Titanic Logistic Regression basic

March 30, 2021

Logistic Regression on Titanic dataset

Sohini Mukherjee

30.03.2021

Importing Libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[2]:
```
train =pd.read_csv('E:/2.PYTHON-ML-BOOTCAMP/resources/13-Logistic-Regression/
 ↪titanic_train.csv')
```

[3]:
```
train.head()
```

[3]:
```
   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 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
```

Exploratory Data Analysis

Checking missing data

```
[4]: train.isnull()
```

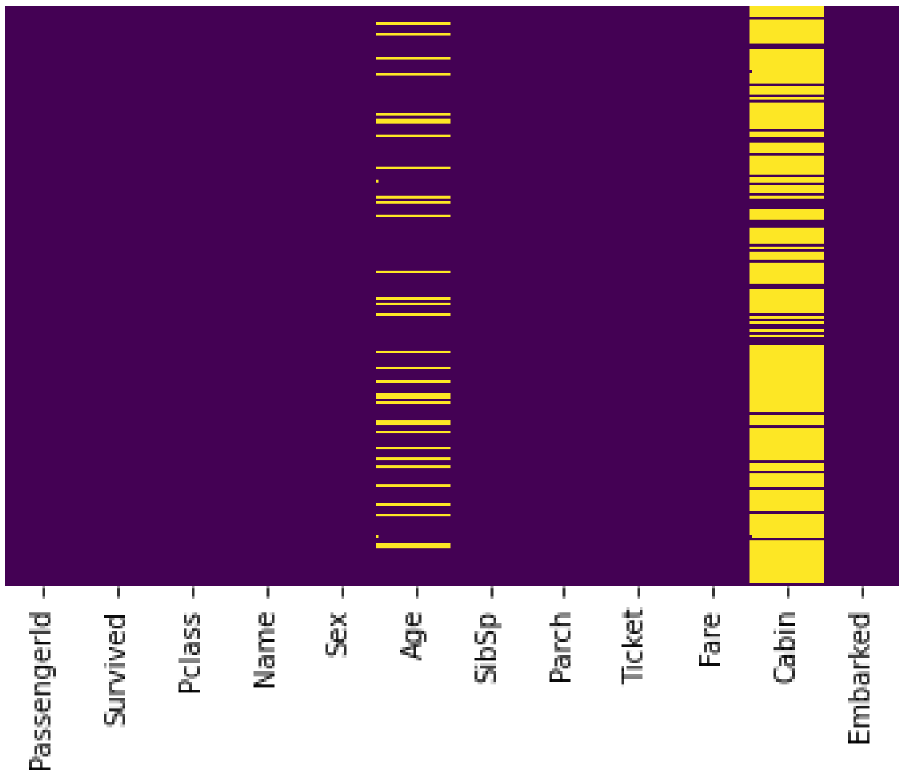
```
[4]:      PassengerId  Survived  Pclass   Name    Sex    Age  SibSp  Parch  Ticket  \
     0          False     False   False  False  False  False  False  False   False
     1          False     False   False  False  False  False  False  False   False
     2          False     False   False  False  False  False  False  False   False
     3          False     False   False  False  False  False  False  False   False
     4          False     False   False  False  False  False  False  False   False
     ..           …         …      …     …     …     …     …     …      …
     886        False     False   False  False  False  False  False  False   False
     887        False     False   False  False  False  False  False  False   False
     888        False     False   False  False  False   True  False  False   False
     889        False     False   False  False  False  False  False  False   False
     890        False     False   False  False  False  False  False  False   False

           Fare  Cabin  Embarked
     0    False   True     False
     1    False  False     False
     2    False   True     False
     3    False  False     False
     4    False   True     False
     ..     …      …        …
     886  False   True     False
     887  False  False     False
     888  False   True     False
     889  False  False     False
     890  False   True     False

     [891 rows x 12 columns]
```

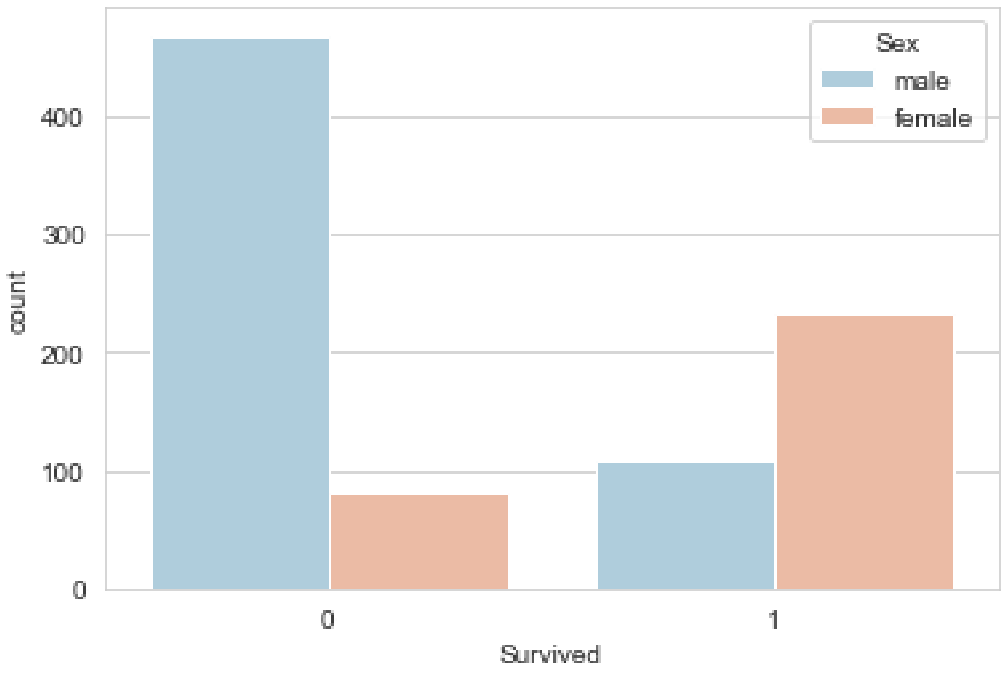
```
[5]: sns.heatmap(train.isnull(), cbar=False, yticklabels=False, cmap='viridis')
```

```
[5]: <AxesSubplot:>
```



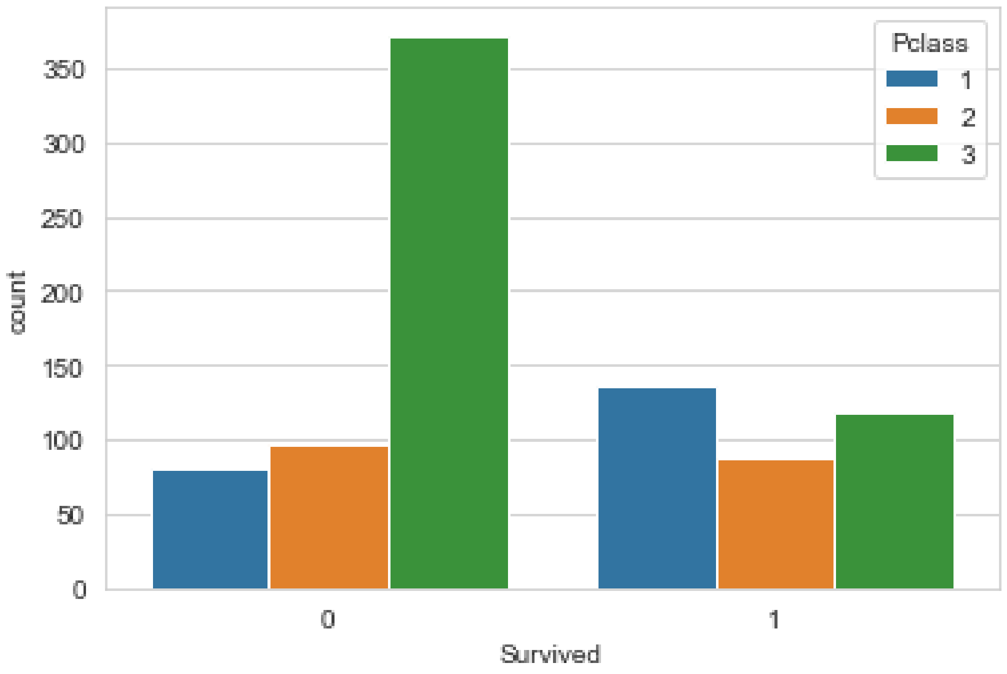
```
[10]: #for classsification problems its a good idea to see the ratio of target labels.
      #checking who survived
      sns.set_style('whitegrid')
      sns.countplot(x='Survived', data=train,hue='Sex', palette='RdBu_r')
```

```
[10]: <AxesSubplot:xlabel='Survived', ylabel='count'>
```



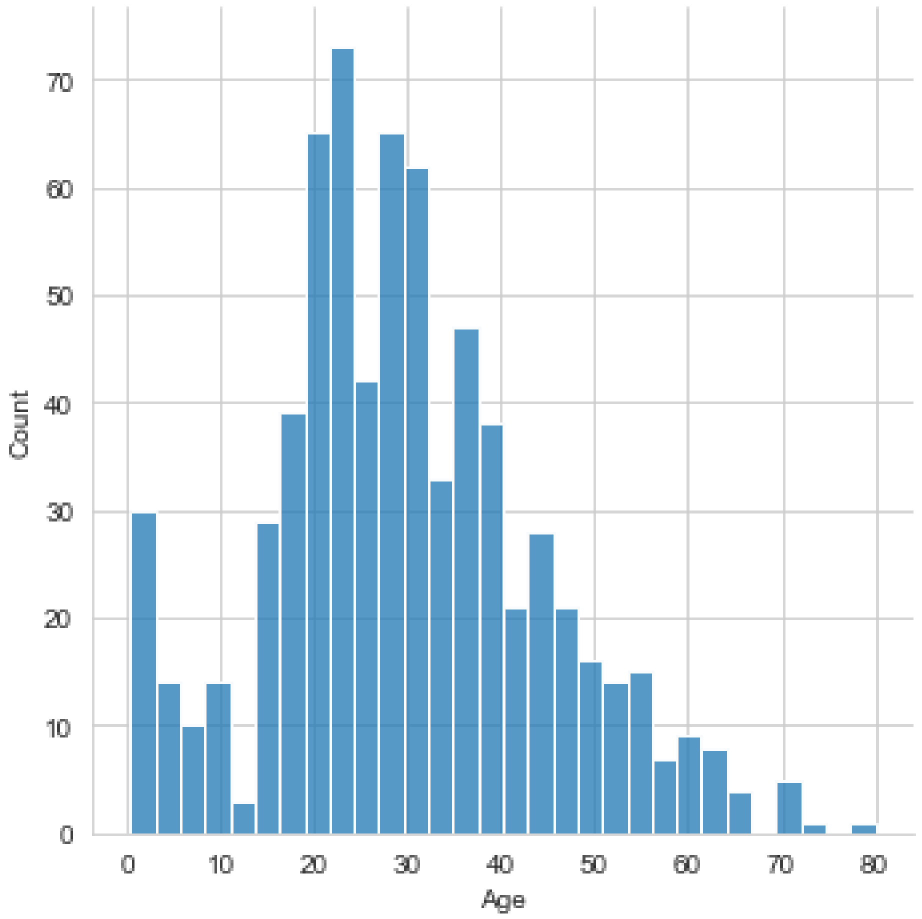
```
[11]: sns.countplot(x='Survived', data= train, hue='Pclass')
```

```
[11]: <AxesSubplot:xlabel='Survived', ylabel='count'>
```



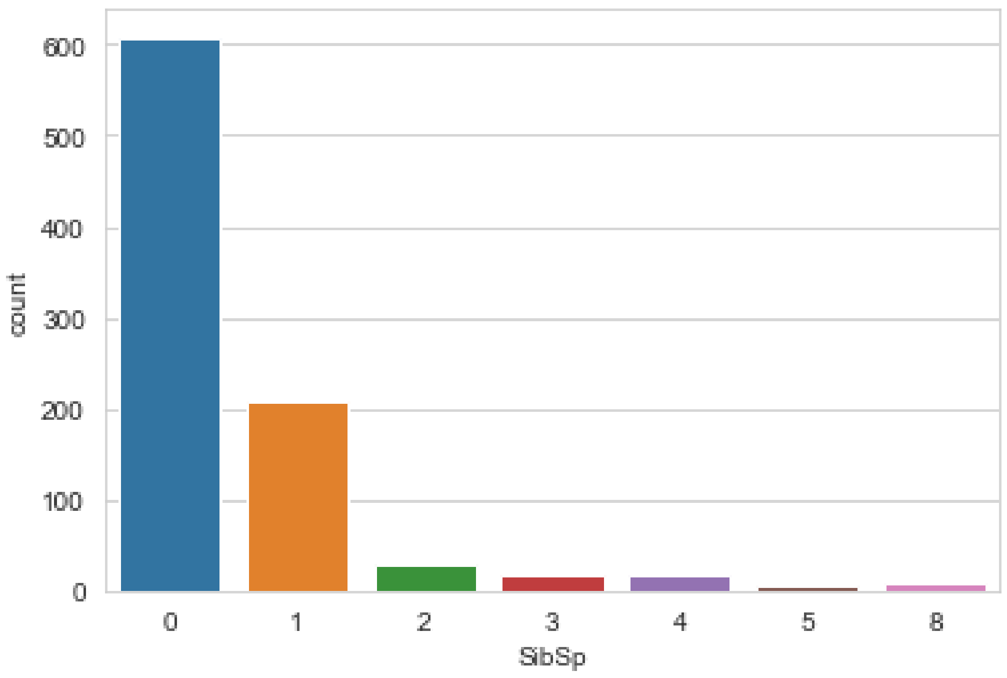
```
[13]: sns.displot(train['Age'].dropna(),bins=30)
```

```
[13]: <seaborn.axisgrid.FacetGrid at 0x2434881b3a0>
```



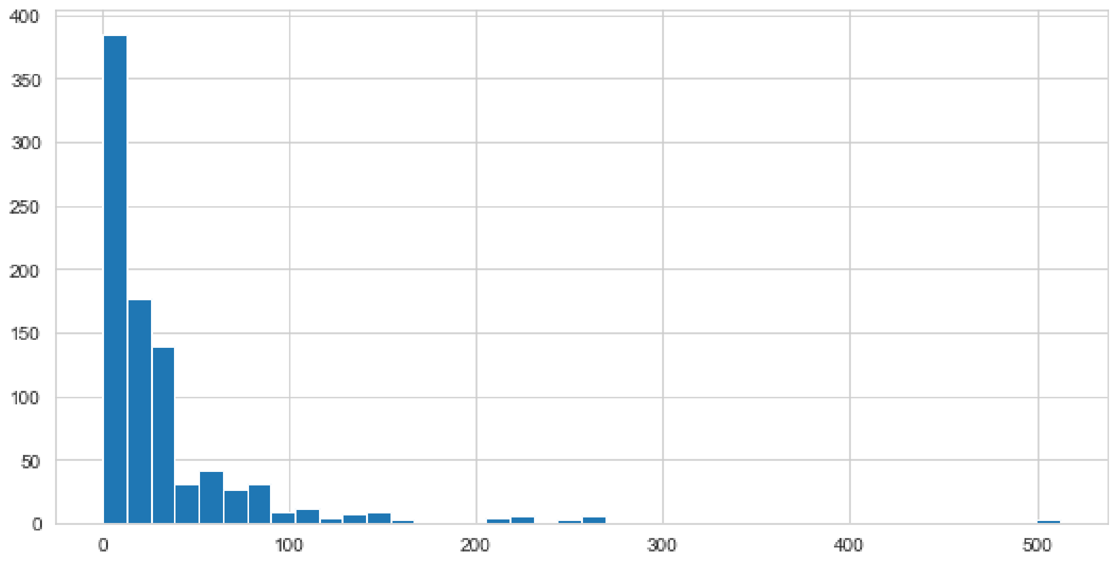
```
[14]: sns.countplot(x='SibSp', data=train)
```

```
[14]: <AxesSubplot:xlabel='SibSp', ylabel='count'>
```



```
[18]: train['Fare'].hist(bins=40,figsize=(10,5))
```

```
[18]: <AxesSubplot:>
```



```
[19]: import cufflinks as cf
      cf.go_offline()
```

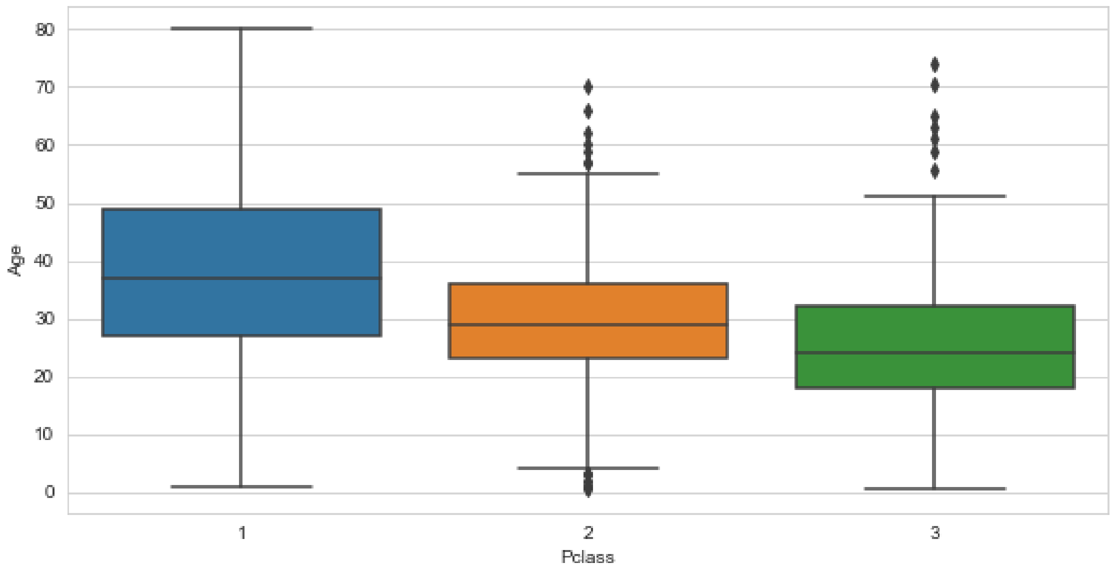
[21]:
```
train['Fare'].iplot(kind='hist',bins=40)
```

Cleaning Data

From the heatmap it is evident that Age and Cabin columns have a lot of missin data. It is not wise to drop the Age column altogether. So we will fill in the mmissing data.

[24]:
```
plt.figure(figsize=(10,5))
sns.boxplot(x='Pclass', y='Age', data=train)
```

[24]:
```
<AxesSubplot:xlabel='Pclass', ylabel='Age'>
```



The people in the first class and second class are older than third class.

[30]:
```
means = train.groupby('Pclass')['Age'].mean()
means
```

[30]:
```
Pclass
1    38.233441
2    29.877630
3    25.140620
Name: Age, dtype: float64
```

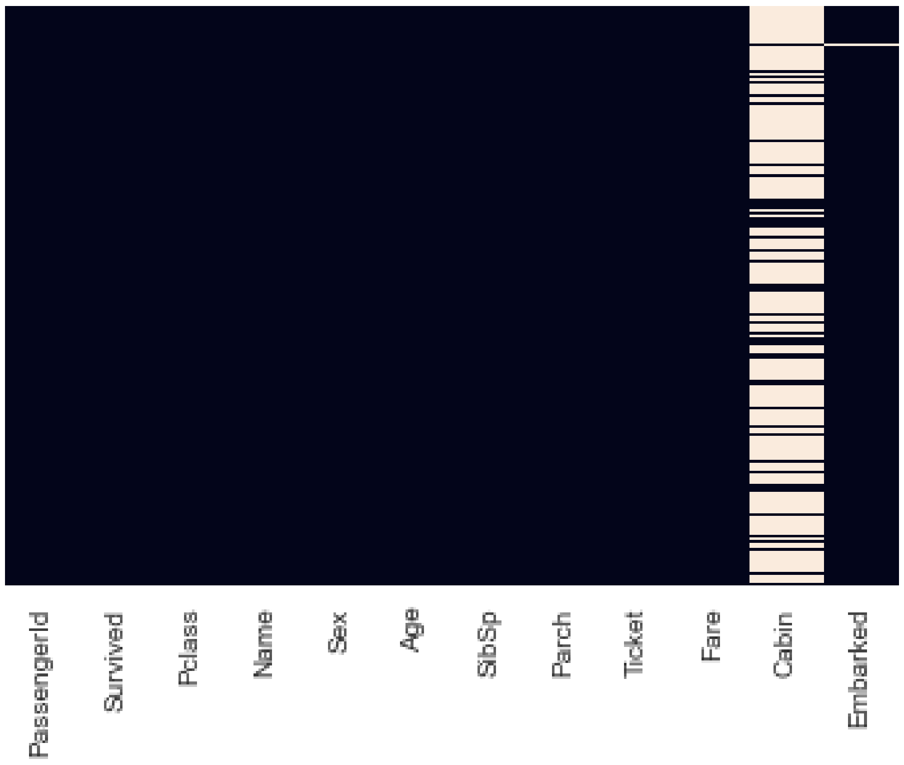
[32]:
```
def impute_age(cols):
    age = cols[0]
    pclass = cols[1]

    if pd.isnull(age):
        if pclass == 1:
            return 38
        elif pclass == 2:
            return 29
        else:
            return 25
    else:
        return age
```

[33]:
```
train['Age'] = train[['Age','Pclass']].apply(impute_age, axis=1)
```

[34]:
```
sns.heatmap(train.isnull(), yticklabels=False, cbar=False)
```

[34]:
```
<AxesSubplot:>
```



Cabin column has too many missing values so we are going to drop it.

[35]:
```
train.drop('Cabin', axis= 1, inplace = True)
```

[36]:
```
train.head()
```

[36]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
```

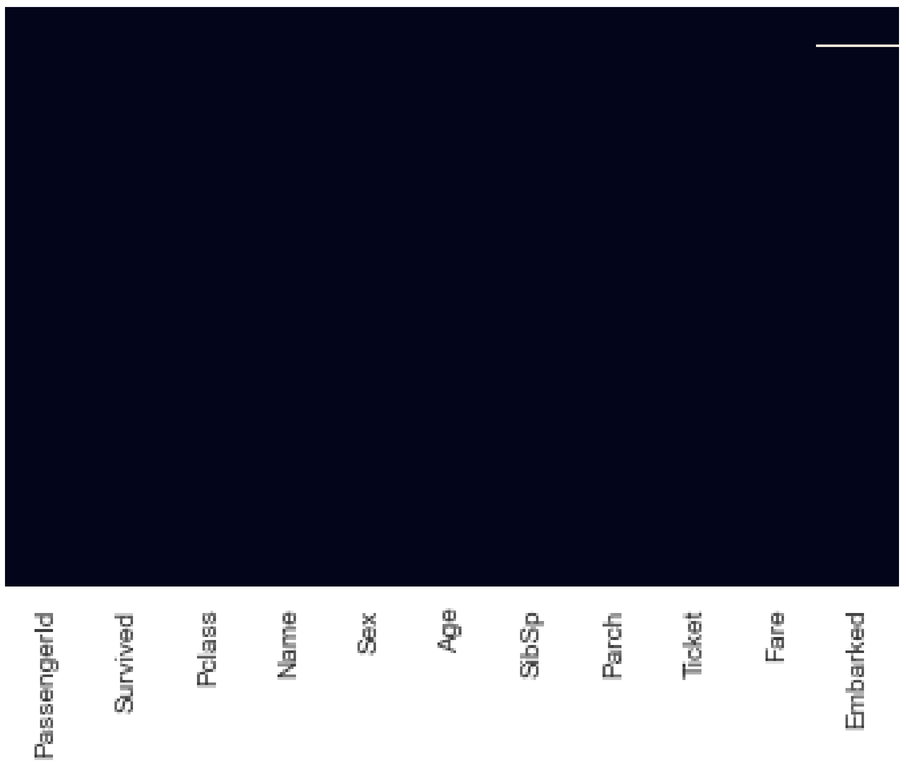
```
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                    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
```

[37]:
```python
sns.heatmap(train.isnull(), yticklabels=False, cbar=False)
```

[37]: `<AxesSubplot:>`

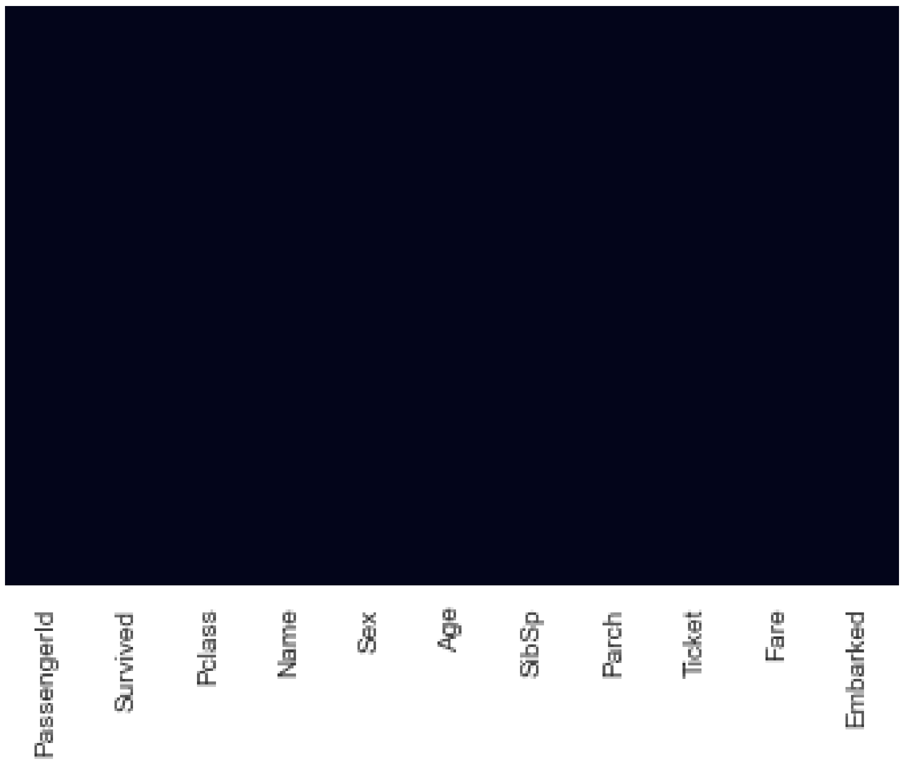


Dropping any remaining missing values.

[38]:
```
train.dropna(inplace=True)
```

[39]:
```
sns.heatmap(train.isnull(), yticklabels=False, cbar=False)
```

[39]:
```
<AxesSubplot:>
```



Creating Dummy Variables for Sex and Embarked columns to apply Machine Learning

[42]:
```
sex = pd.get_dummies(train['Sex'], drop_first=True)  #using drop_first to avoid␣
 ↪multicolinearity problems
```

[43]:
```
sex.head()
```

[43]:
```
   male
0     1
1     0
2     0
3     0
4     1
```

[44]:
```
embark = pd.get_dummies(train['Embarked'], drop_first=True)
```

```
[45]: embark.head()
```

```
[45]:    Q  S
      0  0  1
      1  0  0
      2  0  1
      3  0  1
      4  0  1
```

```
[46]: train = pd.concat([train, sex, embark], axis=1)
```

```
[47]: train.head(1)
```

```
[47]:    PassengerId  Survived  Pclass                     Name   Sex   Age  SibSp  \
      0            1         0       3  Braund, Mr. Owen Harris  male  22.0      1

         Parch     Ticket  Fare Embarked  male  Q  S
      0      0  A/5 21171  7.25        S     1  0  1
```

Dropping columns that are not usable

```
[51]: train.drop(['Sex','Embarked','Name','Ticket'], axis=1, inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-51-926d501f47cf> in <module>
----> 1 train.drop(['Sex','Embarked','Name','Ticket'], axis=1, inplace=True)

~\anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis,␣
 ↪index, columns, level, inplace, errors)
   4161                 weight  1.0     0.8
   4162         """
-> 4163         return super().drop(

   4164             labels=labels,
   4165             axis=axis,

~\anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis␣
 ↪index, columns, level, inplace, errors)
   3885         for axis, labels in axes.items():
   3886             if labels is not None:
-> 3887                 obj = obj._drop_axis(labels, axis, level=level,␣
 ↪errors=errors)
   3888
   3889         if inplace:

~\anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels␣
 ↪axis, level, errors)
```

```
   3919                 new_axis = axis.drop(labels, level=level, errors=errors
   3920             else:
-> 3921                 new_axis = axis.drop(labels, errors=errors)
   3922             result = self.reindex(**{axis_name: new_axis})
   3923

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels,
 ↪errors)
   5280         if mask.any():
   5281             if errors != "ignore":
-> 5282                 raise KeyError(f"{labels[mask]} not found in axis")
   5283             indexer = indexer[~mask]
   5284         return self.delete(indexer)

KeyError: "['Sex' 'Embarked' 'Name' 'Ticket'] not found in axis"
```

[55]:
```
train.drop('PassengerId', axis=1, inplace=True)
```

[56]:
```
train.head(1)
```

[56]:
```
   Survived  Pclass   Age  SibSp  Parch  Fare  male  Q  S
0         0       3  22.0      1      0  7.25     1  0  1
```

Machine Learning

[57]:
```
X= train.drop('Survived', axis=1)
y= train['Survived']
```

[58]:
```
from sklearn.model_selection import train_test_split
```

[59]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=101)
```

[62]:
```
from sklearn.linear_model import LogisticRegression
```

[66]:
```
lg = LogisticRegression()
```

[67]:
```
lg.fit(X_train, y_train)
```

```
C:\Users\ADMIN\anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:762: 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
```

```
[67]: LogisticRegression()
```

```
[68]: predictions = lg.predict(X_test)
```

```
[71]: from sklearn.metrics import classification_report
```

```
[72]: print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.83      0.91      0.87       163
           1       0.84      0.70      0.76       104

    accuracy                           0.83       267
   macro avg       0.83      0.81      0.82       267
weighted avg       0.83      0.83      0.83       267
```

```
[73]: from sklearn.metrics import confusion_matrix
```

```
[74]: confusion_matrix(y_test, predictions)
```

```
[74]: array([[149,  14],
             [ 31,  73]], dtype=int64)
```

```
[ ]:
```