In [1]:

```
# Python 3.7
# Using famous Titanic data to predict the people who survived or did not survive
# Using the relatively clean data as compared to real original data

# Usual imports
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [62]:

```
# Load the data
titanic = pd.read_csv('titanic_train.csv')
```

In [3]:

```
# Check the head
titanic.head()
```

Out[3]:

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

In [4]:

```
# check the info() method
titanic.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
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]:

```
# This is a train set of titanic data so titanic is not a good name of the dataframe
titanic_train = titanic
# We find from info method that there are quit a null values
# to visualize these null values there are two methods
# Method No. 1
titanic_train.isnull().sum()
```

Out[7]:

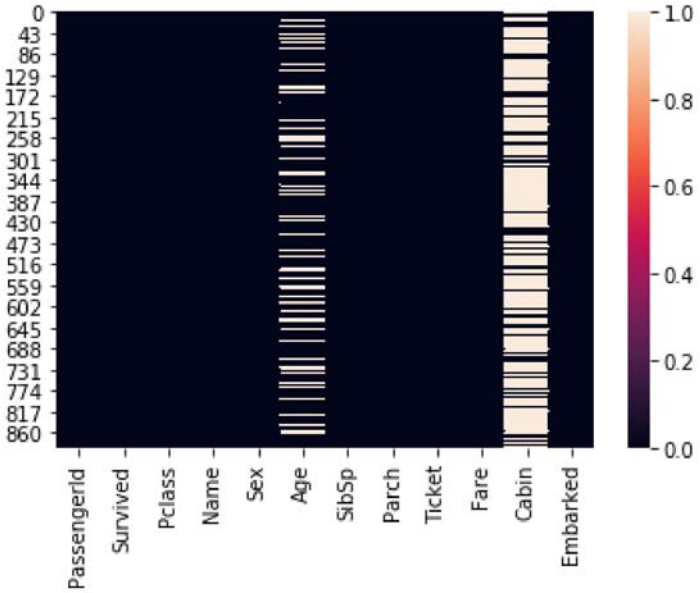
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [8]:

```
# Method No. 2
sns.heatmap(titanic_train.isnull())
# Use some useful attributes to make it look pretty. Check the next cell out
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e625e9c50>
```

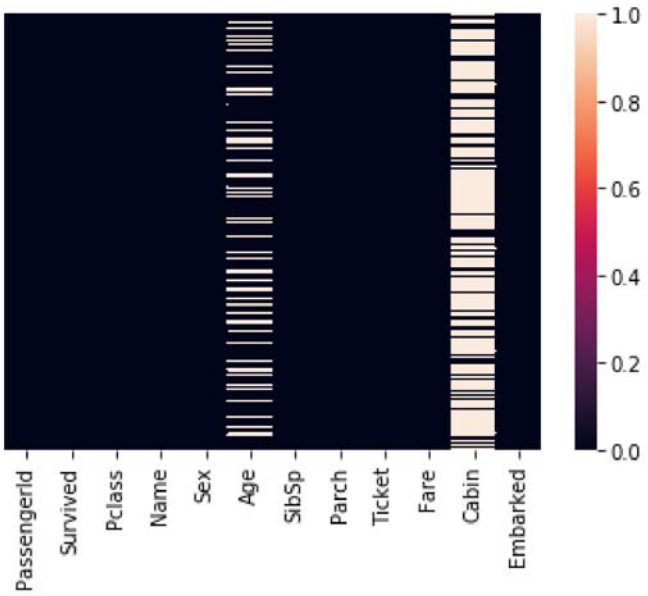


In [10]:

```
sns.heatmap(titanic_train.isnull(), yticklabels=False)
# check out the next cell
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e6196d8d0>
```

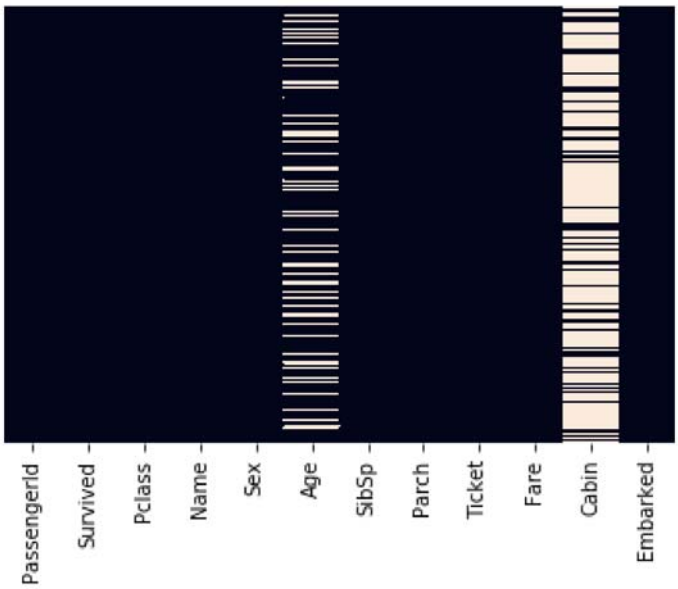


In [11]:

```
sns.heatmap(titanic_train.isnull(), yticklabels=False, cbar=False)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e622873c8>
```

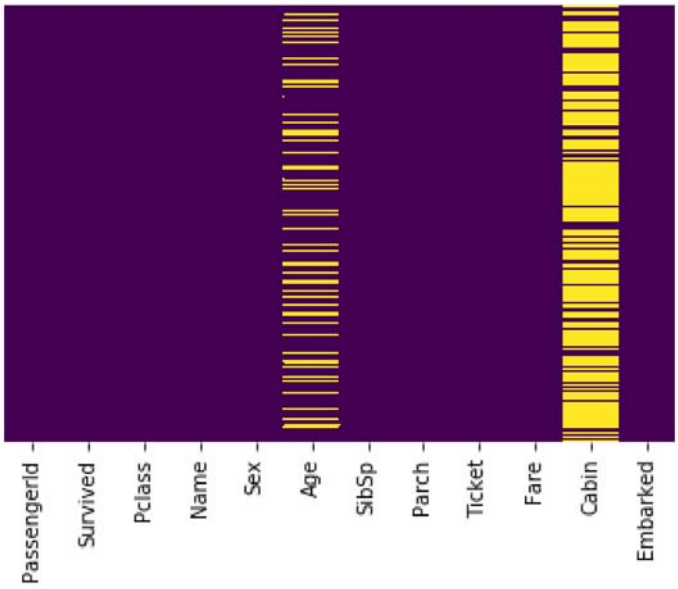


In [13]:

```
sns.heatmap(titanic_train.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e62932940>
```



In [14]:

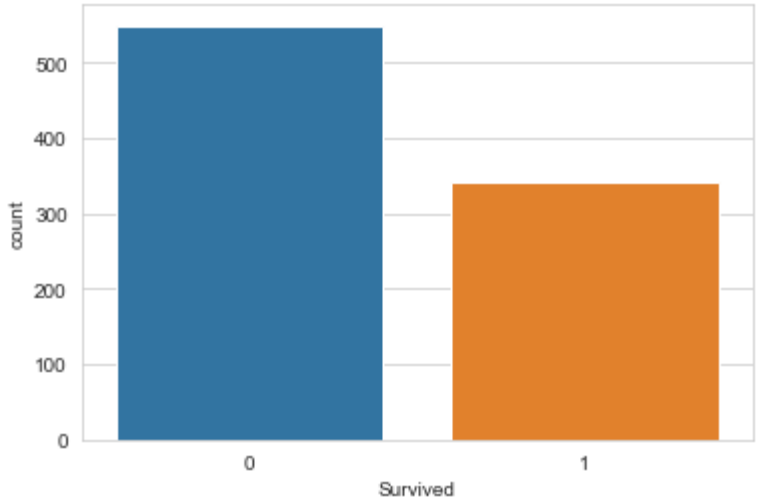
```
# Set the format for seaborn
sns.set_style('whitegrid')
```

In [15]:

```
# let's make count plot for the people survived
sns.countplot(x=titanic_train['Survived'])
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e62011160>
```



In [18]:

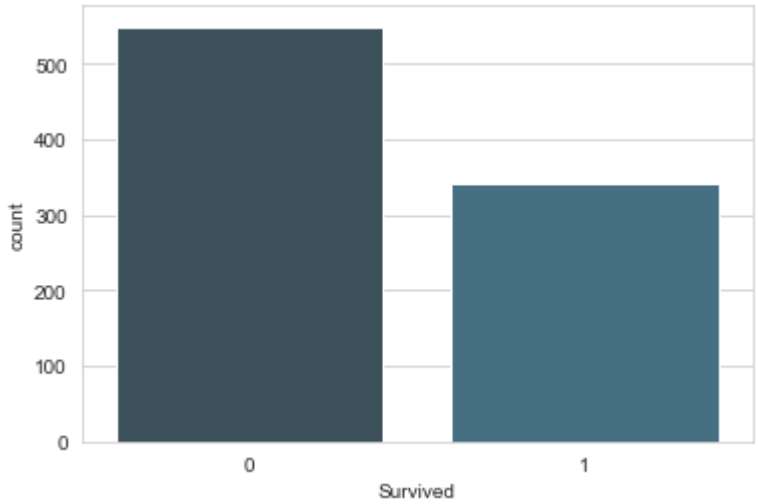
```
# Let's set palette in sns and run teh above command again to check the difference
sns.set_palette("GnBu_d")
sns.set_style('whitegrid')
```

In [19]:

```
# BTW another way to feed the 'survived' data to the seaborn is to do it as following
sns.countplot(x='Survived',data=titanic_train)
# I do like the default version of the palette settings. How can I get it back???
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e62fde9b0>
```

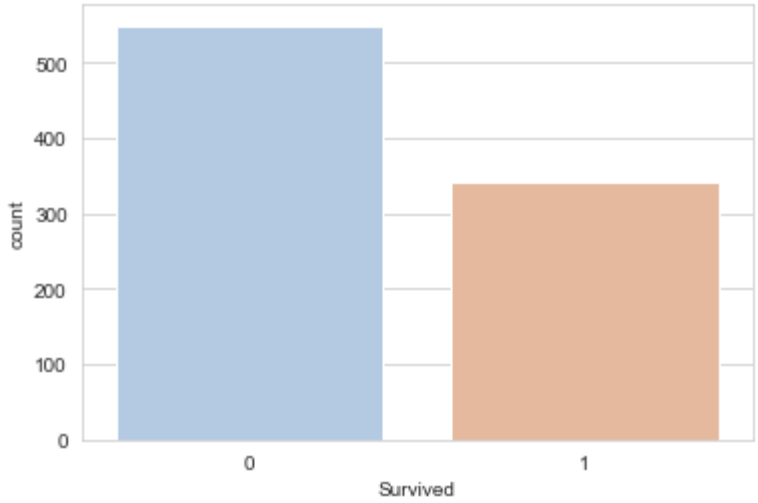


In [38]:

```
# Pass an arguments 'pastel' in the set_palette() method
sns.countplot(x='Survived',data=titanic_train)
# But the default brightness is missing
```

Out[38]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e629c89e8>
```

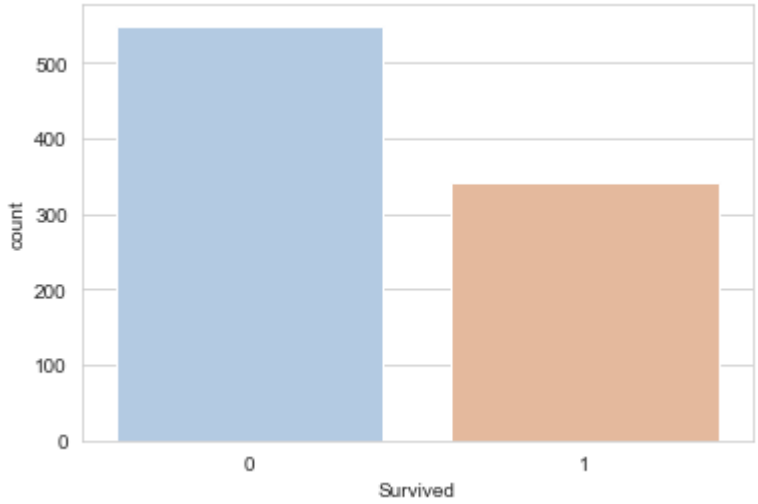


In [33]:

```
sns.set_palette('pastel', 10, .75)
sns.countplot(x='Survived',data=titanic_train)
```

Out[33]:

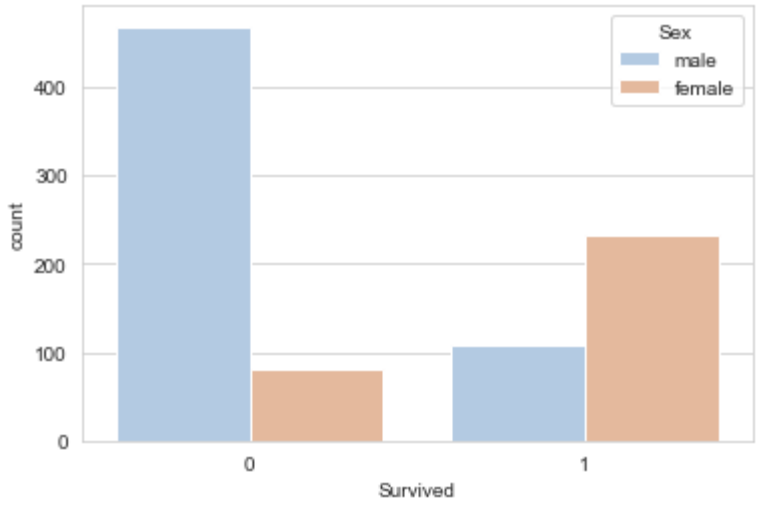
```
<matplotlib.axes._subplots.AxesSubplot at 0x29e62b12588>
```



In [35]:

```
# Moving on....
# add another dimention by passing the hue argument
sns.countplot(x='Survived',data=titanic_train, hue='Sex')
# It seems you have a bad luck while surviving from the titanic disaster if you are mal
e
# Is above statement true? ratios are better way of addressing
# roughly looking at the plot suggests
total_male = 480 + 105
total_female = 90 + 220
male_survival_rate = 105/total_male
female_survival_rate = 220/total_female
print('male_survival_rate: ', male_survival_rate)
print('female_survival_rate: ', female_survival_rate)
```

```
male_survival_rate:  0.1794871794871795
female_survival_rate:  0.7096774193548387
```

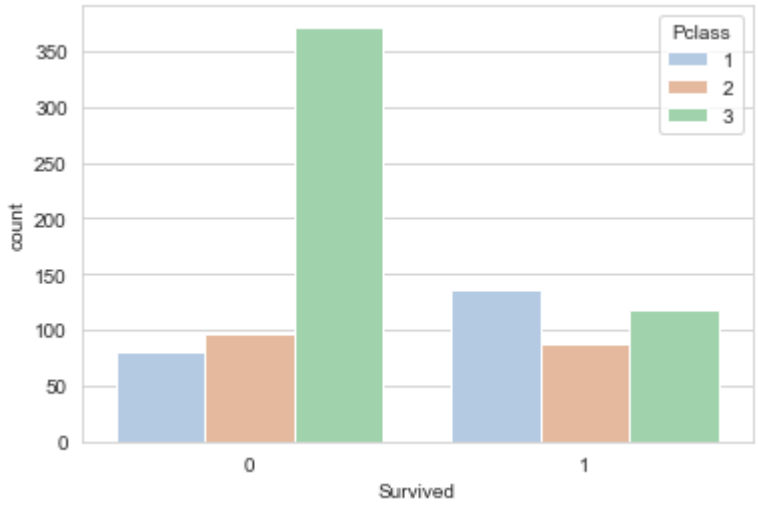


In [36]:

```
# Let's use class as the hue
sns.countplot(x='Survived',data=titanic_train, hue='Pclass')
# Clearly if you are a female in 1st class you have the highest chances of surviving
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e63081e48>
```

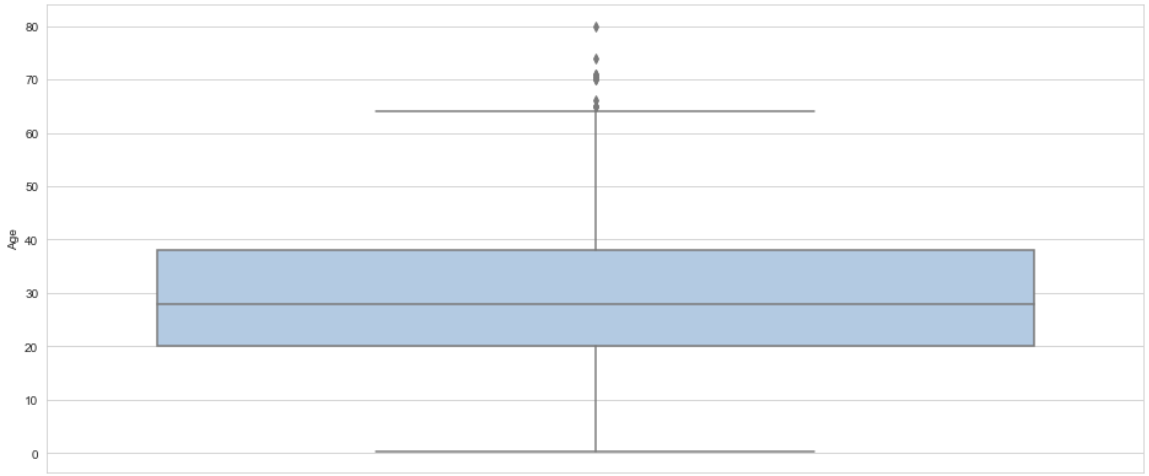


In [45]:

```
# Let's check out the age spread of the people on titanic
plt.figure(figsize=(16,7))
sns.boxplot(data=titanic_train, y='Age')
# We have 28 as a average age of a passanger
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e64709d68>
```



In [57]:

```
# Let's say we decide to add this to each of the null value present in the 'Age' column
# we would have to build a function and then apply that function to the 'Age' column
# let's first do this with one average value like above which is 28
# First built a function

def mean_age(age):
    if pd.isnull(age):
        return 28
    else:
        return age



titanic_train['Age_with_mean'] = titanic_train['Age'].apply(mean_age)
```

In [55]:

```
print(titanic_train[['Age', 'Age_with_mean']])
```

	Age	Age_with_mean
0	22.0	22.0
1	38.0	38.0
2	26.0	26.0
3	35.0	35.0
4	35.0	35.0
5	NaN	28.0
6	54.0	54.0
7	2.0	2.0
8	27.0	27.0
9	14.0	14.0
10	4.0	4.0
11	58.0	58.0
12	20.0	20.0
13	39.0	39.0
14	14.0	14.0
15	55.0	55.0
16	2.0	2.0
17	NaN	28.0
18	31.0	31.0
19	NaN	28.0
20	35.0	35.0
21	34.0	34.0
22	15.0	15.0
23	28.0	28.0
24	8.0	8.0
25	38.0	38.0
26	NaN	28.0
27	19.0	19.0
28	NaN	28.0
29	NaN	28.0
..	...	...
861	21.0	21.0
862	48.0	48.0
863	NaN	28.0
864	24.0	24.0
865	42.0	42.0
866	27.0	27.0
867	31.0	31.0
868	NaN	28.0
869	4.0	4.0
870	26.0	26.0
871	47.0	47.0
872	33.0	33.0
873	47.0	47.0
874	28.0	28.0
875	15.0	15.0
876	20.0	20.0
877	19.0	19.0
878	NaN	28.0
879	56.0	56.0
880	25.0	25.0
881	33.0	33.0
882	22.0	22.0
883	28.0	28.0
884	25.0	25.0
885	39.0	39.0
886	27.0	27.0
887	19.0	19.0
888	NaN	28.0
889	26.0	26.0

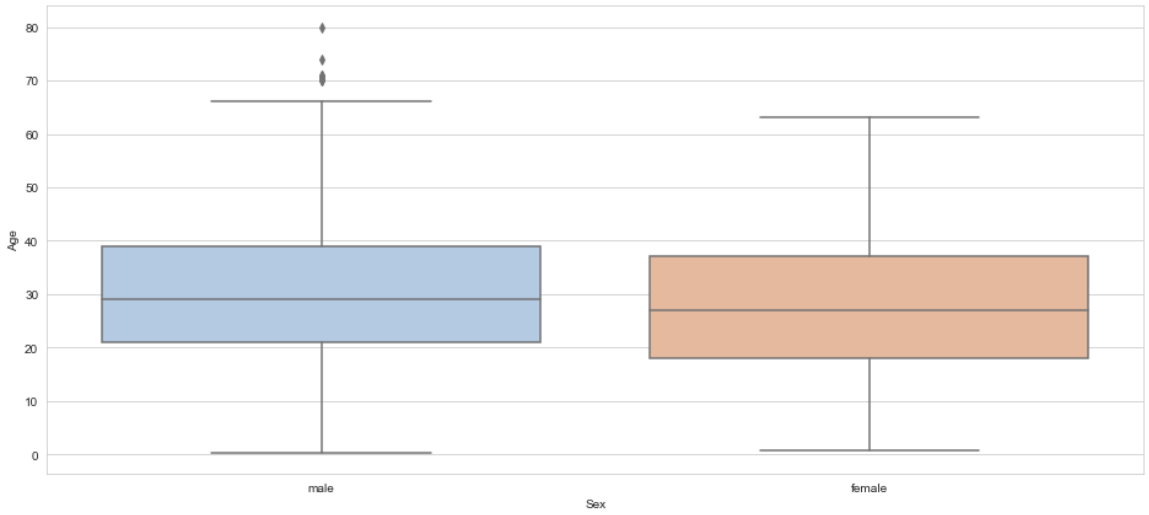
```
890  32.0           32.0

[891 rows x 2 columns]
```

In [58]:

```
# Let's make this more complicated
# Instead of taking one mean let's change it to two means one for men and one for women
plt.figure(figsize=(16,7))
sns.boxplot(data=titanic_train, x='Sex',y='Age')
```

Out[58]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e6465ea90>
```



In [65]:

```
def mean_age_perSex(oranges):
    if pd.isnull(oranges[0]):
        if oranges[1]=='male':
            return 29
        elif oranges[1]=='female':
            return 27
    else:
        return oranges[0]


titanic_train['Age_with_mean_per_sex'] = titanic_train[['Age','Sex']].apply(mean_age_pe
rSex, axis=1)
titanic_train[['Age_with_mean_per_sex','Age']].head(10)
```

Out[65]:

