In [11]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
df = pd.read_csv("C:/Users/ABHISHEK/Desktop/titanic_train.csv")
df.head()
```

Out[11]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
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 [12]:

```
df.isnull().sum()
```

Out[12]:

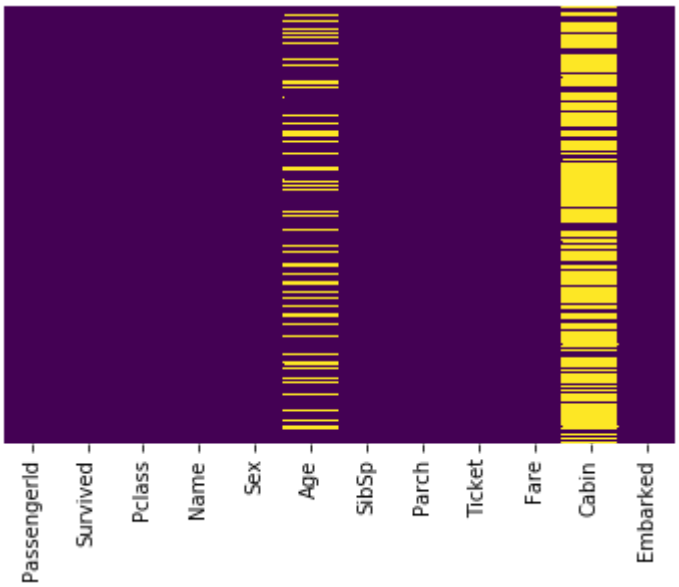
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [13]:

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

Out[13]:

<matplotlib.axes._subplots.AxesSubplot at 0x1f6797bb5f8>

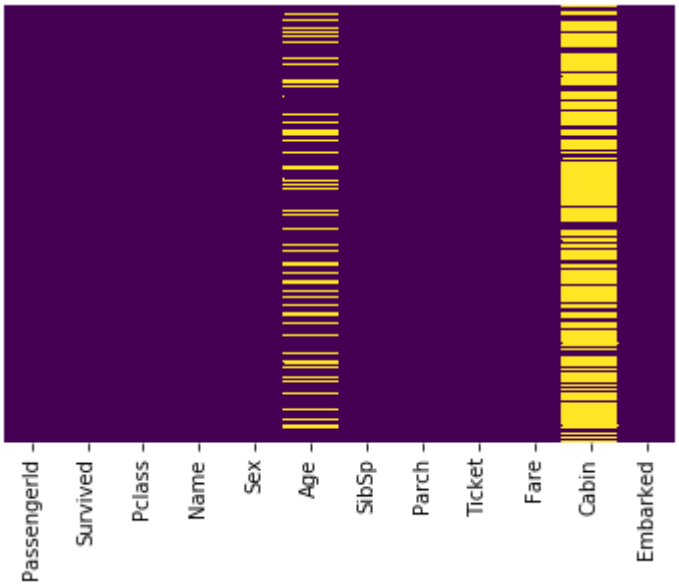


In [14]:

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

Out[14]:

<matplotlib.axes._subplots.AxesSubplot at 0x1f6798103c8>



In [15]:

```
# mean age
print('The mean of "Age" is %.2f' %(df["Age"].mean(skipna=True)))
# median age
print('The median of "Age" is %.2f' %(df["Age"].median(skipna=True)))
```

The mean of "Age" is 29.70
The median of "Age" is 28.00

In [16]:

```
# percent of missing "Cabin"
print('Percent of missing "Cabin" records is %.2f%%' %((df['Cabin'].isnull().sum()/df.shape[0])*100))
```

Percent of missing "Cabin" records is 77.10%

In [17]:

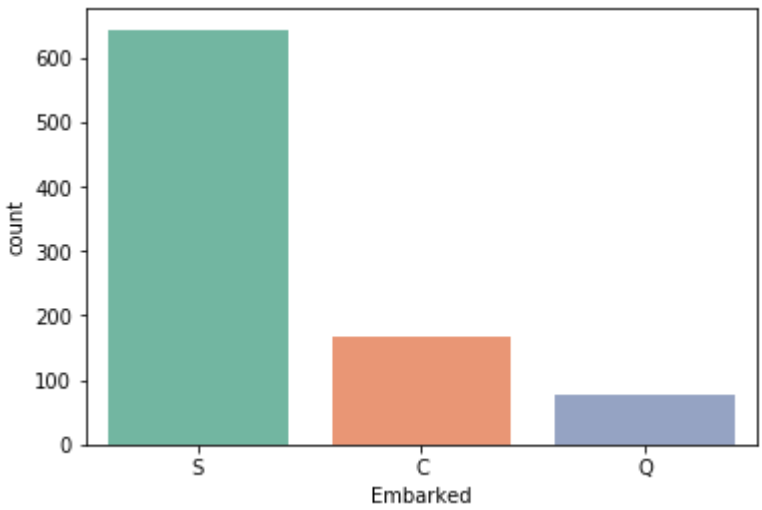
```
print('Percent of missing "Embarked" records is %.2f%%' %((df['Embarked'].isnull().sum()/df.shape[0])*10
```

Percent of missing "Embarked" records is 0.22%

In [18]:

```
print('Boarded passengers grouped by port of embarkation (C = Cherbourg, Q = Queenstown, S = Southamp
print(df['Embarked'].value_counts())
sns.countplot(x='Embarked', data=df, palette='Set2')
plt.show()
```

Boarded passengers grouped by port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton):
S 644
C 168
Q 77
Name: Embarked, dtype: int64



In [25]:

```
df["Age"].fillna(df["Age"].median(skipna=True), inplace=True)
df["Embarked"].fillna(df['Embarked'].value_counts().idxmax(), inplace=True)
df.head()
```

Out[25]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	I
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 [26]:

```
df.isnull().sum()
```

Out[26]:

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

In [27]:

```
sex = pd.get_dummies(df['Sex'],drop_first=True)
embark = pd.get_dummies(df['Embarked'],drop_first=True)
pclass = pd.get_dummies(df['Pclass'],drop_first=True)

df.drop(['PassengerId','Pclass','Sex','Embarked','Name','Ticket'], axis=1, inplace=True)

df.head()
```

Out[27]:

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

In [28]:

```
df = pd.concat([df, pclass, sex, embark],axis=1)
df.head()
```

Out[28]:

	Survived	Age	SibSp	Parch	Fare	2	3	male	Q	S
0	0	22.0	1	0	7.2500	0	1	1	0	1
1	1	38.0	1	0	71.2833	0	0	0	0	0
2	1	26.0	0	0	7.9250	0	1	0	0	1
3	1	35.0	1	0	53.1000	0	0	0	0	1
4	0	35.0	0	0	8.0500	0	1	1	0	1

In [39]:

```
x = df.drop("Survived",axis=1)
y = df["Survived"]
print(x)
```

```
     Age  SibSp  Parch     Fare  2  3  male  Q  S
0   22.0      1      0   7.2500  0  1     1  0  1
1   38.0      1      0  71.2833  0  0     0  0  0
2   26.0      0      0   7.9250  0  1     0  0  1
3   35.0      1      0  53.1000  0  0     0  0  1
4   35.0      0      0   8.0500  0  1     1  0  1
5   28.0      0      0   8.4583  0  1     1  1  0
6   54.0      0      0  51.8625  0  0     1  0  1
7    2.0      3      1  21.0750  0  1     1  0  1
8   27.0      0      2  11.1333  0  1     0  0  1
9   14.0      1      0  30.0708  1  0     0  0  0
10   4.0      1      1  16.7000  0  1     0  0  1
11  58.0      0      0  26.5500  0  0     0  0  1
12  20.0      0      0   8.0500  0  1     1  0  1
13  39.0      1      5  31.2750  0  1     1  0  1
14  14.0      0      0   7.8542  0  1     0  0  1
15  55.0      0      0  16.0000  1  0     0  0  1
16   2.0      4      1  29.1250  0  1     1  1  0
17  28.0      0      0  13.0000  1  0     1  0  1
18  31.0      1      0  18.0000  0  1     0  0  1
19  28.0      0      0   7.2250  0  1     0  0  0
20  35.0      0      0  26.0000  1  0     1  0  1
21  34.0      0      0  13.0000  1  0     1  0  1
22  15.0      0      0   8.0292  0  1     0  1  0
23  28.0      0      0  35.5000  0  0     1  0  1
24   8.0      3      1  21.0750  0  1     0  0  1
25  38.0      1      5  31.3875  0  1     0  0  1
26  28.0      0      0   7.2250  0  1     1  0  0
27  19.0      3      2  263.0000 0  0     1  0  1
28  28.0      0      0   7.8792  0  1     0  1  0
29  28.0      0      0   7.8958  0  1     1  0  1
..   ...    ...    ...      ... .. ..   ... .. ..
861 21.0      1      0  11.5000  1  0     1  0  1
862 48.0      0      0  25.9292  0  0     0  0  1
863 28.0      8      2  69.5500  0  1     0  0  1
864 24.0      0      0  13.0000  1  0     1  0  1
865 42.0      0      0  13.0000  1  0     0  0  1
866 27.0      1      0  13.8583  1  0     0  0  0
867 31.0      0      0  50.4958  0  0     1  0  1
868 28.0      0      0   9.5000  0  1     1  0  1
869  4.0      1      1  11.1333  0  1     1  0  1
870 26.0      0      0   7.8958  0  1     1  0  1
871 47.0      1      1  52.5542  0  0     0  0  1
872 33.0      0      0   5.0000  0  0     1  0  1
873 47.0      0      0   9.0000  0  1     1  0  1
874 28.0      1      0  24.0000  1  0     0  0  0
875 15.0      0      0   7.2250  0  1     0  0  0
876 20.0      0      0   9.8458  0  1     1  0  1
877 19.0      0      0   7.8958  0  1     1  0  1
878 28.0      0      0   7.8958  0  1     1  0  1
879 56.0      0      1  83.1583  0  0     0  0  0
880 25.0      0      1  26.0000  1  0     0  0  1
881 33.0      0      0   7.8958  0  1     1  0  1
882 22.0      0      0  10.5167  0  1     0  0  1
```

```
     883  28.0      0      0   10.5000  1  0    1  0  1
     884  25.0      0      0    7.0500  0  1    1  0  1
     885  39.0      0      5   29.1250  0  1    0  1  0
     886  27.0      0      0   13.0000  1  0    1  0  1
     887  19.0      0      0   30.0000  0  0    0  0  1
     888  28.0      1      2   23.4500  0  1    0  0  1
     889  26.0      0      0   30.0000  0  0    1  0  0
     890  32.0      0      0    7.7500  0  1    1  1  0

     [891 rows x 9 columns]
```

In [40]:

```
print(y)
```

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

```
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

In [32]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 40)

from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(x_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: Futu
reWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence thi
s warning.
  FutureWarning)
```

Out[32]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, 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 [42]:

```
predictions = logmodel.predict(x_test)
```

In [44]:

```
from sklearn.metrics import confusion_matrix
accuracy=confusion_matrix(y_test, predictions)
accuracy
```

Out[44]:

```
array([[134,  22],
       [ 27,  85]], dtype=int64)
```

In [45]:

```
from sklearn.metrics import accuracy_score
accuracy=accuracy_score(y_test, predictions)
accuracy
```

Out[45]:

```
0.8171641791044776
```

In [46]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.85       156
           1       0.79      0.76      0.78       112

   micro avg       0.82      0.82      0.82       268
   macro avg       0.81      0.81      0.81       268
weighted avg       0.82      0.82      0.82       268
```

In [ ]:

```
```