Survival Rate in Titanic Disaster

Author: Jason(Ye) Wang

In [1]:
```
cd /Users/yewang/Desktop/UMD Courses/Term B 2/758X/Project Titanic
```

```
/Users/yewang/Desktop/UMD Courses/Term B 2/758X/Project Titanic
```

In [2]:
```
from pandas import Series, DataFrame
import numpy as np
import pandas as pd
```

In [3]:
```
data = pd.read_csv("train.csv")
print data.head(10)
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3
5            6         0       3
6            7         0       1
7            8         0       3
8            9         1       3
9           10         1       2

                                                Name     Sex  Age  SibSp  \
0                            Braund, Mr. Owen Harris    male   22      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female   38      1
2                             Heikkinen, Miss. Laina  female   26      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female   35      1
4                           Allen, Mr. William Henry    male   35      0
5                                   Moran, Mr. James    male  NaN      0
6                            McCarthy, Mr. Timothy J    male   54      0
7                     Palsson, Master. Gosta Leonard    male    2      3
8  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female   27      0
9                Nasser, Mrs. Nicholas (Adele Achem)  female   14      1

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
5      0            330877   8.4583   NaN        Q
6      0             17463  51.8625   E46        S
7      1            349909  21.0750   NaN        S
8      2            347742  11.1333   NaN        S
9      0            237736  30.0708   NaN        C
```

1. Data Munging

1) Clean the Name column

I splited the Name into FirstName column and LastName column. Also, I picked out the prefix as a new variable, Prefix.

In [4]:
```
names = list(data.Name)
first = []
last = []
prex =[]
c_last = []
for i in range(0,len(names)):
    tem = names[i].split(",")
    first.append(tem[0])
    last.append(tem[1])
for i in range(0,len(last)):
    tem = last[i].split(".")
    if tem[0] in [" Mr"," Dr"," Master"," Miss"," Mrs"," Ms"," Sir"]:
        prex.append(tem[0].strip())
        c_last.append(tem[1].strip())
    elif tem[0] not in [" Mr"," Dr"," Master"," Miss"," Mrs"," Ms"," Sir"]:
        prex.append("None")
        c_last.append(last[i].strip())
```

In [5]:
```
data["FirstName"] = first
data["Prefix"] = prex
data["LastName"] = c_last
c_data = data.drop("Name",axis=1)
```

In the following output, we can see that three columns - FirstName, LastName and Prefix, replaced original Name column.

In [6]:
```
print c_data.head(5)
```

```
   PassengerId  Survived  Pclass     Sex  Age  SibSp  Parch            Ticket  \
0            1         0       3    male   22      1      0         A/5 21171
1            2         1       1  female   38      1      0          PC 17599
```

```
2           3         1       3  female   26      0       0  STON/O2. 3101282
3           4         1       1  female   35      1       0            113803
4           5         0       3    male   35      0       0            373450

      Fare Cabin Embarked  FirstName Prefix  \
0   7.2500   NaN        S     Braund     Mr
1  71.2833   C85        C    Cumings    Mrs
2   7.9250   NaN        S  Heikkinen   Miss
3  53.1000  C123        S   Futrelle    Mrs
4   8.0500   NaN        S      Allen     Mr

                               LastName
0                           Owen Harris
1  John Bradley (Florence Briggs Thayer)
2                                 Laina
3          Jacques Heath (Lily May Peel)
4                         William Henry
```

2) Clean the Ticket column

First Step: Splited the the Ticket column into TicketMark column and TicketNum column because TicketMark might be a useful predictor to predict passengers' survived rates.

In [7]:
```python
tickets = list(c_data.Ticket)
ticket_mark = []
ticket_num =[]

for n in range(0,len(tickets)):
    tem = tickets[n].split(" ")
    if len(tem)==2:
        ticket_mark.append(tem[0])
        ticket_num.append(tem[1])
    elif len(tem)==3:
        ticket_mark.append(tem[0]+" "+tem[1])
        ticket_num.append(tem[2])
    else:
        ticket_mark.append("None")
        ticket_num.append(tickets[i])

c_data["TicketMark"]=ticket_mark
c_data["TicketNum"] = ticket_num
c_data = c_data.drop("Ticket",axis=1)
c_data.pivot_table("PassengerId",rows="TicketMark",aggfunc="count")
```

