Lab2 Solution: Linear Classification and Logistic Regression

```python
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 = x.dot(theta)
    y = 1.0 * (h >= 0) - 1.0 * (h < 0)
    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)
    '''
    J = ((y_true - y_pred) ** 2).sum() / (2 * len(y_true))
    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)
    dJ = x.T.dot((y_pred - y_true)) / len(x)
    return dJ

def sigmoid(z):
    '''
    input z: np.ndaray of shape (m, n)
    output s: np.ndarray of shape (m, n) where s[i, j] = g(z[i, j])
    '''
    s = 1.0 / ( 1.0 + np.exp(- z) )
    return s
```

```
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
    '''
    J = - (y_true * np.log(y_pred) + \
           (1.0 - y_true) * np.log(1.0 - y_pred)).mean()
    return J

def predict_lr(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)
    '''
    z = x.dot(theta)
    h = sigmoid(z)
    y = np.round(h)
    return y
```