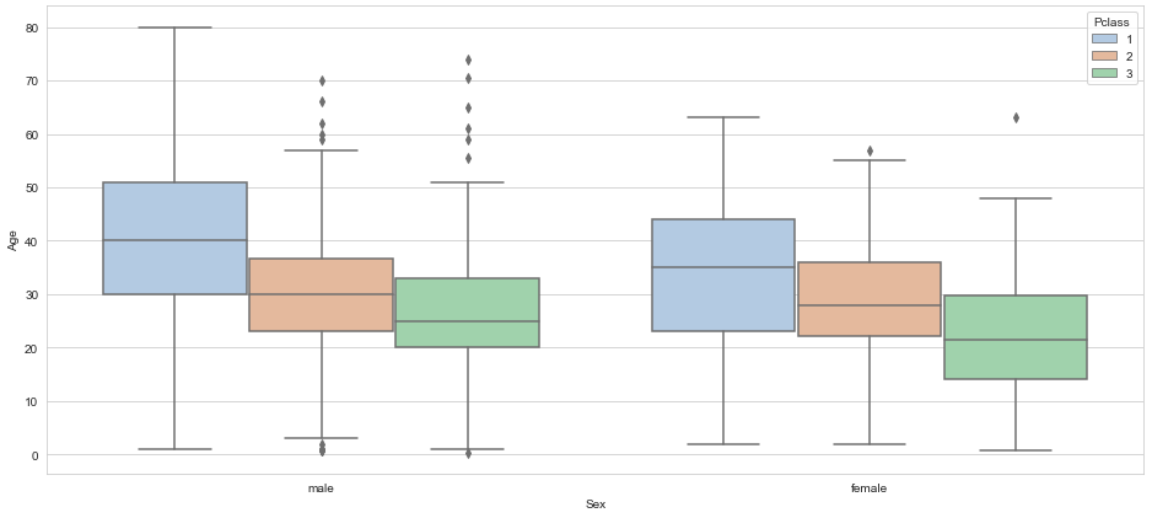
	Age_with_mean_per_sex	Age
0	22.0	22.0
1	38.0	38.0
2	26.0	26.0
3	35.0	35.0
4	35.0	35.0
5	29.0	NaN
6	54.0	54.0
7	2.0	2.0
8	27.0	27.0
9	14.0	14.0

In [66]:

```
# Now further add the hue=Pclass and find average for each of six categories
plt.figure(figsize=(16,7))
sns.boxplot(data=titanic_train, x='Sex',y='Age',hue='Pclass')
# Averages are Male(Pclass1 : 40; Pclass2 : 30, Pclass3: 24) and for Female(Pclass1 : 3
5; Pclass2 : 28, Pclass3: 22)
```

Out[66]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e64b71f28>
```



In [67]:

```
# Averages are Male(Pclass1 : 40; Pclass2 : 30, Pclass3: 24) and for Female(Pclass1 : 3
5; Pclass2 : 28, Pclass3: 22)
def mean_age_perSex_perPclass(oranges):
    if pd.isnull(oranges[0]):
        if oranges[1]=='male':
            if oranges[2]==1:
                return 40
            elif oranges[2]==2:
                return 30
            elif oranges[2]==3:
                return 24
        elif oranges[1]=='female':
            if oranges[2]==1:
                return 35
            elif oranges[2]==2:
                return 28
            elif oranges[2]==3:
                return 22
    else:
        return oranges[0]


titanic_train['Age_with_mean_per_sex_perPclass'] = titanic_train[['Age','Sex','Pclass'
]].apply(mean_age_perSex_perPclass, axis=1)
titanic_train[['Age_with_mean_per_sex_perPclass','Age']].head(10)
```

Out[67]:

	Age_with_mean_per_sex_perPclass	Age
0	22.0	22.0
1	38.0	38.0
2	26.0	26.0
3	35.0	35.0
4	35.0	35.0
5	24.0	NaN
6	54.0	54.0
7	2.0	2.0
8	27.0	27.0
9	14.0	14.0

In [83]:

```
import numpy as np
```

In [93]:

```
# To get the values of NaNs from the 'Age' column use...
# ... np.isnan() method on the 'Age' column
np.isnan(titanic_train[['Age']]) # one set of square bracket will return series
```

Out[93]:

	Age
0	False
1	False
2	False
3	False
4	False
5	True
6	False
7	False
8	False
9	False
10	False
11	False
12	False
13	False
14	False
15	False
16	False
17	True
18	False
19	True
20	False
21	False
22	False
23	False
24	False
25	False
26	True
27	False
28	True
29	True
...	...
861	False
862	False
863	True
864	False
865	False
866	False

	Age
867	False
868	True
869	False
870	False
871	False
872	False
873	False
874	False
875	False
876	False
877	False
878	True
879	False
880	False
881	False
882	False
883	False
884	False
885	False
886	False
887	False
888	True
889	False
890	False

891 rows × 1 columns

In [102]:

```
# Pass this series or DF in titanic_train
titanic_train[np.isnan(titanic_train[['Age']])]
# it turns out Passing a DF does not work
#WHY???? Remember passing series works
```

Out[102]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
12	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
19	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
21	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
22	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
23	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
24	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
26	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
27	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
28	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
861	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
862	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
863	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
864	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
865	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
866	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
867	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
868	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
869	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
870	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
871	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
872	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
873	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
874	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
875	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
876	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
877	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
878	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
879	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
880	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
881	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
882	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
883	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
884	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
885	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
886	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
887	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
888	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
889	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
890	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

891 rows × 15 columns

In [103]:

```
np.isnan(titanic_train['Age'])
```

Out[103]:

```
0      False
1      False
2      False
3      False
4      False
5       True
6      False
7      False
8      False
9      False
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17      True
18     False
19      True
20     False
21     False
22     False
23     False
24     False
25     False
26      True
27     False
28      True
29      True
       ...
861    False
862    False
863     True
864    False
865    False
866    False
867    False
868     True
869    False
870    False
871    False
872    False
873    False
874    False
875    False
876    False
877    False
878     True
879    False
880    False
881    False
882    False
883    False
884    False
885    False
886    False
887    False
888     True
```

```
889    False
890    False
Name: Age, Length: 891, dtype: bool
```

In [90]:

