
         prediction = clf.predict(test_data)
```

```
In [84]: submission = pd.DataFrame({
                 "PassengerId": test["PassengerId"],
                 "Survived": prediction
             })

         submission.to_csv('submission.csv', index=False)
```

```
In [85]: submission = pd.read_csv('submission.csv')
         submission.head()
```

```
Out[85]:    PassengerId  Survived
         0          892         0
         1          893         1
         2          894         0
         3          895         0
         4          896         1
```

1.8 References

https://www.youtube.com/watch?v=3eTSVGY_fIE