

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4
5 def add_intercept(x):
6     """Add intercept to matrix x.
7
8     Args:
9         x: 2D NumPy array.
10
11     Returns:
12         New matrix same as x with 1's in the 0th column.
13     """
14     new_x = np.zeros((x.shape[0], x.shape[1] + 1), dtype=x.dtype)
15     new_x[:, 0] = 1
16     new_x[:, 1:] = x
17
18     return new_x
19
20
21 def load_dataset(csv_path, label_col='y', add_intercept=False):
22     """Load dataset from a CSV file.
23
24     Args:
25         csv_path: Path to CSV file containing dataset.
26         label_col: Name of column to use as labels (should be 'y' or 't').
27         add_intercept: Add an intercept entry to x-values.
28
29     Returns:
30         xs: Numpy array of x-values (inputs).
31         ys: Numpy array of y-values (labels).
32     """
33
34     def add_intercept_fn(x):
35         global add_intercept
36         return add_intercept(x)
37
38     # Validate label_col argument
39     allowed_label_cols = ('y', 't')
40     if label_col not in allowed_label_cols:
41         raise ValueError('Invalid label_col: {} (expected {})'.format(
42             label_col, allowed_label_cols))
43
44     # Load headers
45     with open(csv_path, 'r') as csv_fh:
46         headers = csv_fh.readline().strip().split(',')
47
48     # Load features and labels
49     x_cols = [i for i in range(len(headers)) if headers[i].startswith('x')]
50     l_cols = [i for i in range(len(headers)) if headers[i] == label_col]
51     inputs = np.loadtxt(csv_path, delimiter=',', skiprows=1, usecols=x_cols)
52     labels = np.loadtxt(csv_path, delimiter=',', skiprows=1, usecols=l_cols)
53
54     if inputs.ndim == 1:
55         inputs = np.expand_dims(inputs, -1)
56
57     if add_intercept:
58         inputs = add_intercept_fn(inputs)
59
60     return inputs, labels
61
62
63 def plot(x, y, theta, save_path, correction=1.0):
64     """Plot dataset and fitted logistic regression parameters.
65
66     Args:
67         x: Matrix of training examples, one per row.
68         y: Vector of labels in {0, 1}.
69         theta: Vector of parameters for logistic regression model.
70         save_path: Path to save the plot.
71         correction: Correction factor to apply, if any.
72     """

```

```
73 # Plot dataset
74 plt.figure()
75 plt.plot(x[y == 1, -2], x[y == 1, -1], 'bx', linewidth=2)
76 plt.plot(x[y == 0, -2], x[y == 0, -1], 'go', linewidth=2)
77
78 # Plot decision boundary (found by solving for  $\theta^T x = 0$ )
79 x1 = np.arange(min(x[:, -2]), max(x[:, -2]), 0.01)
80 x2 = -(theta[0] / theta[2] + theta[1] / theta[2] * x1
81         + np.log((2 - correction) / correction) / theta[2])
82 plt.plot(x1, x2, c='red', linewidth=2)
83 plt.xlim(x[:, -2].min()-.1, x[:, -2].max()+.1)
84 plt.ylim(x[:, -1].min()-.1, x[:, -1].max()+.1)
85
86 # Add labels and save to disk
87 plt.xlabel('x1')
88 plt.ylabel('x2')
89 plt.savefig(save_path)
```