Regression Models

You guys have seen the usage of fitting linear model into data. In this project, you will need to explore logistic regression using `scikit-learn` .

Objective: Predicting who will survive on the Titanic with logistic regression

In [1]:
```
import numpy as np
import pandas as pd
import sklearn
import sklearn.linear_model as lm
import sklearn.model_selection as ms # IF you can not import this module, upgr
ade your sklearn to the newest version
import matplotlib.pyplot as plt
%matplotlib inline
```

1 Load the data into dataframe

In [2]:
```
train = pd.read_csv('titanic_train.csv')
train[train.columns[[2,4,5,1]]].head()
```

Out[2]:

	Pclass	Sex	Age	Survived
0	3	male	22.0	0
1	1	female	38.0	1
2	3	female	26.0	1
3	1	female	35.0	1
4	3	male	35.0	0

These four attributes are the ones that we are interested

2. You should always remember to clean the data before fitting any model into the data, please do the following:

- Create a new dataframe 'data' that stores only these four columns. (use `.copy()` method)
- Convert the sex field to a binary variable, so that it can be handled correctly by NumPy and scikit-learn.
- Remove the rows containing NaN values.

In [135]:
```
data = train[['Pclass','Sex','Age','Survived']].copy()
#male=1, female=0
data['Sex'].replace(['male', 'female'], [1, 0], inplace=True)
data.dropna(inplace=True)
data.head()
```

Out[135]:

	Pclass	Sex	Age	Survived
0	3	1	22.0	0
1	1	0	38.0	1
2	3	0	26.0	1
3	1	0	35.0	1
4	3	1	35.0	0

3. Extract the values from 'data' and put them into `numpy.array` (name it data_np)(we need to plug it into sklearn functions), convert the data types to `np.int32` , create the a variable `X` to store the first three columns: "Pclass, Sex, Age", create another variable `y` to store the last column: Survived.

In [164]:
```
data_np = data.values
data_np.astype(np.int32)
X = data_np[:, [0, 1, 2]]
y = data_np[:, [3]]
print "X: Pclass, Sex, Age"
print X.view()
print "Number of rows: ", len(X)
print
print "y: Survived"
yprint = y[:5,:]
print yprint
print "..."
print "Number of rows: ", len(y)
```

```
X: Pclass, Sex, Age
[[ 3.  1. 22.]
 [ 1.  0. 38.]
 [ 3.  0. 26.]
 ...
 [ 1.  0. 19.]
 [ 1.  1. 26.]
 [ 3.  1. 32.]]
Number of rows:  714

y: Survived
[[0.]
 [1.]
 [1.]
 [1.]
 [0.]]
...
Number of rows:  714
```

3. Now we need to split the data into two parts, one part is for training, one part for testing. Let's try split X and y evenly into two parts using the `ms.train_test_split()` method, you should now get four sets: `X_train, X_test, y_train, y_test` .

In [171]:
```
X_train, X_test, y_train, y_test = ms.train_test_split(X, y, test_size=0.05)
print "Number of rows in X_train and y_train: ", len(X_train)
print "Number of rows in X_test and y_test: ", len(X_test)
```

```
Number of rows in X_train and y_train:  678
Number of rows in X_test and y_test:  36
```

4. Now we need to plug the training data set into the logistic regression model, you can find the method `lm.LogisticRegression()` , it's similar to the linear regression method. Try fit `X_train, y_train` into the model, and predict `y_predicted` from `x_test`

In [172]:
```
logreg = lm.LogisticRegression()
logreg.fit(X_train, y_train)
y_predicted = logreg.predict(X_test)
print "Predicted y values: ", y_predicted

#for visualization
y_test = y_test.reshape((1, -1))
```

```
Predicted y values:  [0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 0. 0. 1. 0. 1.
1. 1. 0. 0. 0. 1.
 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 1.]
```

5. The following code will help you visualize the result

In [173]:
```
plt.figure(figsize=(8, 3));
plt.imshow(np.vstack((y_test, y_predicted)),
           interpolation='none', cmap='bone');
plt.xticks([]); plt.yticks([]);
plt.title(("Actual and predicted survival outcomes"
          " on the test set"));
print "Actual y values: ", y_test
print
print "Black is 0, white is 1"
```

```
Actual y values:  [[0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 1.
1. 0. 0. 0. 1.
  0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 1. 1.]]

Black is 0, white is 1
```