Logistic Regression with Python

We are using the Titanic Dataset from Kaggle

Importing Libraries

In [1]:
```python
import pandas as pd
import numpy as np
```

In [2]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

The Data

Let's start by reading in the titanic_train.csv file into a pandas dataframe.

In [3]:
```python
train = pd.read_csv('titanic_train.csv')
```

In [4]:
```python
train.head()
```

Out[4]:

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

Exploratory Data Analysis

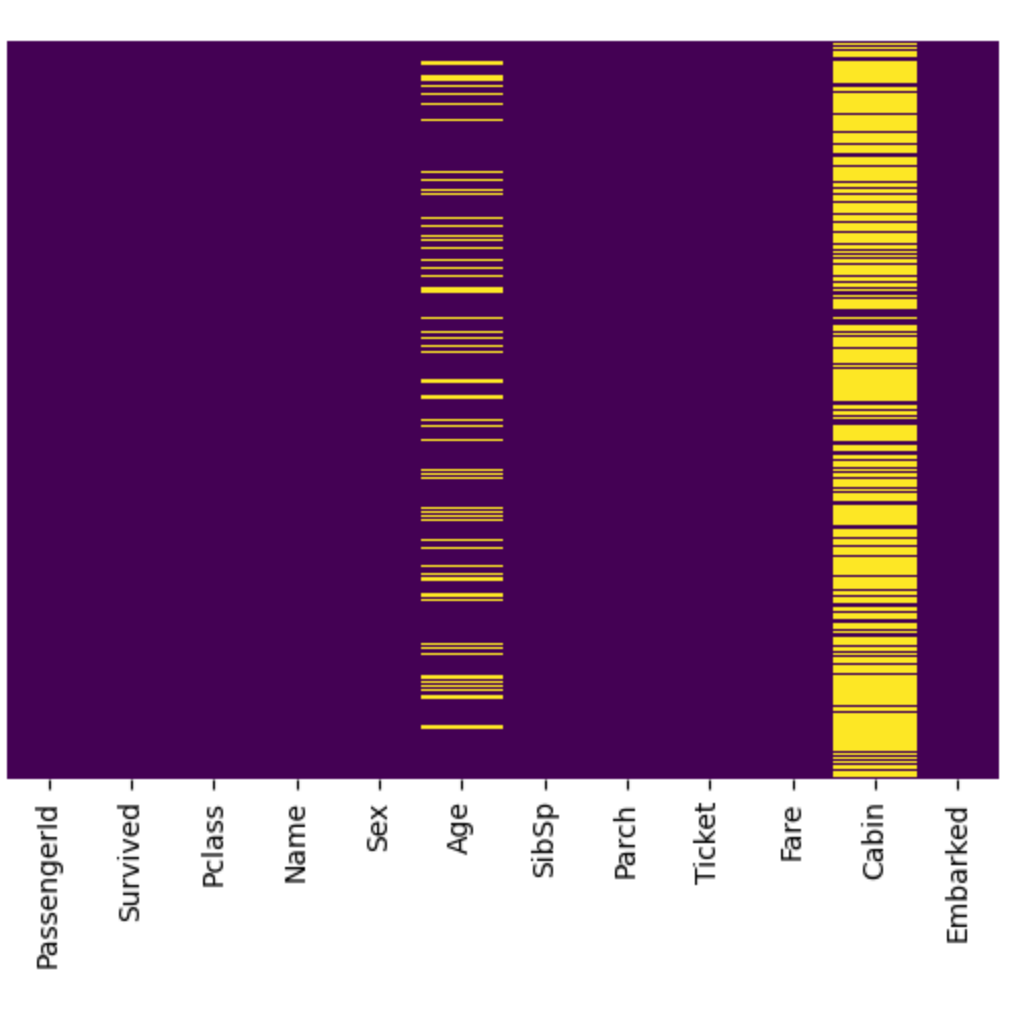
Let's begin some exploratory data analysis! We'll start by checking out missing data!

Missing Data

We can use seaborn to create a simple heatmap to see where we are missing data!

