In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from patsy import dmatrices
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split, cross_val_score
from sklearn import metrics

%matplotlib inline
```

Pre-Task: Describe the goals of your study

Part 1: Aquire the Data

In [ ]:
```
psql -h dsi.c20gkj5cvu3l.us-east-1.rds.amazonaws.com -p 5432 -U dsi_stud
ent titanic
password: gastudents
```

1. Connect to the remote database

In [ ]:

In [ ]:

2. Query the database and aggregate the data

In [ ]:

5. What are the risks and assumptions of our data?

Part 2: Exploratory Data Analysis

1. Describe the Data

```
In [ ]:
```

2. Visualize the Data

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

Part 3: Data Wrangling

1. Create Dummy Variables for *Sex*

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

Part 4: Logistic Regression and Model Validation

1. Define the variables that we will use in our classification analysis

```
In [ ]:
```

2. Transform "Y" into a 1-Dimensional Array for SciKit-Learn

```
In [ ]:
```

3. Conduct the logistic regression

```
In [ ]:
```

In [ ]:

4. Examine the coefficients to see our correlations

In [ ]:

6. Test the Model by introducing a *Test* or *Validaton* set

In [ ]:

7. Predict the class labels for the *Test* set

In [ ]:

8. Predict the class probabilities for the *Test* set

In [ ]:

9. Evaluate the *Test* set

In [ ]:

10. Cross validate the test set

In [ ]:

11. Check the Classification Report

In [ ]:

12. What do the classification metrics tell us?

13. Check the Confusion Matrix

```
In [ ]:
```

14. What does the Confusion Matrix tell us?

15. Plot the ROC curve

```
In [ ]:
```

16. What does the ROC curve tell us?

Part 5: Gridsearch

1. Use GridSearchCV with logistic regression to search for optimal parameters

- Use the provided parameter grid. Feel free to add if you like (such as n_jobs).
- Use 5-fold cross-validation.

```
In [ ]: logreg_parameters = {
            'penalty':['l1','l2'],
            'C':np.logspace(-5,1,50),
            'solver':['liblinear']
        }
```

2. Print out the best parameters and best score. Are they better than the vanilla logistic regression?

```
In [ ]:
```

3. Explain the difference between the difference between the L1 (Lasso) and L2 (Ridge) penalties on the model coefficients.

4. What hypothetical situations are the Ridge and Lasso penalties useful?

5. [BONUS] Explain how the regularization strength (C) modifies the regression loss function. Why do the Ridge and Lasso penalties have their respective effects on the coefficients?

In [ ]:

```

```

6.a. [BONUS] You decide that you want to minimize false positives. Use the predicted probabilities from the model to set your threshold for labeling the positive class to need at least 90% confidence. How and why does this affect your confusion matrix?

In [ ]:

```

```

Part 6: Gridsearch and kNN

1. Perform Gridsearch for the same classification problem as above, but use KNeighborsClassifier as your estimator

At least have number of neighbors and weights in your parameters dictionary.

In [ ]:

```

```

2. Print the best parameters and score for the gridsearched kNN model. How does it compare to the logistic regression model?

In [ ]:

```

```

3. How does the number of neighbors affect the bias-variance tradeoff of your model?

[BONUS] Why?

In [ ]:

```

```

4. In what hypothetical scenario(s) might you prefer logistic regression over kNN, aside from model performance metrics?

In [ ]:

```

```

5. Fit a new kNN model with the optimal parameters found in gridsearch.

In [ ]:

6. Construct the confusion matrix for the optimal kNN model. Is it different from the logistic regression model? If so, how?

In [ ]:

7. [BONUS] Plot the ROC curves for the optimized logistic regression model and the optimized kNN model on the same plot.

In [ ]:

Part 7: [BONUS] Precision-recall

1. Gridsearch the same parameters for logistic regression but change the scoring function to 'average_precision'

`'average_precision'` will optimize parameters for area under the precision-recall curve instead of for accuracy.

In [ ]:

2. Examine the best parameters and score. Are they different than the logistic regression gridsearch in part 5?

In [ ]:

3. Create the confusion matrix. Is it different than when you optimized for the accuracy? If so, why would this be?

In [ ]:

4. Plot the precision-recall curve. What does this tell us as opposed to the ROC curve?

[See the sklearn plotting example here. (http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html)](http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html)

In [ ]:

Part 8: [VERY BONUS] Decision trees, ensembles, bagging

1. Gridsearch a decision tree classifier model on the data, searching for optimal depth. Create a new decision tree model with the optimal parameters.

In [ ]:

2. Compare the performace of the decision tree model to the logistic regression and kNN models.

In [ ]:

3. Plot all three optimized models' ROC curves on the same plot.

In [ ]:

4. Use sklearn's BaggingClassifier with the base estimator your optimized decision tree model. How does the performance compare to the single decision tree classifier?

In [ ]:

5. Gridsearch the optimal n_estimators, max_samples, and max_features for the bagging classifier.

In [ ]:

6. Create a bagging classifier model with the optimal parameters and compare it's performance to the other two models.

In [ ]: