Logistic Regression with Python - titanic dataset

In [73]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

The Data

In [74]:

```
train = pd.read_csv('titanic_train.csv')
```

In [75]:

```
train.head()
```

Out[75]:

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

Exploratory Data Analysis

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

Missing Data

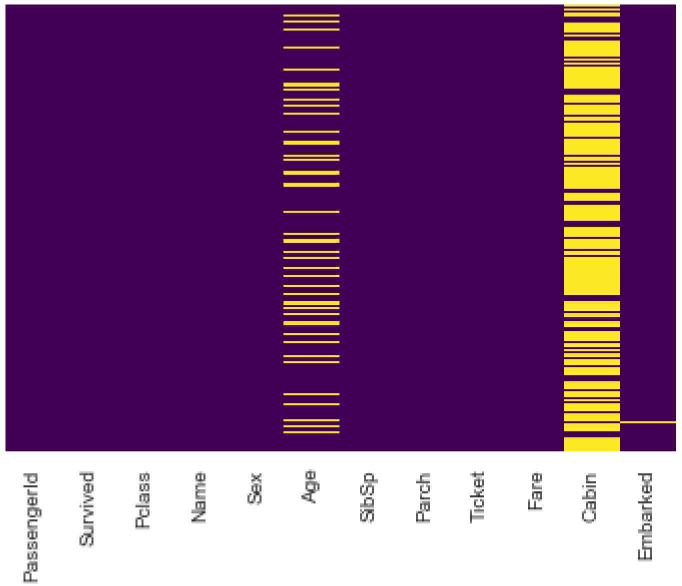
We can use seaborn to create a simple heatmap to see where we are missing data!

In [76]:

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

Out[76]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11a56f7b8>
```



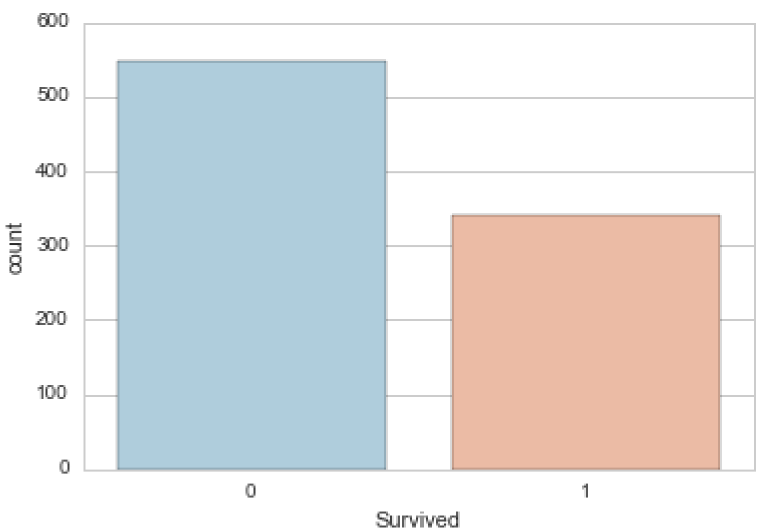
Roughly 20 percent of the Age data is missing. The proportion of Age missing is likely small enough for reasonable replacement with some form of imputation. Looking at the Cabin column, it looks like we are just missing too much of that data to do something useful with at a basic level. We'll probably drop this later, or change it to another feature like "Cabin Known: 1 or 0"

In [77]:

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

Out[77]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11afae630>
```

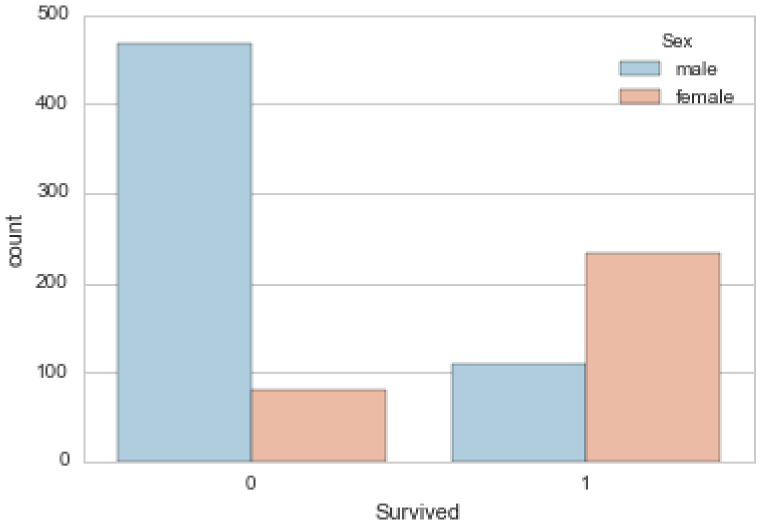


In [78]:

