EDA with Python and applying Logistic Regression

For this lecture we will be working with the Titanic data set from Kaggle. This is a very famouse data set and very oftenis a student first step in machine learning

we'll be try to predict a classification-survival or deceased. Let's begin our understand of implementing logistic Regression in python for classification

we will use a 'semi-cleaned' version of the titanic data set.If you use the data set hosted directly on kaggle. you may need to do some additional cleaning and shown in this lecture notebook.

Import Libraries

Let's import some libraries to get started!

In [69]:
```
import pandas as pd
import numpy as np
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 [70]:
```
train=pd.read_csv('train.csv')
```

In [71]:
```
train.head()
```

Out[71]:

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

In [72]:
```
train.shape
```

Out[72]:
```
(891, 12)
```

In [73]:
```
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 [74]:
```
train.head()
```

Out[74]:

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

In [75]:
```
train.nunique()
```

Out[75]:
```
PassengerId    891
Survived         2
Pclass           3
Name           891
Sex              2
Age             88
SibSp            7
Parch            7
Ticket         681
Fare           248
Cabin          147
Embarked         3
dtype: int64
```

In [76]:
```
train.isnull()
```

Out[76]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

In [77]:
```
train.head(2)
```

Out[77]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embar
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	

Exploratory Data Analysis

Let's begin some exploratory data analysist we will 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 [78]:
```
train.isnull()
```

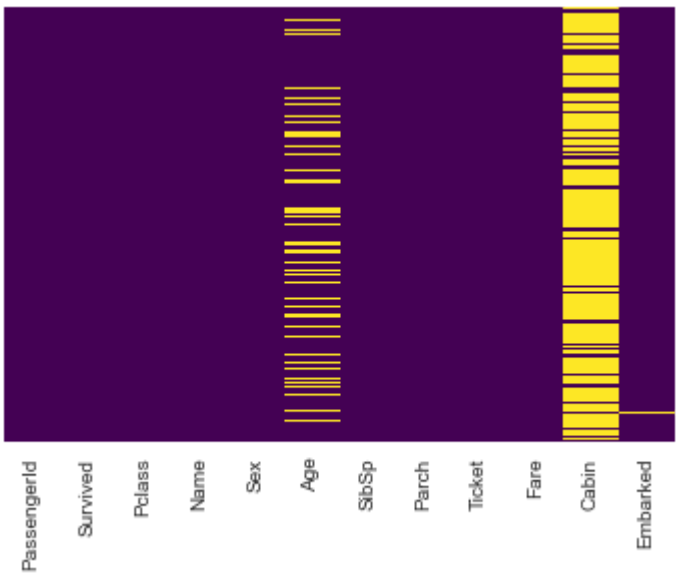
Out[78]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

In [79]:
```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[79]:
```
<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 looking at the cabin column. it looks like we are just missing to much of that data to something useful with at a basic level. we will probably drop this later, or change it to another feature like 'cabine known. 1or 0'

let's continue on by visualizing some more of the data! check out the video for full explations over these points. this code is just to serve as reference.

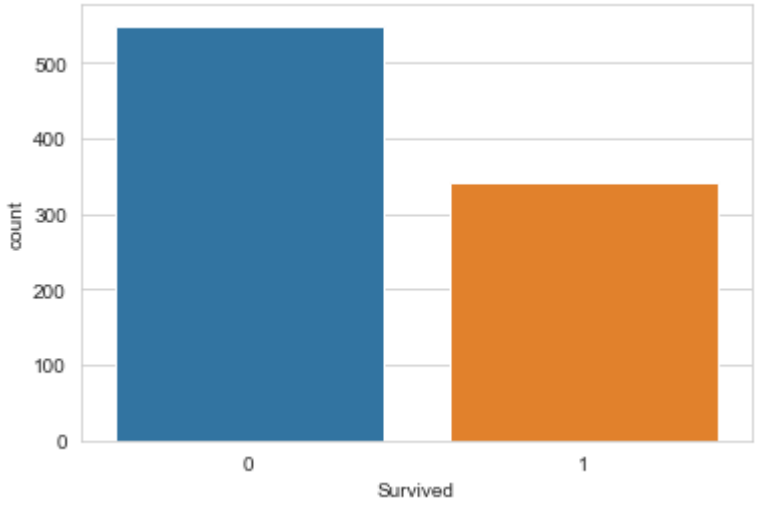
In [80]:
```
train.head(2)
```

Out[80]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embar
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	

