Logistic Regression -mtcars dataset

In [1]:
```
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

import scipy
from scipy.stats import spearmanr

import matplotlib.pyplot as plt
from pylab import rcParams
import seaborn as sb

import sklearn
from sklearn.preprocessing import scale
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
from sklearn import metrics
from sklearn import preprocessing
```

```
C:\Users\by3001pm\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
\cross_validation.py:41: DeprecationWarning: This module was deprecated in ve
rsion 0.18 in favor of the model_selection module into which all the refactor
ed classes and functions are moved. Also note that the interface of the new C
V iterators are different from that of this module. This module will be remov
ed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [2]:
```
%matplotlib inline
rcParams['figure.figsize'] = 5, 4
sb.set_style('whitegrid')
```

Logistic regression on mtcars

In [7]:
```
data = 'mtcars.csv'
cars = pd.read_csv(data)
cars.head()
```

Out[7]:

	Unnamed: 0	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

In [9]:
```
cars.columns = ['car_names','mpg','cyl','disp', 'hp', 'drat', 'wt', 'qsec', 'v
s', 'am', 'gear', 'carb']
cars.head()
```

Out[9]:

	car_names	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

Drat describes the rear axle ratio and carb describes the number of carburetors a car has. Let's check the model assumptions these variable to predict am (whether the car has automatic transmission). Create a sub-set cars_data for this.

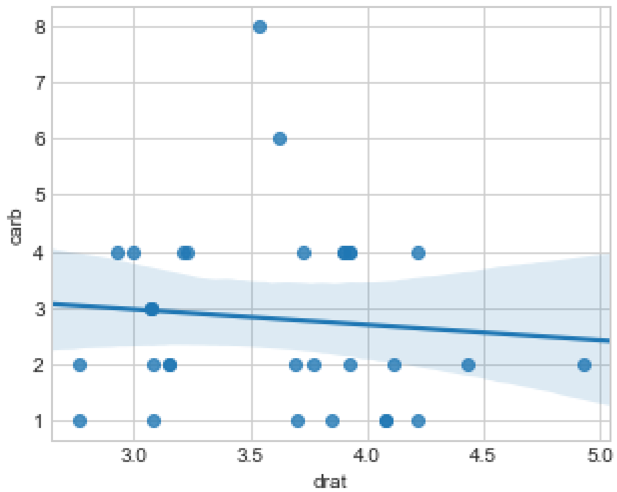
In [19]:
```
cars_data = cars.ix[:,(5,11)].values  #Use special indexer, .ix and we'll sele
ct column with index value five and 11
cars_data_names = ['drat','carb']  #create a list with the names for those col
umns

y = cars.ix[:,9].values #Target variable
```

Checking for independence between features

The first thing we're going to check is for independence between features. Are our predictor variables ordinal? Remember that an ordinal variable is a numeric variable that can be grouped into only a limited number of subcategories. We want to pick ordinal variables for logistic regression analysis. In order to do that, we use a scatter plot. Use Seaborn (sb.regplot). We'll set x equal to our drat variable, and y equal to our carb variable.

```
In [20]: sb.regplot(x='drat', y='carb', data=cars, scatter=True)
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x296b882bc88>



Neither of these variables take on an infinite number of positions, they only take on set positions.

Let's see if these features are independent of each other. First, let's isolate the variables into a variable called drat and a variable called carb, and we'll say drat is equal to cars and then just select the drat variable. Spearman's rank is used for checking relationships between two ordinal variables. You can learn more about Spearman's r (vs. Pearson's r) at http://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/regression/supporting-topics/basics/a-comparison-of-the-pearson-and-spearman-correlation-methods/ (http://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/regression/supporting-topics/basics/a-comparison-of-the-pearson-and-spearman-correlation-methods/)

```
In [24]: drat = cars['drat']
         carb = cars['carb']

         spearmanr_coefficient, p_value =  spearmanr(drat, carb)
         print (spearmanr_coefficient)
```

```
-0.12522293992
```

It shows almost no relationship. So, we can use them in our logistic regression model.

Checking for missing values

Next, we need to check the assumption that there are no missing values in the data set.

