```python
import numpy as np
import util
import sys

sys.path.append('../linearclass')

### NOTE : You need to complete logreg implementation first!

from logreg import LogisticRegression

# Character to replace with sub-problem letter in plot_path/save_path
WILDCARD = 'X'


def main(train_path, valid_path, test_path, save_path):
    """Problem 2: Logistic regression for incomplete, positive-only labels.

    Run under the following conditions:
        1. on t-labels,
        2. on y-labels,
        3. on y-labels with correction factor alpha.

    Args:
        train_path: Path to CSV file containing training set.
        valid_path: Path to CSV file containing validation set.
        test_path: Path to CSV file containing test set.
        save_path: Path to save predictions.
    """
    output_path_true = save_path.replace(WILDCARD, 'true')
    output_path_naive = save_path.replace(WILDCARD, 'naive')
    output_path_adjusted = save_path.replace(WILDCARD, 'adjusted')

    # *** START CODE HERE ***
    plot_path = save_path.replace('.txt', '.png')
    plot_path_true = plot_path.replace(WILDCARD, 'true')
    plot_path_naive = plot_path.replace(WILDCARD, 'naive')
    plot_path_adjusted = plot_path.replace(WILDCARD, 'adjusted')

    # Part (a): Train and test on true labels
    # Make sure to save predicted probabilities to output_path_true using np.savetxt()

    x_train, t_train = util.load_dataset(train_path, label_col='t',
                                         add_intercept=True)
    clf = LogisticRegression()
    clf.fit(x_train, t_train)

    x_test, t_test = util.load_dataset(test_path, label_col='t',
                                       add_intercept=True)
    p_test = clf.predict(x_test)
    np.savetxt(output_path_true, p_test)
    util.plot(x_test, t_test, clf.theta, plot_path_true)

    # Part (b): Train on y-labels and test on true labels
    # Make sure to save predicted probabilities to output_path_naive using np.savetxt()

    x_train, y_train = util.load_dataset(train_path, label_col='y',
                                         add_intercept=True)
    clf = LogisticRegression()
    clf.fit(x_train, y_train)
    x_test, t_test = util.load_dataset(test_path, label_col='t',
                                       add_intercept=True)
    p_test = clf.predict(x_test)
    np.savetxt(output_path_naive, p_test)
    util.plot(x_test, t_test, clf.theta, plot_path_naive)

    # Part (f): Apply correction factor using validation set and test on true labels

    x_valid, y_valid = util.load_dataset(valid_path, label_col='y')
    x_valid = x_valid[y_valid == 1, :]  # Restrict to just the labeled examples
    x_valid = util.add_intercept(x_valid)
    y_pred = clf.predict(x_valid)
    alpha = np.mean(y_pred)
    print('Found alpha = {}'.format(alpha))
    x_test, t_test = util.load_dataset(test_path, label_col='t',
                                       add_intercept=True)

    # Plot and use np.savetxt to save outputs to output_path_adjusted
    np.savetxt(output_path_adjusted, p_test / alpha)
    util.plot(x_test, t_test, clf.theta, plot_path_adjusted, correction=alpha)

    # *** END CODER HERE

if __name__ == '__main__':
    main(train_path='train.csv',
        valid_path='valid.csv',
        test_path='test.csv',
        save_path='posonly_X_pred.txt')
```