Hertie School, Spring 2024
Machine Learning (Professor Dimmery)
Problem Set 1, Due Monday, February 26

In this problem set, we will use some of the tools you've learned in the first few lectures and labs. You should prepare your solutions in a Jupyter notebook, writing and running the appropriate code in the notebook (please no separate script files, as I want to see all of your work in a single document). Provide this document to me on Moodle in a single pdf.

You should prepare all of your writeup and code in this document, formatting it nicely: as if you were going to be sharing the results with your colleagues and boss. What does this mean?

- Clearly label your solutions.
- Solutions which require explanation should use complete sentences.
- If you use code or materials from anywhere else, that should be clearly labeled and cited.
- If you are asked to prepare graphs or tables of output, they should be labeled clearly. A good rule of thumb is that charts should be self-explanatory: they should not require reading the text in order to understand.

To export a Jupyter notebook to a pdf, the easiest solution is to run all of your code and then Print the page to a pdf. Depending on your computer's setup, you may be able to explicitly export using Jupyter's tools through the File - Download as - pdf via LaTeX (.pdf) menu (or one of the other exporters that ends in (.pdf). If you can't make this work (for further details, this blog has some additional instructions that might help), then just print it to pdf. This won't affect your grade, what I care about here is the content.

Note that each question in this problem set has a number of points associated with it. The maximum number of points on this assignment is 15 which, unsurprisingly, corresponds to the number of points on your final grade.

As a reminder, I don't mind if you work together with classmates when you get stuck, but the work you submit should be purely your own.

We're going to start by looking at linear regression, going over what we talked about in class. For the exercise, we're going to use a dataset about the US presidential election in 1996. There is further detail on this dataset in the You can load this data using the following Python code:

```python
import statsmodels.api as sm
data = sm.datasets.anes96.load_pandas()
```

This code will give you a Pandas dataset. Use what you've learned in the labs to manipulate this data into the appropriate formats for carrying out the tasks in this exercise. The particular task we will be considering is predicting `vote` based on five features: `TVnews`, `PID`, `age`, `educ`, and `popul`.

1. (1 point) Calculate summary statistics for the label and five features described above. Pay attention to the meaning of each variable and present a summary of it that makes sense given how it is coded.

2. (1 point) What is the formula for the closed form estimate of the coefficient vector in ordinary least squares regression? Estimate the coefficients using numpy in Python by performing the matrix operations from the closed form solution we worked out in class (Page 14 of Lecture 2). (Be sure to include an intercept!)

3. (1 point) Estimate the coefficients using the `statsmodels` package (sm.OLS documentation). Compare them.

4. (1 point) Now, think about the model you just estimated. In class, we talked about two assumptions we could use to motivate estimation of the variance of this coefficient vector. Which would you choose and why?

5. (1 point) Estimate the variance of these coefficients using the matrix formula. Note: We derived this together on Pages 16-17 of Lecture 2.

6. Create a table showing, for each feature, j, the estimate ($\hat{\beta}_j$), the standard error $\sqrt{\hat{\text{Var}}(\hat{\beta})_{jj}}$, and the upper and lower bounds of the 95% confidence interval ($\hat{\beta}_j \pm z_\alpha \sqrt{\hat{\text{Var}}(\hat{\beta})_{jj}}$). Compare the variance to what you got from statsmodels. What assumption are they using on the variance?

   Note: you can get the constant $z_{1-\frac{\alpha}{2}}$ from scipy with `scipy.stats.norm.ppf(0.975)`. This assumes you have `import scipy` at the top of your notebook with the rest of your imports.

   Up to now, we were doing something that might seem weird: we were estimating a linear regression on labels that were binary. If we care about the task we said we did, shouldn't we treat it as classification? That's what we'll do next.

7. (1 point) Write a function with three arguments:

   - `beta`: A 1D numpy array representing a particular value of your coefficients, β.
   - `label`: A 1D numpy array of the labels in your dataset.
   - `features`: A 2D numpy array representing the features in your dataset.

   This function should output a single number, the negative log-likelihood evaluated at the chosen value of β.

8. (1 point) Using the SciPy library, minimize the objective function we discussed in class for logistic regression. See documentation here. For a function called `nll`, and with your label saved as a numpy array in a variable named `y` and your feature matrix saved as a numpy array in a variable named `X`, the function call would look like:

```
    opt_result = scipy.optimize.minimize(
        nll, args=(X, y), x0 = [0] * 6, method='BFGS'
    )
    beta_logistic = opt_result.x
```

These lines of code will minimize your negative log likelihood and save the optimal value of beta as a numpy array in a variable named `beta_logistic`.

Note: you will run into problems here due to floating point! You should see `RuntimeWarnings` complaining about division by zero. Division by zero means $\frac{1}{0} = \infty$. Why might you get an ∞ in this code? Well, we know that $\log(0) = -\infty$[1]. To resolve this problem, you will need to ensure in your `nll` function that μ_i is always at least as large as the smallest floating point number. Numpy will tell you what this smallest representable fraction is with the command `np.finfo(float).eps`. You can use this to construct a `nll` function that is robust to this division-by-zero bug.

9. (1 point) Now you can construct your predictions by taking the dot-product between `beta_logistic` and your feature matrix and then passing that dot-product through the sigmoid function (defined on page 27 of Lecture 2). This provides an estimate of the probability of class membership. Also calculate the most likely class for each unit by predicting a 1 when $p(y_i|x_i; \beta) > 0.5$ (i.e. the Heaviside function).

   Note: If you absolutely cannot get the previous question working, you can alternatively scikit-learn to construct your estimates of logistic regression in lieu of calculating it by hand: sklearn's logistic regression.

10. Construct class estimates for your OLS predictions as well by calculating $1(X\beta_{ols} > 0.5)$ (i.e. output a 1 if the OLS predicted value is greater than 0.5).

11. (1 point) Calculate the full confusion matrix for the logistic regression and the OLS model.

    Note: An easy way to construct a contingency table is by using a function in SciPy: `scipy.stats.contingency.crosstab` .

12. (1 point) Plot the relationship between the predictions from the linear regression in Question 1 (on the x-axis) and the predictions from the logistic regression (on the y-axis). What do you see?

For the next three questions, say whether the following problem sounds more like a supervised learning problem, an unsupervised learning problem or a reinforcement learning problem. Look back at the definitions and examples from the first Lecture if you aren't sure.

For these scenarios, imagine that you are data scientists working at a social media company interested in how users interact with the platform. You're particularly interested in how users interact with "stories" on the platform. Stories might be anything that shows up on their feed: links, videos, pictures or updates.

[1] Technically, we should only talk about this in terms of limits, i.e. $\lim_{t\downarrow 0} \log(t) = -\infty$.

13. (1 point) Separating users into communities based on the kind of stories they engage with and other users they interact with.
14. (1 point) Predicting whether users will click on a story or not based on their past behavior.
15. (1 point) Choosing which story to show a user in order to keep them active on the platform for longer.