Logistic Regression

Ajeet K. Jain

Using Titanic Data Set

The wreck of the RMS Titanic was one of the worst shipwrecks in history, and is certainly the most well-known. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding With an iceberg, killing 1502 out of 2224 passengers and crew. **This sensational tragedy shocked the international community and lead to better safety regulations for ships.**

One of the reasons that the shipwreck lead to such loss of life is that were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, like women, children, and the upper-class.

In this exercise , we are trying to do is the analysis of " what sorts of people were likely to survive " . In particular, we are trying to ask to apply the tools of Machine Learning to predict which passengers survived the tragedy.

Data Set consists of 1310 rows and 1 2 columns as

```
In [2]: %matplotlib inline
rcParams['figure.figsize'] = 10, 8
sb.set_style('whitegrid')
```

```
In [3]: url = 'https://raw.githubusercontent.com/BigDataGal/Python-for-Data-Science/master/titanic-train.csv'
titanic = pd.read_csv(url)
titanic.columns = ['PassengerId','Survived','Pclass','Name','Sex','Age','SibSp','Parch','Ticket','Fare','Cabin','Embarked']
titanic.head()
```

Out[3]:

	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

In [ ]:

Activate Windows
Go to Settings to activate Windo

titanic3 [Compatibility Mode] - Microsoft Excel

Home | Insert | Page Layout | Formulas | Data | Review | View

G1 | fx | parch

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	ho
2	1	1	Allen, Miss. Elisabeth Walton	female	29	0	0	24160	211.3375	B5	S	2		St Louis, MO
3	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11		Montreal, PQ / Chesterv
4	1	0	Allison, Miss. Helen Loraine	female	2	1	2	113781	151.5500	C22 C26	S			Montreal, PQ / Chesterv
5	1	0	Allison, Mr. Hudson Joshua Creighton	male	30	1	2	113781	151.5500	C22 C26	S		135	Montreal, PQ / Chesterv
6	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25	1	2	113781	151.5500	C22 C26	S			Montreal, PQ / Chesterv
7	1	1	Anderson, Mr. Harry	male	48	0	0	19952	26.5500	E12	S	3		New York, NY
8	1	1	Andrews, Miss. Kornelia Theodosia	female	63	1	0	13502	77.9583	D7	S	10		Hudson, NY
9	1	0	Andrews, Mr. Thomas Jr	male	39	0	0	112050	0.0000	A36	S			Belfast, NI
10	1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53	2	0	11769	51.4792	C101	S	D		Bayside, Queens, NY
11	1	0	Artagaveytia, Mr. Ramon	male	71	0	0	PC 17609	49.5042		C		22	Montevideo, Uruguay
12	1	0	Astor, Col. John Jacob	male	47	1	0	PC 17757	227.5250	C62 C64	C		124	New York, NY
13	1	1	Astor, Mrs. John Jacob (Madeleine Talmadge Force)	female	18	1	0	PC 17757	227.5250	C62 C64	C	4		New York, NY
14	1	1	Aubart, Mme. Leontine Pauline	female	24	0	0	PC 17477	69.3000	B35	C	9		Paris, France
15	1	1	Barber, Miss. Ellen "Nellie"	female	26	0	0	19877	78.8500		S	6		
16	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80	0	0	27042	30.0000	A23	S	B		Hessle, Yorks
17	1	0	Baumann, Mr. John D	male		0	0	PC 17318	25.9250		S			New York, NY
18	1	0	Baxter, Mr. Quigg Edmond	male	24	0	1	PC 17558	247.5208	B58 B60	C			Montreal, PQ
19	1	1	Baxter, Mrs. James (Helene DeLaudeniere Chaput)	female	50	0	1	PC 17558	247.5208	B58 B60	C	6		Montreal, PQ
20	1	1	Bazzani, Miss. Albina	female	32	0	0	11813	76.2917	D15	C	8		
21	1	0	Beattie, Mr. Thomson	male	36	0	0	13050	75.2417	C6	C	A		Winnipeg, MN
22	1	1	Beckwith, Mr. Richard Leonard	male	37	1	1	11751	52.5542	D35	S	5		New York, NY
23	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47	1	1	11751	52.5542	D35	S	5		New York, NY
24	1	1	Behr, Mr. Karl Howell	male	26	0	0	111369	30.0000	C148	C	5		New York, NY
25	1	1	Bidois, Miss. Rosalie	female	42	0	0	PC 17757	227.5250		C	4		
26	1	1	Bird, Miss. Ellen	female	29	0	0	PC 17483	221.7792	C97	S	8		
27	1	0	Birnbaum, Mr. Jakob	male	25	0	0	13905	26.0000		C		148	San Francisco, CA
28	1	1	Bishop, Mr. Dickinson H	male	25	1	0	11967	91.0792	B49	C	7		Dowagiac, MI
29	1	1	Bishop, Mrs. Dickinson H (Helen Walton)	female	19	1	0	11967	91.0792	B49	C	7		Dowagiac, MI
30	1	1	Bissette, Miss. Amelia	female	35	0	0	PC 17760	135.6333	C99	S	8		
31	1	1	Bjornstrom-Steffansson, Mr. Mauritz Hakan	male	28	0	0	110564	26.5500	C52	S	D		Stockholm, Sweden / W
32	1	0	Blackwell, Mr. Stephen Weart	male	45	0	0	113784	35.5000	T	S			Trenton, NJ
33	1	1	Blank, Mr. Henry	male	40	0	0	112277	31.0000	A31	C	7		Glen Ridge, NJ
34	1	1	Bonnell, Miss. Caroline	female	30	0	0	36928	164.8667	C7	S	8		Youngstown, OH
35	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.5500	C103	S	8		Birkdale, England Clevel
36	1	0	Borebank, Mr. John James	male	42	0	0	110489	26.5500	D22	S			London / Winnipeg, MB
37	1	1	Bowen, Miss. Grace Scott	female	45	0	0	PC 17608	262.3750		C	4		Cooperstown, NY
38	1	1	Bowerman, Miss. Elsie Edith	female	22	0	1	113505	55.0000	E33	S	6		St Leonards-on-Sea, En
39	1	1	Bradley, Mr. George ("George Arthur Brayton")	male		0	0	111427	26.5500		S	9		Los Angeles, CA
40	1	0	Brady, Mr. John Bertram	male	41	0	0	113054	30.5000	A21	S			Pomeroy, WA
41	1	0	Brandeis, Mr. Emil	male	48	0	0	PC 17591	50.4958	B10	C		208	Omaha, NE
42	1	0	Brewe, Dr. Arthur Jackson	male		0	0	112379	39.6000		C			Philadelphia, PA
43	1	1	Brown, Mrs. James Joseph (Margaret Tobin)	female	44	0	0	PC 17610	27.7208	B4	C	6		Denver, CO
44	1	1	Brown, Mrs. John Murray (Caroline Lane Lamson)	female	59	2	0	11769	51.4792	C101	S	D		Belmont, MA
45	1	1	Bucknell, Mrs. William Robert (Emma Eliza Ward)	female	60	0	0	11813	76.2917	D15	C	8		Philadelphia, PA

Activate Windows
Go to Settings to activate Windows.

VARIABLE DESCRIPTIONS

Survived - Survival (0 = No; 1 = Yes)

Pclass - Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)

Name - Name

Sex - Sex

Age - Age

SibSp - Number of Siblings/Spouses Aboard

Parch - Number of Parents/Children Aboard

Ticket - Ticket Number

Fare - Passenger Fare (British pound)

Cabin - Cabin

Embarked - Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

New package we need to know

Seaborn: statistical data visualization.

Seaborn is a **Python** visualization library based on **matplotlib**.

It provides a high-level interface for drawing attractive statistical graphics.

Seaborn is complimentary to **Matplotlib .**
Seaborn extends **Matplotlib** and can address two biggest things to (frustrations of working with Matplotlib) .
If matplotlib "tries to make easy things easy and hard things possible",
seaborn tries to make a well-defined set of hard things easy too."

