# INFO251 – Applied Machine Learning

Lab 3

**Suraj R. Nair**

# Announcements

- Problem Set 2 due Feb 6!

# Today

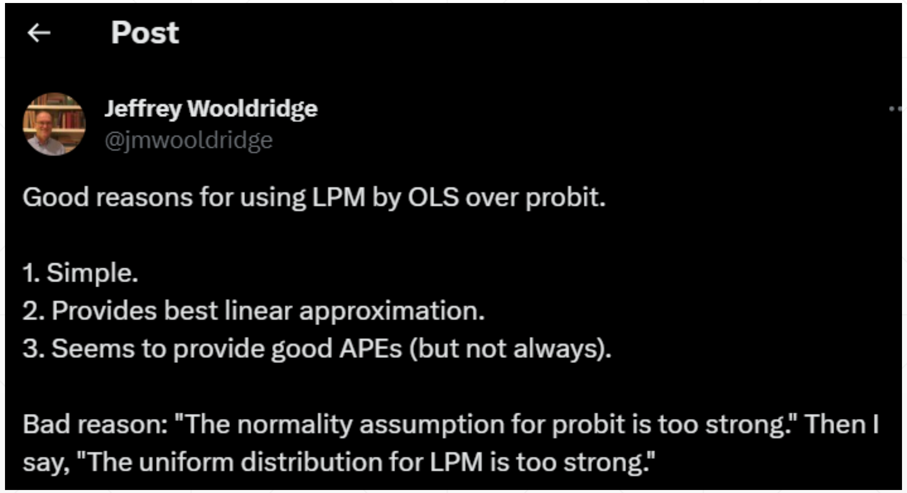
- Wrap up pending material from Lab 2
- Vectorized computation + Matrix handling
- Today’s programming tool: `numpy`

# Review: Linear Probability Model

- Suppose we run a regression of the form (where Y is binary)

$$Y_i = \alpha + \beta x_i + u_i$$

- β expresses the change in P(Y = 1) for a unit change in x
- Concerns:
  - Heteroskedasticity
  - Predicted values can lie outside [0, 1]



Jeffrey Wooldridge
@jmwooldridge

Good reasons for using LPM by OLS over probit.

1. Simple.
2. Provides best linear approximation.
3. Seems to provide good APEs (but not always).

Bad reason: "The normality assumption for probit is too strong." Then I say, "The uniform distribution for LPM is too strong."

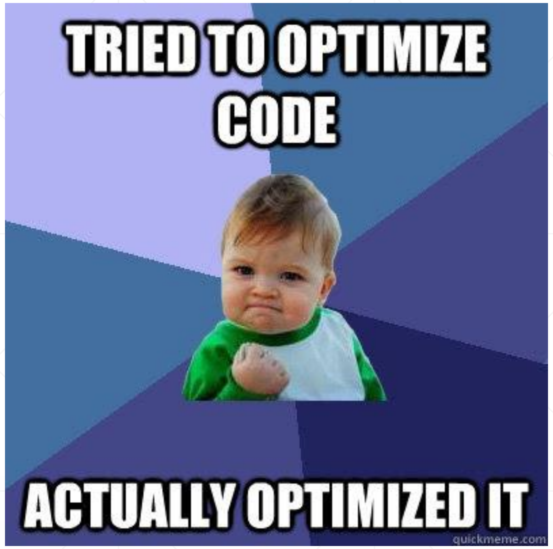
# Review: Dummy Variables / Interactions

- Suppose we have two cities (New York, San Francisco), and data on the quantity and price of bagels from various bagel shops (indexed by ***i***) in each city.
- Compare the following regressions:
  - $Quantity_i = \alpha + \beta_1 Price_i + e_i$ (separate regression for each city)
  - $Quantity_i = \alpha + \beta_2 Price_i + \gamma SF_i + u_i$ (single regression, SF is a dummy)
  - $Quantity_i = \alpha + \beta_3 Price_i * SF_i + \delta_1 Price_i + \delta_2 SF_i + v_i$ (add an interaction term)

# Vectorized Computation

- Efficient vectorized computation
- Creating and manipulating matrices in Python
- Matrix operations: Addition, multiplication, dot product

Today’s programming tool: `numpy`

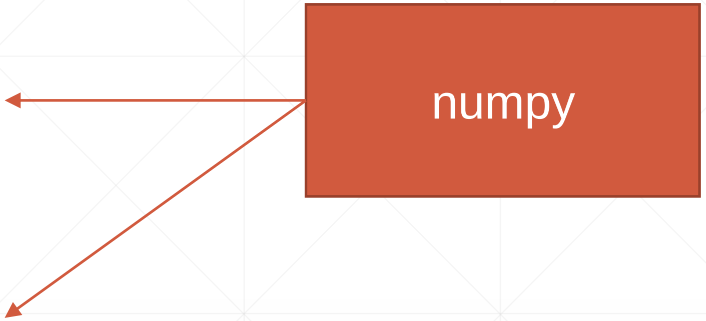


# How to make a program run fast

- Programming language
  - **Fast:** C, C++, Java, Lisp/OCaml
  - **Slow:** Python
  - **Very slow:** R
- Writing efficient code
  - For loops vs. vectorized computation
- Hardware and parallelization
  - Run parts of a program in parallel on separate cores -- on a single machine or in a distributed system
  - Software packages for parallelizing data analysis in python: `pyspark, dask`
  - For more: **CS267**

# How to make a program run fast

- Programming language
  - **Fast:** C, C++, Java, Lisp/OCaml
  - **Slow:** Python
  - **Very slow:** R
- Writing efficient code
  - For loops vs. vectorized computation

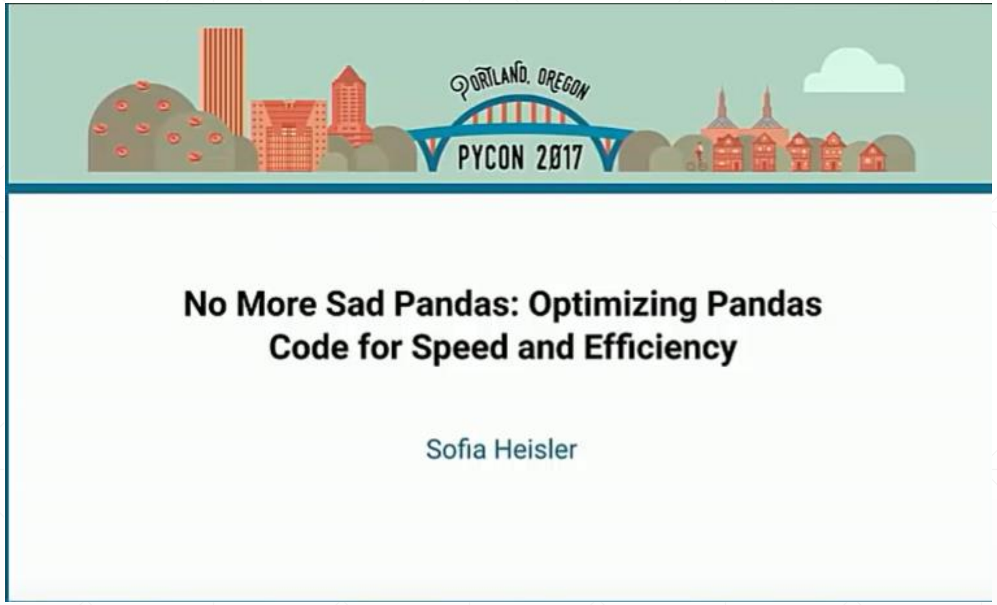


numpy

- Hardware and parallelization
  - Run parts of a program in parallel on separate cores -- on a single machine or in a distributed system
  - Software packages for parallelizing data analysis in python: pyspark, pandaral.lel, dask
  - For more: **CS267**

# Pandas Optimization

- Avoid for loops / .iterrows()
- If looping is a must, use apply.
- Pandas series vectorization
- Vector operations on NumPy arrays are more efficient than on native Pandas series



No More Sad Pandas: Optimizing Pandas Code for Speed and Efficiency

Sofia Heisler

Video