Introduction to Machine Learning (CS 135)
Project 01 (80 points)

For this assignment, you will explore the use of logistic regression for image classification. You will hand in a PDF containing results and analysis, along with the usual collaborators file; you ***do not*** hand in code for this assignment. You will also submit a text-file containing predictions of a classifier that you built; these results will be used to score your code on a leaderboard, based upon accuracy on a provided testing set (for which you do not know the correct outputs).

Part Zero: Collaborators file (5 points)

Provide the usual file containing your name, the amount of time you worked on the assignment, and any resources or individuals you consulted in your work.

Part One: Logistic Regression for Digit Classification (25 points)

You have been given data (in `data_digits_8_vs_9_noisy`) corresponding to images of handwritten digits (8 and 9 in particular).* As before, this data has been split into various training, and testing sets; each set is given in CSV form, and is divided into inputs (`x`) and outputs (`y`).

Each row of the input data consists of pixel data from a (28×28) image with gray-scale values between 0.0 (black) and 1.0 (white); this pixel data is thus represented as a single feature-vector of length $28^2 = 784$ such values. The output data is a binary label, with `0` representing an `8`, and `1` representing a `9`.

1. (*5 pts.*) You will fit logistic regression models to the training data, using `sklearn`'s implementation of the model, with the `liblinear` solver:

   https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

   Leaving all other parameters with default values, you will explore what happens when we limit the iterations allowed for the solver to converge on its solution.

   For the values $i = 1, 2, \ldots, 40$, build a logistic regression model with the `max_iter` set to i. Fit each such model to the training data, and keep track of the *accuracy* of the resulting model (via the model's own `score()` function) along with the *logistic loss*, each measured on the training data.†

   Produce two plots, each with the values of i as x-axis and with the accuracy/loss, respectively, as y. Place these plots into your PDF document, with captions labeling each appropriately. Below the plots, discuss the results you are seeing; what do they show, and why?

*This is based upon the popular MNIST data-set, from the work of LeCun, Cortes, and Burges. To add to the challenge, the data has been preprocessed to add some random noise to each image.

†When doing this, you will probably see warnings about non-convergence for lower values of parameter i. You can ignore these warnings, as they are expected. You can measure the loss using https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html

2. (*5 pts.*) After fitting a logistic model, you can access the weights it assigns to each feature in the data using its `coef_` attribute. For each of the i models you generated, record the first such weight, which is the one the model applies to feature `pixel000` in the input data. Produce a plot with the values of i as x-axis and with the feature weight as y. Place this plot into your PDF document, with a caption labeling it appropriately. Below the plot, discuss the results you are seeing; what do they show, and why?

3. (*5 pts.*) As in prior homework assignments, you will explore different values of regularization penalty for the logistic model.[‡] Your code should explore a range of values for this parameter, using a regularly-spaced grid of values:

```
C_grid = np.logspace(-9, 6, 31)

for C in C_grid:
    # Build and evaluate model for each value C
```

   For each such value of `C` create a model and fit it to the training data, and then compute the log loss of that model on the test data. Determine which value gives you the least loss on the test data. Record that value, along with the accuracy score of the model, in your PDF. Also include a table for the confusion matrix of that model on the test data.

4. (*5 pts.*) Analyze some of the mistakes that your best model makes. Produce two plots, one consisting of 9 sample images that are *false positives* in the test set, and one consisting of 9 *false negatives*. You can display the images by converting the pixel data using the `matplotlib` function `imshow()`, using the `Grey` colormap, with `vmin=0.0` and `vmax=1.0`. Place each plot into your PDF as a properly captioned figure. Below the figures, discuss the results you are seeing. What mistakes is the classifier making?

5. (*5 pts.*) Analyze all of the final weights produced by your classifier. Reshape the weight coefficients into a (28×28) matrix, corresponding to the pixels of the original images, and plot the result using `imshow()`, with colormap `RdYlBu`, `vmin=-0.5`, and `vmax=0.5`.[§] Place this plot into your PDF as a properly captioned figure. Below it, discuss what it shows. Which pixels correspond to an `8` (have negative weights), and which correspond to a `9` (have positive weights)? Why do you think this is the case?

[‡] As always should be the case, make sure to read the model documentation—in particular, note that the regularization parameter, `C`, is an *inverse* penalty (knowing this is important to interpreting and discussing your results).

[§] The use of that particular colormap will allow us to compare results across different submissions more easily. If you have a visual impairment that makes the output difficult to parse, please do feel free to replace it with another one that is more conducive to analysis (grayscale is always a possibility).

Part Two: Trousers v. Dresses (45 points)

We have also provided some image data, in the same format as before, of trousers (output label `1`) and dresses (output label `0`).[¶] We have again given you input and output data for a training set, along with ***input data only*** for a test set. Your task is to build a logistic regression classifier for this data. Your PDF for this part will describe your approach and the results you see.

When doing regression, you should explore different ***feature transformations***, transforming the input features (in any way you see fit) that are given to the regression classifier. Your PDF should explain, as completely as you are able, what feature transformations you tried, and what processes you used to build your classifier (parameters you tried like regularization penalty, etc.), along with the reasoning behind your decisions. Your work should contain at least two figures comparing the results you get by regression using the original features of the data and some modified features. Your discussion should include the error rate on the testing data for various models, provided when you submit the model's predictions on that data (see next section). Overall, the entire write-up for this part of the assignment should take 2–3 pages, including figures.

For this part of the assignment, ***process*** is more important than raw results. A well-designed set of tests, with coherent explanation and careful comparison of results will be worth more than something that achieves 0 error, but is not explained clearly.

You may use any transformations of the existing data you like. ***Do not*** use any additional sources of data; use only the input sets provided, and feature transformations on those sets. Be creative in thinking about how to transform data; some ideas you might consider (we encourage you to try other things as well):

- Consider things like histograms of parts of the data.
- Consider adding features that count overall numbers of white or black pixels.
- Consider adding features that capture spatial patterns in the original data.
- Consider exploring data augmentation, where you add to your data set via transformations of the data. For example, if you flip each image horizontally, you can double the training set size without needing fundamentally new data.

[¶]These are taken from the Fashion MNIST data-set, originally released by Zalando research. Some noise has been added to the original data.

Part Three: Prediction submission (5 points)

To test your regression classifiers for the clothing data, you can use them to generate the predicted probabilities for the data in the testing input set. You can then upload those predictions in the form of a text-file to the relevant Gradescope link, where the autograder will compare your results to the correct results (which it knows, while you don't) and compute and display the overall quality of your predictions.

You can update your submission at any time up to the deadline, in attempts to improve predictive accuracy. A leaderboard will display the results, in order from best to worst. The autograder will also assign points based on overall leaderboard position.

The submission should be in the form of a text-file, `yproba1_test.txt`, containing one probability value (the probability of a positive binary label, `1`) per example in the test input. Code like the following can be used to produce that file:

```
x_test = np.loadtxt('data_trouser_dress/troudress_test_x.csv')
yproba1_test = model.predict_proba(x_test)[:, 1]
np.savetxt('yproba1_test.txt', yproba1_test)
```

Note: to make the automated scoring work, your file-name must match the one given here exactly. You can upload the file directly by dragging and dropping into Gradescope.

Submission details

There are ***three*** Gradescope submission links for this assignment. You will upload the following:

- **Collaborators information**: Submit the usual `COLLABORATORS.txt` file, containing your name, amount of time spent on the project, and persons/resources consulted.
- **PDF**: Submit the PDF containing all the required figures and discussion.
- **Prediction**: Submit a text-file containing your predictions on the sneakers/sandals test data. This will be auto-graded, and results will be placed onto a leaderboard.