```
missing_age = titanic_train[np.isnan(titanic_train['Age'])][['Age', 'Age_with_mean', 'A
ge_with_mean_per_sex', 'Age_with_mean_per_sex_perPclass' ]]
```

In [91]:

```
# This dataframe gives missing age values based on different methods calculate
missing_age
```

Out[91]:

	Age	Age_with_mean	Age_with_mean_per_sex	Age_with_mean_per_sex_perPclass
5	NaN	28.0	29.0	24.0
17	NaN	28.0	29.0	30.0
19	NaN	28.0	27.0	22.0
26	NaN	28.0	29.0	24.0
28	NaN	28.0	27.0	22.0
29	NaN	28.0	29.0	24.0
31	NaN	28.0	27.0	35.0
32	NaN	28.0	27.0	22.0
36	NaN	28.0	29.0	24.0
42	NaN	28.0	29.0	24.0
45	NaN	28.0	29.0	24.0
46	NaN	28.0	29.0	24.0
47	NaN	28.0	27.0	22.0
48	NaN	28.0	29.0	24.0
55	NaN	28.0	29.0	40.0
64	NaN	28.0	29.0	40.0
65	NaN	28.0	29.0	24.0
76	NaN	28.0	29.0	24.0
77	NaN	28.0	29.0	24.0
82	NaN	28.0	27.0	22.0
87	NaN	28.0	29.0	24.0
95	NaN	28.0	29.0	24.0
101	NaN	28.0	29.0	24.0
107	NaN	28.0	29.0	24.0
109	NaN	28.0	27.0	22.0
121	NaN	28.0	29.0	24.0
126	NaN	28.0	29.0	24.0
128	NaN	28.0	27.0	22.0
140	NaN	28.0	27.0	22.0
154	NaN	28.0	29.0	24.0
...	...	...	...	...
718	NaN	28.0	29.0	24.0
727	NaN	28.0	27.0	22.0
732	NaN	28.0	29.0	30.0
738	NaN	28.0	29.0	24.0
739	NaN	28.0	29.0	24.0
740	NaN	28.0	29.0	40.0

	Age	Age_with_mean	Age_with_mean_per_sex	Age_with_mean_per_sex_perPclass
760	NaN	28.0	29.0	24.0
766	NaN	28.0	29.0	40.0
768	NaN	28.0	29.0	24.0
773	NaN	28.0	29.0	24.0
776	NaN	28.0	29.0	24.0
778	NaN	28.0	29.0	24.0
783	NaN	28.0	29.0	24.0
790	NaN	28.0	29.0	24.0
792	NaN	28.0	27.0	22.0
793	NaN	28.0	29.0	40.0
815	NaN	28.0	29.0	40.0
825	NaN	28.0	29.0	24.0
826	NaN	28.0	29.0	24.0
828	NaN	28.0	29.0	24.0
832	NaN	28.0	29.0	24.0
837	NaN	28.0	29.0	24.0
839	NaN	28.0	29.0	40.0
846	NaN	28.0	29.0	24.0
849	NaN	28.0	27.0	35.0
859	NaN	28.0	29.0	24.0
863	NaN	28.0	27.0	22.0
868	NaN	28.0	29.0	24.0
878	NaN	28.0	29.0	24.0
888	NaN	28.0	27.0	22.0

177 rows × 4 columns

In [104]:

```
titanic_train.columns
```

Out[104]:

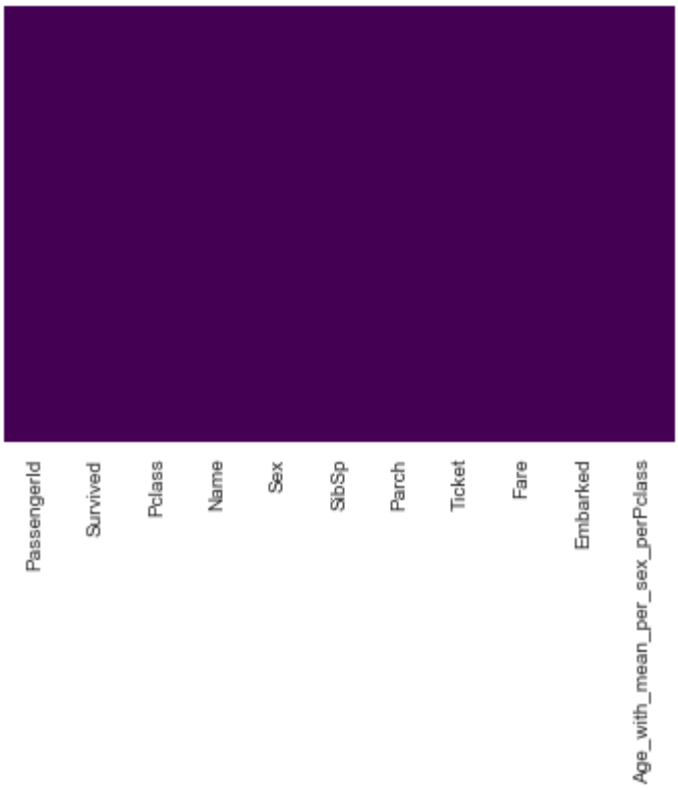
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked', 'Age_with_mean',
       'Age_with_mean_per_sex', 'Age_with_mean_per_sex_perPclass'],
      dtype='object')
```

In [112]:

```
titanic_train1 = titanic_train[['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Embarked', 'Age_with_mean_per_sex_perPclass']]
titanic_train1
sns.heatmap(data=titanic_train1.isnull(), yticklabels=False, cbar=False, cmap='viridis')
# The map below is like a bird-eye-view to find if there is any null values
```

Out[112]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29e65052358>
```



In [113]:

```
titanic_train1.info()
# There are four columns below with datatype object(or string) that the machine learnin
g algoritham cannot interpret
# Thses are
# 1. Name: This colunm can be dropped
# 2. Sex: This is a categorical column and can be converted to dummy variable column
# 3. Ticket: This column can be dropped
# 4. Embarked: This is a categorical column and can be converted to dummy variable colu
mn
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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
SibSp                               891 non-null int64
Parch                               891 non-null int64
Ticket                              891 non-null object
Fare                                891 non-null float64
Embarked                            889 non-null object
Age_with_mean_per_sex_perPclass     891 non-null float64
dtypes: float64(2), int64(5), object(4)
memory usage: 76.6+ KB
```

In [117]:

```
sex = pd.get_dummies(titanic_train1['Sex'])
embarked = pd.get_dummies(titanic_train1['Embarked'])
print(sex)
print(embarked)
```

	female	male
0	0	1
1	1	0
2	1	0
3	1	0
4	0	1
5	0	1
6	0	1
7	0	1
8	1	0
9	1	0
10	1	0
11	1	0
12	0	1
13	0	1
14	1	0
15	1	0
16	0	1
17	0	1
18	1	0
19	1	0
20	0	1
21	0	1
22	1	0
23	0	1
24	1	0
25	1	0
26	0	1
27	0	1
28	1	0
29	0	1
..	...	...
861	0	1
862	1	0
863	1	0
864	0	1
865	1	0
866	1	0
867	0	1
868	0	1
869	0	1
870	0	1
871	1	0
872	0	1
873	0	1
874	1	0
875	1	0
876	0	1
877	0	1
878	0	1
879	1	0
880	1	0
881	0	1
882	1	0
883	0	1
884	0	1
885	1	0
886	0	1
887	1	0
888	1	0
889	0	1

```
890       0     1

[891 rows x 2 columns]
     C  Q  S
0    0  0  1
1    1  0  0
2    0  0  1
3    0  0  1
4    0  0  1
5    0  1  0
6    0  0  1
7    0  0  1
8    0  0  1
9    1  0  0
10   0  0  1
11   0  0  1
12   0  0  1
13   0  0  1
14   0  0  1
15   0  0  1
16   0  1  0
17   0  0  1
18   0  0  1
19   1  0  0
20   0  0  1
21   0  0  1
22   0  1  0
23   0  0  1
24   0  0  1
25   0  0  1
26   1  0  0
27   0  0  1
28   0  1  0
29   0  0  1
..  .. .. ..
861  0  0  1
862  0  0  1
863  0  0  1
864  0  0  1
865  0  0  1
866  1  0  0
867  0  0  1
868  0  0  1
869  0  0  1
870  0  0  1
871  0  0  1
872  0  0  1
873  0  0  1
874  1  0  0
875  1  0  0
876  0  0  1
877  0  0  1
878  0  0  1
879  1  0  0
880  0  0  1
881  0  0  1
882  0  0  1
883  0  0  1
884  0  0  1
885  0  1  0
886  0  0  1
```

```
887  0  0  1
888  0  0  1
889  1  0  0
890  0  1  0

[891 rows x 3 columns]
```

In [125]:

```
sex = pd.get_dummies(titanic_train1['Sex'],drop_first=True)
embarked = pd.get_dummies(titanic_train1['Embarked'],drop_first=True)
print(sex)
print(embarked)
```

	male
0	1
1	0
2	0
3	0
4	1
5	1
6	1
7	1
8	0
9	0
10	0
11	0
12	1
13	1
14	0
15	0
16	1
17	1
18	0
19	0
20	1
21	1
22	0
23	1
24	0
25	0
26	1
27	1
28	0
29	1
..	...
861	1
862	0
863	0
864	1
865	0
866	0
867	1
868	1
869	1
870	1
871	0
872	1
873	1
874	0
875	0
876	1
877	1
878	1
879	0
880	0
881	1
882	0
883	1
884	1
885	0
886	1
887	0
888	0
889	1

```
890    1

[891 rows x 1 columns]
     Q  S
0    0  1
1    0  0
2    0  1
3    0  1
4    0  1
5    1  0
6    0  1
7    0  1
8    0  1
9    0  0
10   0  1
11   0  1
12   0  1
13   0  1
14   0  1
15   0  1
16   1  0
17   0  1
18   0  1
19   0  0
20   0  1
21   0  1
22   1  0
23   0  1
24   0  1
25   0  1
26   0  0
27   0  1
28   1  0
29   0  1
..  .. ..
861  0  1
862  0  1
863  0  1
864  0  1
865  0  1
866  0  0
867  0  1
868  0  1
869  0  1
870  0  1
871  0  1
872  0  1
873  0  1
874  0  0
875  0  0
876  0  1
877  0  1
878  0  1
879  0  0
880  0  1
881  0  1
882  0  1
883  0  1
884  0  1
885  1  0
886  0  1
```

```
887  0  1
888  0  1
889  0  0
890  1  0

[891 rows x 2 columns]
```

In [135]:

```
sex_embarked = pd.concat([sex, embarked], axis=1)
sex_embarked
```

Out[135]:

	male	Q	S
0	1	0	1
1	0	0	0
2	0	0	1
3	0	0	1
4	1	0	1
5	1	1	0
6	1	0	1
7	1	0	1
8	0	0	1
9	0	0	0
10	0	0	1
11	0	0	1
12	1	0	1
13	1	0	1
14	0	0	1
15	0	0	1
16	1	1	0
17	1	0	1
18	0	0	1
19	0	0	0
20	1	0	1
21	1	0	1
22	0	1	0
23	1	0	1
24	0	0	1
25	0	0	1
26	1	0	0
27	1	0	1
28	0	1	0
29	1	0	1
...	...	...	...
861	1	0	1
862	0	0	1
863	0	0	1
864	1	0	1
865	0	0	1
866	0	0	0

	male	Q	S
867	1	0	1
868	1	0	1
869	1	0	1
870	1	0	1
871	0	0	1
872	1	0	1
873	1	0	1
874	0	0	0
875	0	0	0
876	1	0	1
877	1	0	1
878	1	0	1
879	0	0	0
880	0	0	1
881	1	0	1
882	0	0	1
883	1	0	1
884	1	0	1
885	0	1	0
886	1	0	1
887	0	0	1
888	0	0	1
889	1	0	0
890	1	1	0

891 rows × 3 columns

In [119]:

