titanic-classification

October 9, 2023

Project Title:

Titanic Survival Prediction System

Problem Statement:

The sinking of the RMS Titanic in 1912 was a tragic event that resulted in a significant loss of life. This project aims to develop a machine learning system that predicts whether a person aboard the Titanic would have survived the disaster based on various factors. The primary goal is to identify the most influential factors that led to survival, such as socio-economic status, age, gender, and more.

Description: The Titanic Survival Prediction System utilizes a dataset containing passenger information, including socio-economic class (Pclass), age, gender, family size, ticket fare, and more.

Conclusion:

Summarize the project's findings and key results. Highlight the factors that were most likely to lead to survival on the Titanic. Discuss the machine learning model's accuracy and its ability to predict survival based on passenger information. Emphasize the importance of understanding historical events like the Titanic disaster through data analysis and predictive modeling.

Libraries

[79]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
```

[81]:
```
train = pd.read_csv('/content/train.csv')
train.head()
train.tail()
```

[81]:
```
     PassengerId  Survived  Pclass                                      Name  \
886          887         0       2                     Montvila, Rev. Juozas
887          888         1       1              Graham, Miss. Margaret Edith
888          889         0       3  Johnston, Miss. Catherine Helen "Carrie"
889          890         1       1                     Behr, Mr. Karl Howell
890          891         0       3                       Dooley, Mr. Patrick
```

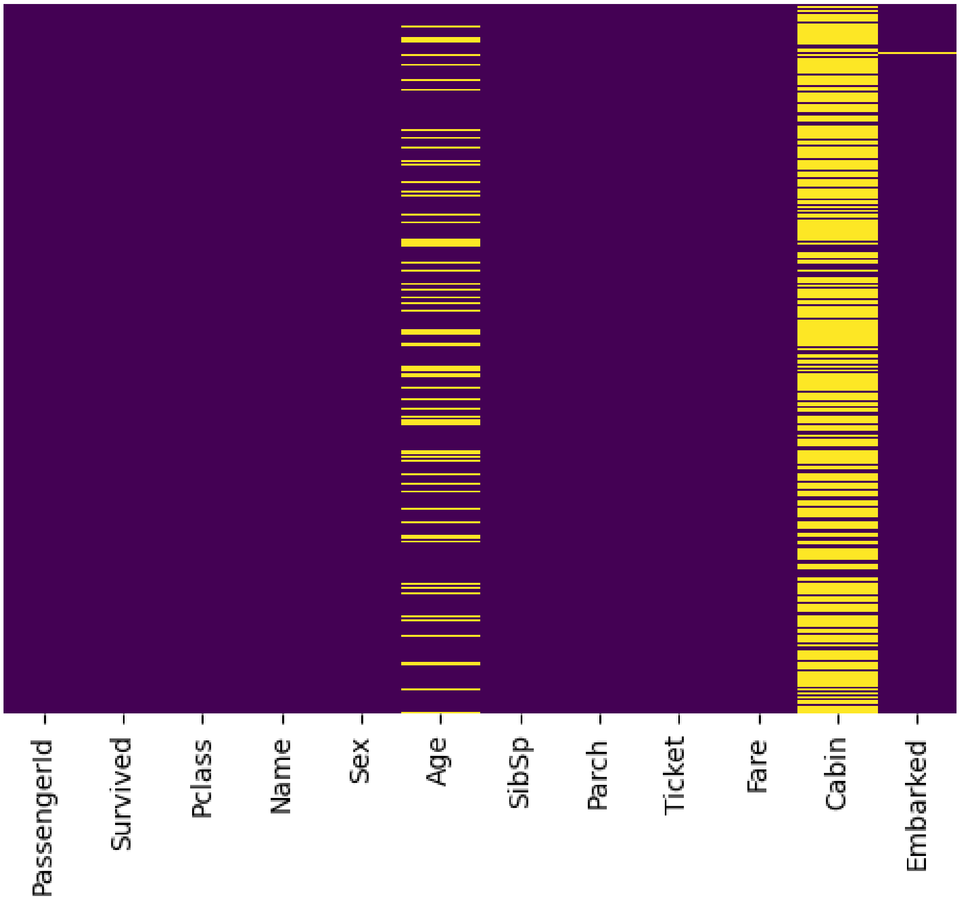
```
        Sex   Age  SibSp  Parch      Ticket   Fare Cabin Embarked
886    male  27.0      0      0      211536  13.00   NaN        S
887  female  19.0      0      0      112053  30.00   B42        S
888  female   NaN      1      2  W./C. 6607  23.45   NaN        S
889    male  26.0      0      0      111369  30.00  C148        C
890    male  32.0      0      0      370376   7.75   NaN        Q
```

Exploratory Data Analysis

Missing valus

[5]:
```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

[5]:
```
<Axes: >
```

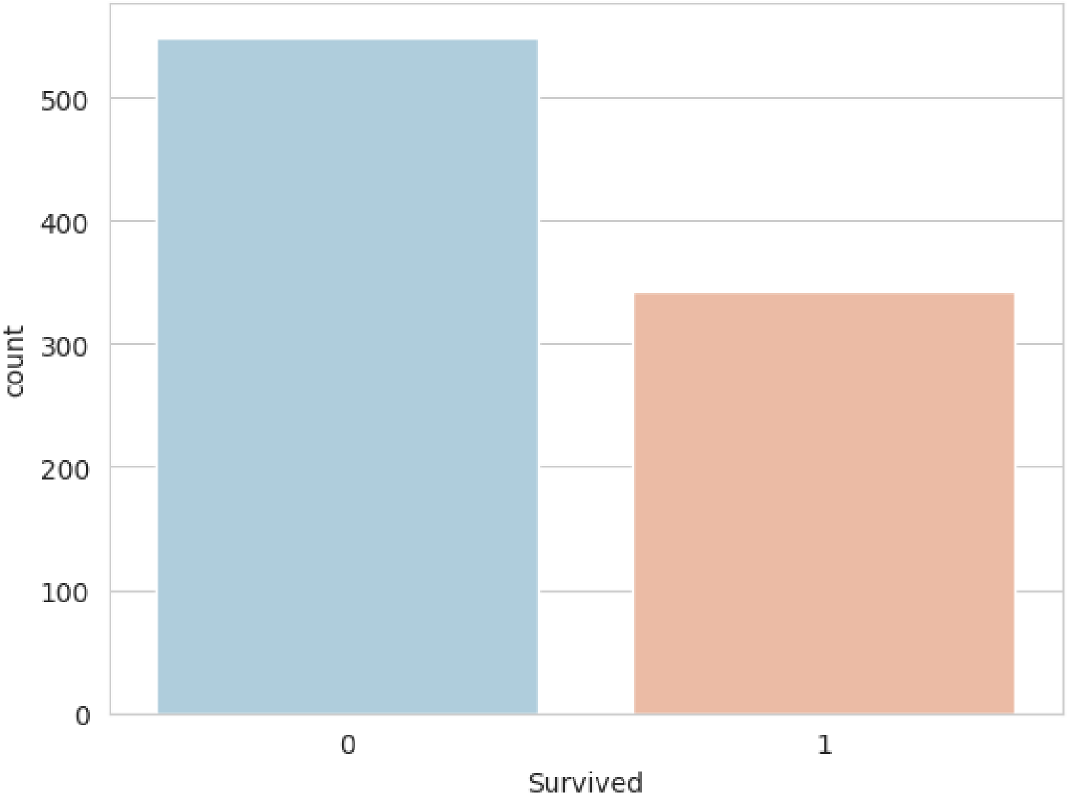


[6]:
```
train.isnull().sum().sort_values(ascending=False)
```

[6]:
```
Cabin          687
Age            177
Embarked         2
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
SibSp            0
Parch            0
Ticket           0
Fare             0
dtype: int64
```

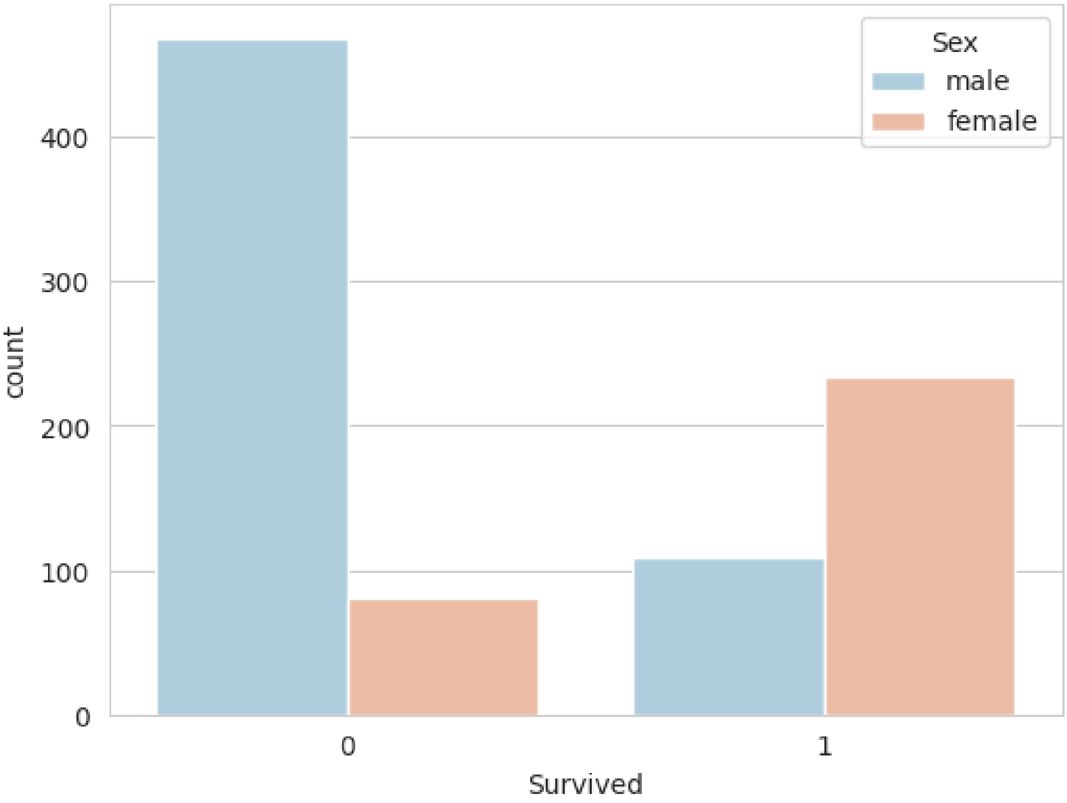
[7]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived',data=train,palette='RdBu_r')
```

[7]:
```
<Axes: xlabel='Survived', ylabel='count'>
```



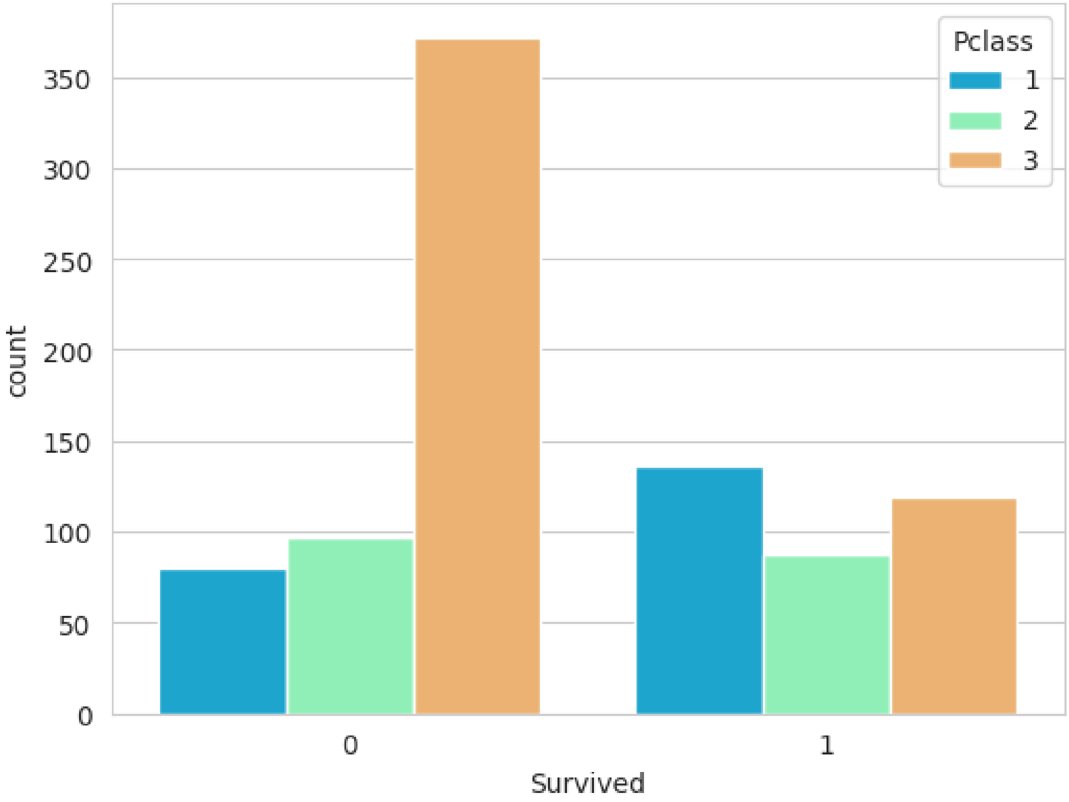
[8]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Sex',data=train,palette='RdBu_r')
```

```
[8]: <Axes: xlabel='Survived', ylabel='count'>
```



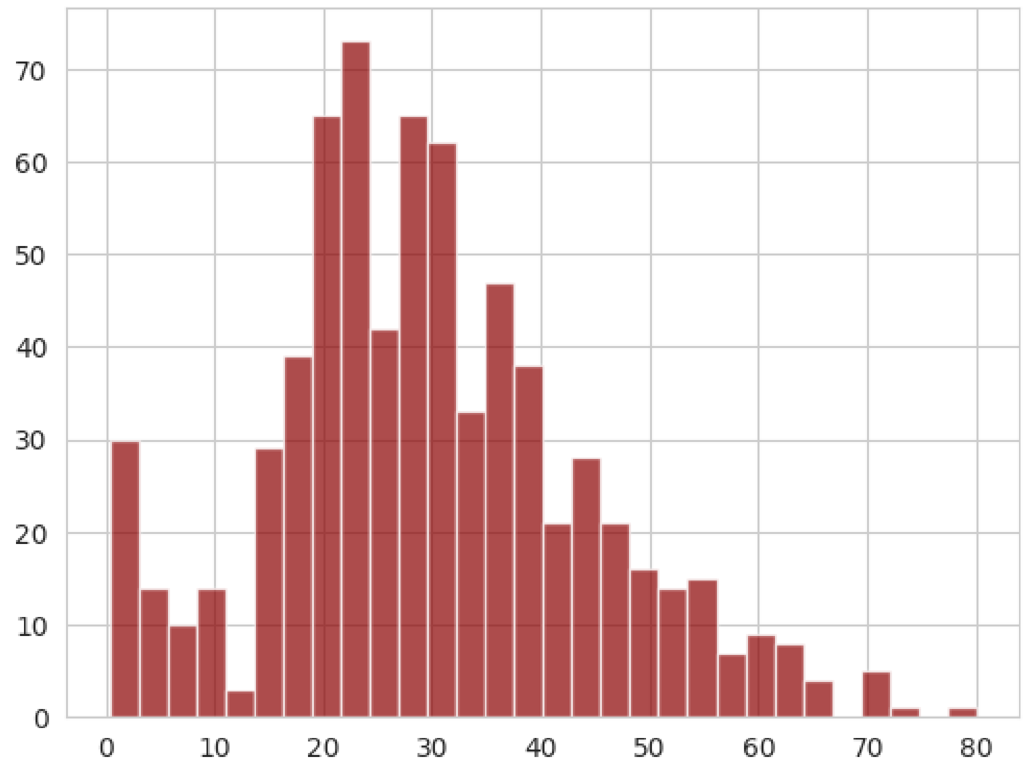
```
[9]: sns.set_style('whitegrid')
     sns.countplot(x='Survived',hue='Pclass',data=train,palette='rainbow')
```

```
[9]: <Axes: xlabel='Survived', ylabel='count'>
```



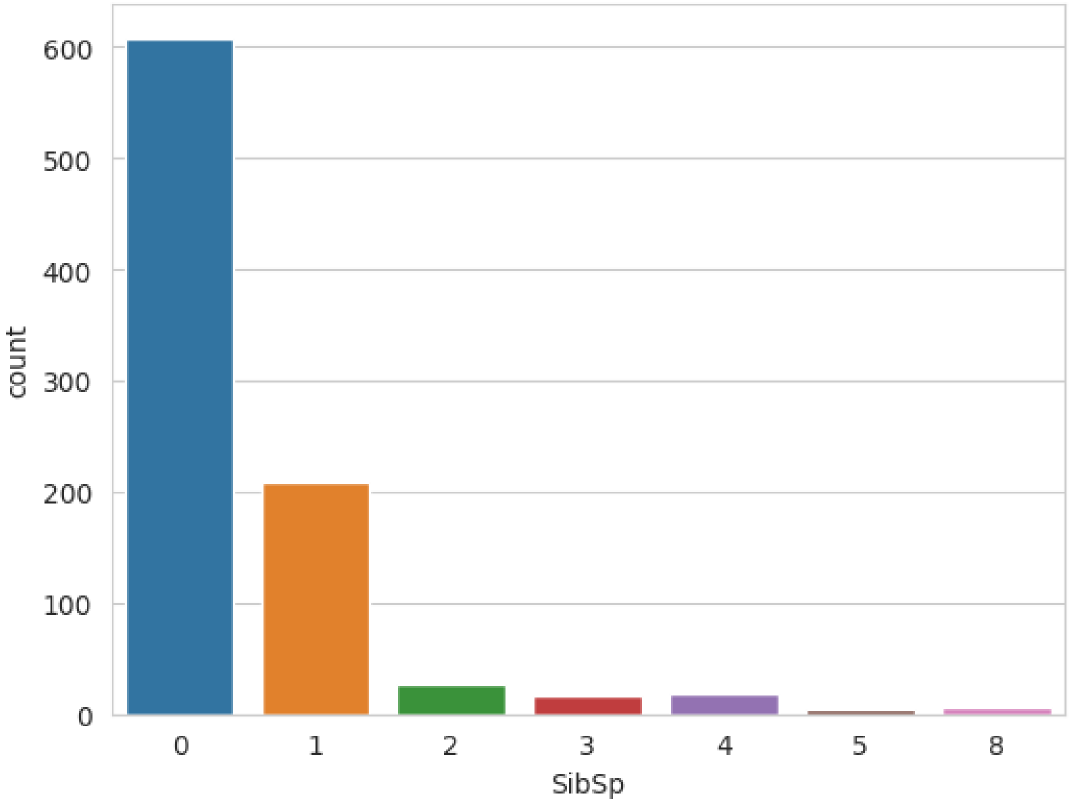
```
[10]: train['Age'].hist(bins=30,color='darkred',alpha=0.7)
```

```
[10]: <Axes: >
```



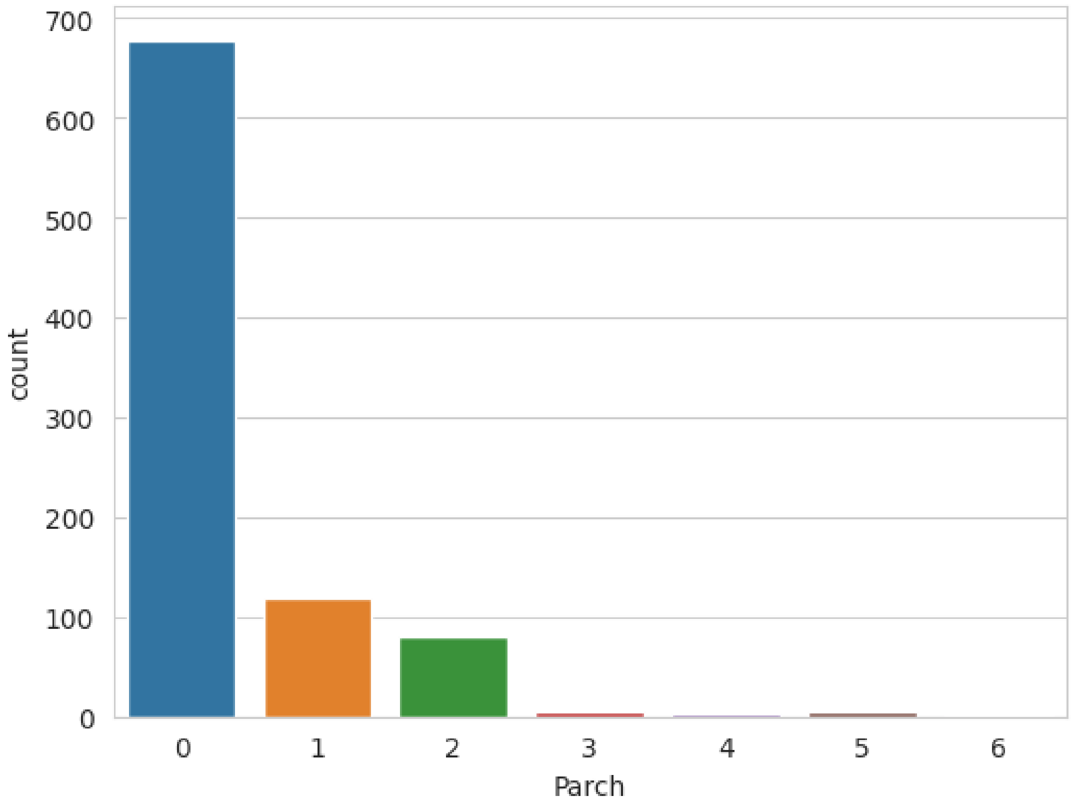
[11]:
```
sns.countplot(x='SibSp',data=train)
```

[11]:
```
<Axes: xlabel='SibSp', ylabel='count'>
```



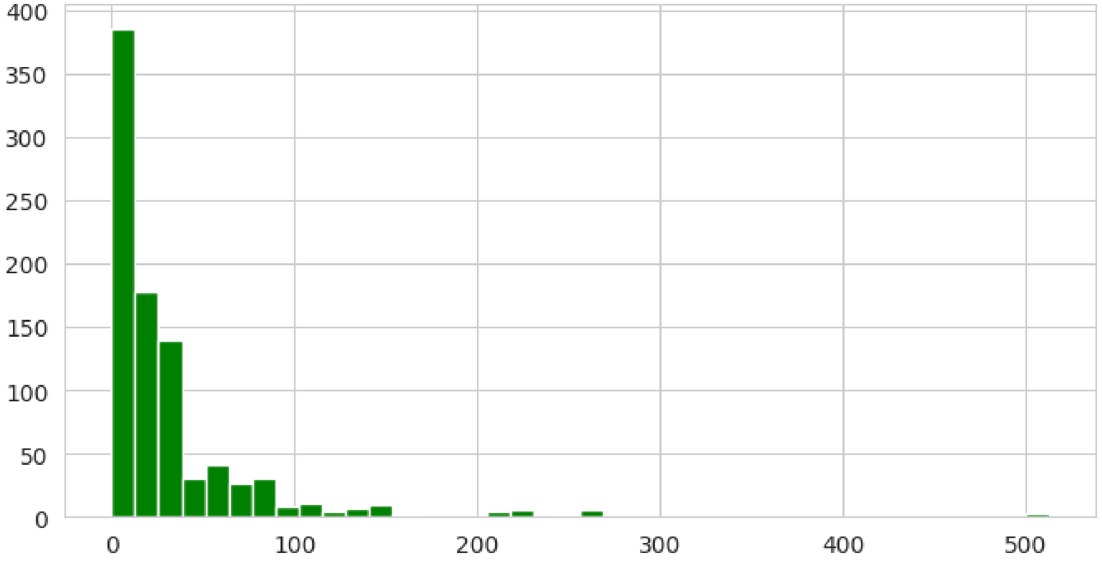
```
[12]: sns.countplot(x='Parch',data=train)
```

```
[12]: <Axes: xlabel='Parch', ylabel='count'>
```



```
[13]: train['Fare'].hist(color='green',bins=40,figsize=(8,4))
```

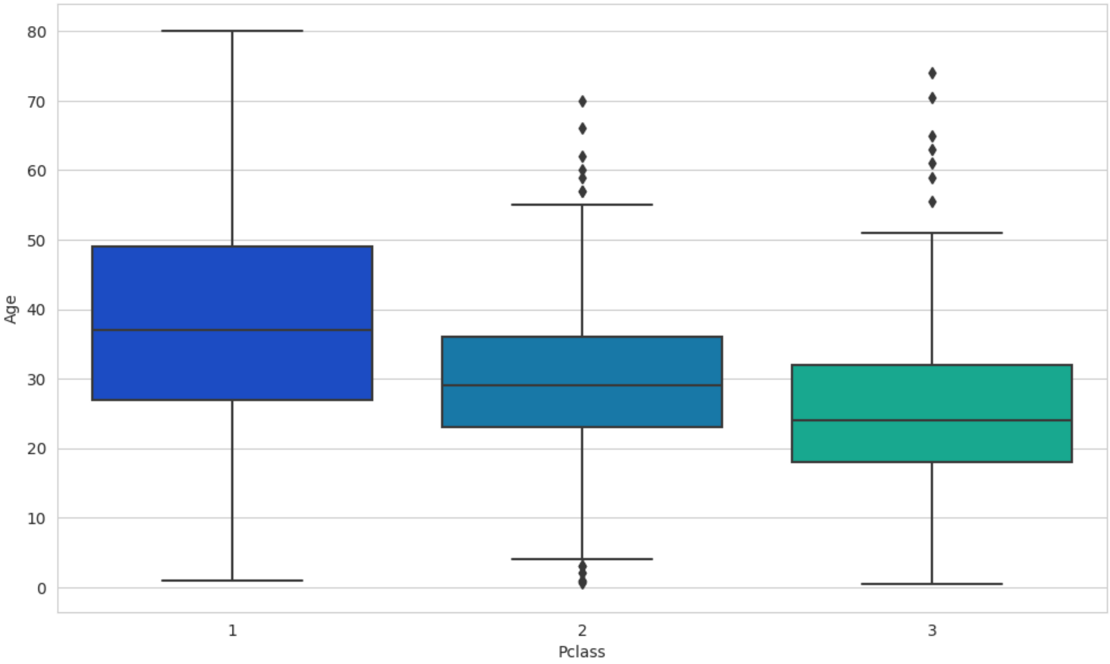
```
[13]: <Axes: >
```



Data Cleaning

```
[14]: plt.figure(figsize=(12, 7))
      sns.boxplot(x='Pclass',y='Age',data=train,palette='winter')
```

```
[14]: <Axes: xlabel='Pclass', ylabel='Age'>
```



```
[15]: def impute_age(cols):
          Age = cols[0]
          Pclass = cols[1]

          if pd.isnull(Age):

              if Pclass == 1:
                  return 37

              elif Pclass == 2:
                  return 29

              else:
                  return 24

          else:
              return Age
```

[16]:

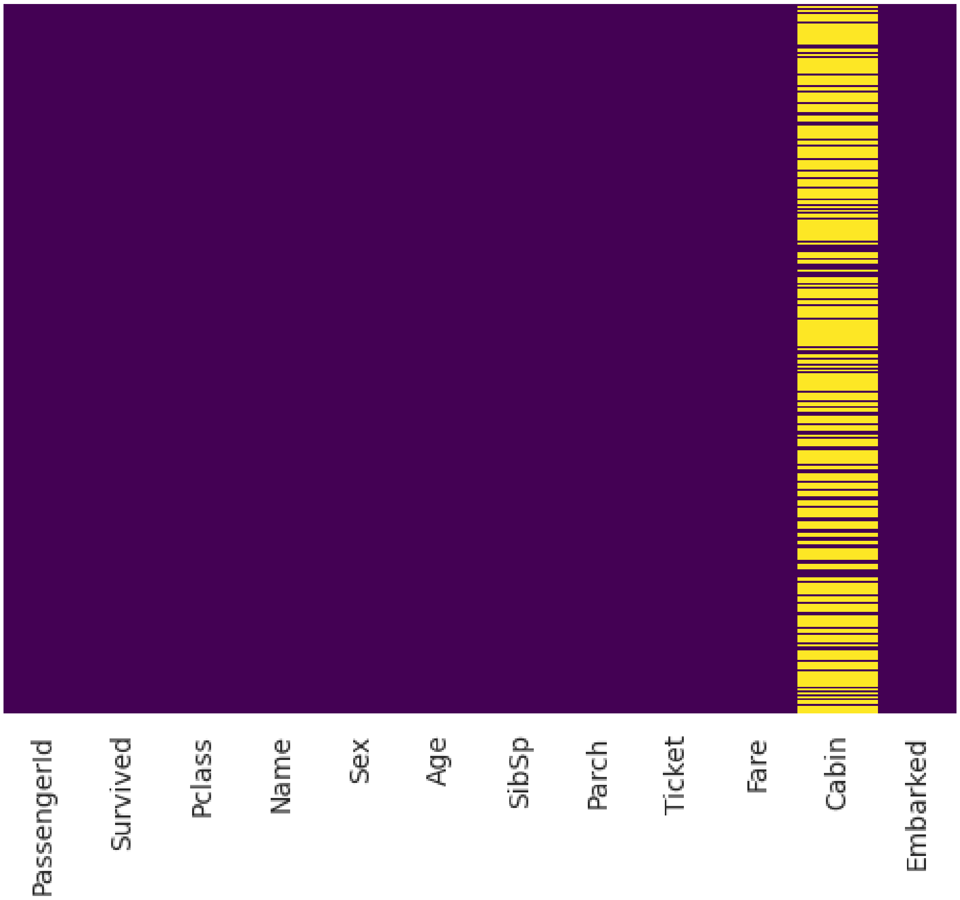
```
train['Age'] = train[['Age','Pclass']].apply(impute_age,axis=1)
train['Embarked'] = train['Embarked'].fillna('S')
```

[17]:

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

[17]:

```
<Axes: >
```



[18]:

```
train.drop('Cabin',axis=1,inplace=True)
train.head()
```

[18]:

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                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

   Parch            Ticket     Fare Embarked
0      0         A/5 21171   7.2500        S
1      0          PC 17599  71.2833        C
2      0  STON/O2. 3101282   7.9250        S
3      0            113803  53.1000        S
4      0            373450   8.0500        S
```

[19]:
```
train.dropna(inplace=True)
```

Converting Categorcal Figures

[20]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

[21]:
```
sex = pd.get_dummies(train['Sex'],drop_first=True)
embark = pd.get_dummies(train['Embarked'],drop_first=True)
train.drop(['Sex','Embarked','Name','Ticket'],axis=1,inplace=True)
train = pd.concat([train,sex,embark],axis=1)
train.head()
```

[21]:
```
   PassengerId  Survived  Pclass   Age  SibSp  Parch     Fare  male  Q  S
0            1         0       3  22.0      1      0   7.2500     1  0  1
1            2         1       1  38.0      1      0  71.2833     0  0  0
```

```
2            3          1         3  26.0        0        0   7.9250      0  0  1
3            4          1         1  35.0        1        0  53.1000      0  0  1
4            5          0         3  35.0        0        0   8.0500      1  0  1
```

Train Test Split

[22]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(train.
 ↪drop(['Survived'],axis=1),
                                                    train['Survived'],␣
 ↪test_size=0.10,
                                                    random_state=101)
```

Training and Predicting

[23]:
```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[23]:
```
LogisticRegression()
```

[24]:
```
predictions = logmodel.predict(X_test)
X_test.head()
```

[24]:
```
     PassengerId  Pclass   Age  SibSp  Parch     Fare  male  Q  S
331          332       1  45.5      0      0   28.500     1  0  1
700          701       1  18.0      1      0  227.525     0  0  0
748          749       1  19.0      1      0   53.100     1  0  1
751          752       3   6.0      0      1   12.475     1  0  1
481          482       2  29.0      0      0    0.000     1  0  1
```

[26]:
```
predictions
```

[26]:
```
array([0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
           0, 1])
```

[82]:
```
X_train_prediction = logmodel.predict(X_train)
X_train_prediction
```

[82]:
```
array([0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1,
       1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 1, 0])
```

[83]:
```
training_data_accuracy = accuracy_score(y_train, X_train_prediction)
print('Accuracy score of training data : ', training_data_accuracy)
```

```
Accuracy score of training data :  0.8002496878901373
```

```
[84]: X_test_prediction = logmodel.predict(X_test)
      X_test_prediction
```

```
[84]: array([0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
             1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
             0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
             1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
             0, 1])
```

```
[85]: test_data_accuracy = accuracy_score(y_test, X_test_prediction)
      print('Accuracy score of test data : ', test_data_accuracy)
```

```
Accuracy score of test data :  0.7333333333333333
```