Machine Intelligence

End-to-End Machine Learning Project Part 1

We will do a machine learning project!

1. Look at the big picture
2. Get the data
3. Discover and visualize the data to gain insights
4. Prepare the data for Machine Learning algorithms
5. Select a model and train it
6. Fine-tune your model
7. Present your solutions
8. Launch, monitor, and maintain your system

Chapter 2 in Sci-Kit has a number of public datasets

- We are going to use the California Housing Prices dataset from the StatLib repository
- The data is based on the 1990 California census – not exactly new

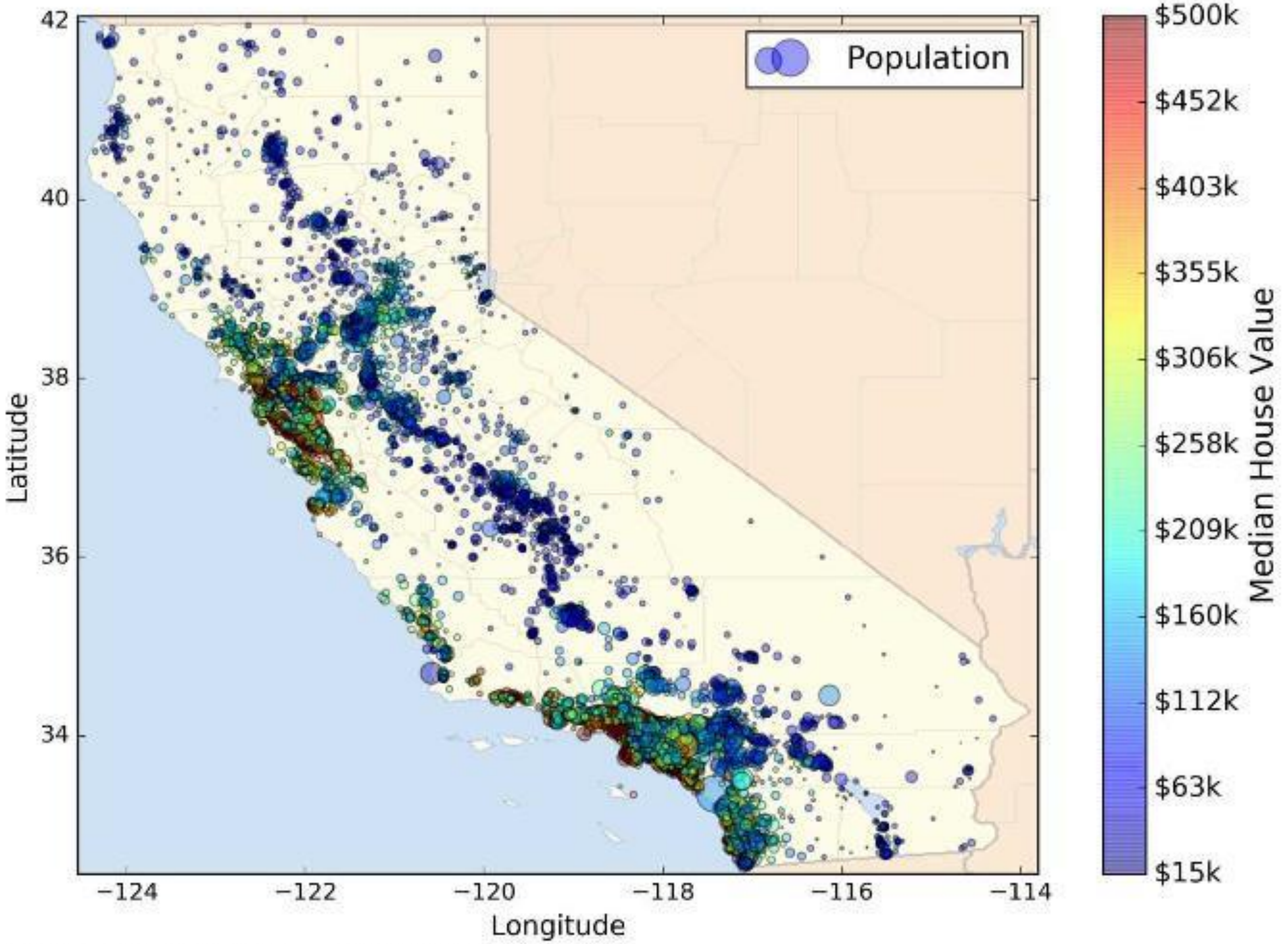

Figure 2-1. California housing prices

Looking at the Big Picture

- Welcome to the Machine Learning Housing Corporation

- ***Your first task is to build a model of housing prices in California*** using the California census data

- The ***data includes metrics*** such as population, median income, median housing price, etc.

- ***Block groups are the smallest geographical unit*** for which the US Census Bureau publishes sample data (600 to 3000 people)
  - Let’s call them ‘districts’

- Your model is to ***learn from this data and be able to predict the median housing price in any district***, given all the other metrics

Frame the Problem – Question 1

- ***First question for the boss***: What is the business objective?
  - Building a model is not the end goal. That's just the beginning.
- This '***frames the problem***'
  - What algorithm(s) to select?
  - What performance measure will we use to evaluate the model?
  - How much effort to spend 'tweaking' the model?
- ***Boss says***: your model's output will be fed to another Machine Learning system along with many other ***signals***
- The downstream system will determine whether it is worth investing in a given area or not

Frame the Problem

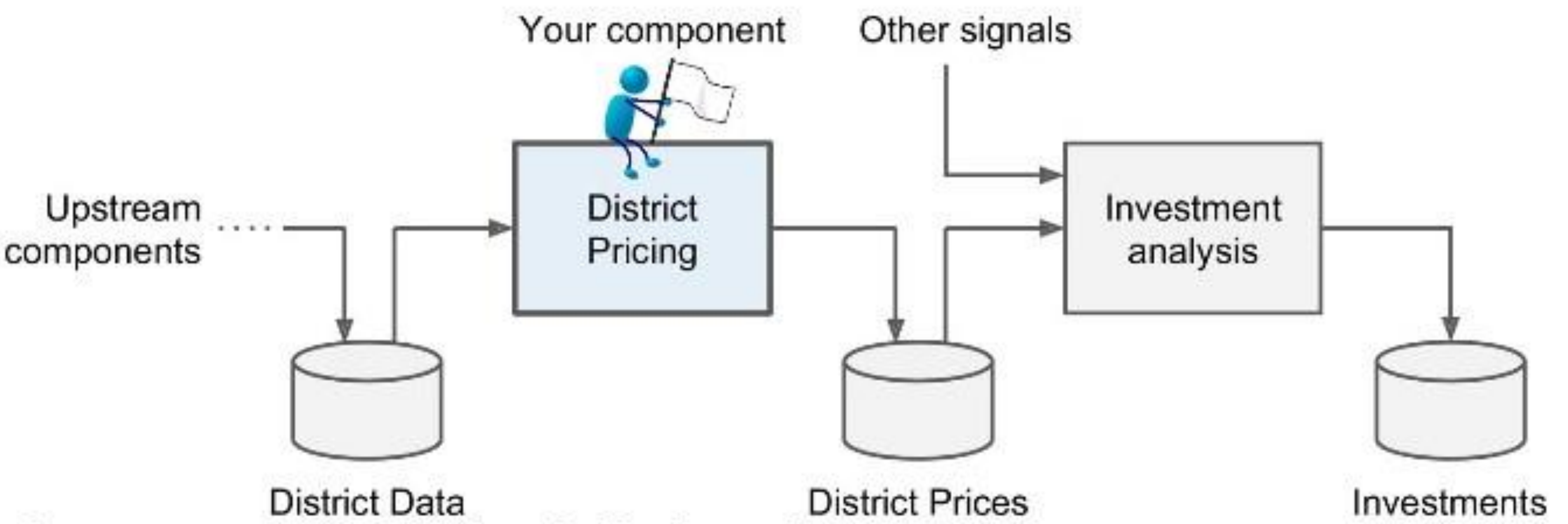

Figure 2-2. A Machine Learning pipeline for real estate investments

Pipelines

- A ***sequence of data processing components*** is called a data pipeline
- ***Pipelines are very common in ML systems***, since there is a lot of data to manipulate and many data transformations to apply
- Components typically run asynchronously
- Each component pulls in a large amount of data, processes it, and passes the results to a data store
  - Each component is fairly self-contained
  - The data store is the interface between components
  - Different teams can work on different components

Frame the Problem - Question 2

- What does the current solution for investing in properties looks like?

- This question will often provide a reference performance, as well as insights for how to solve the problem

- ***Boss answers***: the district housing prices are currently estimated manually by experts
  - This is not working well. They can’t always get median housing prices, so they estimate with complex rules
  - Sometimes they are off by more than 20%
  - The census data looks like a great dataset to exploit for this purpose

Example

- If the ***first district in the dataset*** is located at longitude -118.29º, latitude 33.91º, and it has 1416 inhabitants with a median income of $38,372, and the median house value is $156,400, then:

$$\mathbf{x}^{(1)} = \begin{pmatrix} -118.29 \\ 33.91 \\ 1,416 \\ 38,372 \end{pmatrix}$$

and:

$$y^{(1)} = 156,400$$

Example

- ***X*** is a matrix containing all the feature values (without labels) of all instances in the dataset. There is one row per instance

- ***h*** is your system's prediction function – a hypothesis

- When your system is given an instance's feature vector $x^{(i)}$, it outputs a predicted value $\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$ for that instance ($\hat{y}$ is pronounced "y-hat").

$$\mathbf{X} = \begin{pmatrix} (\mathbf{x}^{(1)})^T \\ (\mathbf{x}^{(2)})^T \\ \vdots \\ (\mathbf{x}^{(1999)})^T \\ (\mathbf{x}^{(2000)})^T \end{pmatrix} = \begin{pmatrix} -118.29 & 33.91 & 1,416 & 38,372 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Check the Assumptions

- The district prices that our system outputs are going to be fed into a downstream ML system
- We assume that these prices are going to be used. What if the downstream system actually converts the prices into categories (cheap, medium, expensive)?
- If so, then our problem should have been framed as a classification task, not a regression task
- Best to find this out in the beginning

Get the Data

- Create a new folder (not a project) in your Pycharm/VS Code/etc. Call it whatever you want – firstML, GeronChap2, etc.
- ***Download housing.csv*** from Brightspace it is with today's lecture slides. Put it in your new folder.
- ***Download 'chapter 2 load data.py'*** from Brightspace. Put in your new folder, as well. Open it up.
- We need to download a number of Python packages. At the import statements at the top of your Python script, click on the package name and hit Alt-Enter (for Windows); for Mac, just click on the name and then click on the red lightbulb
  - numpy, pandas, matplotlib, sklearn

Take a quick look at the data

- Top 5 rows of the data – use a spreadsheet app
- Each row represents one district. There are 10 attributes in total
- Time to load the data – run ‘chapter 2 load data.py'. Let’s look at the output and the code.
- Uncomment line 17, run the program and let’s look at the output again
- Uncomment line 19 and rerun the program. Let’s look at the output again
- Now uncomment lines 21 and 22 to see the plots

About the histogram data

- ***Medium Income Attribute***: Is not dollars. It is actually about $10,000
- Housing median age and median house value were capped
  - Our ML algorithm will learn that prices never go beyond that limit
  - Check with the downstream ML team. If they need predictions above $500,000, we have two options
    - Collect proper labels for the districts whose labels were capped
    - Remove those districts from the training set (and test set)
- Finally, ***many histograms are tail heavy*** – they extend much farther to the right of the median than to the left
  - Might make it harder for some ML algorithms to detect patterns