Lab 2: Linear Classification and Logistic Regression

```
def predict_lc(x, theta):
'''
input x: np.ndarray of shape (m, 3)
input theta: np.ndarray of shape (3, 1)
output y: np.ndarray of shape (m, 1)
'''
h = np.dot(x,theta) # dim (m,1)
y = np.sign(h) # dim (m,1)
return y
```

```
def mse(y_true, y_pred):
'''
input y_true: np.ndarray of shape (m, 1)
input y_pred: np.ndarray of shape (m, 1)
'''
# Insert your code here ~ 1-3 lines
m = y_true.shape[0]
res2 = (y_true - y_pred)**2 # dim (m,1)
J = np.sum(res2) / (2*m)
return J
```

```
def gradient(y_true, y_pred, x):
'''
input y_true: np.ndarray of shape (m,)
input y_pred: np.ndarray of shape (m,)
input x: np.ndarray of shape (m, 3)
output dJ: np.array of shape (3, 1)
'''
# Reshape arrays
y_true = y_true.reshape(-1, 1) # now shape (m, 1)
y_pred = y_pred.reshape(-1, 1) # now shape (m, 1)

# Your code here ~ 1-6 lines
m = y_true.shape[0]
res = y_pred - y_true # dim (m,1)
dJ = np.dot(x.T, res) / m # dim (3,1)
return dJ
```

```python
def sigmoid(z):
'''
input z: np.ndaray of shape (m, 3)
output s: np.ndarray of shape (m, 3) where s[i, j] = g(z[i, j])
'''
# Insert your code here ~ 1-6 line
s = 1.0 / (1.0 + np.exp(-z))
return s
```

```python
def xent(y_true, y_pred):
'''
input y_true: np.ndarray of shape (m,)
input y_pred: np.ndarray of shape (m,)
output J: float
'''
# Insert your code here ~ 1-6 lines
m = y_true.shape[0]
a = y_true*np.log(y_pred) # dim (m,)
b = (1-y_true)*np.log(1-y_pred) # dim (m,)
c = -a -b # dim (m,)
J = np.sum(c) / m
return J
```

```python
def predict_lr(x, theta):
# Insert your code here ~ 1-3 lines
h = sigmoid(np.dot(x,theta)) # dim (m,1)
y_pred = np.round(h) # dim (m,1)
return y_pred
```