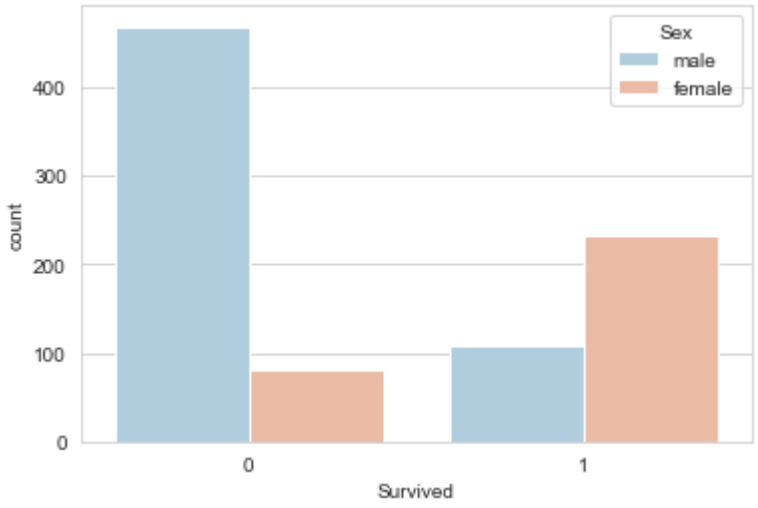
In [81]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived',data=train)
```

Out[81]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



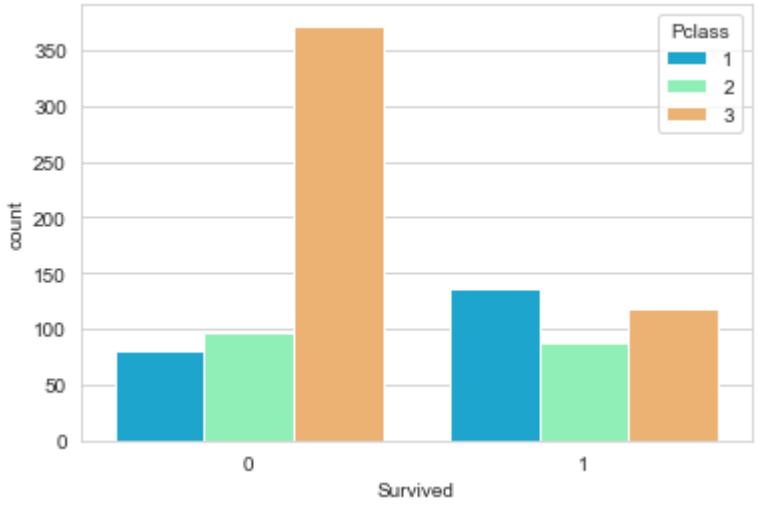
In [82]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Sex',data=train,palette='RdBu_r')
```

Out[82]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



In [83]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Pclass',data=train,palette='rainbow')
```

Out[83]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



In [84]:
```
train.head(2)
```

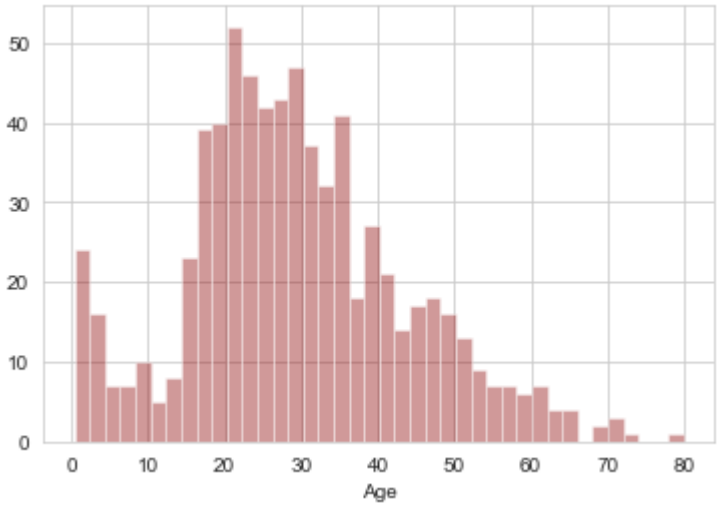
Out[84]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embar
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	

In [85]:
```
sns.distplot(train['Age'].dropna(),kde=False,color='darkred',bins=40)
```

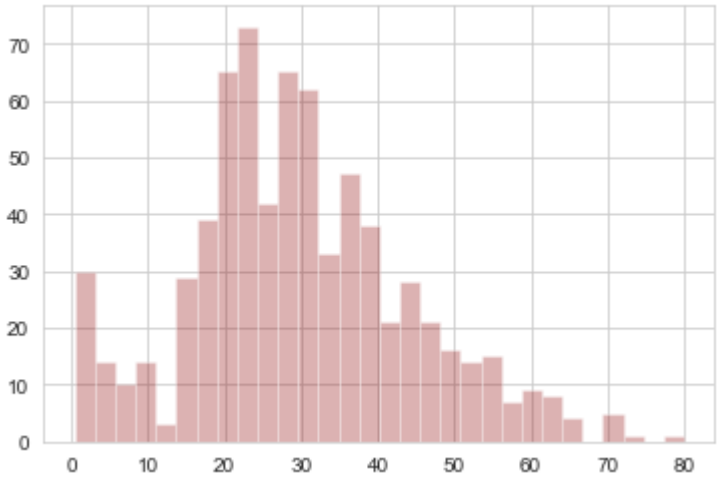
```
E:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot
` is a deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility) or `h
istplot` (an axes-level function for histograms).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

Out[85]:
```
<AxesSubplot:xlabel='Age'>
```



In [86]:
```
train['Age'].hist(bins=30,color='darkred',alpha=0.3)
```

Out[86]:
```
<AxesSubplot:>
```



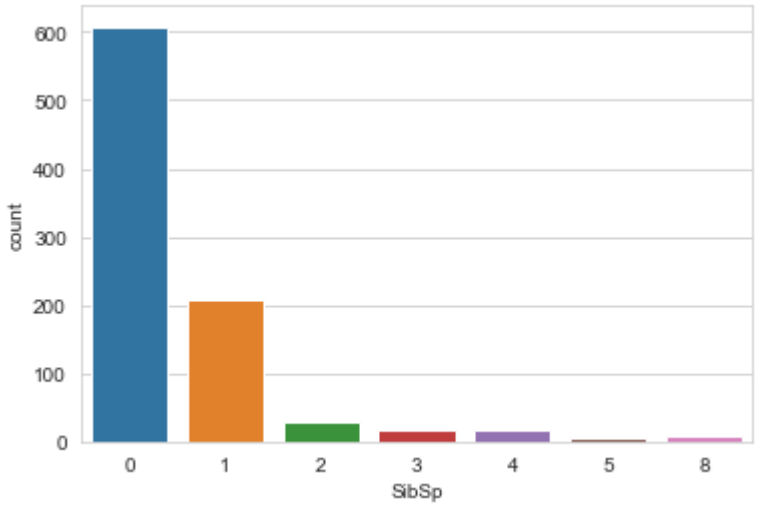
In [87]:
```
train.head(1)
```

Out[87]:

	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	7.25	NaN	S

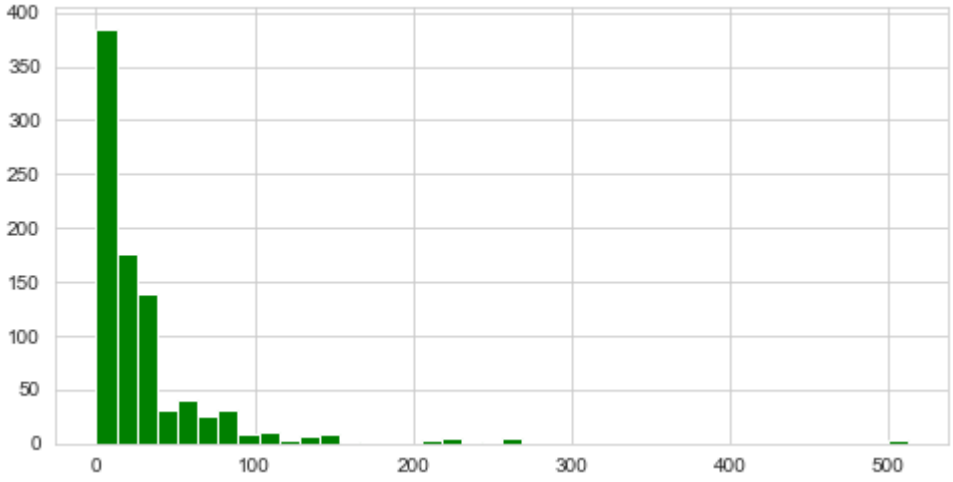
In [88]:
```
sns.countplot(x='SibSp',data=train)
```

