This assignment is based on implementing L1 and L2 loss functions using Logistic Regression

In [2]:

```
%matplotlib inline

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [26]:

```
dataset = pd.read_csv("C:/Users/ddalv/Downloads/data.csv")
X = dataset.iloc[:,[3,10]].values
y = dataset.iloc[:,11].values
dataset.head()
```

Out[26]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	s
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0

In [27]:

```
from sklearn.cross_validation import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

In [28]:

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Now we will implement L1 Loss Function

In [29]:

```
from sklearn.linear_model import LogisticRegression
classifier1 = LogisticRegression("l1",random_state=0)
classifier1.fit(X_train,y_train)
```

Out[29]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l1', random_state=0, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [30]:

```
y_pred1 = classifier1.predict(X_test)
y_pred1
```

Out[30]:

```
array([6, 5, 6, 5, 6, 5, 5, 6, 5, 5, 5, 5, 6, 6, 5, 6, 6, 6, 6, 5, 6, 5,
6,
       7, 5, 5, 5, 6, 5, 6, 5, 6, 6, 5, 6, 6, 5, 5, 6, 6, 5, 6, 7, 6, 6,
5,
       5, 6, 6, 6, 5, 5, 5, 7, 5, 5, 5, 5, 6, 5, 5, 6, 6, 6, 5, 6, 5, 6,
6,
       6, 5, 5, 5, 5, 5, 6, 5, 5, 5, 6, 6, 5, 6, 6, 5, 5, 7, 5, 5, 5, 5,
5,
       6, 5, 6, 5, 6, 5, 5, 5, 7, 6, 6, 6, 6, 5, 6, 5, 6, 5, 6, 5, 6, 5,
6,
       5, 5, 6, 6, 6, 5, 6, 5, 5, 6, 6, 5, 5, 5, 6, 5, 5, 6, 6, 5, 5, 5,
5,
       5, 5, 6, 5, 5, 5, 5, 5, 6, 6, 6, 5, 6, 6, 5, 5, 5, 6, 5, 5, 6, 6,
6,
       5, 6, 5, 6, 5, 6, 6, 5, 6, 6, 5, 5, 6, 7, 6, 6, 7, 6, 5, 5, 7, 5,
5,
       7, 5, 5, 6, 5, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 5, 5, 5, 5,
5,
       6, 6, 5, 6, 6, 5, 7, 5, 5, 5, 6, 5, 5, 5, 6, 6, 5, 5, 5, 6, 6, 5,
5,
       5, 6, 5, 6, 6, 6, 5, 6, 6, 6, 5, 5, 5, 5, 6, 5, 5, 5, 5, 6, 5, 5,
5,
       6, 5, 5, 5, 5, 5, 5, 5, 7, 5, 5, 5, 5, 5, 5, 5, 6, 5, 5, 5, 7, 6,
6,
       6, 5, 6, 6, 5, 6, 5, 6, 5, 5, 5, 5, 6, 5, 5, 6, 5, 5, 5, 6, 5, 5,
5,
       5, 6, 6, 5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 5, 5, 5, 6, 6, 6, 5, 6], dty
pe=int64)
```

In [31]:

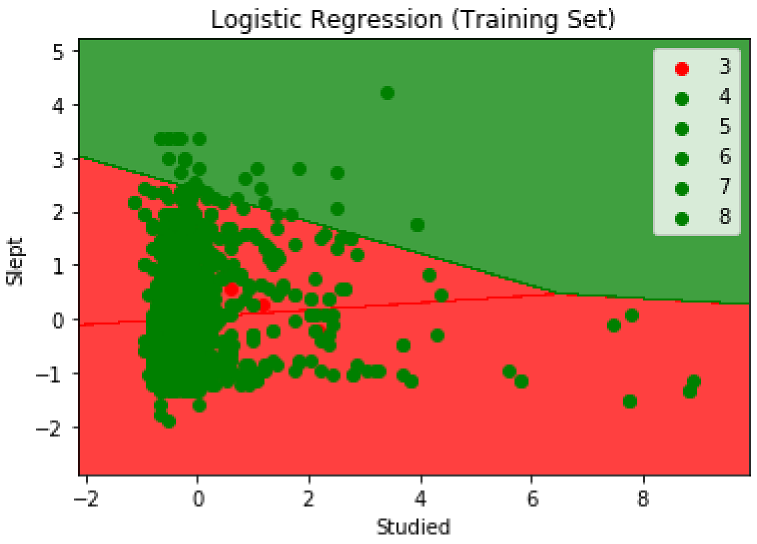
```
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test,y_pred1)
cm1
```

Out[31]:

```
array([[  0,   0,   2,   0,   0,   0],
       [  0,   0,   7,   2,   2,   0],
       [  0,   0, 111,  22,   2,   0],
       [  0,   0,  60,  77,   5,   0],
       [  0,   0,   2,  23,   2,   0],
       [  0,   0,   1,   1,   1,   0]], dtype=int64)
```

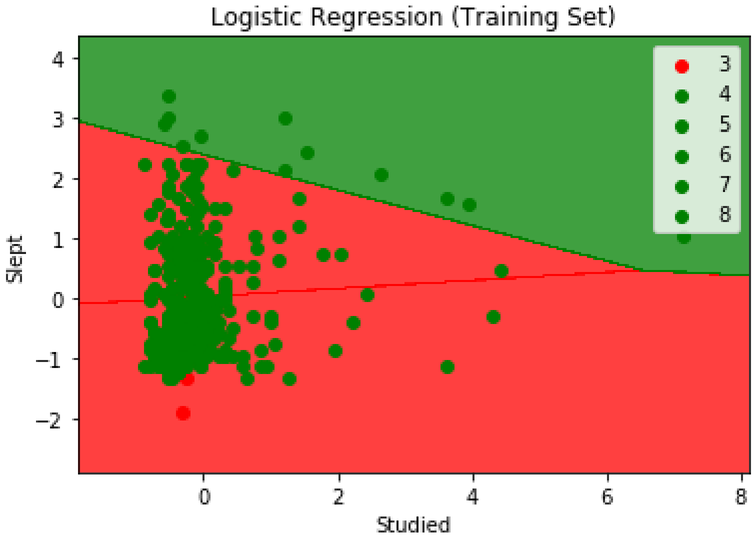
In [32]:

```
from matplotlib.colors import ListedColormap
X_set,y_set = X_train,y_train

X1,X2 = np.meshgrid(np.arange(start=X_set[:,0].min()-1, stop=X_set[:,0].max()+1, step=
0.01),
                   np.arange(start=X_set[:,1].min()-1, stop=X_set[:,1].max()+1, step=0.
01))

plt.contourf(X1,X2,classifier1.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.
shape),
            alpha=0.75,cmap=ListedColormap(('red','green')))

plt.xlim(X1.min(),X1.max())
plt.ylim(X2.min(),X2.max())

for i,j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set==j,0],X_set[y_set==j,1],
               c=ListedColormap(('red','green'))(i),label=j)

plt.title('Logistic Regression (Training Set)')
plt.xlabel('Studied')
plt.ylabel('Slept')
plt.legend()
plt.show()
```



In [33]:

```
from matplotlib.colors import ListedColormap
X_set,y_set = X_test,y_test

X1,X2 = np.meshgrid(np.arange(start=X_set[:,0].min()-1, stop=X_set[:,0].max()+1, step=
0.01),
                   np.arange(start=X_set[:,1].min()-1, stop=X_set[:,1].max()+1, step=0.
01))

plt.contourf(X1,X2,classifier1.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.
shape),
            alpha=0.75,cmap=ListedColormap(('red','green')))

plt.xlim(X1.min(),X1.max())
plt.ylim(X2.min(),X2.max())

for i,j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set==j,0],X_set[y_set==j,1],
               c=ListedColormap(('red','green'))(i),label=j)

plt.title('Logistic Regression (Training Set)')
plt.xlabel('Studied')
plt.ylabel('Slept')
plt.legend()
plt.show()
```



In [34]:

```
accuracy1 = 1.0 - (float(np.count_nonzero(y_pred1-y_test)) / len(y_pred1-y_test))
accuracy1
```

Out[34]:

```
0.59375
```

Now we will implement L2 Loss Function

In [35]:

```
from sklearn.linear_model import LogisticRegression
classifier2 = LogisticRegression("l2",random_state=0)
classifier2.fit(X_train,y_train)
```

Out[35]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [36]:

```
y_pred2 = classifier2.predict(X_test)
y_pred2
```

Out[36]:

```
array([6, 5, 6, 5, 6, 5, 5, 6, 5, 5, 5, 5, 6, 6, 5, 6, 6, 6, 6, 5, 6, 5,
6,
       7, 5, 5, 5, 6, 5, 6, 5, 6, 6, 5, 6, 6, 5, 5, 6, 6, 5, 6, 7, 6, 6,
5,
       5, 6, 6, 6, 5, 5, 5, 7, 5, 5, 5, 5, 6, 5, 5, 6, 6, 6, 5, 6, 5, 6,
6,
       6, 5, 5, 5, 5, 5, 6, 5, 5, 5, 6, 6, 5, 6, 6, 5, 5, 7, 5, 5, 5, 5,
5,
       6, 5, 6, 5, 6, 5, 5, 5, 7, 6, 6, 6, 6, 5, 6, 5, 6, 5, 6, 5, 6, 5,
6,
       5, 5, 6, 6, 6, 5, 6, 5, 5, 6, 6, 5, 5, 5, 6, 5, 5, 6, 6, 5, 5, 5,
5,
       5, 5, 6, 5, 5, 5, 5, 5, 6, 6, 6, 5, 6, 6, 5, 5, 5, 6, 5, 5, 6, 6,
6,
       5, 6, 5, 6, 5, 6, 6, 5, 6, 6, 5, 5, 6, 7, 6, 6, 7, 6, 5, 5, 7, 5,
5,
       7, 5, 5, 6, 5, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 5, 5, 5, 5,
5,
       6, 6, 5, 6, 6, 5, 7, 5, 5, 5, 6, 5, 5, 5, 6, 6, 5, 5, 5, 6, 6, 5,
5,
       5, 6, 5, 6, 6, 6, 5, 6, 6, 6, 5, 5, 5, 5, 6, 5, 5, 5, 5, 6, 5, 5,
5,
       6, 5, 5, 5, 5, 5, 5, 5, 7, 5, 5, 5, 5, 5, 5, 5, 6, 5, 5, 5, 7, 6,
6,
       6, 5, 6, 6, 5, 6, 5, 6, 5, 5, 5, 5, 6, 5, 5, 6, 5, 5, 5, 6, 5, 5,
5,
       5, 6, 6, 5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 5, 5, 5, 6, 6, 6, 5, 6], dty
pe=int64)
```

In [37]:

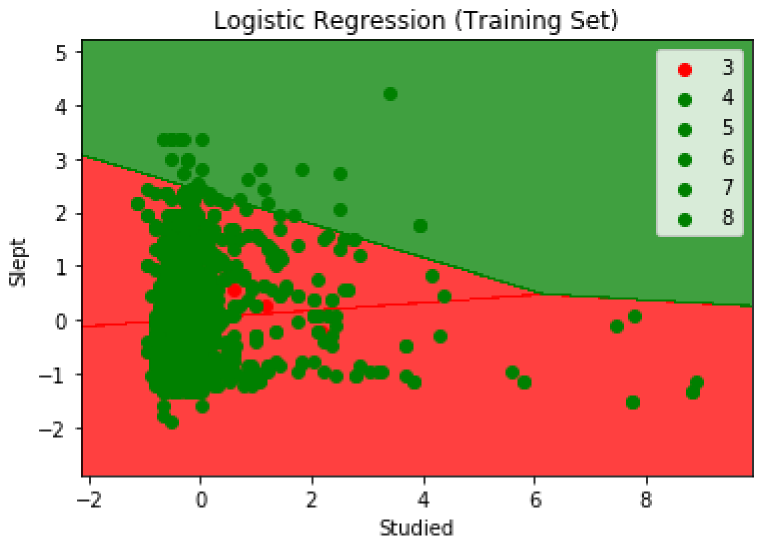
```
from sklearn.metrics import confusion_matrix
cm2 = confusion_matrix(y_test,y_pred2)
cm2
```

Out[37]:

```
array([[  0,   0,   2,   0,   0,   0],
       [  0,   0,   7,   2,   2,   0],
       [  0,   0, 111,  22,   2,   0],
       [  0,   0,  60,  77,   5,   0],
       [  0,   0,   2,  23,   2,   0],
       [  0,   0,   1,   1,   1,   0]], dtype=int64)
```

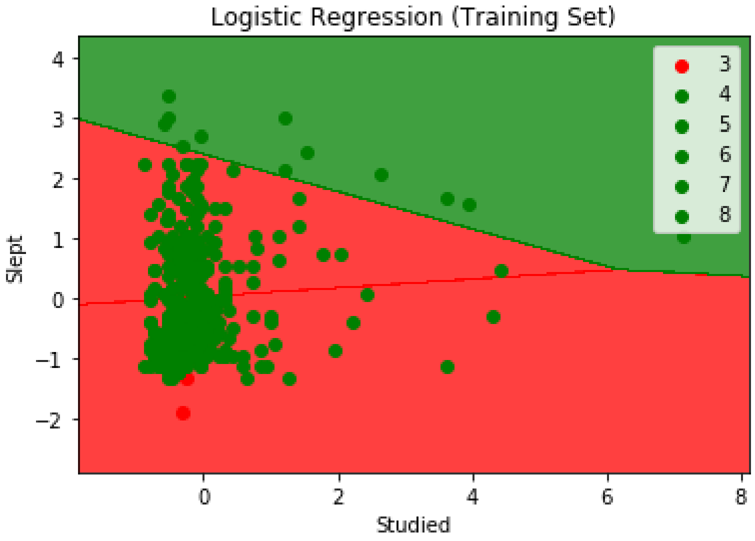
In [38]:

```
from matplotlib.colors import ListedColormap
X_set,y_set = X_train,y_train

X1,X2 = np.meshgrid(np.arange(start=X_set[:,0].min()-1, stop=X_set[:,0].max()+1, step=
0.01),
                   np.arange(start=X_set[:,1].min()-1, stop=X_set[:,1].max()+1, step=0.
01))

plt.contourf(X1,X2,classifier2.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.
shape),
            alpha=0.75,cmap=ListedColormap(('red','green')))

plt.xlim(X1.min(),X1.max())
plt.ylim(X2.min(),X2.max())

for i,j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set==j,0],X_set[y_set==j,1],
               c=ListedColormap(('red','green'))(i),label=j)

plt.title('Logistic Regression (Training Set)')
plt.xlabel('Studied')
plt.ylabel('Slept')
plt.legend()
plt.show()
```



In [39]:

```
from matplotlib.colors import ListedColormap
X_set,y_set = X_test,y_test

X1,X2 = np.meshgrid(np.arange(start=X_set[:,0].min()-1, stop=X_set[:,0].max()+1, step=
0.01),
                   np.arange(start=X_set[:,1].min()-1, stop=X_set[:,1].max()+1, step=0.
01))

plt.contourf(X1,X2,classifier2.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.
shape),
            alpha=0.75,cmap=ListedColormap(('red','green')))

plt.xlim(X1.min(),X1.max())
plt.ylim(X2.min(),X2.max())

for i,j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set==j,0],X_set[y_set==j,1],
               c=ListedColormap(('red','green'))(i),label=j)

plt.title('Logistic Regression (Training Set)')
plt.xlabel('Studied')
plt.ylabel('Slept')
plt.legend()
plt.show()
```



In [40]:

```
accuracy2 = 1.0 - (float(np.count_nonzero(y_pred2-y_test)) / len(y_pred2-y_test))
accuracy2
```

Out[40]:

```
0.59375
```

Though the accuracy of both the models are same, after observing the graphs we can say that L1 function fits best to our data set than L2 loss function.