```
titanic_train1.columns
```

Out[119]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'SibSp', 'Parc
h',
       'Ticket', 'Fare', 'Embarked', 'Age_with_mean_per_sex_perPclass'],
      dtype='object')
```

In [120]:

```
# we drop 'Name', 'Sex', 'Ticket', 'Embarked' columns or make a new DF without these co
ulumns
```

In [133]:

```
train = titanic_train1[['PassengerId', 'Survived', 'Pclass', 'SibSp', 'Parch', 'Fare',
'Age_with_mean_per_sex_perPclass']]
```

In [136]:

```
train.head()
```

Out[136]:

	PassengerId	Survived	Pclass	SibSp	Parch	Fare	Age_with_mean_per_sex_perPclass
0	1	0	3	1	0	7.2500	22.0
1	2	1	1	1	0	71.2833	38.0
2	3	1	3	0	0	7.9250	26.0
3	4	1	1	1	0	53.1000	35.0
4	5	0	3	0	0	8.0500	35.0

In [138]:

```
# Now concatenate dummy 'sex' and dummy 'embarked' DFs with train
train = pd.concat([train, sex_embarked],axis=1)
train
```

Out[138]:

	PassengerId	Survived	Pclass	SibSp	Parch	Fare	Age_with_mean_per_sex_perPcl
0	1	0	3	1	0	7.2500	2
1	2	1	1	1	0	71.2833	3
2	3	1	3	0	0	7.9250	2
3	4	1	1	1	0	53.1000	3
4	5	0	3	0	0	8.0500	3
5	6	0	3	0	0	8.4583	2
6	7	0	1	0	0	51.8625	5
7	8	0	3	3	1	21.0750	
8	9	1	3	0	2	11.1333	2
9	10	1	2	1	0	30.0708	1
10	11	1	3	1	1	16.7000	
11	12	1	1	0	0	26.5500	5
12	13	0	3	0	0	8.0500	2
13	14	0	3	1	5	31.2750	3
14	15	0	3	0	0	7.8542	1
15	16	1	2	0	0	16.0000	5
16	17	0	3	4	1	29.1250	
17	18	1	2	0	0	13.0000	3
18	19	0	3	1	0	18.0000	3
19	20	1	3	0	0	7.2250	2
20	21	0	2	0	0	26.0000	3
21	22	1	2	0	0	13.0000	3
22	23	1	3	0	0	8.0292	1
23	24	1	1	0	0	35.5000	2
24	25	0	3	3	1	21.0750	
25	26	1	3	1	5	31.3875	3
26	27	0	3	0	0	7.2250	2
27	28	0	1	3	2	263.0000	1
28	29	1	3	0	0	7.8792	2
29	30	0	3	0	0	7.8958	2
...	...	...	...	...	...	...	
861	862	0	2	1	0	11.5000	2
862	863	1	1	0	0	25.9292	4
863	864	0	3	8	2	69.5500	2
864	865	0	2	0	0	13.0000	2
865	866	1	2	0	0	13.0000	4
866	867	1	2	1	0	13.8583	2

	PassengerId	Survived	Pclass	SibSp	Parch	Fare	Age_with_mean_per_sex_perPcl
867	868	0	1	0	0	50.4958	3
868	869	0	3	0	0	9.5000	2
869	870	1	3	1	1	11.1333	
870	871	0	3	0	0	7.8958	2
871	872	1	1	1	1	52.5542	4
872	873	0	1	0	0	5.0000	3
873	874	0	3	0	0	9.0000	4
874	875	1	2	1	0	24.0000	2
875	876	1	3	0	0	7.2250	1
876	877	0	3	0	0	9.8458	2
877	878	0	3	0	0	7.8958	1
878	879	0	3	0	0	7.8958	2
879	880	1	1	0	1	83.1583	5
880	881	1	2	0	1	26.0000	2
881	882	0	3	0	0	7.8958	3
882	883	0	3	0	0	10.5167	2
883	884	0	2	0	0	10.5000	2
884	885	0	3	0	0	7.0500	2
885	886	0	3	0	5	29.1250	3
886	887	0	2	0	0	13.0000	2
887	888	1	1	0	0	30.0000	1
888	889	0	3	1	2	23.4500	2
889	890	1	1	0	0	30.0000	2
890	891	0	3	0	0	7.7500	3

891 rows × 10 columns

Building a logistic regression model

Train Test Split

In [139]:

```
from sklearn.model_selection import train_test_split
```

Out[139]:

```
<function sklearn.model_selection._split.train_test_split(*arrays, **optio
ns)>
```

In [140]:

```
X_train, X_test, y_train, y_test = train_test_split(train.drop('Survived', axis=1),
                                                    train['Survived'], test_size=0.3,
                                                    random_state=101)
```

Training and Predicting

In [141]:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

In [142]:

```
lr.fit(X_train, y_train)
```

```
C:\Users\vtaor\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.p
y:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. S
pecify a solver to silence this warning.
  FutureWarning)
```

Out[142]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=
0,
                   warm_start=False)
```

In [143]:

```
predictions = lr.predict(X_test)
```

Evaluation

In [144]:

```
# We can check precision, recall and f1-score using classification report
from sklearn.metrics import classification_report
```

In [145]:

```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.77      0.88      0.82       154
           1       0.79      0.64      0.71       114

    accuracy                           0.78       268
   macro avg       0.78      0.76      0.76       268
weighted avg       0.78      0.78      0.77       268
```