Logistic Regression - Titanic

October 26, 2016

---

1 Logistic Regression Titanic Projecrt

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

1.1 The Data

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

```
In [19]: train = pd.read_csv('titanic_train.csv')
```

```
In [20]: train['x']=train['Age']*2 /train['Pclass']
         train.head()
```

```
Out[20]:    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          x
         0      0         A/5 21171   7.2500   NaN        S  14.666667
         1      0          PC 17599  71.2833   C85        C  76.000000
         2      0  STON/O2. 3101282   7.9250   NaN        S  17.333333
         3      0            113803  53.1000  C123        S  70.000000
         4      0            373450   8.0500   NaN        S  23.333333
```

```
In [21]: train.head(8)
```

```
Out[21]:    PassengerId  Survived  Pclass  \
         0            1         0       3
```

```
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3
5            6         0       3
6            7         0       1
7            8         0       3
```

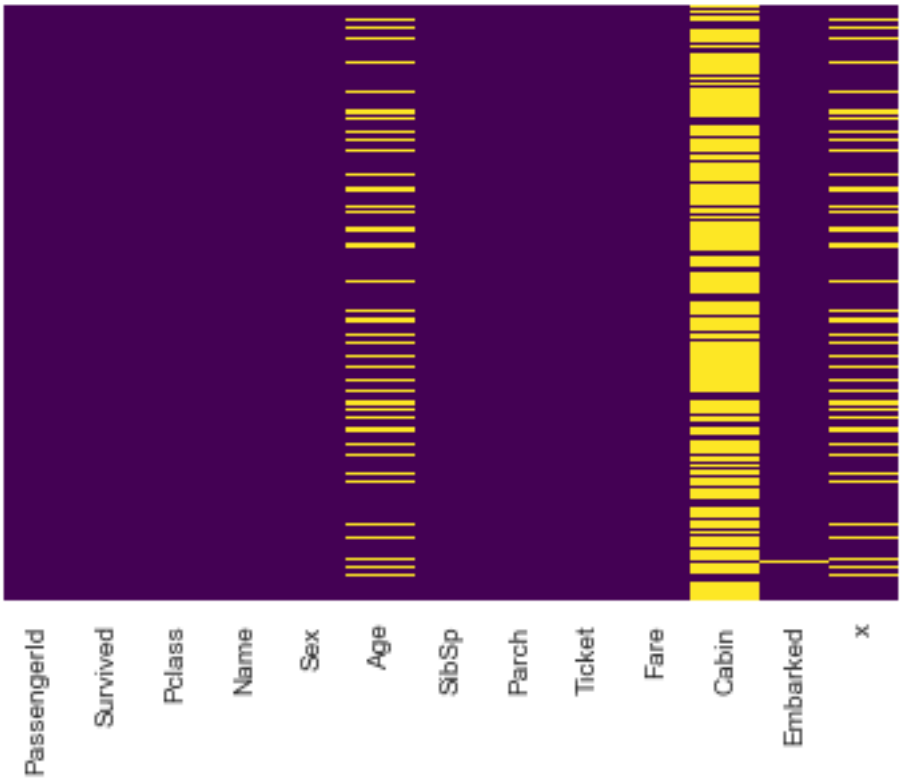
```
                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0
5                                    Moran, Mr. James    male   NaN      0
6                             McCarthy, Mr. Timothy J    male  54.0      0
7                      Palsson, Master. Gosta Leonard    male   2.0      3

   Parch            Ticket     Fare Cabin Embarked           x
0      0         A/5 21171   7.2500   NaN        S   14.666667
1      0          PC 17599  71.2833   C85        C   76.000000
2      0  STON/O2. 3101282   7.9250   NaN        S   17.333333
3      0            113803  53.1000  C123        S   70.000000
4      0            373450   8.0500   NaN        S   23.333333
5      0            330877   8.4583   NaN        Q         NaN
6      0             17463  51.8625   E46        S  108.000000
7      1            349909  21.0750   NaN        S    1.333333
```