Out[88]:
```
<AxesSubplot:xlabel='SibSp', ylabel='count'>
```



In [89]:
```
train['Fare'].hist(color='green',bins=40,figsize=(8,4))
```

Out[89]:
```
<AxesSubplot:>
```

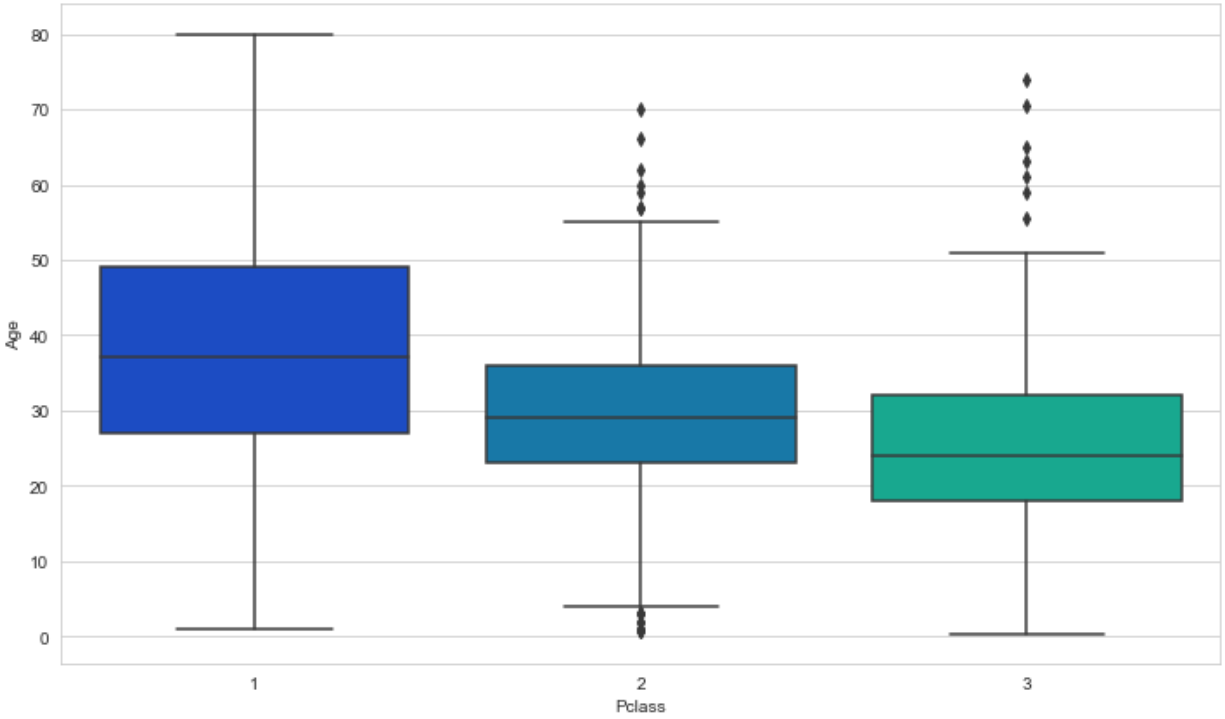


Data Cleaning

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 [90]:
```
plt.figure(figsize=(12,7))
sns.boxplot(x='Pclass',y='Age',data=train,palette='winter')
```

Out[90]:
```
<AxesSubplot:xlabel='Pclass', ylabel='Age'>
```



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

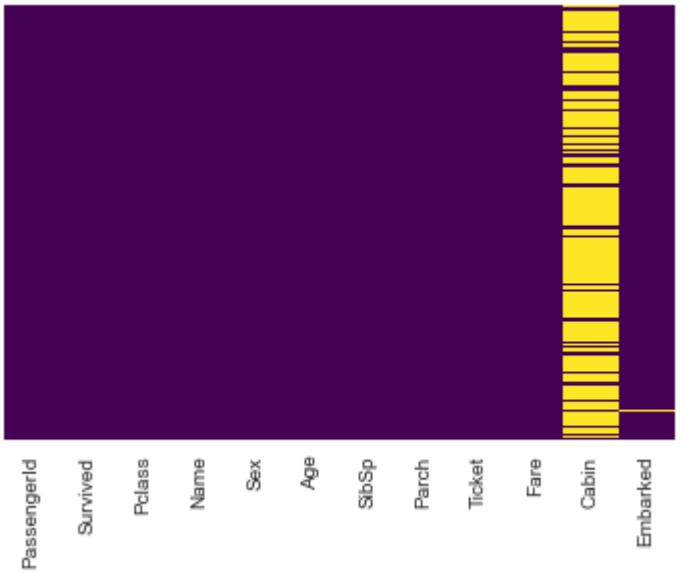
In [91]:
```
def impute_age(cols):
    Age = cols[0]
    Pclass = cols[1]

    if pd.isnull(Age):

        if Pclass == 1:
            return 37

        elif Pclass == 2:
            return 29

        else:
            return 24

    else:
        return Age

    ## Now apply that function!
```

In [92]:
```
train['Age'] = train[['Age','Pclass']].apply(impute_age,axis=1)
```

Now lets check that heat map again!

In [93]:
```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[93]:
```
<AxesSubplot:>
```



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

In [94]:
```
train.drop('Cabin',axis=1,inplace=True)
```

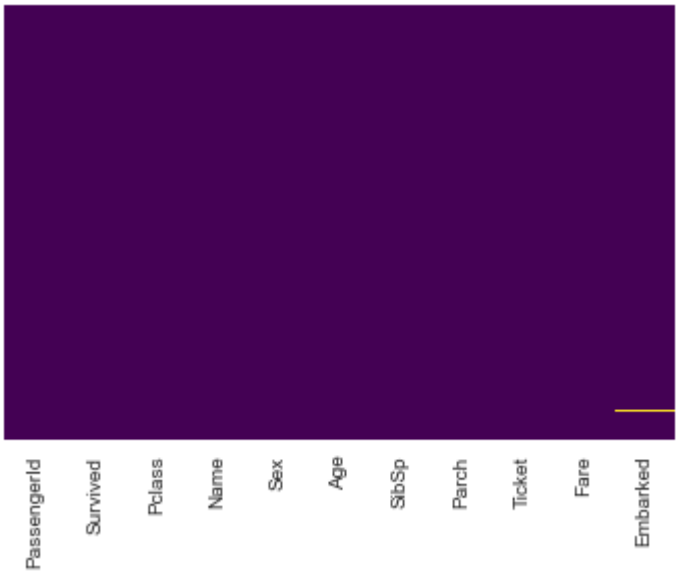
In [95]:
```
train.head()
```

Out[95]:

	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

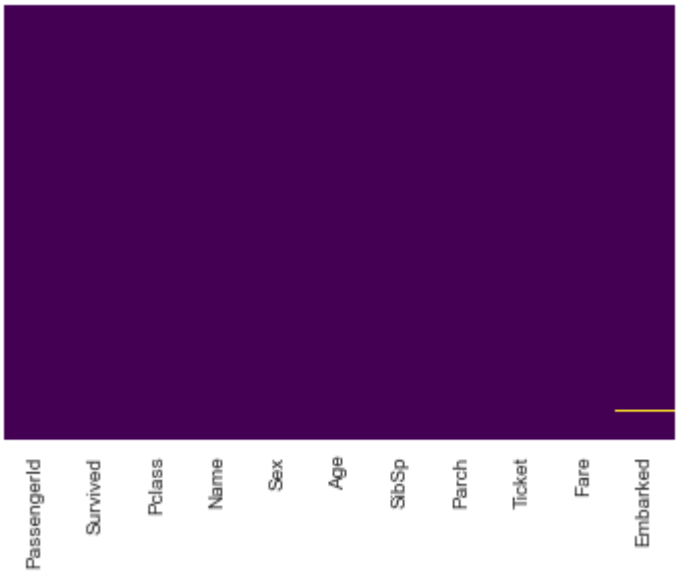
In [96]:
```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[96]:
```
<AxesSubplot:>
```



In [97]:
```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[97]:
```
<AxesSubplot:>
```



In [98]:
```
train.head(1)
```

Out[98]:

	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	7.25	S

In [99]:
```
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 [100…
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  889 non-null    int64  
 1   Survived     889 non-null    int64  
 2   Pclass       889 non-null    int64  
 3   Name         889 non-null    object 
 4   Sex          889 non-null    object 
 5   Age          889 non-null    float64
 6   SibSp        889 non-null    int64  
 7   Parch        889 non-null    int64  
 8   Ticket       889 non-null    object 
 9   Fare         889 non-null    float64
 10  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 83.3+ KB
```

In [101…
```
pd.get_dummies(train['Embarked'],drop_first=True).head()
```

Out[101]:

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

In [102…
```
sex = pd.get_dummies(train['Sex'],drop_first=True)
embark = pd.get_dummies(train['Embarked'],drop_first=True)
```

In [103…
```
train.drop(['Sex','Embarked','Name','Ticket'],axis=1,inplace=True)
```

In [104…
```
train.head()
```

Out[104]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
0	1	0	3	22.0	1	0	7.2500
1	2	1	1	38.0	1	0	71.2833
2	3	1	3	26.0	0	0	7.9250
3	4	1	1	35.0	1	0	53.1000
4	5	0	3	35.0	0	0	8.0500

In [105…
```
train = pd.concat([train,sex,embark],axis=1)
```

In [106…
```
train.head()
```

Out[106]:

	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

Great! Our data is ready for our model!

Building a Logistic Regression model

Let's start by splitting our data into a training set and test set (there is another test.csv file that you can play around with in case you want to use all this data for training).

Train Test Split

In [107…
```
train.drop('Survived',axis=1).head()
```

Out[107]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	male	Q	S
0	1	3	22.0	1	0	7.2500	1	0	1
1	2	1	38.0	1	0	71.2833	0	0	0
2	3	3	26.0	0	0	7.9250	0	0	1
3	4	1	35.0	1	0	53.1000	0	0	1
4	5	3	35.0	0	0	8.0500	1	0	1

In [108…
```
train['Survived'].head()
```

Out[108]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

In [109…
```
from sklearn.model_selection import train_test_split
```

In [111…
```
X_train, X_test, y_train, y_test = train_test_split(train.drop('Survived',axis=1),
                                                    train['Survived'], test_size=0.30,
                                                    random_state=101)
```

Training and Predicting

In [114…
```
from sklearn.linear_model import LogisticRegression
```

In [115…
```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarni
ng: 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(
```

Out[115]:
```
LogisticRegression()
```

In [116…
```
predictions = logmodel.predict(X_test)
```

In [117…
```
from sklearn.metrics import confusion_matrix
```

In [118…
```
accuracy=confusion_matrix(y_test,predictions)
```

In [119…
```
accuracy
```

Out[119]:
```
array([[150,  13],
       [ 39,  65]], dtype=int64)
```

In [120…
```
from sklearn.metrics import accuracy_score
```

In [121…
```
accuracy=accuracy_score(y_test,predictions)
accuracy
```

Out[121]:
```
0.8052434456928839
```

In [122…
```
predictions
```

Out[122]:
```
array([0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 1], dtype=int64)
```

Let's move on to evaluate our model!

Evaluation

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

In [123…
```
from sklearn.metrics import classification_report
```

In [124…
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.79      0.92      0.85       163
           1       0.83      0.62      0.71       104

    accuracy                           0.81       267
   macro avg       0.81      0.77      0.78       267
weighted avg       0.81      0.81      0.80       267
```

Not so bad! You might want to explore other feature engineering and the other titanic_text.csv file, some suggestions for feature engineering:

Try grabbing the Title (Dr.,Mr.,Mrs,etc..) from the name as a feature Maybe the Cabin letter could be a feature Is there any info you can get from the ticket

In [ ]: