Logistic Regression with Python

we will be working with the Titanic Data Set from Kaggle (https://www.kaggle.com/c/titanic).

We'll be trying to predict a classification- survival or deceased.

Import Libraries

Let's import some libraries to get started!

In [77]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

The Data

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

In [78]:
```
train=pd.read_csv('titanic_train.csv')
```

In [79]:
```
train.head()
```

Out[79]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Nal
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Nal
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Nal

Exploratory Data Analysis

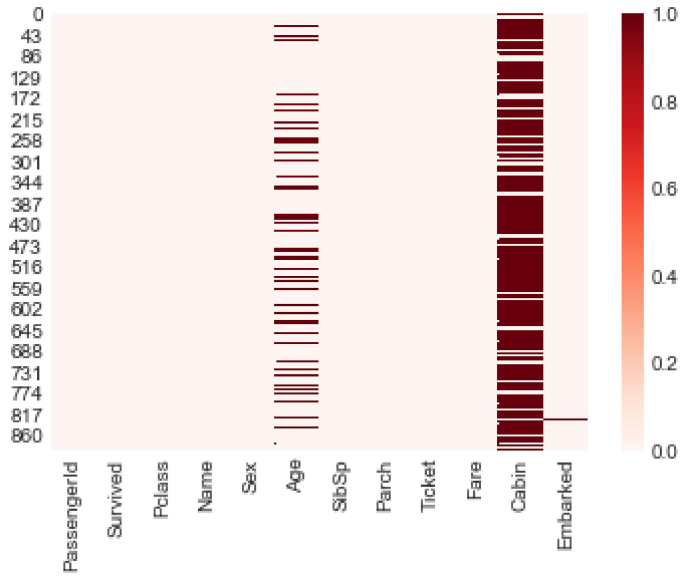
Let's begin some exploratory data analysis! We'll start by checking out missing data!

Missing Data

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

In [56]:
```
sns.heatmap(data=train.isnull(),cmap='Reds')
```

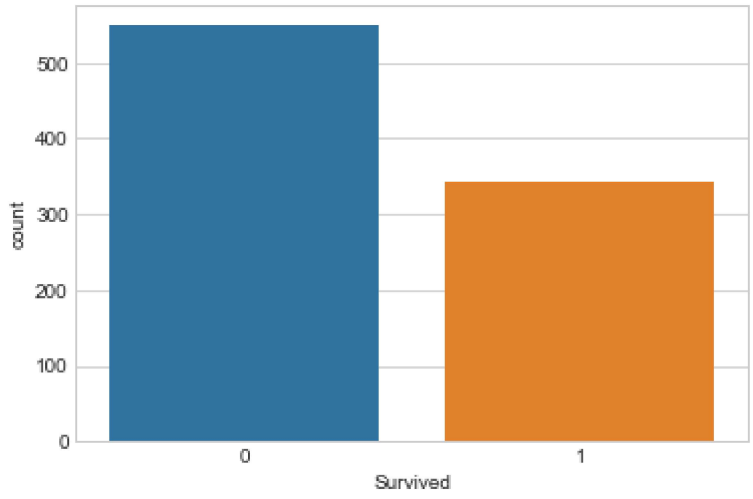
Out[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fd6523e400>
```



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"

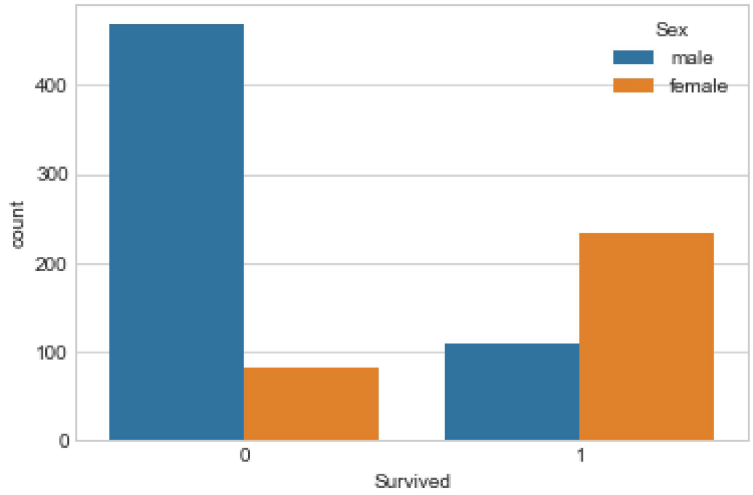
In [57]:
```
sns.countplot(x='Survived',data=train)
# there is 0= not servived and 1= servived
```

Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fd640355f8>
```