Out[7]:
```
TicketMark
A./5.            2
A.5.             2
A/4              3
A/4.             3
A/5             10
A/5.             7
A/S              1
A4.              1
C                5
C.A.            27
C.A./SOTON       1
CA               6
CA.              8
F.C.             1
F.C.C.           5
Fa               1
None           665
P/PP             2
PC              60
PP               3
S.C./A.4.        1
S.C./PARIS       2
S.O./P.P.        3
S.O.C.           5
S.O.P.           1
S.P.             1
S.W./PP          1
SC               1
SC/AH            2
SC/AH Basle      1
SC/PARIS         5
SC/Paris         4
SCO/W            1
SO/C             1
SOTON/O.Q.       8
SOTON/O2         2
SOTON/OQ         7
STON/O 2.       12
STON/O2.         6
SW/PP            1
W./C.            9
W.E.P.           1
W/C              1
WE/P             2
Name: PassengerId, dtype: int64
```

Second Step: to clean some misspell and typo errors in the TicketMark column such as STON/O 2. and STON/O2.. Also, I deleted all periods(".")

a. remove period in the TicketMark

In [8]:
```python
marks = c_data["TicketMark"]
for i in range(0,len(marks)):
```

```
    c_data["TicketMark"][i]=marks[i].replace(".","")
print c_data.pivot_table("PassengerId",rows="TicketMark",aggfunc="count")
```

```
TicketMark
A/4             6
A/5            19
A/S             1
A4              1
A5              2
C               5
CA             41
CA/SOTON        1
FC              1
FCC             5
Fa              1
None          665
P/PP            2
PC             60
PP              3
SC              1
SC/A4           1
SC/AH           2
SC/AH Basle     1
SC/PARIS        7
SC/Paris        4
SCO/W           1
SO/C            1
SO/PP           3
SOC             5
SOP             1
SOTON/O2        2
SOTON/OQ       15
SP              1
STON/O 2       12
STON/O2         6
SW/PP           2
W/C            10
WE/P            2
WEP             1
Name: PassengerId, dtype: int64
```

b. clean some typos

In [9]:
```
tem2 = list(c_data["TicketMark"])
import re
regex1 = re.compile("\D+\s+\d")
tem3 = []
check = []
for i in range(0,len(marks)):
    tem3.append(regex1.findall(tem2[i]))
    if len(tem3[i]) != 0:
        check.append(i)

# finish clean "STON/O 2"
for i in range(0,len(check)):
    c_data["TicketMark"][check[i]] = c_data["TicketMark"][check[i]].replace(" ","")

regex2 = re.compile("\D+\s+")
tem4 = []
check2 = []
for i in range(0,len(marks)):
    tem4.append(regex2.findall(tem2[i]))
    if len(tem4[i]) != 0:
        check.append(i)

c_data["TicketMark"][check[0]] = c_data["TicketMark"][check[0]].strip()
print c_data.pivot_table("PassengerId",rows="TicketMark",aggfunc="count")
```

```
TicketMark
A/4             6
A/5            19
A/S             1
A4              1
A5              2
C               5
CA             41
CA/SOTON        1
FC              1
FCC             5
Fa              1
None          665
P/PP            2
PC             60
PP              3
SC              1
SC/A4           1
SC/AH           2
SC/AH Basle     1
SC/PARIS        7
SC/Paris        4
SCO/W           1
SO/C            1
SO/PP           3
SOC             5
SOP             1
SOTON/O2        2
SOTON/OQ       15
SP              1
STON/O2        18
SW/PP           2
W/C            10
WE/P            2
WEP             1
Name: PassengerId, dtype: int64
```

3) Clean and cut the Age column

In this procssing, I classified Age variable into 4 categories- Unknown, Teen, Mid and Old according to different age interval.

In [10]:
```
c_data.Age[pd.isnull(c_data.Age)] = 0
age_gap = pd.cut(c_data.Age, [-1,0, 17, 40, 100],labels=["Unknown","Teen","Mid","Old"])
c_data["AgeLevel"] = age_gap

c_data.pivot_table("PassengerId",rows="AgeLevel",aggfunc="count")
```

Out[10]:
```
AgeLevel
Mid         451
Old         150
Teen        113
Unknown     177
Name: PassengerId, dtype: int64
```

4) Clean the Cabin column

I only kept the the alphabet instead of number in the column. Those different cabins can be used in the logistic regression model.

In [11]:
```
c_data.Cabin[pd.isnull(c_data.Cabin)] = "None"

cab =list(c_data["Cabin"])
for i in range(0,len(c_data["Cabin"])):
    if cab[i] != "None":
        c_data["Cabin"][i] = cab[i][0]

print c_data.pivot_table("PassengerId",rows="Cabin",aggfunc="count")
```

```
Cabin
A        15
B        47
C        59
D        33
E        32
F        13
G         4
None    687
T         1
Name: PassengerId, dtype: int64
```

5) Create Fare interval

I made the FareLevel, a new variable produced by dividing continuous Fare variable into 4 different categories (Low, LowMiddle, Middle and High).

In [12]:
```
print c_data["Fare"].describe()
fare_gap = pd.cut(c_data.Fare, [0, 10, 50, 100, 520],labels=["Low","LowMid","Mid","High"])
c_data["FareLevel"] = fare_gap
print ""
print c_data.pivot_table("PassengerId",rows="FareLevel",aggfunc="count")
```

```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
dtype: float64

FareLevel
High         53
Low         321
LowMid      395
Mid         107
Name: PassengerId, dtype: int64
```

6) Clean the number of Sibling

In the processing, I found that most individuals only had 0 or 1 sibling. Therefore, I grouped the number of sibling which is more than 2.

In [13]:
```
print c_data.pivot_table("PassengerId",rows="SibSp",aggfunc="count")
```

```
SibSp
0       608
1       209
2        28
3        16
4        18
5         5
8         7
Name: PassengerId, dtype: int64
```

In [14]:
```
sib = list(c_data["SibSp"])
tem = []
for i in range(0,len(sib)):
```

```
    if sib[i] >= 2:
        tem.append("2+")
    else:
        tem.append(str(sib[i]))

c_data = c_data.drop("SibSp",axis=1)
c_data["SibSp"] = tem
```

In [15]:
```
print c_data.pivot_table("PassengerId",rows="SibSp",aggfunc="count")
```

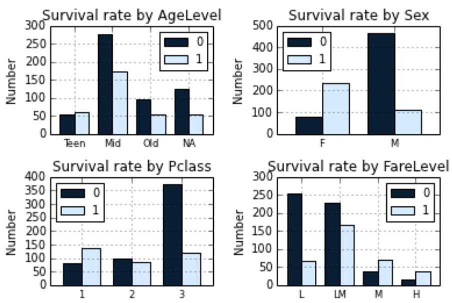
```
SibSp
0        608
1        209
2+        74
Name: PassengerId, dtype: int64
```

2. Exploratory Analysis

In [16]:
```
import matplotlib.pyplot as plt
```

1) Survival rate histograms on different categorical variables

In [17]:
```
fig,axes = plt.subplots(2,2)

#Survived by AgeLevel
age_int = c_data.pivot_table("PassengerId",rows="AgeLevel",cols= "Survived",aggfunc="count")
age_int = age_int.reindex(index=["Teen","Mid","Old","Unknown"])
age_int.plot(kind="bar",color=("#0A1F33","#D6EBFF"),ax=axes[0,0],fontsize="small",title="Survival rate by AgeLevel")
axes[0,0].set_xticklabels(["Teen","Mid","Old","NA"],rotation=0,fontsize="small")

axes[0,0].legend(loc="upper right",prop={"size":10})
axes[0,0].set_ylabel("Number")

#Survived by Gender
sex_int = c_data.pivot_table("PassengerId",rows="Sex",cols= "Survived",aggfunc="count")
sex_int.plot(kind="bar",color=("#0A1F33","#D6EBFF"),ax=axes[0,1],fontsize="small",title="Survival rate by Sex")
axes[0,1].set_xticklabels(["F","M"],rotation=0,fontsize="small")
axes[0,1].legend(loc="upper left",prop={"size":10})
axes[0,1].set_xlabel("")
axes[0,1].set_ylabel("Number")

#Survived by Class
pclass_int = c_data.pivot_table("PassengerId",rows="Pclass",cols= "Survived",aggfunc="count")
pclass_int.plot(kind="bar",color=("#0A1F33","#D6EBFF"),ax=axes[1,0],fontsize="small", title="Survival rate by Pclass")
axes[1,0].set_xticklabels(["1","2","3"],rotation=0,fontsize="small")
axes[1,0].legend(loc="upper left",prop={"size":10})
axes[1,0].set_xlabel("")
axes[1,0].set_ylabel("Number")

#Survived by Fare
fare_int = c_data.pivot_table("PassengerId",rows="FareLevel",cols= "Survived",aggfunc="count")
fare_int = fare_int.reindex(index=["Low","LowMid","Mid","High"])
fare_int.plot(kind="bar",color=("#0A1F33","#D6EBFF"),ax=axes[1,1],fontsize="small", title="Survival rate by FareLevel")
axes[1,1].set_xticklabels(["L","LM","M","H"],rotation=0,fontsize="small")
axes[1,1].legend(loc="upper right",prop={"size":10})
axes[1,1].set_ylabel("Number")

plt.subplots_adjust(wspace=0.4,hspace=0.4)
```



2) Death rate on different categorical variables

In [18]:
```
print "-----Survived by Pclass-----"
pclass_int["DeathRate"] = pclass_int[0]/float(sum(pclass_int[0]+pclass_int[1])) *100
pclass_int["DeathRate"] = pclass_int["DeathRate"].round(3)
print pclass_int
print ""

print "-----Survived by AgeLevel-----"
age_int["DeathRate"] = age_int[0]/float(sum(age_int[0]+age_int[1])) *100
age_int["DeathRate"] = age_int["DeathRate"].round(3)
print age_int
print ""

print "-----Survived by Gender-----"
sex_int["DeathRate"] = sex_int[0]/float(sum(sex_int[0]+sex_int[1])) *100
sex_int["DeathRate"] = sex_int["DeathRate"].round(3)
print sex_int
print ""
```

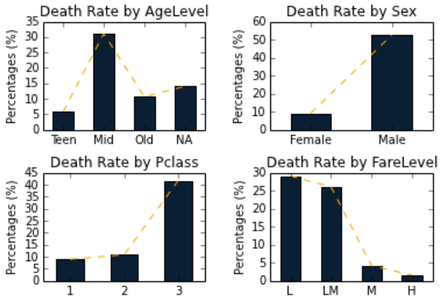
```
print "-----Survived by FareLevel-----"
fare_int["DeathRate"] = fare_int[0]/float(sum(fare_int[0]+fare_int[1])) *100
fare_int["DeathRate"] = fare_int["DeathRate"].round(3)
print fare_int
```

```
-----Survived by Pclass-----
Survived    0    1  DeathRate
Pclass
1          80  136      8.979
2          97   87     10.887
3         372  119     41.751

-----Survived by AgeLevel-----
Survived    0    1  DeathRate
Teen       52   61      5.836
Mid       277  174     31.089
Old        95   55     10.662
Unknown   125   52     14.029

-----Survived by Gender-----
Survived    0    1  DeathRate
Sex
female     81  233      9.091
male      468  109     52.525

-----Survived by FareLevel-----
Survived    0    1  DeathRate
Low       255   66     29.110
LowMid    229  166     26.142
Mid        37   70      4.224
High       14   39      1.598
```

3) Histogram for death rate on different categorical variables

In [19]:

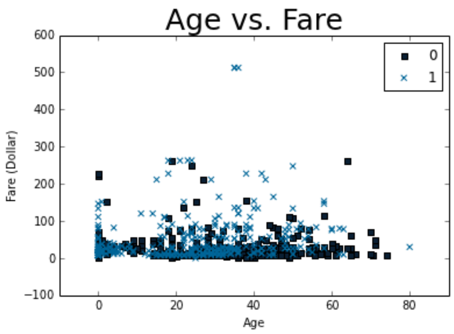
```
fig2,axes2 = plt.subplots(2,2)

#DeathRate by AgeLevel
ind = [0,1,2,3]
width=0.5
axes2[0,0].bar(ind,age_int["DeathRate"],width,color="#0A1F33")
axes2[0,0].set_xticks([0.25,1.25,2.25,3.25])
axes2[0,0].set_xticklabels(["Teen","Mid","Old","NA"],rotation=0,fontsize="medium",)
axes2[0,0].plot([0.25,1.25,2.25,3.25],age_int["DeathRate"],"--",color="orange")
axes2[0,0].set_xlim([-0.25,3.75])
axes2[0,0].set_ylabel("Percentages (%)")
axes2[0,0].set_title('Death Rate by AgeLevel')

#DeathRate by Gender
axes2[0,1].bar([0.5,1.5],sex_int["DeathRate"],0.5,color="#0A1F33")
axes2[0,1].set_xticks([0.75,1.75])
axes2[0,1].set_xlim([0.25,2.25])
axes2[0,1].set_xticklabels(["Female","Male"],rotation=0,fontsize="medium",)
axes2[0,1].plot([0.75,1.75],sex_int["DeathRate"],"--",color="orange")
axes2[0,1].set_ylabel("Percentages (%)")
axes2[0,1].set_title('Death Rate by Sex')

#DeathRate by Class
axes2[1,0].bar([0.75,1.75,2.75],pclass_int["DeathRate"],0.5,color="#0A1F33")
axes2[1,0].set_xticks([1,2,3])
axes2[1,0].set_xlim([0.5,3.5])
axes2[1,0].plot([1,2,3],pclass_int["DeathRate"],"--",color="orange")
axes2[1,0].set_ylabel("Percentages (%)")
axes2[1,0].set_title('Death Rate by Pclass')

#DeathRate by Fare
ind = [0,1,2,3]
width=0.5
axes2[1,1].bar(ind,fare_int["DeathRate"],width,color="#0A1F33")
axes2[1,1].set_xticks([0.25,1.25,2.25,3.25])
axes2[1,1].set_xticklabels(["L","LM","M","H"],rotation=0,fontsize="medium",)
axes2[1,1].plot([0.25,1.25,2.25,3.25],fare_int["DeathRate"],"--",color="orange")
axes2[1,1].set_xlim([-0.25,3.75])
axes2[1,1].set_ylabel("Percentages (%)")
axes2[1,1].set_title('Death Rate by FareLevel')

plt.subplots_adjust(wspace=0.4,hspace=0.4)
```



4) Scatterplot for age and fare

```
In [20]: plt.scatter(c_data["Age"][c_data["Survived"]==0],c_data["Fare"][c_data["Survived"]==0],c=("#0A1F33"),marker=",",s=25)
         plt.scatter(c_data["Age"][c_data["Survived"]==1],c_data["Fare"][c_data["Survived"]==1],c=("#006699"),marker="x",s=25,hold
         plt.title("Age vs. Fare",fontsize=25)
         plt.xlabel("Age")
         plt.ylabel("Fare (Dollar)")
         plt.legend(("0","1"),scatterpoints=1,loc="upper right")
```

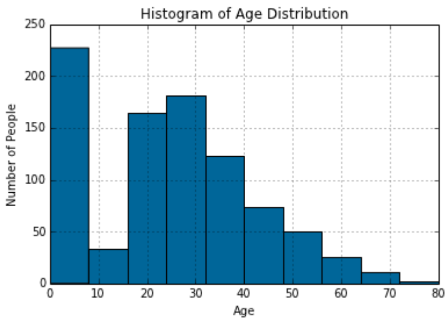
```
Out[20]: <matplotlib.legend.Legend at 0x10fe29c10>
```



5) Distribution of age

```
In [21]: c_data["Age"].hist(color="#006699")
         plt.title("Histogram of Age Distribution")
         plt.xlabel("Age")
         plt.ylabel("Number of People")
```

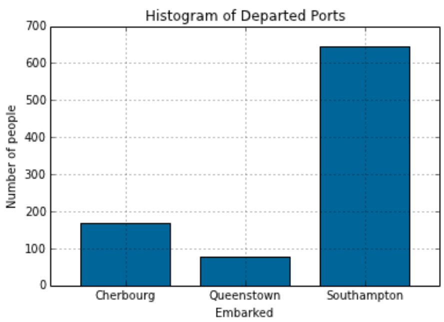
```
Out[21]: <matplotlib.text.Text at 0x10efe2950>
```



6) Distribution of embarked

```
In [22]: fig,ax = plt.subplots(1,1)
         embarked = c_data.pivot_table("PassengerId",rows="Embarked",aggfunc="count")
         embarked.plot(kind="bar",color="#006699",ax=ax)
         ax.set_xticklabels(["Cherbourg","Queenstown","Southampton"],rotation=0)
         ax.set_title("Histogram of Departed Ports")
         ax.set_ylabel("Number of people")
```

```
Out[22]: <matplotlib.text.Text at 0x10fec2cd0>
```



3. Statistical Modeling

Logistic Regression, classification tree and random forest were applied in processed dataset. Because Classificantion tree and random forest were built by R, the results of these two models didn't include in ipython notebook. Plase see the report to find the analysis and figures of classificantion tree and random forest. Also, detail analysis and intepretation were in the PDF report.

1) Dummy for the categorical variables

In [23]:
```python
sex_dummy = pd.get_dummies(c_data["Sex"],prefix="Sex")
pclass_dummy = pd.get_dummies(c_data["Pclass"],prefix="Pclass")
embarked_dummy = pd.get_dummies(c_data["Embarked"],prefix = "Embarked")
pre_dummy = pd.get_dummies(c_data["Prefix"],prefix="P")
ticketmark_dummy = pd.get_dummies(c_data["TicketMark"],prefix="TicketMark")
cabin_dummy=pd.get_dummies(c_data["Cabin"],prefix="Cabin")
sib_dummy = pd.get_dummies(c_data["SibSp"],prefix="Sib")
```

Put those dummy variables into a new dataset to prepare for further analysis

In [24]:
```python
cols_keep = ["Survived","Age","Fare","Parch"]
lr_data = c_data[cols_keep].join(sex_dummy.ix[:,"Sex_male"])
lr_data = lr_data.join(pclass_dummy.ix[:,"Pclass_2":])
lr_data = lr_data.join(embarked_dummy.ix[:,["Embarked_C","Embarked_Q"]])
lr_data = lr_data.join(pre_dummy.ix[:,["P_Dr","P_Master","P_Miss","P_Mr","P_Mrs","P_Ms","P_Sir"]])
lr_data = lr_data.join(sib_dummy.ix[:,"Sib_1":])

#---------dont include the ticketmark due to the large number
#lr_data = lr_data.join(ticketmark_dummy.ix[:,["Embarked_C","Embarked_Q"]])
#-------------------------------------------------------------

lr_data = lr_data.join(cabin_dummy.ix[:,["Cabin_A","Cabin_B","Cabin_C","Cabin_D","Cabin_E","Cabin_F","Cabin_G","Cabin_T"]
```

3) Logistic Regression

In [25]:
```python
import statsmodels.api as sm
```

In [26]:
```python
train_cols = lr_data.columns[1:]

logit = sm.Logit(lr_data["Survived"],lr_data[train_cols])

result = logit.fit()

plt.clf()
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.406915
         Iterations: 35
<matplotlib.figure.Figure at 0x112197450>
```

In [27]:
```python
print result.summary()
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:               Survived   No. Observations:                  891
Model:                          Logit   Df Residuals:                      866
Method:                           MLE   Df Model:                           24
Date:                Mon, 16 Dec 2013   Pseudo R-squ.:                  0.3889
Time:                        19:28:24   Log-Likelihood:                -362.56
converged:                      False   LL-Null:                       -593.33
                                        LLR p-value:                 1.564e-82
==============================================================================
                 coef    std err          z      P>|z|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
Age           -0.0102      0.006     -1.598      0.110        -0.023     0.002
Fare           0.0041      0.003      1.456      0.145        -0.001     0.010
Parch         -0.3708      0.132     -2.808      0.005        -0.630    -0.112
Sex_male      -1.7809      0.880     -2.024      0.043        -3.506    -0.056
Pclass_2       0.0295      0.444      0.066      0.947        -0.841     0.899
Pclass_3      -1.0014      0.439     -2.280      0.023        -1.862    -0.141
Embarked_C     0.5377      0.251      2.138      0.033         0.045     1.031
Embarked_Q     0.1831      0.346      0.530      0.596        -0.494     0.860
P_Dr           1.1274      1.174      0.960      0.337        -1.174     3.428
P_Master       3.8136      0.946      4.030      0.000         1.959     5.668
P_Miss         1.6927      0.488      3.469      0.001         0.736     2.649
P_Mr           0.3926      0.816      0.481      0.630        -1.206     1.991
P_Mrs          2.1774      0.550      3.958      0.000         1.099     3.256
P_Ms          36.4045   6.71e+07   5.42e-07      1.000     -1.32e+08  1.32e+08
P_Sir         37.0345   6.71e+07   5.52e-07      1.000     -1.32e+08  1.32e+08
Sib_1         -0.1800      0.245     -0.735      0.463        -0.660     0.300
Sib_2+        -1.4178      0.389     -3.640      0.000        -2.181    -0.654
Cabin_A        0.8549      0.686      1.246      0.213        -0.490     2.200
Cabin_B        1.1419      0.564      2.024      0.043         0.036     2.248
Cabin_C        0.6485      0.516      1.256      0.209        -0.363     1.660
Cabin_D        1.5883      0.596      2.663      0.008         0.420     2.757
Cabin_E        1.9087      0.608      3.141      0.002         0.718     3.100
Cabin_F        0.9919      0.825      1.202      0.229        -0.625     2.609
Cabin_G       -0.2862      1.024     -0.279      0.780        -2.293     1.721
Cabin_T      -34.5035   7.27e+07  -4.75e-07      1.000     -1.42e+08  1.42e+08
==============================================================================
```

In [28]:
```python
confMax = result.pred_table()
print confMax

accuracy = (confMax[0][0]+confMax[1][1]) / confMax.sum()
specificity = confMax[0][0] / np.sum(confMax[0])
senstivity = confMax[1][1] / np.sum(confMax[1])
ppv = confMax[1][1] / (confMax[1][1]+confMax[0][1])
npv = confMax[0][0] / (confMax[0][0]+confMax[1][0])

print " "
print "Accuracy: " + str(round(accuracy,3)*100)+"%"
print "Senstivity: "+str(round(senstivity,3)*100)+"%"
print "Specificity: " + str(round(specificity,3)*100)+"%"
print "Postive Predictive Value: " + str(round(ppv,3)*100)+"%"
print "Negative Predictive Value: "+str(round(npv,3)*100)+"%"
```

```
[[ 481.   68.]
```

```
[  82.  200.]]

Accuracy: 83.2%
Senstivity: 76.0%
Specificity: 87.6%
Postive Predictive Value: 79.3%
Negative Predictive Value: 85.4%
```

4) Classification tree

In [29]:
```python
from sklearn import tree
```

I ran classification tree here. I set the max_depth = 4 due to overfitting. Plase see following graph of whole classification tree.

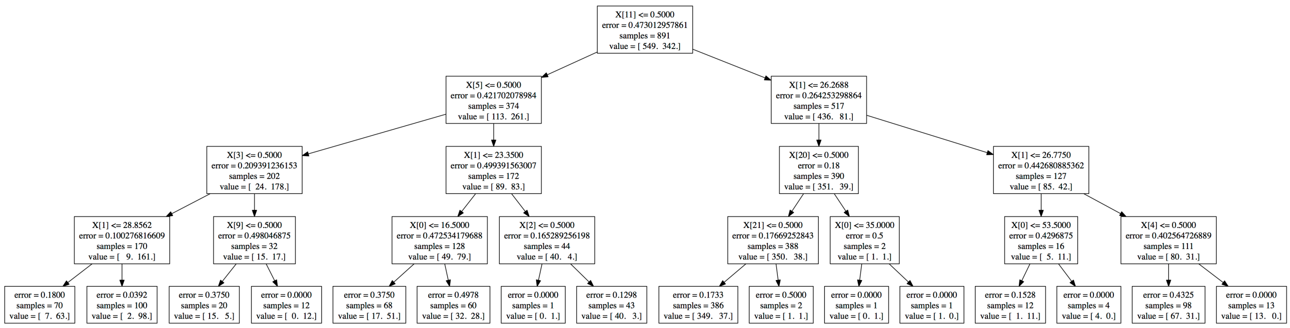
In [30]:
```python
clf = tree.DecisionTreeClassifier(min_samples_split=1,min_density=1,max_depth=4)
clf = clf.fit(lr_data[train_cols],lr_data["Survived"])
```

In this step, I output the dot file named "titanic.dot", and then used Graphviz app to draw the graph. If you want to replicate this step, please download Graphviz app from http://www.graphviz.org/Download_macos.php (http://www.graphviz.org/Download_macos.php). Using graphviz app to open the file "titanic.dot" in order to see the graph.

In [31]:
```python
with open("titanic.dot", 'w') as f:
    f = tree.export_graphviz(clf, out_file=f)
```

In [32]:
```python
from IPython.core.display import Image
Image(filename='classificationtreeInPython.png')
```

Out[32]:



In [33]:
```python
clf_cm = pd.crosstab(lr_data.Survived,clf.predict(lr_data[train_cols]),rownames=['actual'], colnames=['preds'])
clf_cm
```

Out[33]:

preds	0	1
actual		
0	522	27
1	105	237

In [34]:
```python
clf_accuracy = float(clf_cm[0][0]+clf_cm[1][1]) / clf_cm.sum().sum()
clf_specificity = float(clf_cm[0][0]) / np.sum(clf_cm[0])
clf_senstivity = float(clf_cm[1][1]) / np.sum(clf_cm[1])
clf_ppv = float(clf_cm[1][1]) / (clf_cm[1][1]+clf_cm[0][1])
clf_npv = float(clf_cm[0][0]) / (clf_cm[0][0]+clf_cm[1][0])

print "Accuracy: " + str(round(clf_accuracy,3)*100)+"%"

print "Senstivity: "+str(round(clf_ppv,3)*100)+"%"
print "Specificity: " + str(round(clf_npv,3)*100)+"%"
print "Postive Predictive Value: " + str(round(clf_senstivity,3)*100)+"%"
print "Negative Predictive Value: "+ str(round(clf_specificity,3)*100)+"%"
```

```
Accuracy: 85.2%
Senstivity: 69.3%
Specificity: 95.1%
Postive Predictive Value: 89.8%
Negative Predictive Value: 83.3%
```

Classification tree is also run by R, please find more detail in the project report

5) Random forest

In [35]:
```python
from sklearn.ensemble import RandomForestClassifier
```

I ran random forest. In order to avoid overfitting, I also set max_depth = 4 like classification tree.

In [36]:
```python
rfl = RandomForestClassifier(n_estimators=500, max_depth=4, min_samples_split=1, random_state=0)
rfl_results = rfl.fit(lr_data[train_cols],lr_data["Survived"])
```

I made confusion matrix on the random forest and calculated sensitivity and specificity.

In [37]:
```
rf_cm = pd.crosstab(lr_data.Survived,rfl.predict(lr_data[train_cols]),rownames=['actual'], colnames=['preds'])
rf_cm
```

Out[37]:

preds	0	1
actual		
0	495	54
1	90	252

In [38]:
```
rf_accuracy = float(rf_cm[0][0]+rf_cm[1][1]) / rf_cm.sum().sum()
rf_specificity = float(rf_cm[0][0]) / np.sum(rf_cm[0])
rf_senstivity = float(rf_cm[1][1]) / np.sum(rf_cm[1])
rf_ppv = float(rf_cm[1][1]) / (rf_cm[1][1]+rf_cm[0][1])
rf_npv = float(rf_cm[0][0]) / (rf_cm[0][0]+rf_cm[1][0])

print "Accuracy: " + str(round(rf_accuracy,3)*100)+"%"
print "Senstivity: "+ str(round(rf_ppv,3)*100)+"%"
print "Specificity: " + str(round(rf_npv,3)*100)+"%"
print "Postive Predictive Value: " + str(round(rf_senstivity,3)*100)+"%"
print "Negative Predictive Value: "+ str(round(rf_specificity,3)*100)+"%"
```

```
Accuracy: 83.8%
Senstivity: 73.7%
Specificity: 90.2%
Postive Predictive Value: 82.4%
Negative Predictive Value: 84.6%
```

Random forest tree is also run by R. Please see detail in report.

4. Export the dataset

In [39]:
```
c_data.to_csv("processed_titanic.csv")
lr_data.to_csv("dummy_titanic.csv")
```