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

Out[78]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11b004a20>
```

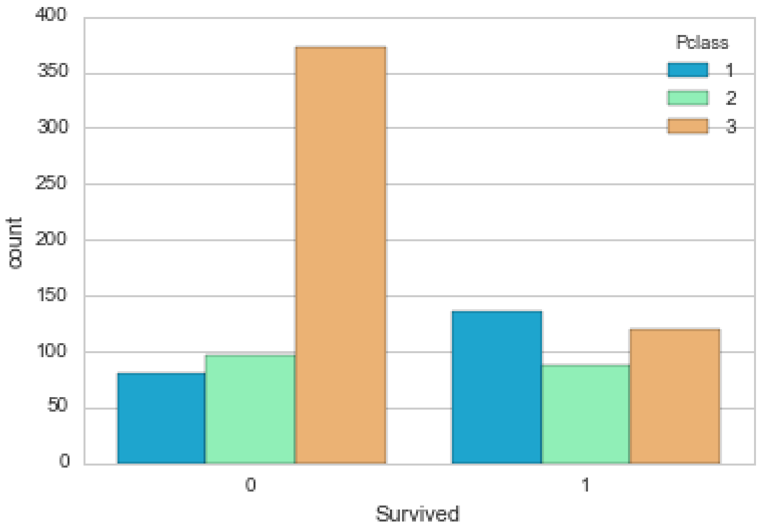


In [79]:

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

Out[79]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11b130f28>
```

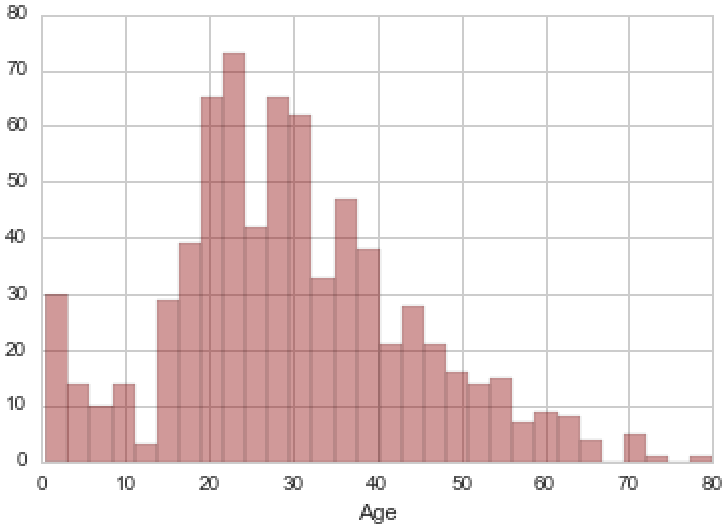


In [80]:

```
sns.distplot(train['Age'].dropna(),kde=False,color='darkred',bins=30)
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11c16f710>
```

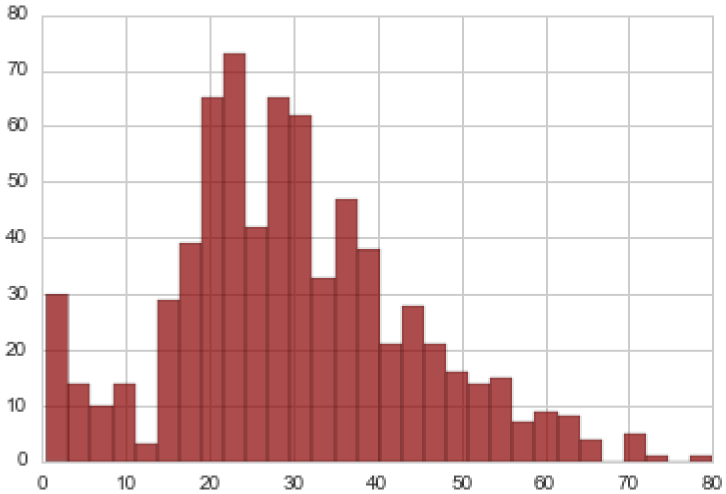


In [81]:

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

Out[81]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11b127ef0>
```

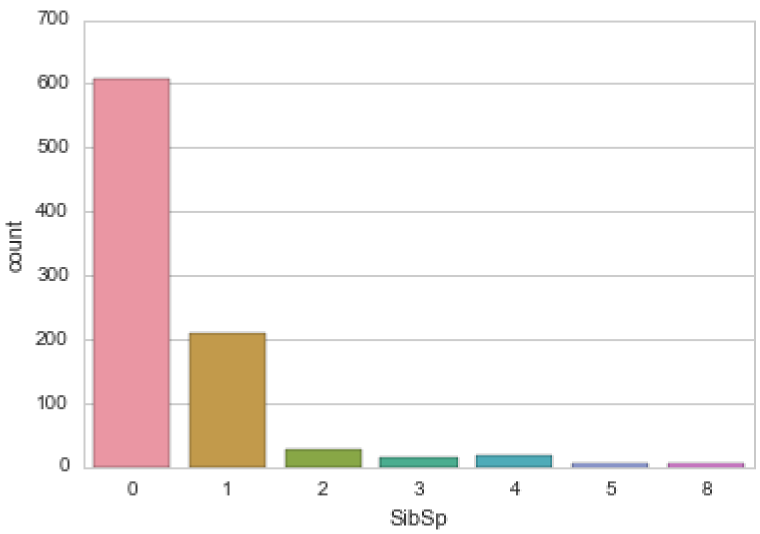


In [82]:

```
sns.countplot(x='SibSp',data=train)
```

Out[82]:

<matplotlib.axes._subplots.AxesSubplot at 0x11c4139e8>

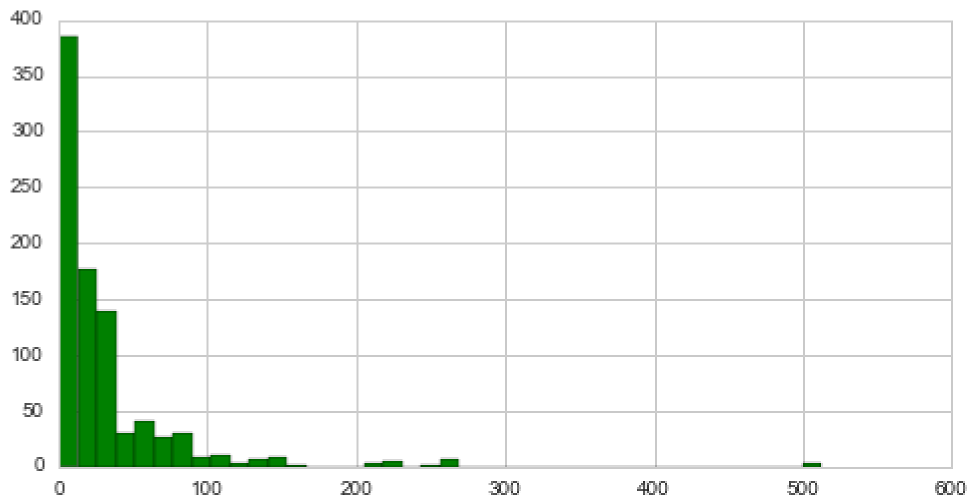


In [83]:

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

Out[83]:

<matplotlib.axes._subplots.AxesSubplot at 0x113893048>



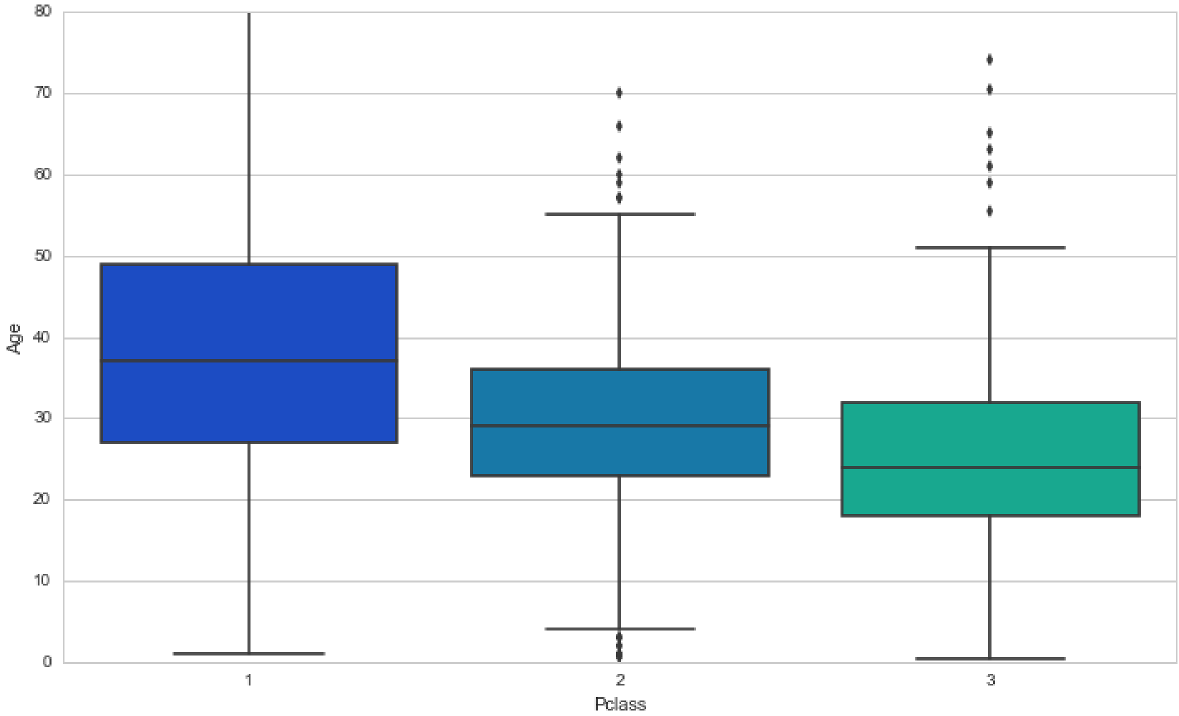
Data Cleaning

In [86]:

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

Out[86]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11c901cc0>
```



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 [87]:

```
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
```

In [88]:

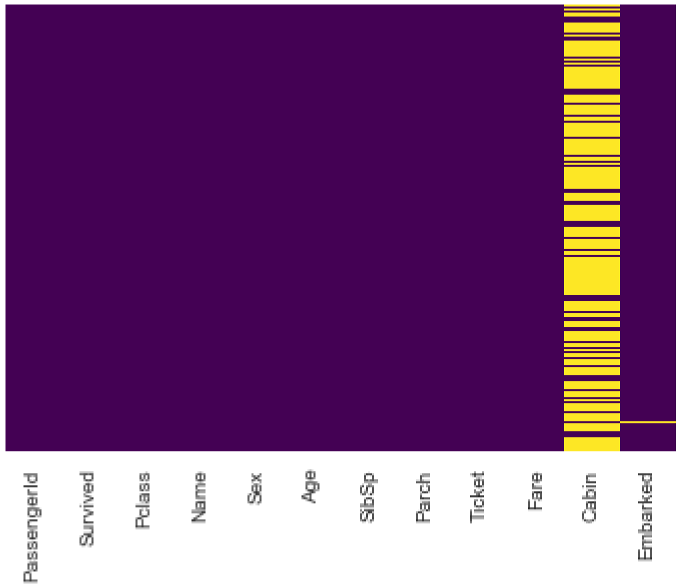
```
train['Age'] = train[['Age','Pclass']].apply(impute_age,axis=1)
```

In [89]:

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

Out[89]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11c4dae10>
```



drop the Cabin column and the row in Embarked that is NaN.

In [90]:

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

In [91]:

```
train.head()
```

Out[91]:

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

In [92]:

```
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 [93]:

```
train.info()
```

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

In [94]:

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

In [95]:

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

In [96]:

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

In [97]:

```
train.head()
```

Out[97]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	male	Q	S
0	1	0	3	22.0	1	0	7.2500	1.0	0.0	1.0
1	2	1	1	38.0	1	0	71.2833	0.0	0.0	0.0
2	3	1	3	26.0	0	0	7.9250	0.0	0.0	1.0
3	4	1	1	35.0	1	0	53.1000	0.0	0.0	1.0
4	5	0	3	35.0	0	0	8.0500	1.0	0.0	1.0

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 [98]:

```
from sklearn.model_selection import train_test_split
```

In [100]:

```
X_train, X_test, y_train, y_test = train_test_split(train.drop('Survived',axis=1),
                                                    train['Survived'], test_size=0.3
                                                    random_state=101)
```

Training and Predicting

In [101]:

```
from sklearn.linear_model import LogisticRegression
```

In [102]:

```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

Out[102]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs
=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0
001,
          verbose=0, warm_start=False)
```

In [103]:

```
predictions = logmodel.predict(X_test)
```

Evaluation

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

In [104]:

```
from sklearn.metrics import classification_report
```

In [105]:

```
print(classification_report(y_test,predictions))
```

```
             precision    recall  f1-score   support

          0       0.81      0.93      0.86       163
          1       0.85      0.65      0.74       104

avg / total       0.82      0.82      0.81       267
```