Apply data preprocessing on the given dataset ,

then split the dataset into train and test and predict

the survival rate using Gaussian Naive Bayes.

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import  train_test_split
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
import seaborn as sns
from sklearn.metrics import accuracy_score
df=pd.read_csv('data.csv')
df.head()
```

Out[1]:

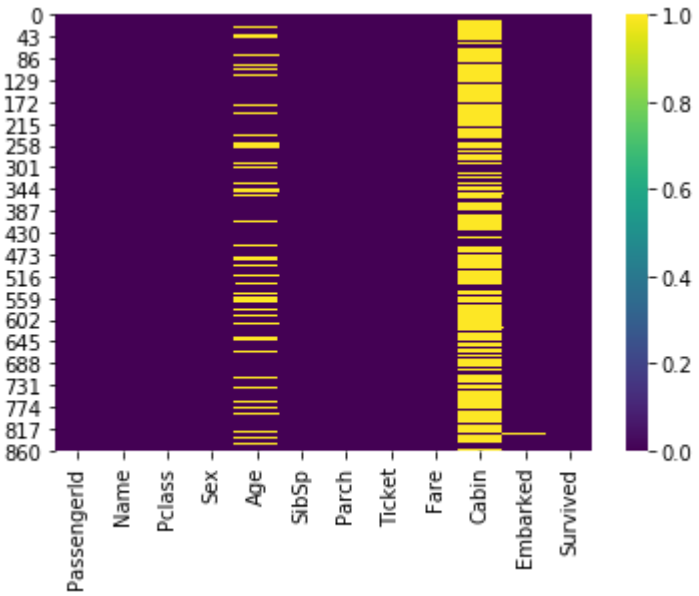
	PassengerId	Name	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embark
0	1	Braund, Mr. Owen Harris	3	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	Heikkinen, Miss. Laina	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN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

In [2]:
```
#Data preprocessing
#Let's check for null values
df.isnull().sum()
```

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

In [3]:
```
sns.heatmap(df.isnull(),cbar=True,cmap='viridis')
```

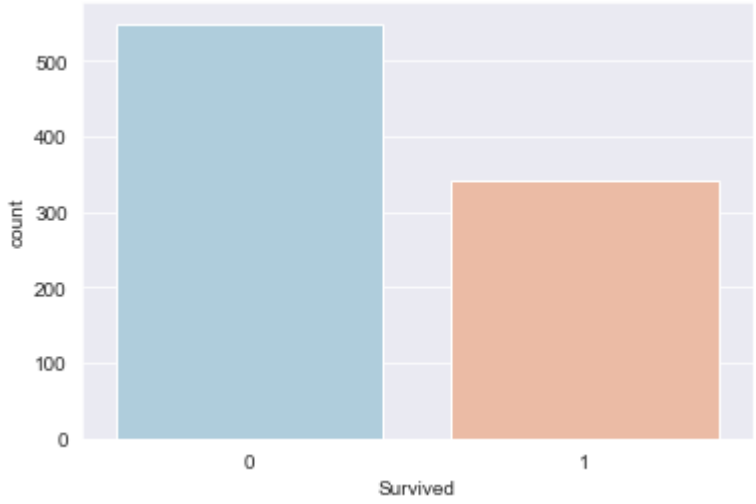
Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a6f5b4bc48>
```



In [4]:
```
#As there are total of 891 records and Cabin column has max
#null values so let's drop the entire cabin column
df.drop('Cabin',axis=1,inplace=True)
```

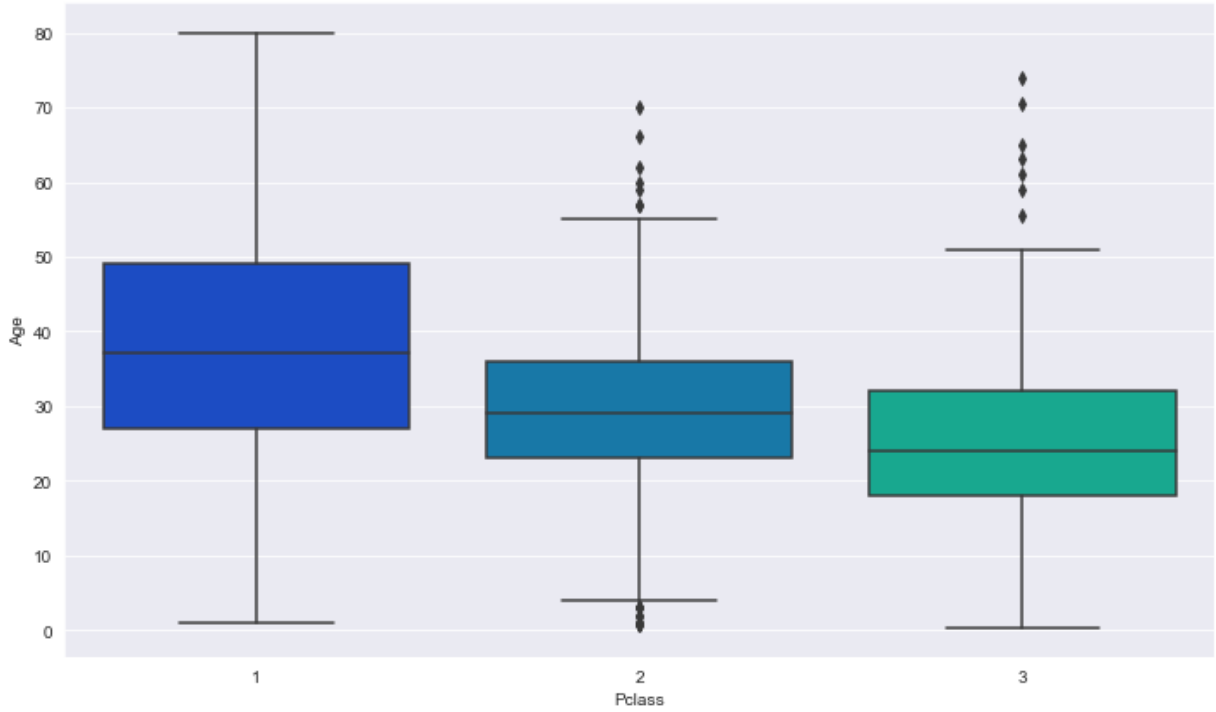
In [5]:
```
#Checking count of observation for each category i.e. 'Survived'
sns.set_style('darkgrid')
sns.countplot(x='Survived',data=df,palette='RdBu_r')
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x1a6f60c0488>



In [6]:
```
#Now let's fill the 'Age' column
#We can replace the missing values with the mean values of age w.r.t. to Pcl
plt.figure(figsize=(12,7))
sns.boxplot('Pclass','Age',data=df,palette='winter')
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x1a6f6113f88>

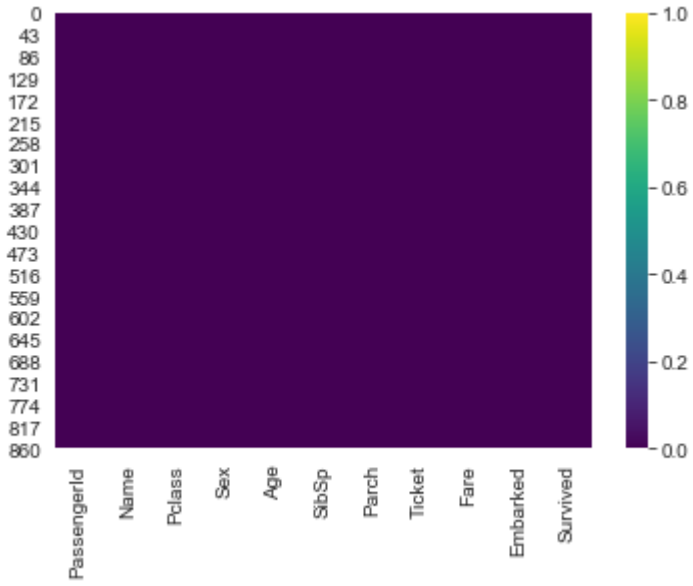


In [7]:
```
#Now lets create a function that could be used to fill the missing values in

def funfill(col):
    Age=col[0]
    Pclass=col[1]
    if pd.isnull(Age):
        if Pclass==1:
            return 37
        elif Pclass==2:
            return 29
        else:
            return 24
    else:
        return Age

df['Age'] = df[['Age','Pclass']].apply(funfill,axis=1)
#Let's check the heatmap and see if the null values of Age are eliminated
sns.heatmap(df.isnull(),cbar=True,cmap='viridis')
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1a6f61e84c8>



In [8]:
```
#Removing the 2 rows of embarked containing null values
df.dropna(inplace=True)
```

In [9]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 11 columns):
PassengerId    889 non-null int64
Name           889 non-null object
Pclass         889 non-null int64
Sex            889 non-null object
Age            889 non-null float64
SibSp          889 non-null int64
Parch          889 non-null int64
Ticket         889 non-null object
Fare           889 non-null float64
Embarked       889 non-null object
Survived       889 non-null int64
dtypes: float64(2), int64(5), object(4)
memory usage: 83.3+ KB
```

In [10]:
```
1 #We can see that Columns 'Name' has unique values let's remove it
2 len(df['Name'].unique()) # gives 889
3 #So let's remove 'Name' column
4 df.drop(columns=['Name'],inplace=True)
```

In [11]:
```
1 #As we can see we have some columns having categorical
2 #data let's apply Label Encoder on them
3 from sklearn import preprocessing
4 label_encoder = preprocessing.LabelEncoder()
5 df['Sex']= label_encoder.fit_transform(df['Sex'])
6 df['Embarked']= label_encoder.fit_transform(df['Embarked'])
7 df['Ticket']= label_encoder.fit_transform(df['Ticket'])
```

In [12]:
```
1 #Train Test Split
2 from sklearn.model_selection import train_test_split
3 x_train,x_test,y_train,y_test=train_test_split(df.drop('Survived',axis=1),df
```

In [13]:
```
1 x_test.shape
```

Out[13]:
```
(300, 9)
```

In [14]:
```
1 #Initialize Gaussian Naive Bayes
2 clf=GaussianNB()
3 clf.fit(x_train,y_train)
```

Out[14]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [15]:
```
#Predicting for test set
pred=clf.predict(x_test)
pred
```

Out[15]:
```
array([0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0], dtype=int64)
```

In [16]:
```
#Reshaping needed to perform the concatenation
pred1=pd.DataFrame(pred.reshape(300,1))

#Naming this column as prediction
pred1.rename(columns={0:'Prediction'},inplace=True)
```

In [17]:
```
#Reshaping test dataset
xtest_df = pd.DataFrame(x_test.values.reshape(300,9))
```

In [18]:
```
#Concatenating pred1 and xtest_df
pred_concat=pd.concat([xtest_df,pred1],axis=1,join_axes=[xtest_df.index])
```

```
C:\Users\Mehakdeep\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureW
arning: The join_axes-keyword is deprecated. Use .reindex or .reindex_like on t
he result to achieve the same functionality.
```

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

Out[19]:

	0	1	2	3	4	5	6	7	8	Prediction
0	170.0	3.0	1.0	28.0	0.0	0.0	79.0	56.4958	2.0	0
1	218.0	2.0	1.0	42.0	1.0	0.0	141.0	27.0000	2.0	0
2	61.0	3.0	1.0	22.0	0.0	0.0	196.0	7.2292	0.0	0
3	54.0	2.0	0.0	29.0	1.0	0.0	238.0	26.0000	2.0	1
4	454.0	1.0	1.0	49.0	1.0	0.0	83.0	89.1042	0.0	1

In [20]:
```
pred_concat.rename(columns={0:'PassengerId',1:'Pclass',2:'Sex',3:'Age',4:'Si
                            5:'Parch',6:'Ticket',7:'Fare',8:'Embarked'},inpla
pred_concat.head()
```

Out[20]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Prediction
0	170.0	3.0	1.0	28.0	0.0	0.0	79.0	56.4958	2.0	0
1	218.0	2.0	1.0	42.0	1.0	0.0	141.0	27.0000	2.0	0
2	61.0	3.0	1.0	22.0	0.0	0.0	196.0	7.2292	0.0	0
3	54.0	2.0	0.0	29.0	1.0	0.0	238.0	26.0000	2.0	1
4	454.0	1.0	1.0	49.0	1.0	0.0	83.0	89.1042	0.0	1

In [21]:
```
#merging the prediction with original dataset
pred_comp = pd.merge(df,pred_concat,on=['PassengerId','Pclass','Sex','Age','
pred_comp.head()
```

Out[21]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Survived	Prediction
0	3	3	0	26.0	0	0	668	7.9250	2	1	0
1	10	2	0	14.0	1	0	131	30.0708	0	1	1
2	14	3	1	39.0	1	5	332	31.2750	2	0	0
3	16	2	0	55.0	0	0	152	16.0000	2	1	1
4	23	3	0	15.0	0	0	277	8.0292	1	1	1

In [22]:
```
print('Accuracy score is:',accuracy_score(y_test,pred))
```

```
Accuracy score is: 0.78
```