One of these hard things has to do with the default Matplotlib parameters.
Seaborn works with different parameters, which undoubtedly speaks to those users that don't use the default looks of the Matplotlib plots.

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import sklearn


from pandas import Series, DataFrame
from pylab import rcParams
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report
```

```
C:\Users\Kmit\Anaconda1\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This module was deprecated in ver
sion 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that
the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

Data Set consists of 1310 rows and 1 2 columns as

```
In [2]: %matplotlib inline
rcParams['figure.figsize'] = 10, 8
sb.set_style('whitegrid')
```

```
In [3]: url = 'https://raw.githubusercontent.com/BigDataGal/Python-for-Data-Science/master/titanic-train.csv'
titanic = pd.read_csv(url)
titanic.columns = ['PassengerId','Survived','Pclass','Name','Sex','Age','SibSp','Parch','Ticket','Fare','Cabin','Embarked']
titanic.head()
```

Out[3]:

	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

```
In [ ]:
```

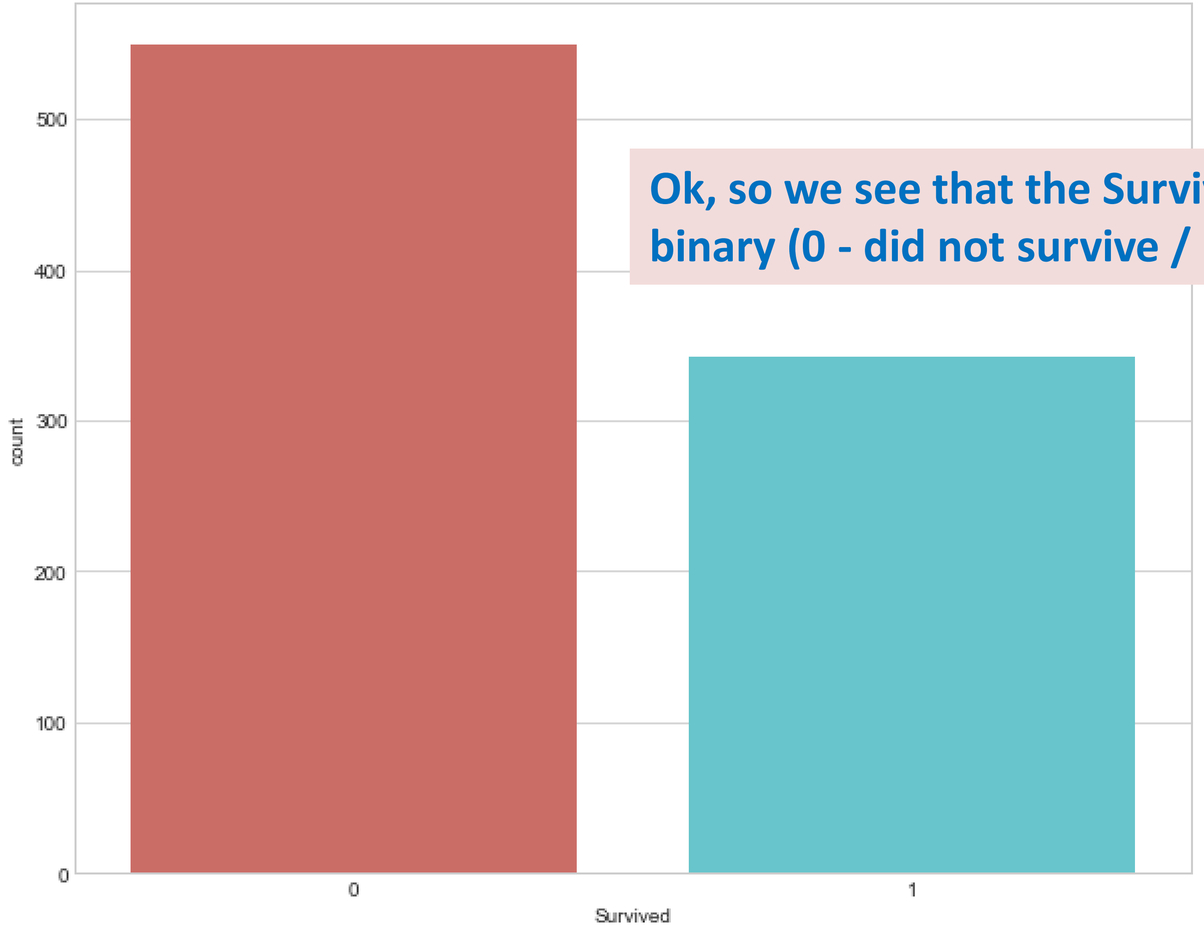
Activate Windows
Go to Settings to activate Windo

Checking that your target variable is binary

Since we are building a model to predict survival of passangers from the Titanic, our target is going to be "Survived" variable from the titanic dataframe. To make sure that it's a binary variable, let's use **Seaborn's countplot()** function.

```
In [4]: sb.countplot(x='Survived',data=titanic, palette='hls')
```

```
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x1ccb909d550>
```



Ok, so we see that the Survived variable is binary (0 - did not survive / 1 - survived)

Checking for missing values

It's easy to check for missing values by calling the **isnull()** method, and the **sum()** method off of that, to return a tally of all the **True** values that are returned by the **isnull()** method.

```
In [5]: titanic.isnull().sum()

Out[5]: PassengerId      0
        Survived         0
        Pclass           0
        Name             0
        Sex              0
        Age            177
        SibSp            0
        Parch            0
        Ticket           0
        Fare             0
        Cabin          687
        Embarked         2
        dtype: int64

In [ ]:
```

Well, how many records are there in the data frame anyway?

In [6]:
```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

In [ ]:

Ok, so there are only 891 rows in the titanic data frame. Cabin is almost all missing values, so we can drop that variable completely, but what about age? Age seems like a relevant predictor for survival right? We'd want to keep the variables, but it has 177 missing values. We are going to need to find a way to approximate for those missing values!

Taking care of missing values

Dropping missing values

So let's just go ahead and drop all the variables that aren't relevant for predicting survival. We should at least keep the following:

- Survived - This variable is obviously relevant.
- Pclass - Does a passenger's class on the boat affect their survivability?
- Sex - Could a passenger's gender impact their survival rate?
- Age - Does a person's age impact their survival rate?
- SibSp - Does the number of relatives on the boat (that are siblings or a spouse) affect a person survivability? Probability
- Parch - Does the number of relatives on the boat (that are children or parents) affect a person survivability? Probability
- Fare - Does the fare a person paid effect his survi vability? Maybe - let's keep it.
- Embarked - Does a person's point of embarkation matter? It depends on how the boat was filled... Let's keep it.

What about a person's name, ticket number, and passenger ID number? They're irrelavant for predicting survivability. And as we recall, the cabin variable is almost all missing values, so we can just drop all of these.

In [7]:
```
titanic_data = titanic.drop(['PassengerId','Name','Ticket','Cabin'], 1)
titanic_data.head()
```

Out[7]:

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

In [ ]:

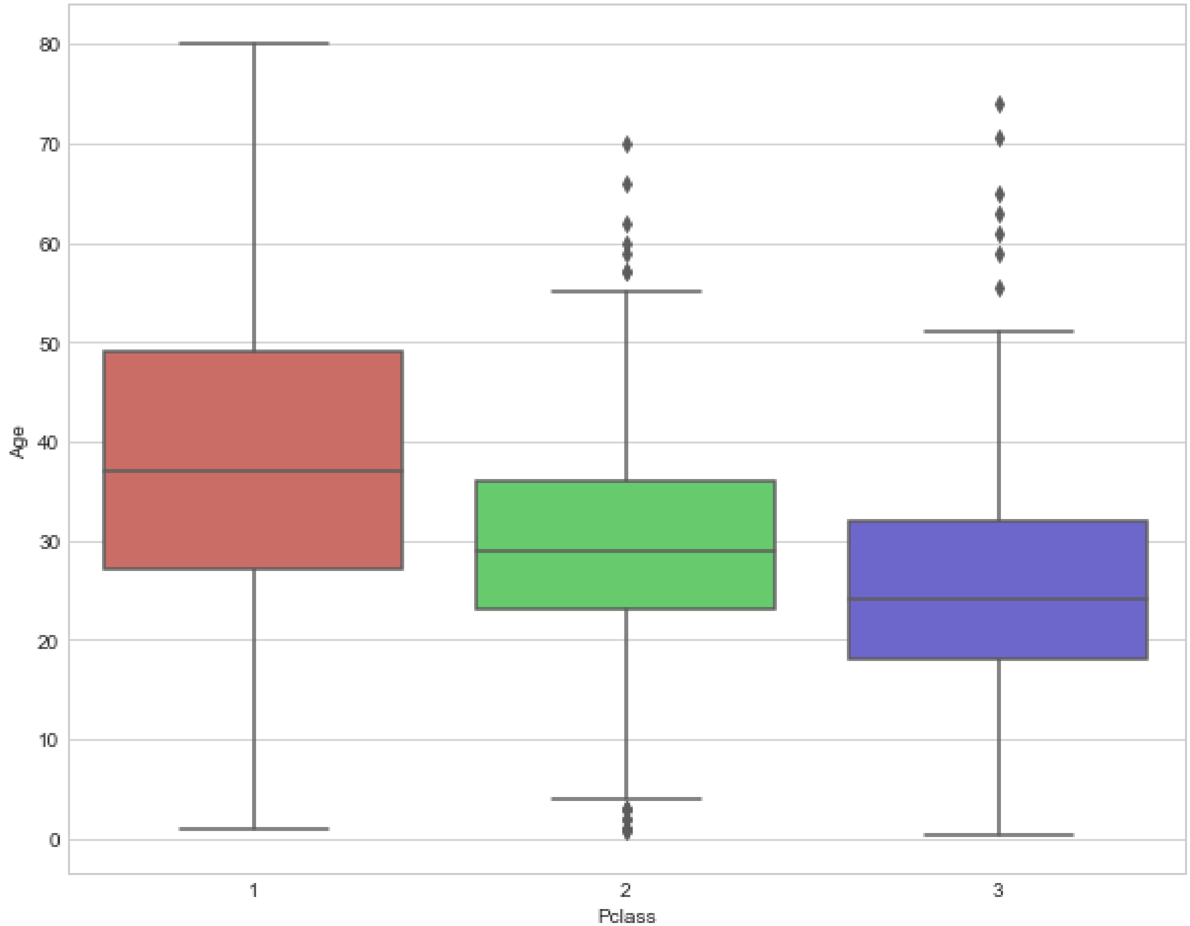
Now we have the **dataframe** reduced down to only relevant variables, but now we need to deal with the missing values in the age variable.

Imputing missing values

Let's look at how passenger age is related to their class as a passenger on the boat.

In [8]:
```
sb.boxplot(x='Pclass', y='Age', data=titanic_data, palette='hls')
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1ccba307588>



In [ ]:

Pclass

In [9]:
```
titanic_data.head()
```

Out[9]:

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

In [ ]:

Speaking roughly, we could say that the younger a passenger is, the more likely it is for them to be in 3rd class. The older a passenger is, the more likely it is for them to be in 1st class. So there is a loose relationship between these variables. So, let's write a function that approximates a passengers age, based on their class.
From the box plot, it looks like the average age of 1st class passengers is about 37, 2nd class passengers is 29, and 3rd class pasengers is 24.

So let's write a function that finds each null value in the Age variable, and for each null, checks the value of the Pclass and assigns an age value according to the average age of passengers in that class.

In [10]:
```
def age_approx(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
```

When we apply the function and check again for null values, we see that there are no more null values in the age variable.

In [11]:
```
titanic_data['Age'] = titanic_data[['Age', 'Pclass']].apply(age_approx, axis=1)
titanic_data.isnull().sum()
```

Out[11]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    2
dtype: int64
```

There are 2 null values in the embarked variable. We can drop those 2 records without loosing too much important information from our dataset, so we will do that.

In [ ]:

In [11]:
```
titanic_data['Age'] = titanic_data[['Age', 'Pclass']].apply(age_approx, axis=1)
titanic_data.isnull().sum()
```

Out[11]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    2
dtype: int64
```

In [12]:
```
titanic_data.dropna(inplace=True)
titanic_data.isnull().sum()
```

Out[12]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

There are 2 null values in the embarked variable. We can drop those 2 records without loosing too much important information from our dataset, so we will do that.

In [ ]:

Converting categorical variables to a dummy indicators

The next thing we need to do is reformat our variables so that they work with the model. Specifically, we need to reformat the Sex and Embarked variables into numeric variables.

In [13]:
```
gender = pd.get_dummies(titanic_data['Sex'],drop_first=True)
gender.head()
```

Out[13]:

	male
0	1
1	0
2	0
3	0
4	1

In [14]:
```
embark_location = pd.get_dummies(titanic_data['Embarked'],drop_first=True)
embark_location.head()
```

Out[14]:

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

In [15]:
```
titanic_data.head()
```

Out[15]:

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

In [16]:
```
titanic_data.drop(['Sex', 'Embarked'],axis=1,inplace=True)
titanic_data.head()
```

Out[16]:

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

In [ ]:

Out[16]:

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

In [17]:

```
titanic_dmy = pd.concat([titanic_data,gender,embark_location],axis=1)
titanic_dmy.head()
```

Out[17]:

	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

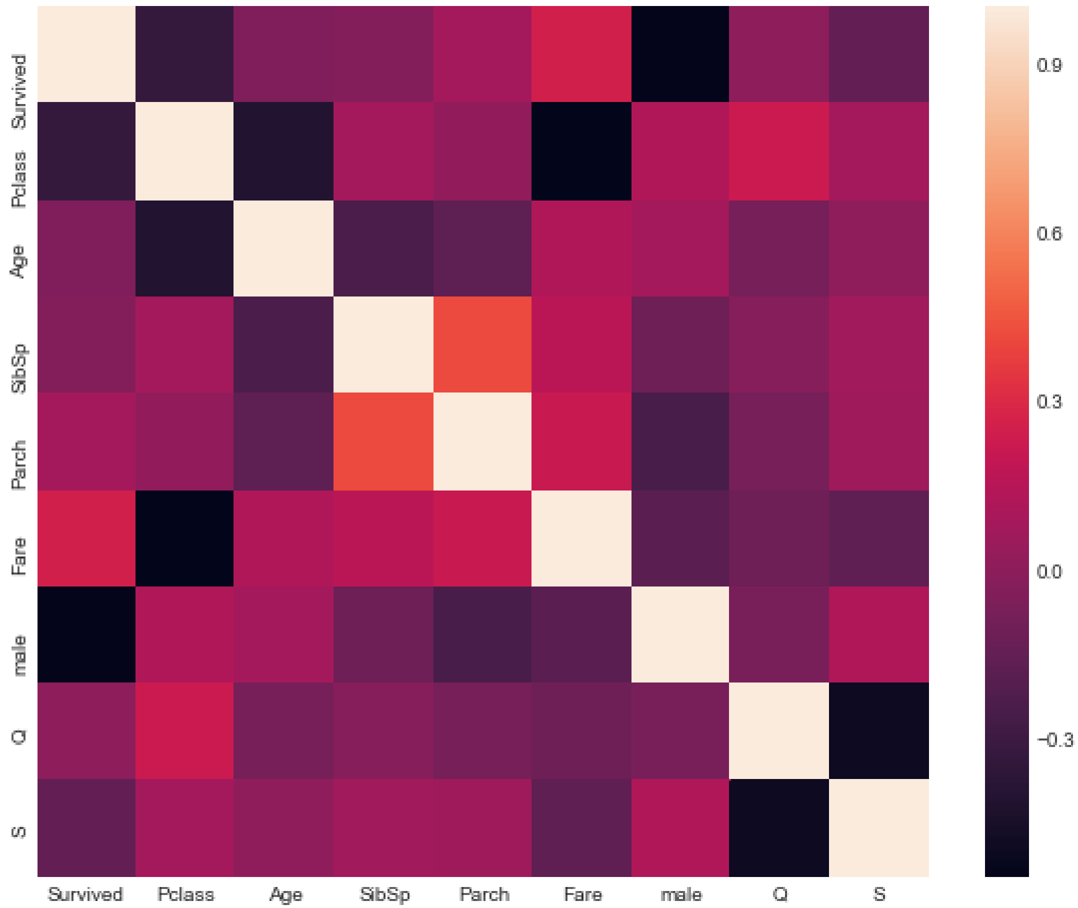
In [ ]:

Now we have a dataset with all the variables in the correct format!

Checking for independence between features

In [18]:
```
sb.heatmap(titanic_dmy.corr())
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ccbab11be0>
```



In [ ]:

Fare and Pclass are not independent of each other, so we are going to drop these.

```
In [19]: titanic_dmy.drop(['Fare', 'Pclass'],axis=1,inplace=True)
         titanic_dmy.head()
```

Out[19]:

	Survived	Age	SibSp	Parch	male	Q	S
0	0	22.0	1	0	1	0	1
1	1	38.0	1	0	0	0	0
2	1	26.0	0	0	0	0	1
3	1	35.0	1	0	0	0	1
4	0	35.0	0	0	1	0	1

```
In [ ]:
```

Checking that your dataset size is sufficient

We have 6 predictive features that remain. The rule of thumb is 50 records per feature... so we need to have at least 300 records in this dataset. Let's check again.

```
In [20]: titanic_dmy.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 7 columns):
Survived    889 non-null int64
Age         889 non-null float64
SibSp       889 non-null int64
Parch       889 non-null int64
male        889 non-null uint8
Q           889 non-null uint8
S           889 non-null uint8
dtypes: float64(1), int64(3), uint8(3)
memory usage: 37.3 KB

In [ ]:
```

Ok, we have 889 records so we are fine.

```
In [21]: X = titanic_dmy.ix[:,(1,2,3,4,5,6)].values
         y = titanic_dmy.ix[:,0].values
```

```
C:\Users\Kmit\Anaconda1\lib\site-packages\ipykernel_launcher.py:1: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  """Entry point for launching an IPython kernel.
C:\Users\Kmit\Anaconda1\lib\site-packages\ipykernel_launcher.py:2: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
```

```
In [ ]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3, random_state=25)
```


```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3, random_state=25)
```

Deploying and evaluating the model

```
In [23]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3, random_state=25)
         LogReg = LogisticRegression()
         LogReg.fit(X_train, y_train)

Out[23]: 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 [25]: y_pred = LogReg.predict(X_test)
         from sklearn.metrics import confusion_matrix
         confusion_matrix = confusion_matrix(y_test, y_pred)
         confusion_matrix

Out[25]: array([[137,  27],
```

The results from the confusion matrix are telling us that 137 and 69 are the number of correct predictions. 34 and 27 are the number of incorrect predictions.

```
Out[23]: 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 [25]: y_pred = LogReg.predict(X_test)
         from sklearn.metrics import confusion_matrix
         confusion_matrix = confusion_matrix(y_test, y_pred)
         confusion_matrix
```

```
Out[25]: array([[137,  27],
                [ 34,  69]], dtype=int64)
```

```
In [26]: print(classification_report(y_test, y_pred))

                      precision    recall  f1-score   support

                   0       0.80      0.84      0.82       164
                   1       0.72      0.67      0.69       103

         avg / total       0.77      0.77      0.77       267
```

```
In [ ]:
```