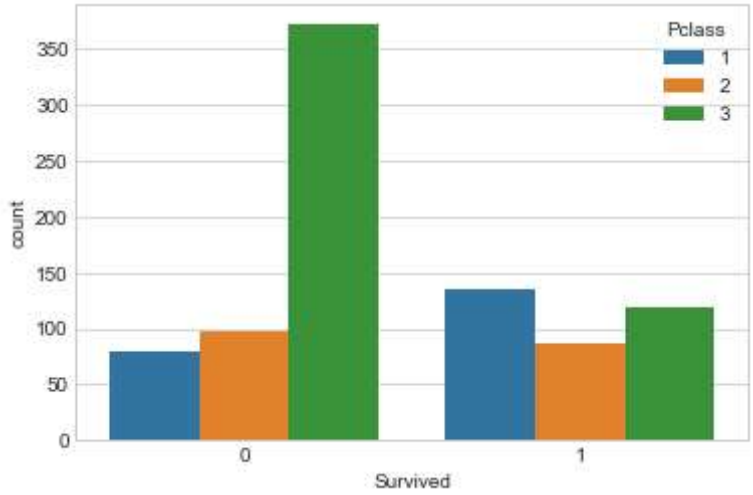
In [58]:
```
sns.countplot(data=train,x='Survived',hue='Sex')
#based on sex servived
#we can see the trend of survived is just opposite of not servived
```

Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x1fd652c1518>



In [59]:
```
sns.set_style('whitegrid')
sns.countplot(data=train,x='Survived',hue='Pclass')
# so 3rd class passenger has mostly not survived
#where 1st class passengers were mostly safe
```

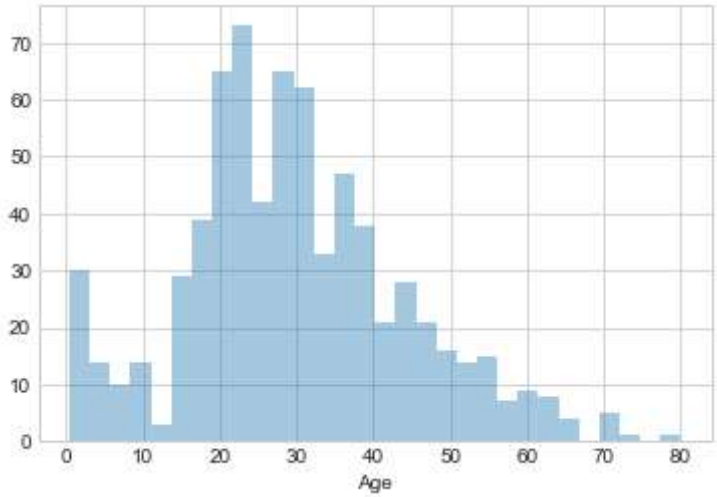
Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x1fd652eccf8>



In [60]:
```
sns.distplot(train['Age'].dropna(),kde=False,bins=30)
```

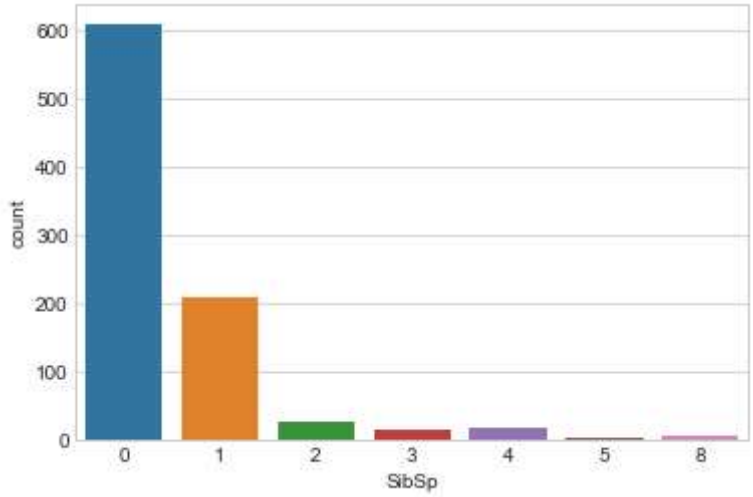
```
C:\Users\INDRAJEET YADAV\Anaconda3\envs\chatbot\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[60]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fd652f0e10>
```



In [61]:
```
sns.countplot(data=train,x='SibSp')
# most of onboard passengers didnt have relative/spouse/childeren
```

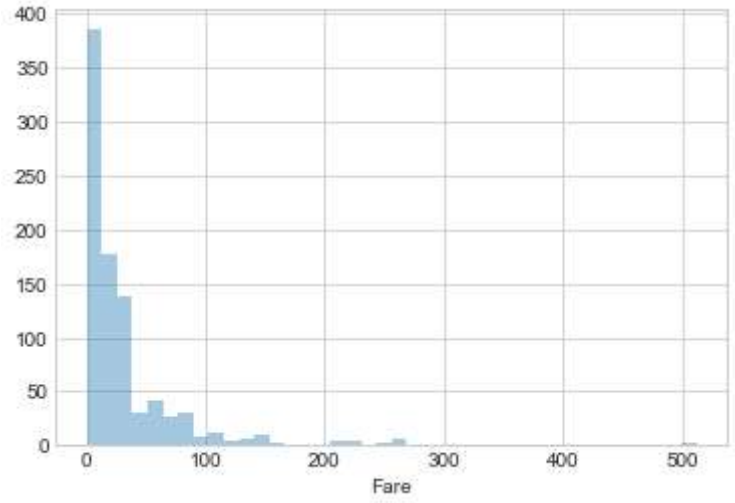
Out[61]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fd653a0a20>
```



In [62]:
```
sns.distplot(train['Fare'],bins=40,kde=False)
```

```
C:\Users\INDRAJEET YADAV\Anaconda3\envs\chatbot\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[62]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fd653deeb8>
```

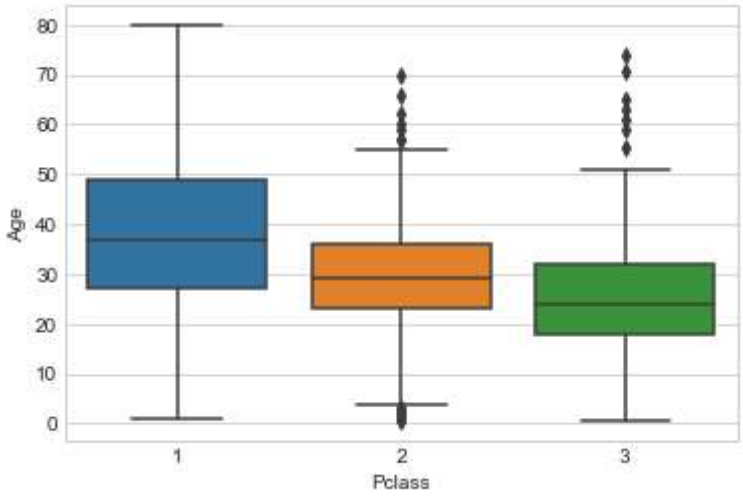


Data Cleaning

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 [63]:
```
sns.boxplot(y='Age',x='Pclass',data=train)
#we can fill age data using below average of class wise age
```

Out[63]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fd65453320>
```



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 [64]:
```
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
```

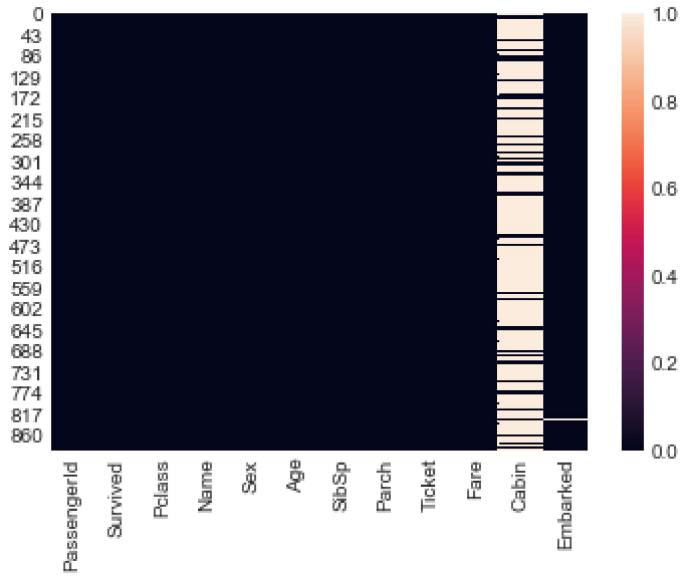
Now apply that function!

In [80]:
```
train['Age']=train[['Age','Pclass']].apply(impute_age,axis=1)
```

Now let's check that heat map again!

In [81]:
```
sns.heatmap(data=train.isnull(),cbar=True )
```

Out[81]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fd6556c400>
```



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

In [82]:
```
train.drop('Cabin', inplace=True, axis=1)
```

In [83]:
```
train.head()
```

Out[83]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Emb
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 [84]:
```
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 [85]:
```
train.head()
```

Out[85]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Emb
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 [87]:
```
sex=pd.get_dummies(train['Sex'],drop_first=True)

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

In [88]:
```
train.drop(['Sex','Embarked','Name','Ticket'],axis=1,inplace=True)
```

In [89]:
```
train=pd.concat([train,sex,embark],axis=1)
train.drop('PassengerId',axis=1,inplace=True)
```

In [90]:
```
train.head()
```

Out[90]:

	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

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 [100]:
```
X=train.drop('Survived',axis=1)

y=train['Survived']
```

In [107]:
```
from sklearn.cross_validation import train_test_split
```

Training and Predicting

In [108]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s
```

In [115]:
```
from sklearn.linear_model import LogisticRegression
```

In [116]:
```
logmodel=LogisticRegression()
```

In [117]:
```
logmodel.fit(X_train,y_train)
```

Out[117]:
```
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 [119]:
```
prediction=logmodel.predict(X_test)
```

Let's move on to evaluate our model!

Evaluation

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

In [120]:
```
from sklearn.metrics import classification_report
```

In [121]:
```
print(classification_report(y_test,prediction))
```

```
             precision    recall  f1-score   support

          0       0.80      0.91      0.85       163
          1       0.82      0.65      0.73       104

avg / total       0.81      0.81      0.80       267
```

In [122]:
```
from sklearn.metrics import confusion_matrix
```

In [123]:
```
confusion_matrix(y_test,prediction)
```

Out[123]:
```
array([[148,  15],
       [ 36,  68]], dtype=int64)
```