In [26]:
```
cars.isnull()
```

Out[26]:

	car_names	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	False	False	False	False	False	False	False	False	False	False	False	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	False	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	False	False
5	False	False	False	False	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False	False	False	False	False	False
11	False	False	False	False	False	False	False	False	False	False	False	False
12	False	False	False	False	False	False	False	False	False	False	False	False
13	False	False	False	False	False	False	False	False	False	False	False	False
14	False	False	False	False	False	False	False	False	False	False	False	False
15	False	False	False	False	False	False	False	False	False	False	False	False
16	False	False	False	False	False	False	False	False	False	False	False	False
17	False	False	False	False	False	False	False	False	False	False	False	False
18	False	False	False	False	False	False	False	False	False	False	False	False
19	False	False	False	False	False	False	False	False	False	False	False	False
20	False	False	False	False	False	False	False	False	False	False	False	False
21	False	False	False	False	False	False	False	False	False	False	False	False
22	False	False	False	False	False	False	False	False	False	False	False	False
23	False	False	False	False	False	False	False	False	False	False	False	False
24	False	False	False	False	False	False	False	False	False	False	False	False
25	False	False	False	False	False	False	False	False	False	False	False	False
26	False	False	False	False	False	False	False	False	False	False	False	False
27	False	False	False	False	False	False	False	False	False	False	False	False
28	False	False	False	False	False	False	False	False	False	False	False	False
29	False	False	False	False	False	False	False	False	False	False	False	False
30	False	False	False	False	False	False	False	False	False	False	False	False
31	False	False	False	False	False	False	False	False	False	False	False	False

Better approach

In [27]:
```
cars.isnull().sum()
```

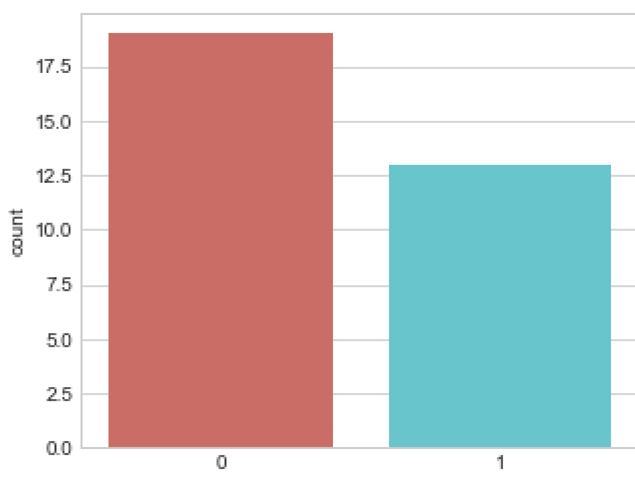
Out[27]:
```
car_names    0
mpg          0
cyl          0
disp         0
hp           0
drat         0
wt           0
qsec         0
vs           0
am           0
gear         0
carb         0
dtype: int64
```

Great, no missing values!

Checking that your target is binary or ordinal

In [28]:
```
sb.countplot(x='am', data=cars, palette='hls')
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x296b8833400>
```



We can see here that our am variable is binary, it only assumes two values, zero or one. So, that's how the suffice the assumptions of the model.

Checking that your dataset size is sufficient

In [30]:
```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 12 columns):
car_names    32 non-null object
mpg          32 non-null float64
cyl          32 non-null int64
disp         32 non-null float64
hp           32 non-null int64
drat         32 non-null float64
wt           32 non-null float64
qsec         32 non-null float64
vs           32 non-null int64
am           32 non-null int64
gear         32 non-null int64
carb         32 non-null int64
dtypes: float64(5), int64(6), object(1)
memory usage: 3.1+ KB
```

We're using two predictors in our model, so we should have 100 observations. But we have only 31 observations. So, we know this model will be somewhat unreliable.

Deploying and evaluating your model

Let's scale our data. We're going to call the scaled data set x and then we're going to call scale and pass in cars_data. The attributes should be of same scale.

In [37]:
```
X = scale(cars_data)
```

Create a logistic regression model. Learn more at http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

In [39]:
```
LogReg = LogisticRegression()

LogReg.fit(X,y)
LogReg.score(X,y)
```

Out[39]:
```
0.8125
```

In [ ]:
```
Good score but we can fully trust it due to insufficient data.
```

Print the confusion matrix.

In [43]:
```
y_pred = LogReg.predict(X)
from sklearn.metrics import classification_report
print(classification_report(y, y_pred))
```

```
             precision    recall  f1-score   support

          0       0.88      0.79      0.83        19
          1       0.73      0.85      0.79        13

avg / total       0.82      0.81      0.81        32
```

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Recall is the ratio of correctly predicted positive observations to the all observations in actual class. F1 Score is the weighted average of Precision and Recall and it takes both false positives and false negatives into account. All scores are pretty high here.

In [ ]:
```
Calculate area under the curve (ROC)
```

In [46]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y, y_pred)
```

Out[46]:
```
0.81781376518218629
```