2 Exploratory Data Analysis

2.1 Missing Data

In [22]:
```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

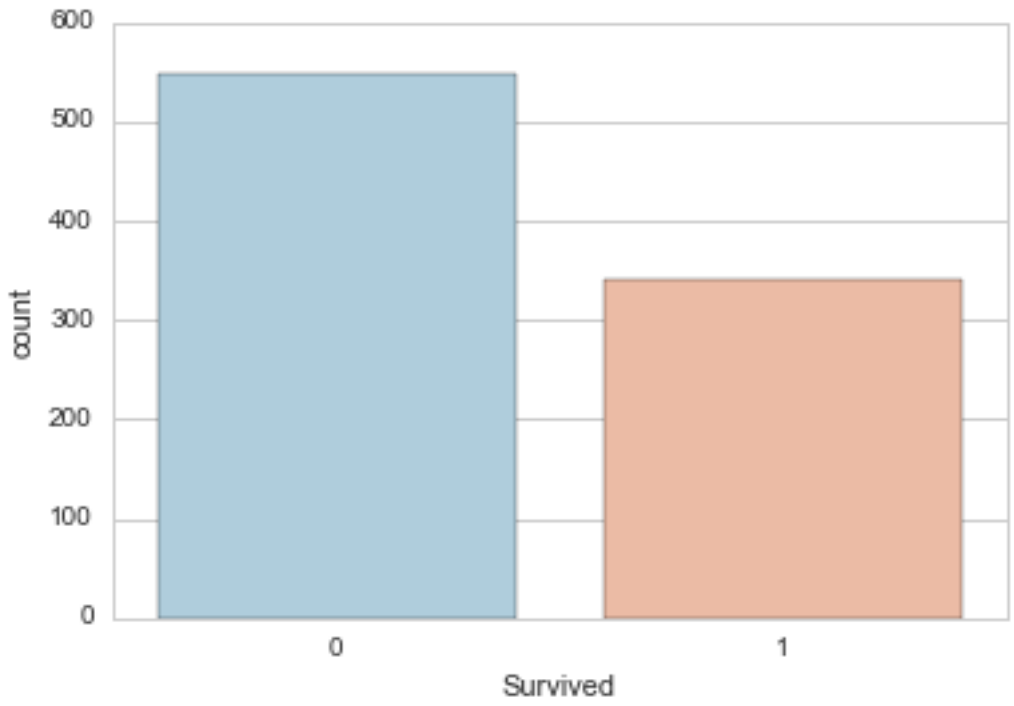
Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xab04e322e8>
```



2.2 Basic Graphs

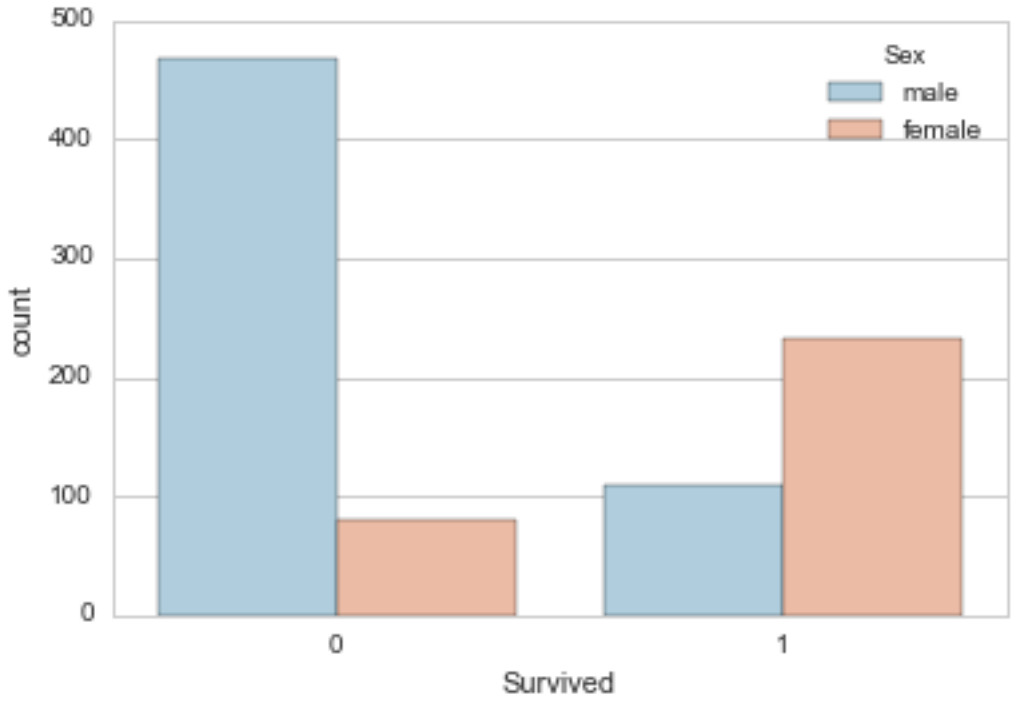
```
In [23]: sns.set_style('whitegrid')
         sns.countplot(x='Survived',data=train,palette='RdBu_r')
```

```
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0xab05e24080>
```



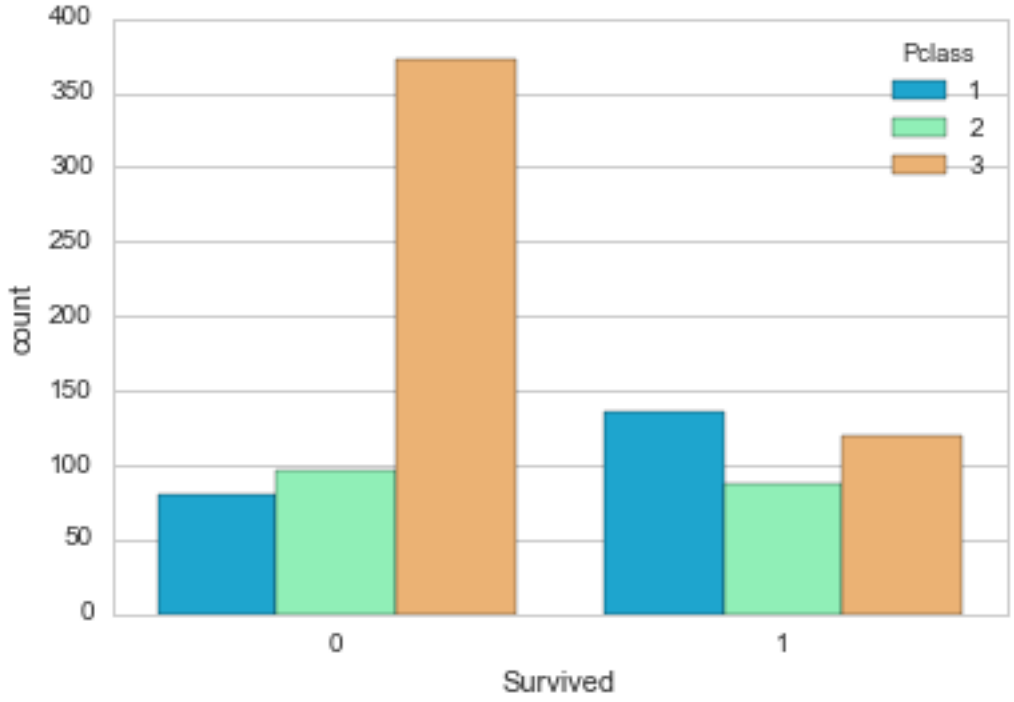
In [24]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Sex',data=train,palette='RdBu_r')
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0xab04e3ae80>



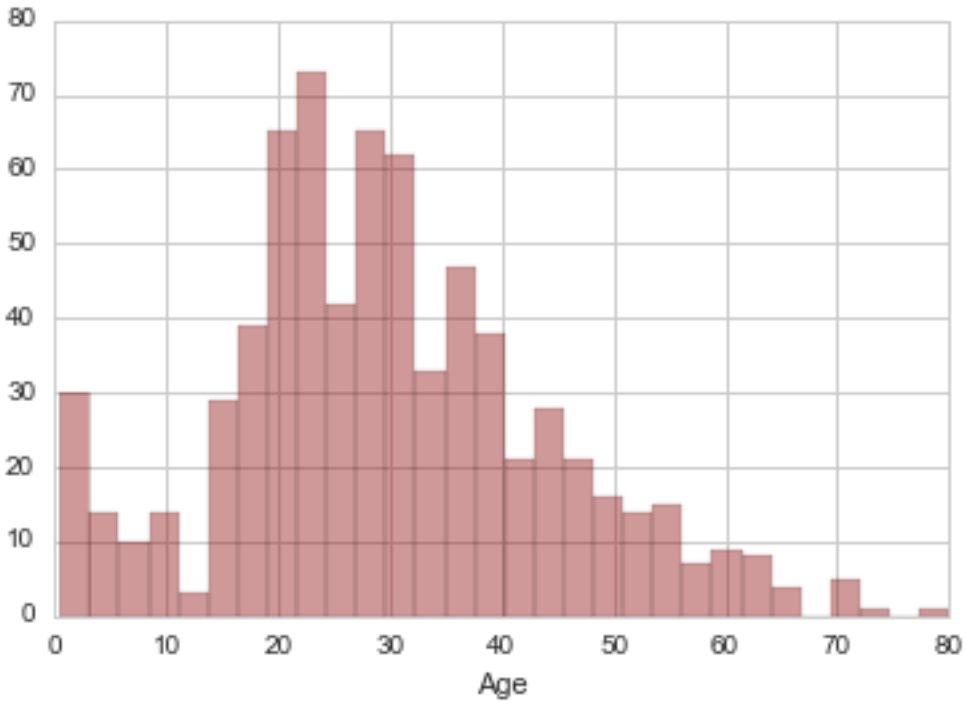
In [25]:
```
sns.set_style('whitegrid')
sns.countplot(x='Survived',hue='Pclass',data=train,palette='rainbow')
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0xab05ed8a90>



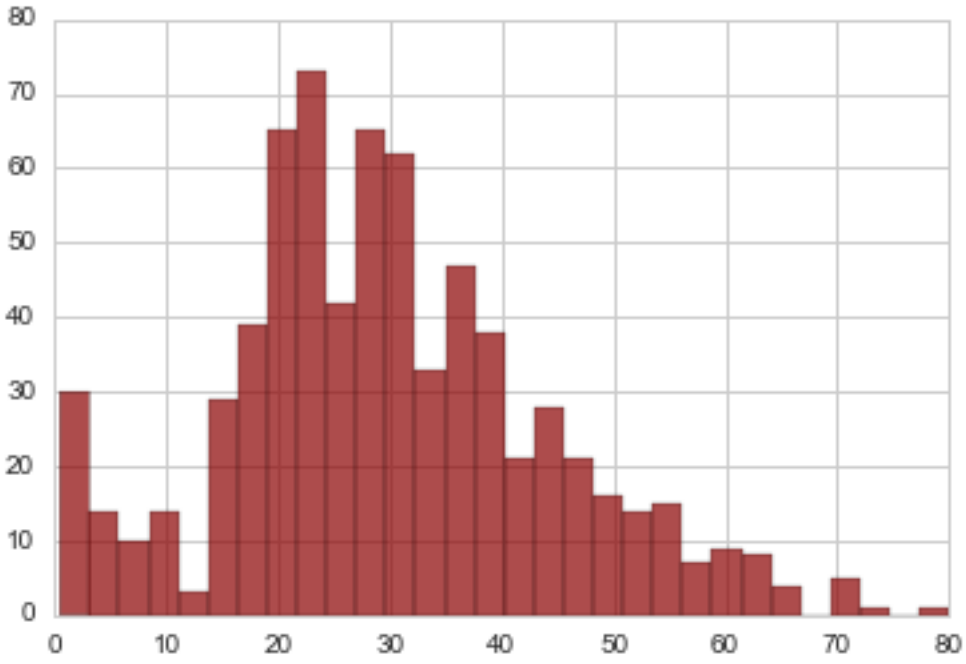
In [26]:
```
sns.distplot(train['Age'].dropna(),kde=False,color='darkred',bins=30)
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0xab05f15f98>



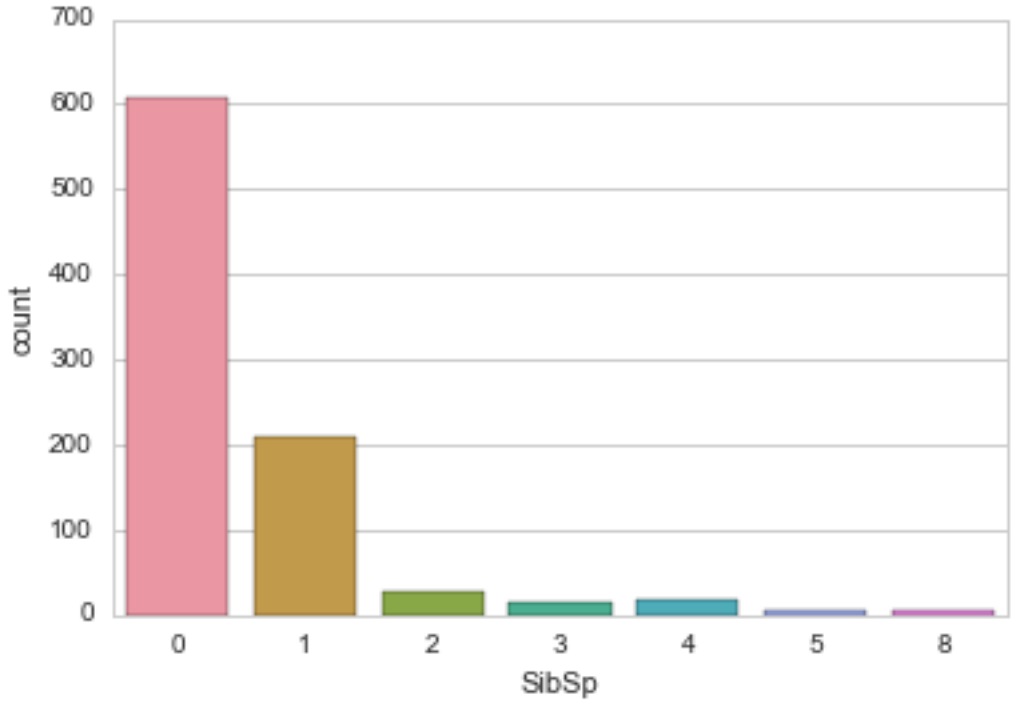
In [27]:
```
train['Age'].hist(bins=30,color='darkred',alpha=0.7)
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xab05f36128>
```



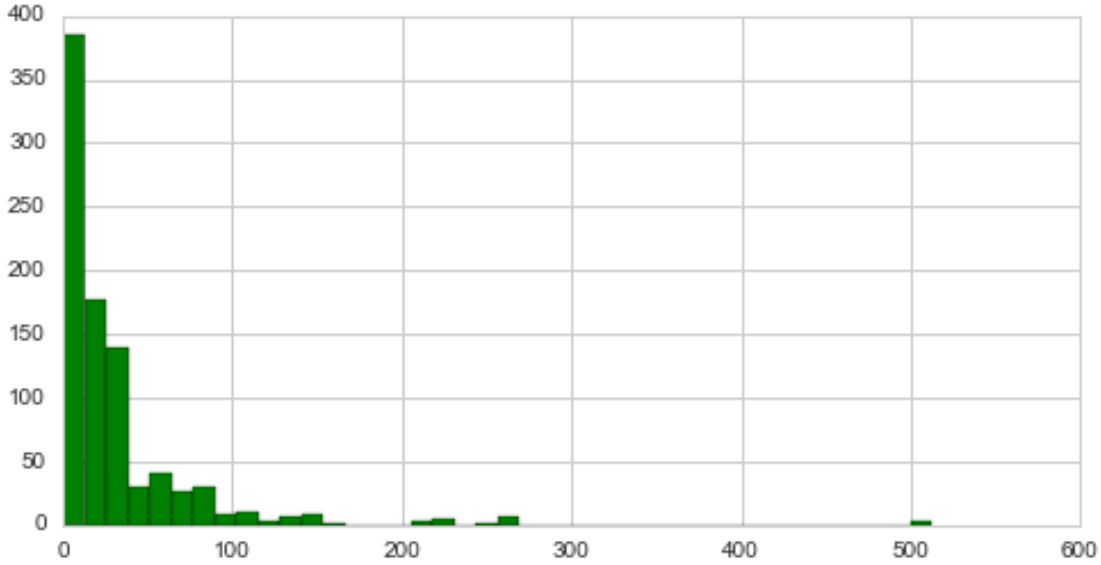
In [28]:
```
sns.countplot(x='SibSp',data=train)
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xab061442b0>
```



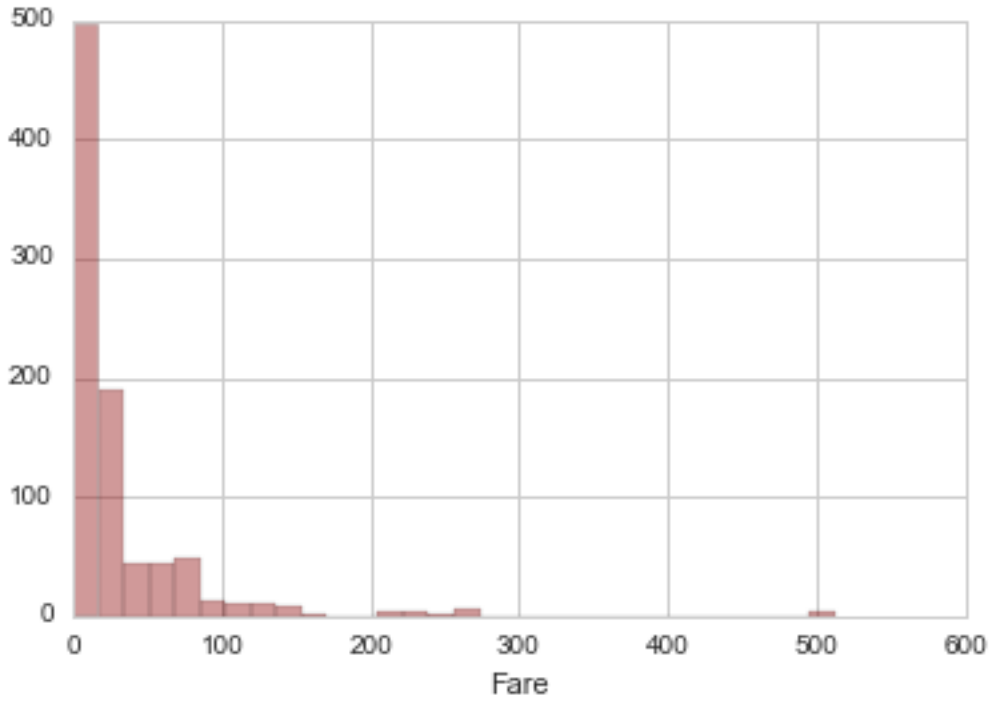
In [29]:
```
train['Fare'].hist(color='green',bins=40,figsize=(8,4))
```

Out[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xab061707b8>
```



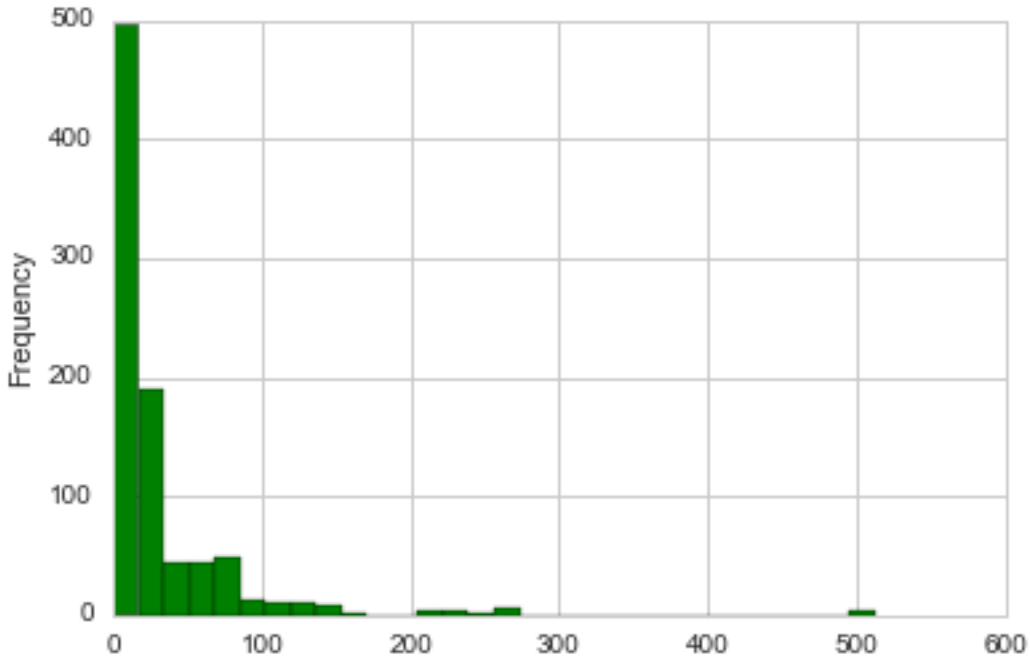
In [30]:
```
sns.distplot(train['Fare'].dropna(),kde=False,color='darkred',bins=30)
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xab061ccf98>
```



In [33]:
```
train['Fare'].plot(kind='hist',bins=30,color='green')
```

Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xab06352358>
```

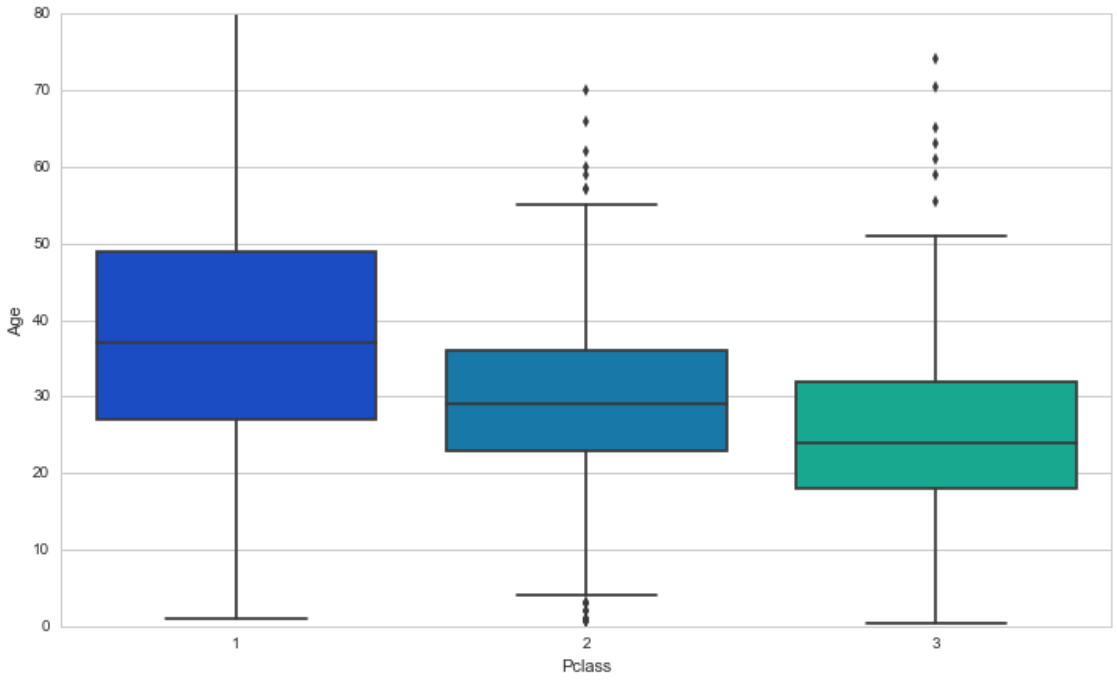


2.3 Data Cleaning

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

```
Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0xab063eacf8>
```



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 [35]: 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
```

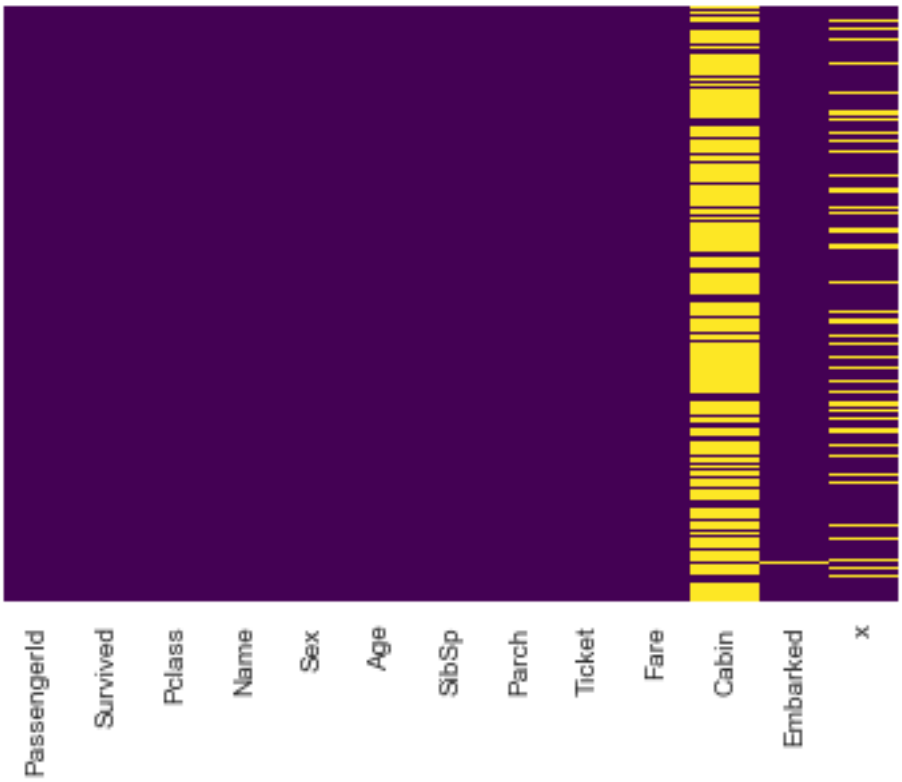
Now apply that function!

```
In [36]: train['Age'] = train[['Age','Pclass']].apply(impute_age,axis=1)
```

Now let's check that heat map again!

```
In [37]: sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0xab064f3080>
```



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

```
In [38]: train.drop('Cabin',axis=1,inplace=True)
```

```
In [39]: train.head()
```

```
Out[39]:    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         x
0      0         A/5 21171   7.2500        S  14.666667
1      0          PC 17599  71.2833        C  76.000000
2      0  STON/O2. 3101282   7.9250        S  17.333333
3      0            113803  53.1000        S  70.000000
4      0            373450   8.0500        S  23.333333
```

```
In [40]: train.dropna(inplace=True)
```

2.4 Converting Categorical Features

2.4.1 Creating dummy variables

```
In [41]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    712 non-null int64
Survived       712 non-null int64
Pclass         712 non-null int64
Name           712 non-null object
Sex            712 non-null object
Age            712 non-null float64
SibSp          712 non-null int64
Parch          712 non-null int64
Ticket         712 non-null object
Fare           712 non-null float64
Embarked       712 non-null object
x              712 non-null float64
dtypes: float64(3), int64(5), object(4)
memory usage: 72.3+ KB
```

```
In [42]: sex = pd.get_dummies(train['Sex'],drop_first=True)
         embark = pd.get_dummies(train['Embarked'],drop_first=True)

In [43]: train.drop(['Sex','Embarked','Name','Ticket'],axis=1,inplace=True)

In [44]: train = pd.concat([train,sex,embark],axis=1)

In [45]: train.head()
```

```
Out[45]:    PassengerId  Survived  Pclass   Age  SibSp  Parch     Fare          x  \
         0            1         0       3  22.0      1      0   7.2500  14.666667
         1            2         1       1  38.0      1      0  71.2833  76.000000
         2            3         1       3  26.0      0      0   7.9250  17.333333
         3            4         1       1  35.0      1      0  53.1000  70.000000
         4            5         0       3  35.0      0      0   8.0500  23.333333
```

```
   male    Q    S
0   1.0  0.0  1.0
1   0.0  0.0  0.0
2   0.0  0.0  1.0
3   0.0  0.0  1.0
4   1.0  0.0  1.0
```

3 Building a Logistic Regression model

3.1 Train Test Split

```
In [46]: from sklearn.cross_validation import train_test_split

In [47]: 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)
```

3.2 Training and Predicting

```
In [48]: from sklearn.linear_model import LogisticRegression

In [49]: logmodel = LogisticRegression()
         logmodel.fit(X_train,y_train)

Out[49]: 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=0.0001,
                   verbose=0, warm_start=False)

In [50]: predictions = logmodel.predict(X_test)
```

Let's move on to evaluate our model!

3.3 Evaluation

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

```
In [51]: from sklearn.metrics import classification_report

In [52]: print(classification_report(y_test,predictions))

precision    recall  f1-score   support

          0       0.80      0.84      0.82       128
          1       0.74      0.69      0.71        86

avg / total       0.77      0.78      0.77       214

In [54]: from sklearn.metrics import confusion_matrix

In [55]: confusion_matrix(y_test,predictions)

Out[55]: array([[107,  21],
                [ 27,  59]])
```