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

Out[5]: <AxesSubplot:>

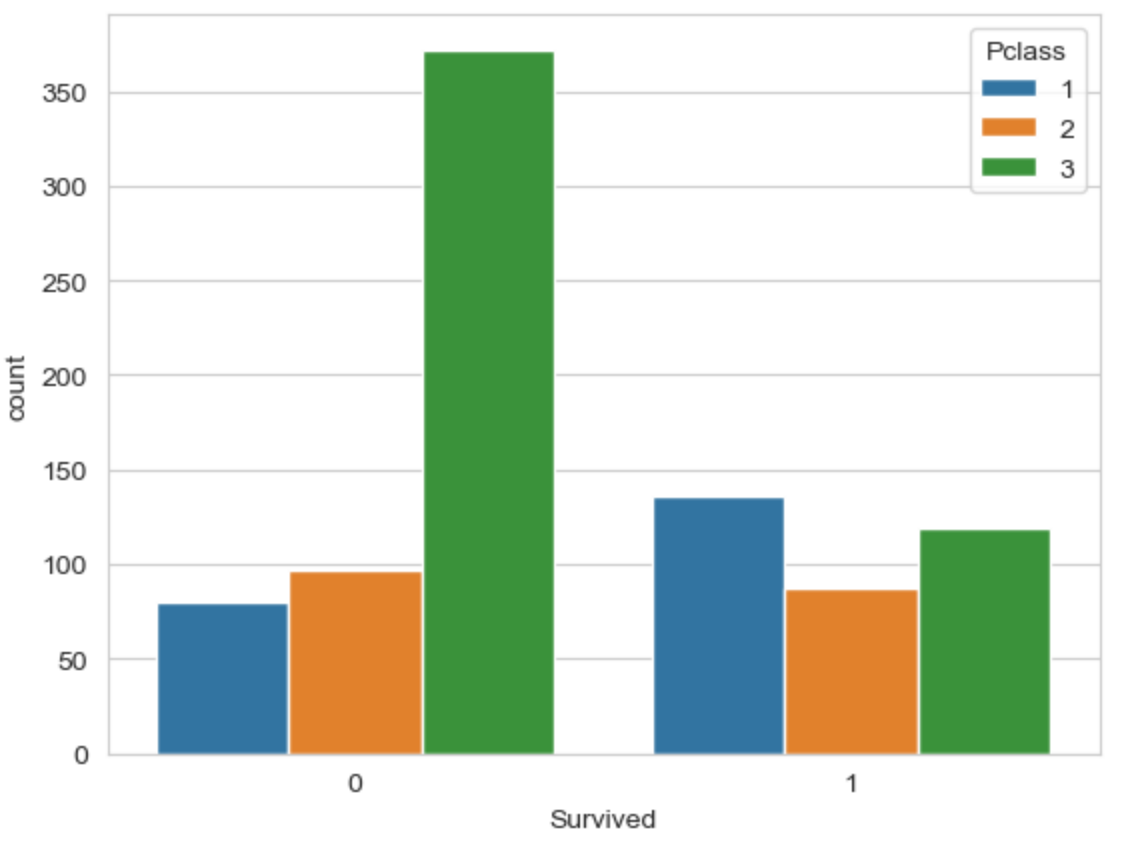


Roughly 20 percent of the Age data is missing. The proportion of Age missing is likely small enough for reasonable replacement with some form of imputation.

In [6]:
```python
sns.set_style('whitegrid')
```

In [7]:
```python
sns.countplot(x='Survived',data=train, hue='Pclass')
```

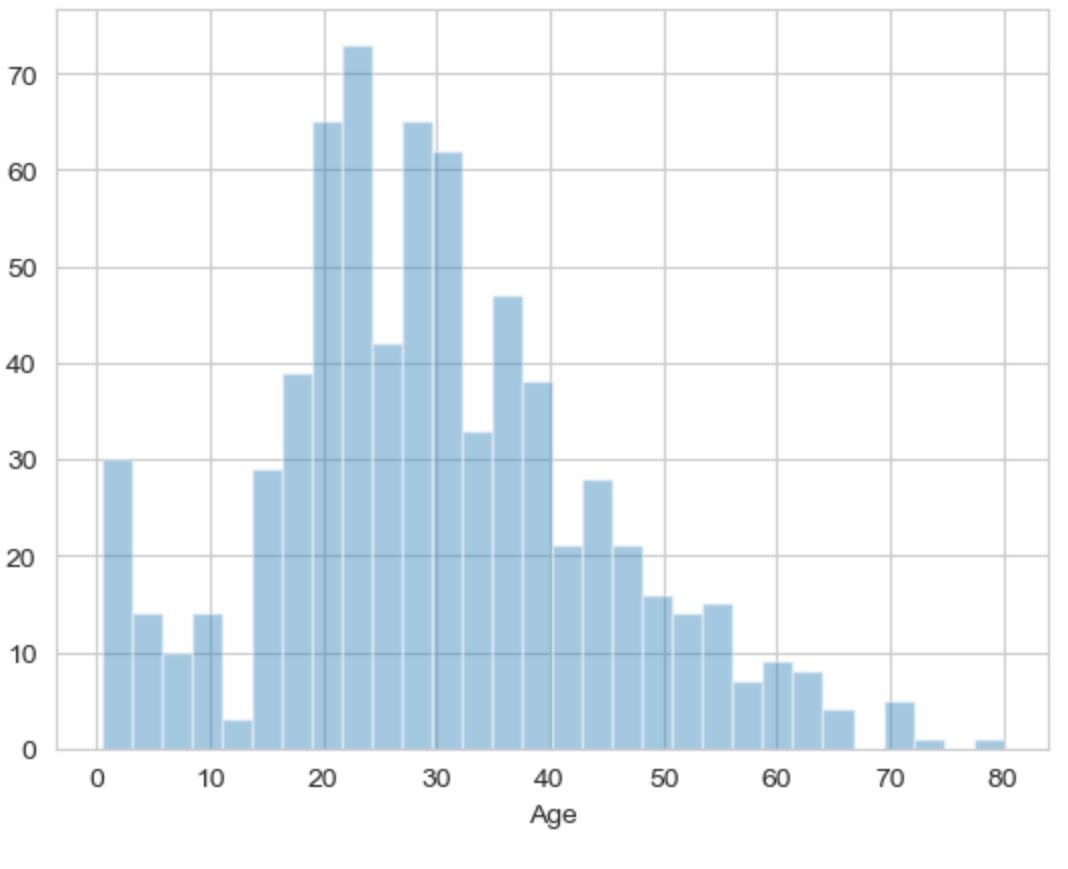
Out[7]: <AxesSubplot:xlabel='Survived', ylabel='count'>



In [8]:
```python
sns.distplot(train['Age'].dropna(),kde=False,bins=30)
```

```
C:\Users\tsaks\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]: <AxesSubplot:xlabel='Age'>

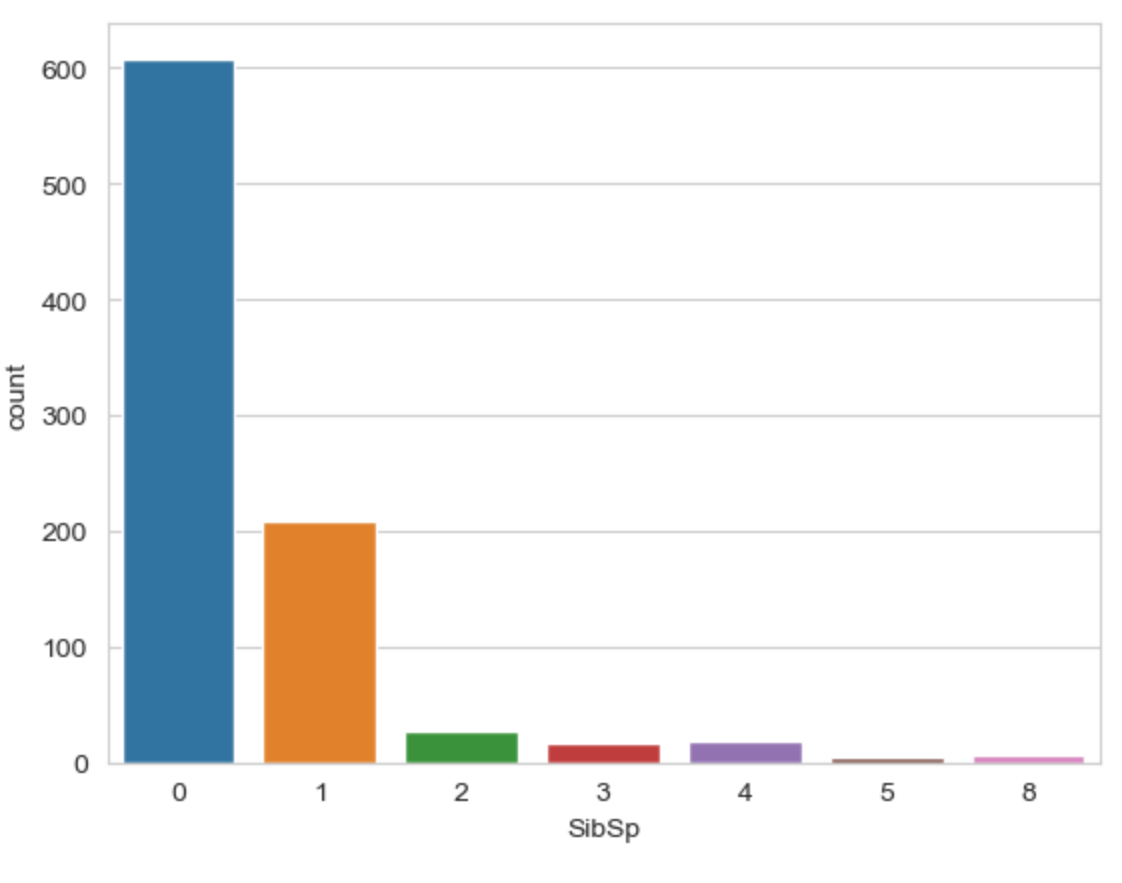


In [9]:
```python
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 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          714 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  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

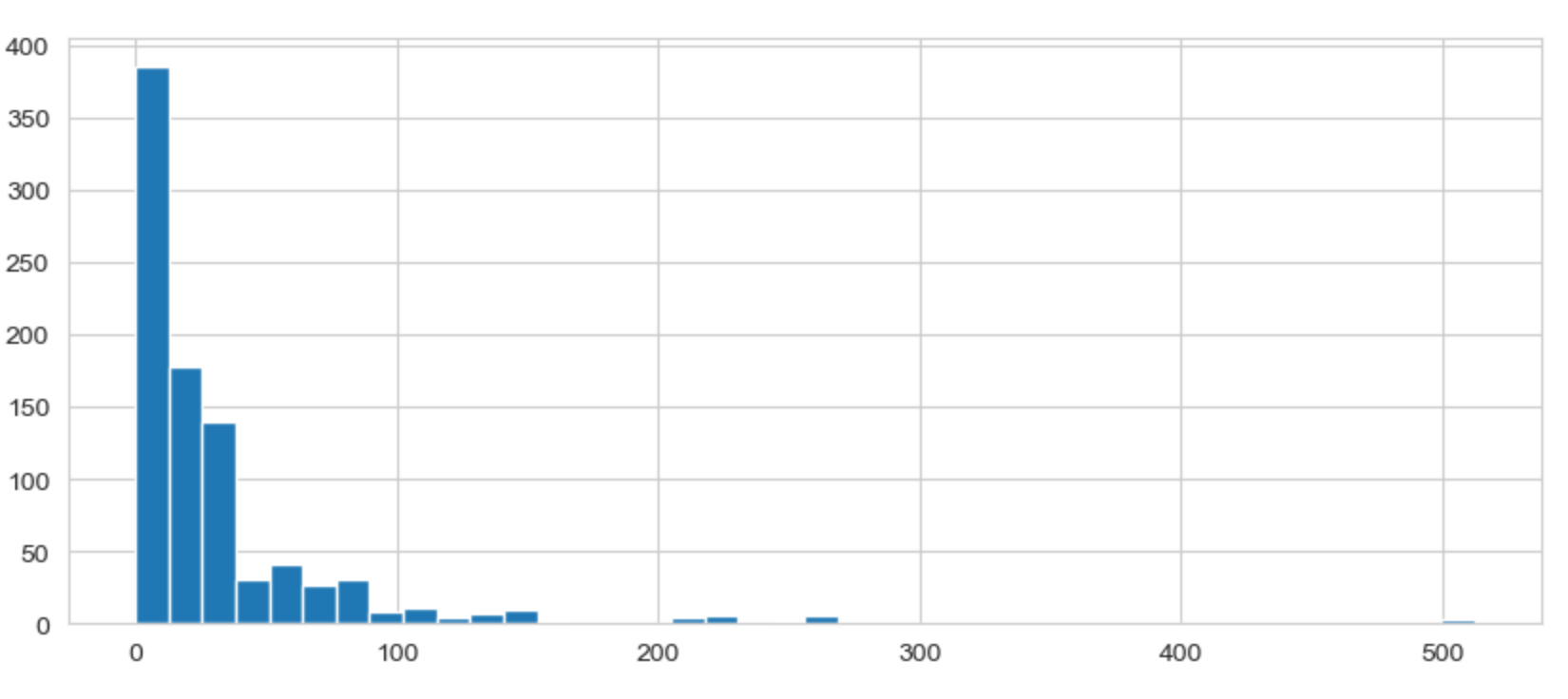
In [10]:
```python
sns.countplot(x='SibSp',data=train)
```

Out[10]: <AxesSubplot:xlabel='SibSp', ylabel='count'>



In [11]:
```python
train['Fare'].hist(bins=40, figsize=(10,4))
```

Out[11]: <AxesSubplot:>



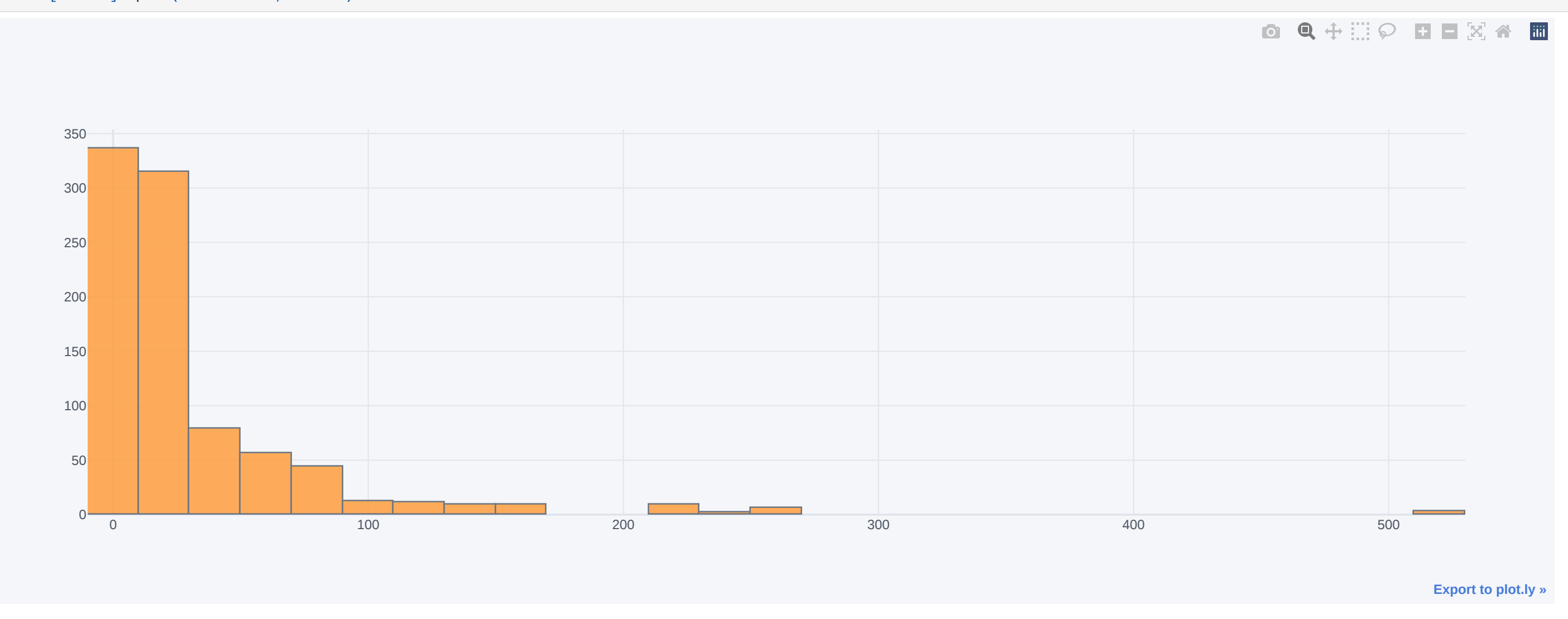
Cufflinks for plots

Let's take a quick moment to show an example of cufflinks!

In [12]:
```python
import cufflinks as cf
```

In [13]:
```python
cf.go_offline()
```

In [14]:
```python
train['Fare'].iplot(kind='hist',bins=30)
```

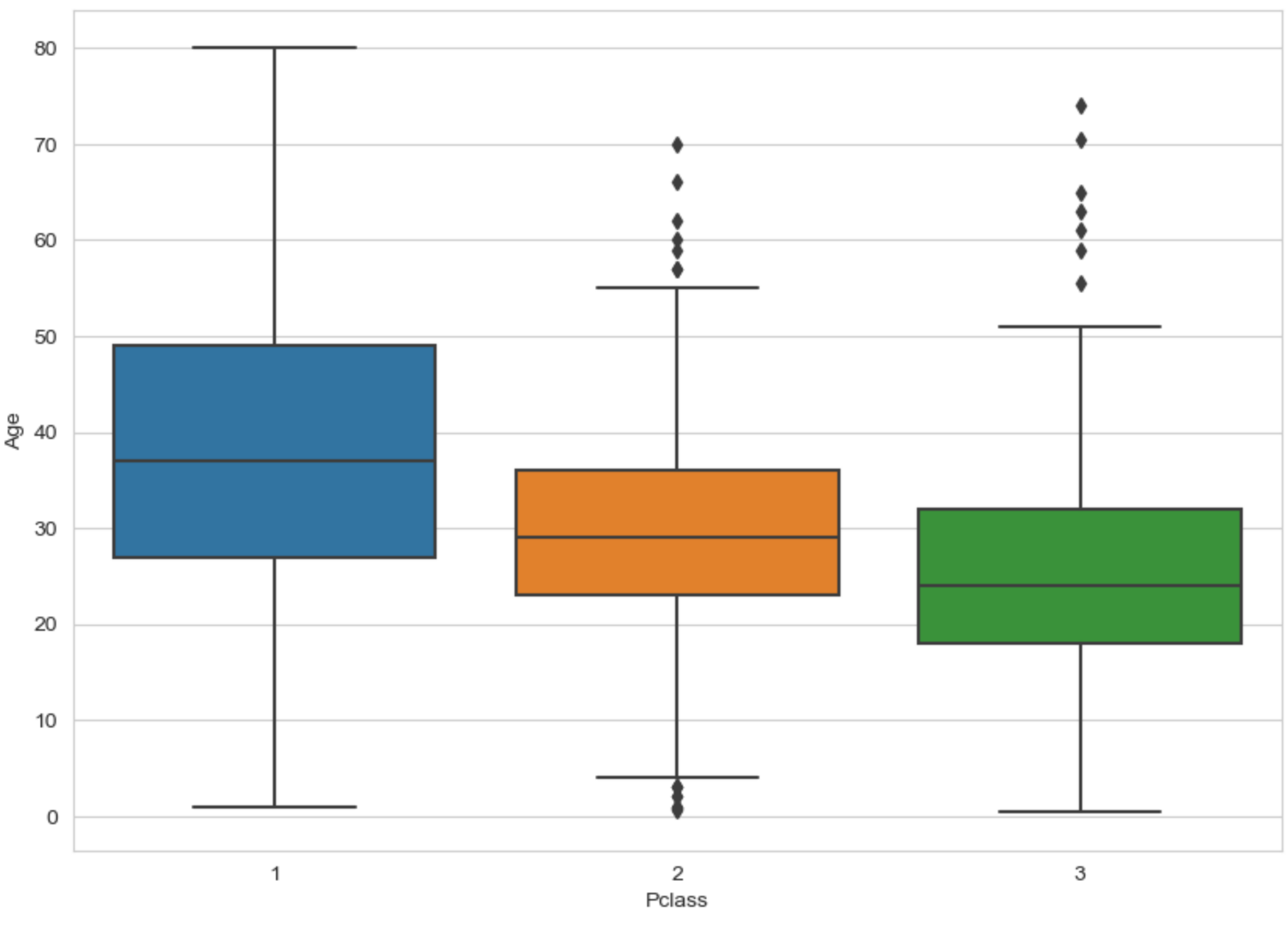


Data Cleaning

We want to fill in missing age data instead of just dropping the missing age data rows. One way to do this is by filling in the mean age of all the passengers (imputation). However we can be smarter about this and check the average age by passenger class. For example:

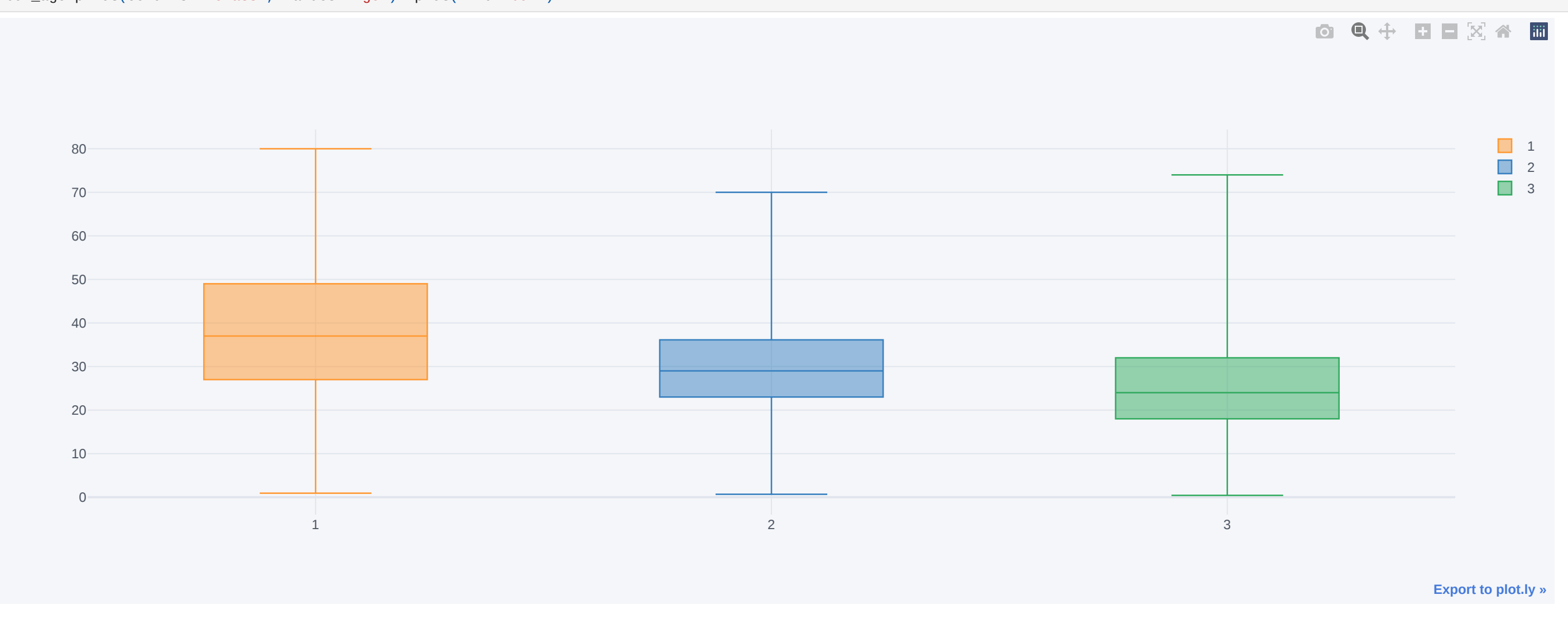
In [15]:
```python
plt.figure(figsize=(10,7))
sns.boxplot(x='Pclass',y='Age', data=train)
```

Out[15]: <AxesSubplot:xlabel='Pclass', ylabel='Age'>



Boxplot using Plotly

In [16]:
```python
box_age = train[['Pclass', 'Age']]
box_age.pivot(columns='Pclass', values='Age').iplot(kind='box')
```



We can see the wealthier passengers in the higher classes tend to be older, which makes sense. We'll use these average age values to impute based on Pclass for Age.

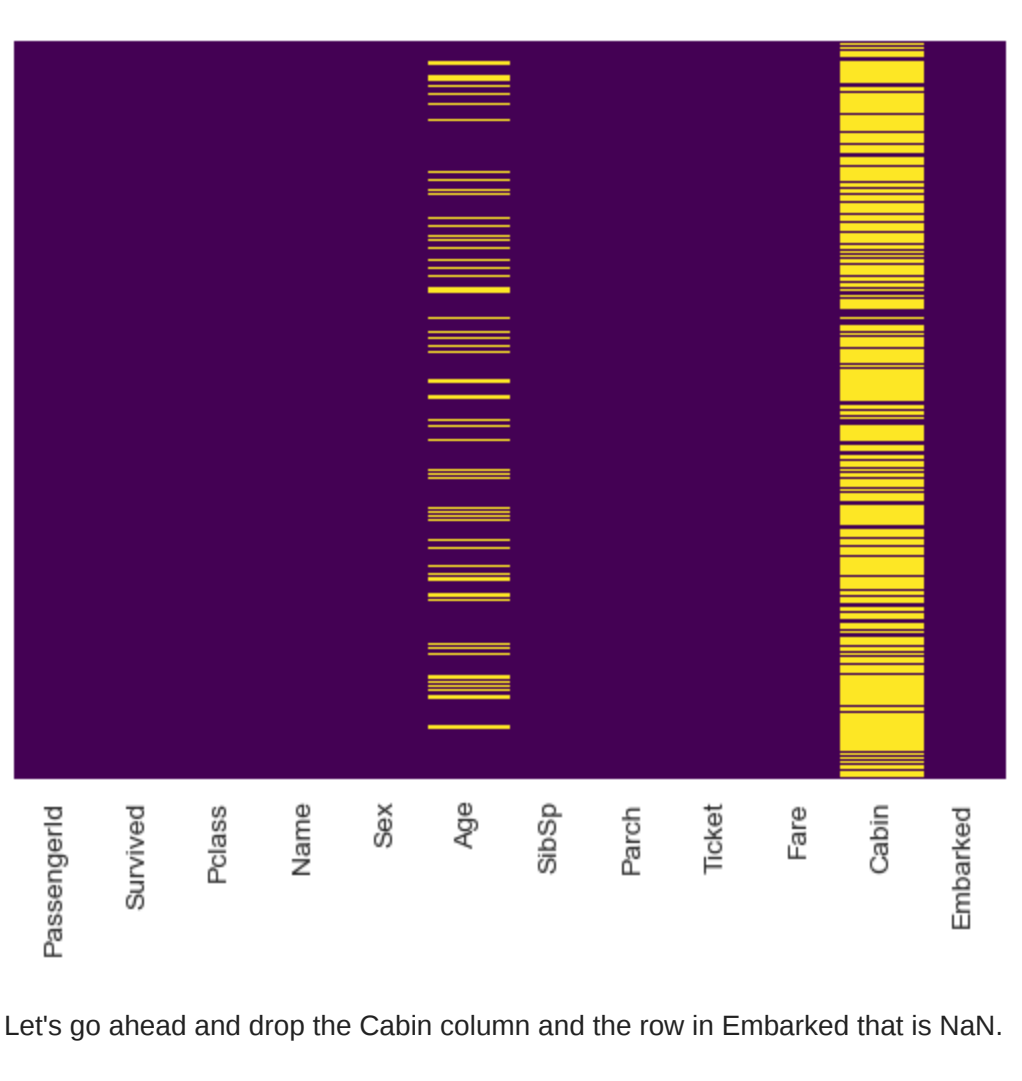
In [17]:
```python
def impute_age(cols):
    Age = cols[0]
    Pclass = cols[1]
    
    if pd.isnull(Age):

        if Pclass == 1:
            return 37
        elif Pclass == 2:
            return 29
        else:
            return 24
    else:
        return Age
```

Now applying that function!

Now let's check that heat map again

In [18]:
```python
sns.heatmap(train.isnull(),yticklabels=False, cbar=False, cmap='viridis')
```

Out[18]: <AxesSubplot:>



Let's go ahead and drop the Cabin column and the row in Embarked that is NaN.

In [19]:
```python
train.drop('Cabin', axis=1, inplace=True)
```

In [20]:
```python
train.head()
```

Out[20]:

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

In [21]:
```python
train.dropna(inplace=True)
```

Converting Categorical Features

We'll need to convert categorical features to dummy variables using pandas! Otherwise our machine learning algorithm won't be able to directly take in those features as inputs.

In [47]:
```python
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  712 non-null    int64  
 1   Pclass    712 non-null    int64  
 2   Age       712 non-null    float64
 3   SibSp     712 non-null    int64  
 4   Parch     712 non-null    int64  
 5   Fare      712 non-null    float64
 6   male      712 non-null    uint8  
 7   Q         712 non-null    uint8  
 8   S         712 non-null    uint8  
dtypes: float64(2), int64(4), uint8(3)
memory usage: 41.0 KB
```

In [22]:
```python
sex = pd.get_dummies(train['Sex'], drop_first= True)
```

In [23]:
```python
embark = pd.get_dummies(train['Embarked'], drop_first = True)
```

In [24]:
```python
embark.head()
```

Out[24]:

	Q	S
0	0	1
1	0	0
2	0	1
3	0	1
4	0	1

In [25]:
```python
train = pd.concat([train, sex, embark], axis = 1)
```

In [26]:
```python
train.head()
```

Out[26]:

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

In [27]:
```python
train.drop(['Sex', 'Embarked', 'Name','Ticket'], axis =1,inplace=True)
```

In [28]:
```python
train.head()
```

Out[28]:

	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

In [29]:
```python
train.drop('PassengerId', axis=1, inplace= True)
```

In [32]:
```python
train.head()
```

Out[32]:

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

In [34]:
```python
X = train.drop('Survived', axis=1)
y = train['Survived']
```

Building a Logistic Regression model

Let's start by splitting our data into a training set and test set.

Train Test Split

In [35]:
```python
from sklearn.model_selection import train_test_split
```

In [36]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

Training and Predicting

In [37]:
```python
from sklearn.linear_model import LogisticRegression
```

In [38]:
```python
logmodel = LogisticRegression()
```

In [39]:
```python
logmodel.fit(X_train, y_train)
```

```
C:\Users\tsaks\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: 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
```

Out[39]: LogisticRegression()

In [40]:
```python
predictions = logmodel.predict(X_test)
```

Evaluation

We can check precision,recall,f1-score using classification report!

In [41]:
```python
from sklearn.metrics import classification_report
```

In [42]:
```python
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.81      0.84      0.83       128
           1       0.75      0.71      0.73        86

    accuracy                           0.79       214
   macro avg       0.78      0.78      0.78       214
weighted avg       0.79      0.79      0.79       214
```

In [43]:
```python
from sklearn.metrics import confusion_matrix
```

In [44]:
```python
confusion_matrix(y_test,predictions)
```

Out[44]:
```
array([[108,  20],
       [ 25,  61]], dtype=int64)
```

In [ ]: