In [69]:

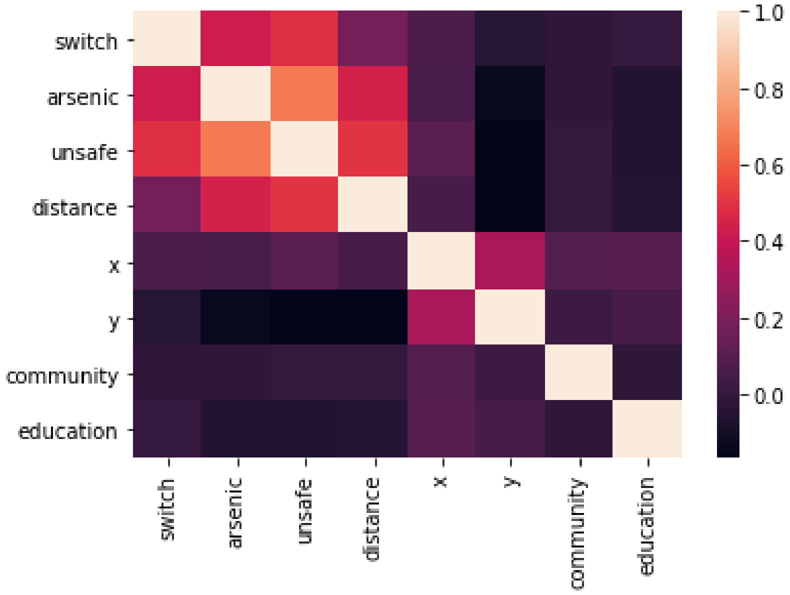
```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn import metrics
from sklearn.model_selection import LeaveOneOut
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
import math as m
```

In [3]:

```
welldata=pd.read_csv('C:\Python27amd64\wells.csv')
```

In [4]:

```
sb.heatmap(welldata.corr())
plt.show()
```



As seen above, there is significant correlation between the pairs of (switch/arsenic), (switch/unsafe) and (arsenic/unsafe). It is likely that arsenic and unsafe will feature coefficients of large values with relatively low p values.

In [5]:

```
welldata=welldata.astype('float')
```

In [6]:

```
response= welldata.switch.tolist()
predictors=welldata.as_matrix(columns=['arsenic', 'unsafe', 'distance', 'x'
, 'y', 'community', 'education'])
```

In [7]:

```
model=sm.Logit(response, predictors)
result=model.fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.525483
         Iterations 6
```

Out[7]:

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	6448
Model:	Logit	**Df Residuals:**	6441
Method:	MLE	**Df Model:**	6
Date:	Thu, 15 Mar 2018	**Pseudo R-squ.:**	0.2151
Time:	14:11:45	**Log-Likelihood:**	-3388.3
converged:	True	**LL-Null:**	-4317.1
		LLR p-value:	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
x1	0.0046	0.000	11.466	0.000	0.004	0.005
x2	1.9309	0.085	22.728	0.000	1.764	2.097
x3	-8.003e-05	9.58e-06	-8.355	0.000	-9.88e-05	-6.13e-05
x4	2.109e-05	1.67e-05	1.260	0.208	-1.17e-05	5.39e-05
x5	-2.775e-06	1.63e-06	-1.698	0.090	-5.98e-06	4.29e-07
x6	-0.0466	0.044	-1.055	0.292	-0.133	0.040
x7	0.0217	0.008	2.882	0.004	0.007	0.036

Referring to the heat map, we can see that there is correlation between 'arsenic' and 'unsafe'. From the summary table, we see that 'unsafe' has a much stronger influence on the 'switch' values than 'arsenic'. Although both have low p values and high absolute z values- for the next model we will drop 'arsenic' and study the impact on R2(to eliminate effects due to colinearity).

In [8]:

```
predictors2=welldata.as_matrix(columns=['unsafe', 'distance', 'x', 'y', 'co
mmunity', 'education'])
model2=sm.Logit(response, predictors2)
result=model2.fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.537634
         Iterations 6
```

Out[8]:

Logit Regression Results

Dep. Variable:	y	No. Observations:	6448
Model:	Logit	Df Residuals:	6442
Method:	MLE	Df Model:	5
Date:	Thu, 15 Mar 2018	Pseudo R-squ.:	0.1970
Time:	14:11:45	Log-Likelihood:	-3466.7
converged:	True	LL-Null:	-4317.1
		LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
x1	2.4956	0.071	34.994	0.000	2.356	2.635
x2	-5.469e-05	9.01e-06	-6.071	0.000	-7.23e-05	-3.7e-05
x3	1.55e-05	1.64e-05	0.942	0.346	-1.67e-05	4.77e-05
x4	-2.204e-06	1.61e-06	-1.372	0.170	-5.35e-06	9.44e-07
x5	-0.0583	0.044	-1.335	0.182	-0.144	0.027
x6	0.0192	0.007	2.601	0.009	0.005	0.034

As expected, the value of R2 has gone down from 0.2151 to 0.1970. We see that the z value of 'unsafe' has gone up. However, the zvalue of 'x' has further decreased, and its p value has gone upt to 0.346. In our next model, we will remove 'x' from the list as well.

In [9]:

```
predictors3=welldata.as_matrix(columns=['unsafe', 'distance', 'y',
'community', 'education'])
model3=sm.Logit(response, predictors3)
result=model3.fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.537703
         Iterations 6
```

Out[9]:

Logit Regression Results

Dep. Variable:	y	No. Observations:	6448
Model:	Logit	Df Residuals:	6443
Method:	MLE	Df Model:	4
Date:	Thu, 15 Mar 2018	Pseudo R-squ.:	0.1969
Time:	14:11:47	Log-Likelihood:	-3467.1
converged:	True	LL-Null:	-4317.1
		LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]

x1	2.5022	0.071	35.241	0.000	2.363	2.641
x2	-5.456e-05	9e-06	-6.060	0.000	-7.22e-05	-3.69e-05
x3	-6.911e-07	2.66e-08	-25.939	0.000	-7.43e-07	-6.39e-07
x4	-0.0542	0.043	-1.249	0.212	-0.139	0.031
x5	0.0201	0.007	2.736	0.006	0.006	0.034

There has been a marginal improvement in the R2 value(from 0.1970 to 0.1969). Given the p values and z values of all the remaining variates, we will retain all the current present parameters defining a confusion matrix and estimating a misclassfication matrix.

In [10]:

```
yval=result.predict(predictors3)
ypred=yval>0.5
ypred=ypred.astype(int)
```

In [11]:

```
confusionm=metrics.confusion_matrix(response, ypred)
confusionm
```

Out[11]:

```
array([[2712, 1210],
       [ 495, 2031]], dtype=int64)
```

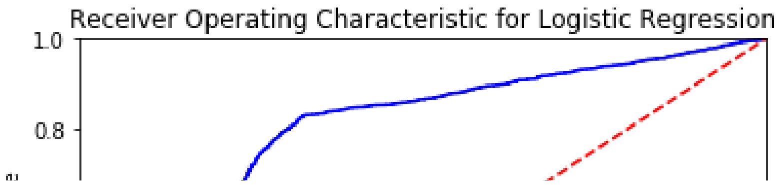
Based on the results of the confusion matrix and a threshold of 0.5, the logistic classifier shows thw following performance- 2712- switch(0) classified as switch(0) 1210- switch(1) classified as switch(0) 495- switch(0) classified as switch(1) 2031- switch(1) classified as switch(1)

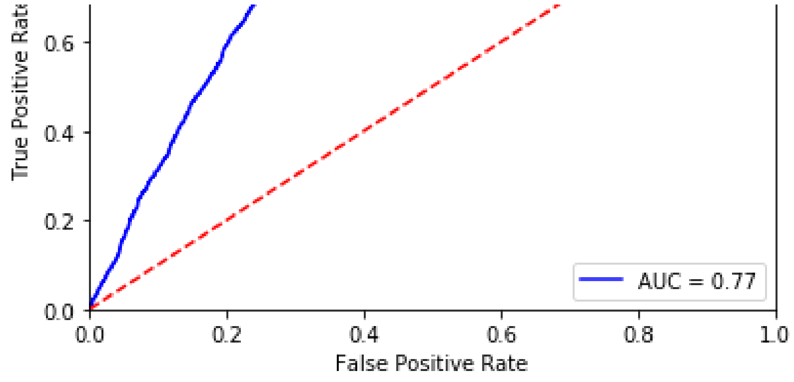
The types are errors are- 1) 1210 false negatives. 2) 495 false positives.

Merely changing the threshold values did not lead to any significant improvement, since it was found that there was a tradeoff involved between balancing the number of false negatives and false positives.

In [76]:

```
fpr, tpr, thresholds=metrics.roc_curve(response, yval)
roc_auc=metrics.auc(fpr, tpr)
plt.title('Receiver Operating Characteristic for Logistic Regression')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```





As it can be seen , the AUC is 0.77. The tradeoff between no. of true positives and the no. of false positives earlier mentioned can be sen in the graph above. The next section deals with examining the quality of the estimators used in the computation of the logistic regression model above, using cross-validation. Given the relatively large size of the dataset, the computationally expensive but more thorough LOOCV method is chosen. The LOOCV will also be used to eliminate points that do not contribute to the actual result of the switch response.

In [68]:

```
mse1=0.0
lgcv=LogisticRegression()
loocv=LeaveOneOut()

welldatacv=welldata.copy()
welldatacv=welldatacv.drop(['arsenic','x'], axis=1)

for trainindex, testindex in loocv.split(welldatacv.loc[:, 'unsafe':]):
    inp_train, inp_test= welldatacv.loc[trainindex, 'unsafe': ], welldatacv
.loc[testindex, 'unsafe': ]
    out_train, out_test= welldatacv.loc[trainindex, 'switch'], welldatacv.l
oc[testindex, 'switch']
    lgcv.fit(inp_train, out_train)
    ycv=lgcv.predict(inp_test)
    mse1=mse1+m.pow((ycv-out_test), 2)

mse1=m.sqrt(mse1)/len(welldatacv)
mse1
```

Out[68]:

```
0.007811513808599894
```

msecounter=0.0 welldataf=pd.DataFrame(columns=['switch', 'unsafe', 'distance', 'y', 'community', 'education']) lgcv.fit(welldatacv.loc[:,'unsafe':], welldatacv.loc[:,'switch'])

for i in range(0, len(welldatacv)-1): ycvf=lgcv.predict([welldatacv.loc[i, 'unsafe':]]) yactual=welldatacv.loc[i,'switch'] diff=m.sqrt(m.pow((ycvf-yactual),2)) if(diff<=1.5*mse1): welldataf=welldataf.append(welldatacv.loc[i]) else: pass

In [71]:

```
clf=LinearDiscriminantAnalysis()
clfx=welldatacv.loc[:, 'unsafe':]
clfy=welldatacv.loc[:, 'switch']
clf.fit(clfx, clfy)
```

Out[71]:

```
LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None,
              solver='svd', store_covariance=False, tol=0.0001)
```

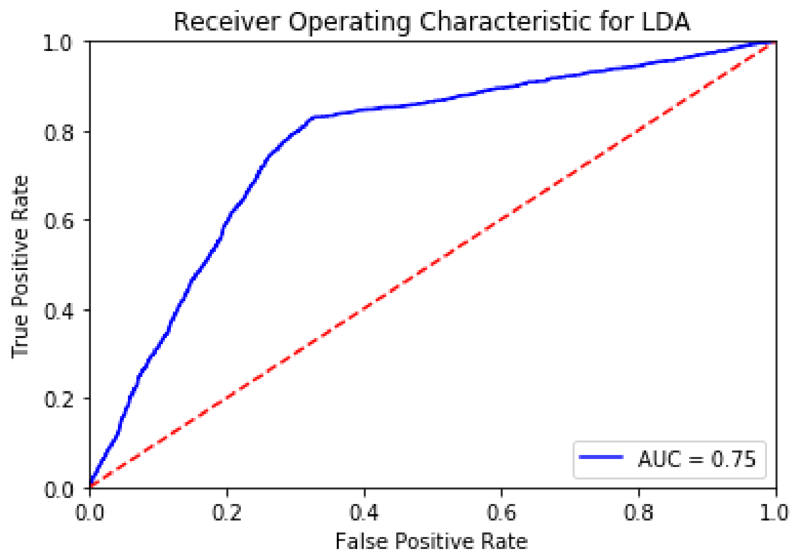
We now run LDA on the dataframe to compare the ROC curve between logistic regression and LDA.

In [77]:

```
clfpred=clf.predict(clfx)
```

In [75]:

```
fprclf, tprclf, thresholdsclf=metrics.roc_curve(clfy, clfpred)
roc_auc=metrics.auc(fprclf, tprclf)
plt.title('Receiver Operating Characteristic for LDA')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



On comparison between the LDA and logistic regression, we see that the latter performs marginally better with a AUC of 0.77 as compared to 0.75.

In [78]:

```
mse2=0.0
for trainindex, testindex in loocv.split(welldatacv.loc[:, 'unsafe':]):
    inp_train, inp_test= welldatacv.loc[trainindex, 'unsafe': ], welldatacv
.loc[testindex, 'unsafe': ]
    out_train, out_test= welldatacv.loc[trainindex, 'switch'], welldatacv.l
oc[testindex, 'switch']
    clf.fit(inp_train, out_train)
    ycv=clf.predict(inp_test)
    mse1=mse1+m.pow((ycv-out_test), 2)

mse2=m.sqrt(mse2)/len(welldatacv)
mse2
```

Out[78]:

```
0.0
```

In [79]:

```
confusionclf=metrics.confusion_matrix(clfy, clfpred)
confusionclf
```

Out[79]:

```
array([[2699, 1223],
       [ 486, 2040]], dtype=int64)
```