Introduction to Machine Intelligence

End-to-end Machine Learning Project Part 2

Two Python files to get from Brightspace

- Chapter 2 load data part 2.py and chapter 2 load data part 2b.py

Let’s continue with our CA Census Bureau Data

- We have a set of data, but we can’t use the entire data set to train our ML algorithm

- We must create a training set and a test set
  - We use the training set of data to train the model
  - We use the test set – not used to train the model – to see how well our model performs on new data

Next step: Create a Test Set

- ***Data Snooping Bias***: The human brain is an incredibly successful pattern detection system.
  - Be careful looking at the data yourself.
  - If you look at the test set, you may stumble upon some seemingly interesting pattern in the test data that leads you to select a particular kind of ML algorithm
  - When you estimate the generalization error using the test set, your estimate may be too optimistic, and you will launch a system that will not perform as well as expected

Next step: Create a Test Set

- Computers are not random
- Let’s look at how NumPy generates random numbers
  - https://www.w3schools.com/python/numpy/numpy_random.asp
- Let’s run ‘chapter 2 load data part 2’. It is printing the first 5 lines of training data and test data. If you run the code multiple times, you will see that the first 5 rows vary.

Next step: Create a Test Set

- Over time, your ML algorithm will see the whole dataset – bad idea – because it is choosing a different set of data for the test set every time
- ***One solution***: Save the test set on the first run then load it in subsequent runs
- ***Another solution***: set the random seed to a fixed number so you always get the same set of data
- ***Problem***: This won’t work the next time we get an updated dataset

How to create a ‘reliable’ test set

- Use each instance’s identifier to decide whether it should go in the test set

- ***Problem***: Our Census Bureau data does not have a unique identifier Can we use the row number?

- ***Algorithm***: Compute a hash of each instance’s identifier and put that instance in the test set if the hash is lower or equal to 20% of the maximum has value (https://en.wikipedia.org/wiki/Hash_function)

How to create a ‘reliable’ test set – even with new data

- A new test set will contain 20% of the new instances, but it ***will not contain any instance that was previously in the training set***
- ***Using the row index as a unique identifier***, you need to make sure that new data gets appended to the end of the dataset, and no row ever gets deleted
- If you can’t guarantee this, use latitude and longitude – they are good for a few million years
- Comment out line 31 and uncomment lines 33 and 34. Run the program several times. The first 5 rows stay the same.

What is random anyway?

- When a company goes to survey 1000 people about something, they don't randomly pick 1000 people

- ***Stratified sampling***: 51.3% of the US population is female, 48.7% are male. A good survey should have 513 women and 487 men in their survey.

Talking to experts about the housing data

- Suppose you chatted with experts who told you that the median income is a very important attribute to predict median housing prices
- Since the ***median income is a continuous numerical attribute***, you first need to create an income category attribute
- Looking back at the histogram, most median income values are clustered around 1.5 to 6 ($15K to $60K), but some median incomes go far beyond 6. ***Uncomment lines 41 and 42 to see the histogram.***
- We need to have enough instances in our dataset for each stratum, or else the estimate of the stratum's importance may be biased

Talking to experts about the housing data

- We need to ensure we don't have too many strata, and each stratum should be large enough
- The code uses the pd.cut() function to create an income category attribute with 5 categories (1 to 5)

  https://pandas.pydata.org/docs/reference/api/pandas.cut.html

- Run ***do_the_cut()***. Comment out lines 41 and 42. Uncomment line 44.
- Do we have sufficient values in each strata?
- Now we are ready to do stratified sampling based on the income category. We will use Scikit-Learn's StratifiedShuffleSplit class

Talking to experts about the housing data

- Time to open ***chapter 2 load data 2b.py***

- Now we are ready to do stratified sampling based on the income category. We will use ***Scikit-Learn's StratifiedShuffleSplit class***

  https://www.investopedia.com/terms/stratified_random_sampling.asp

  - Run the ***2b*** program

Discover and Visualize the Data to Gain Insights

- We’ve only just played at looking at the data. Let’s do this more seriously
- First, let’s remove the income_cat attribute that we created in ‘do_the_cut()’. This puts us back with our original data set
  - Uncomment line 69 and run your code
- First, put the test set aside – we only need the training set for now
- Let’s create a copy so we don’t mess with the original training set
  - Uncomment lines 72 and 73. Comment out lines 104 and 105 – don’t need those print statements. Run your code
  - This replaces all the data in the housing variable to be just our training set – only 16,512 rows now.

Visualizing Geographical Data

- Since there is geographical information (latitude and longitude), it is a good idea to create a scatterplot of all districts to visualize the data
  - Uncomment lines 75 and 76 and run your program
- This looks like California, but it is difficult to see any particular pattern
  - Setting the alpha option to 0.1 makes it easier to visualize the places where there is a high density of data points
  - What does the alpha parameter do? (https://matplotlib.org/3.1.1/gallery/recipes/fill_between_alpha.html)
  - Add ‘, alpha=0.1’ to line 75
  - This makes it easier to see the high density areas: SF, LA, SD, and Central Valley – around Sacramento and Fresno

Now Let's Look at Housing Prices

- The radius of each circle represents the district's population (option s) and the color represents the price (option c)
- We will use a predefined color map (option cmap) called ***jet***, which ranges from blue (low prices) to red (high prices)
  - Comment out lines 75 and 76
  - Uncomment lines 78 through 80
- What does this tell you about housing prices in California?
  - Price is very much related to location (e.g., close to ocean) and to population density

Looking for Correlations

- Since the dataset is not too large, we can easily compute the ***standard correlation coefficient*** (also called Pearson's ***r***) between every pair of attributes using the ***corr()*** method
  - What is this? (https://www.investopedia.com/terms/c/correlationcoefficient.asp)
  - Comment out lines 78 through 80 and uncomment lines 82 and 83
- What do we see? Let's look at the table of values.
  - Median house value tends to go up when median income goes up
  - There is a small negative correlation between latitude and median house value (prices tend to go down as you go north)
  - All the values close to zero mean there is no linear correlation – there may be a more complex correlation

Looking for Correlations

- These correlations only measure linear relationships. The bottom row is an example of non-linear relationships
- Second row shows 1 or -1 relationships
- First row shows a range of correlation numbers

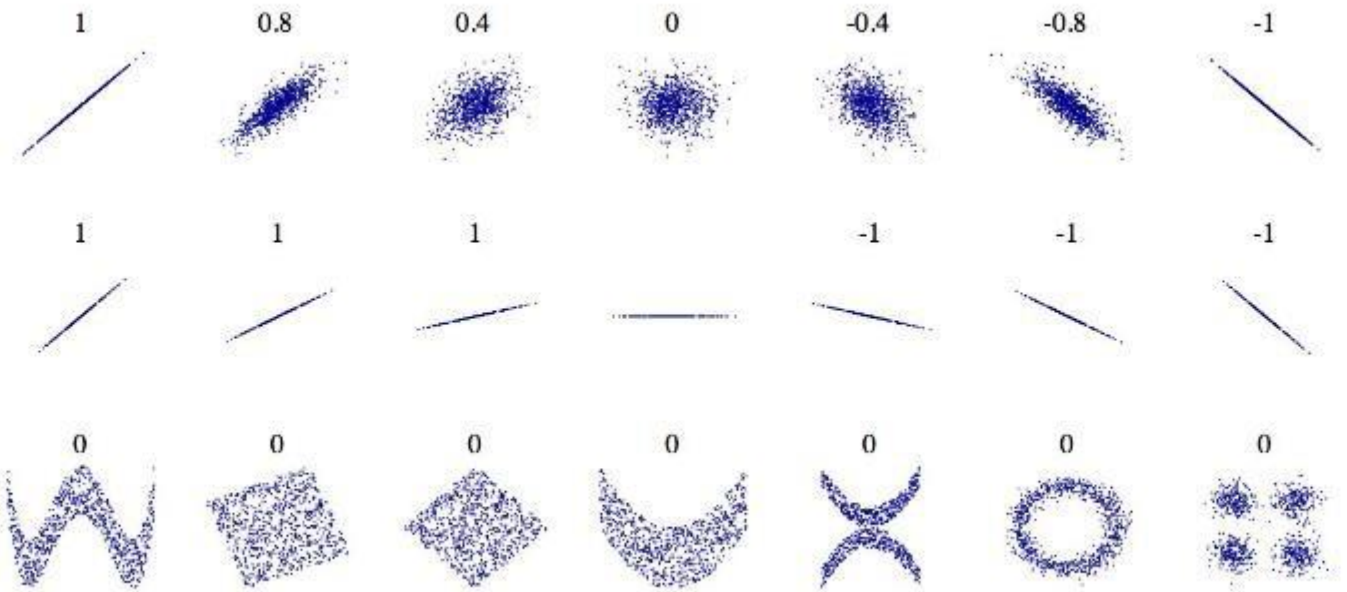

Figure 2-14. Standard correlation coefficient of various datasets (source: Wikipedia; public domain image)

Another way to look for correlations

- We can also use Pandas' ***scatter_matrix function***
  - This plots every numerical attribute against every other numerical attribute
  - Since there are 11 numerical attributes, we would get 121 plots
  - Let's focus on a few promising attributes that seem most correlated with the median housing value
  - You can comment out lines 82 and 83 if you want
  - Uncomment lines 85 to 87
  - Nicely, Pandas knows not to plot attributes against themselves

- The most promising attribute to predict the median house value is the median income, so let's zoom in on their correlation scatterplot
  - Comment out lines 85 to 87
  - Uncomment lines 89 and 90

What do we see?

- Correlation is indeed quite strong – a definite upward trend and the points are not too dispersed

- What's that line at $500,000?
  - That's because our data only goes to $500,000

- What about the lines around $450,000 and $350,000 and maybe around $280,000?
  - We may want to remove corresponding districts to prevent our algorithms from learning to reproduce these data quirks

Experimenting with Attribute Combinations

- ***Total number of rooms in a district*** is not very useful if you don’t know how many households there are
  - Better to have the number of rooms per household
- Similarly, the ***total number of bedrooms*** by itself is not very useful – comparing it to the ***total number of rooms*** might be better
- Also, ***population per household*** seems like an interesting attribute combination to look at
- Let’s create them
  - Comment out lines 89 and 90
  - Remove the quote marks from lines 92 and 98 to uncomment lines 93 – 97
  - Did we gain anything with the new features?

Experimenting with Attribute Combinations

- What did we learn?
  - ***bedrooms_per_room*** is much more negatively correlated with median house value than the total number of rooms or bedrooms
  - Apparently, houses with a lower bedroom/room ratio tend to be more expensive
  - The number of ***rooms per household*** is a little more informative than the ***total number of rooms*** in a district
    - The larger the house the more expensive it is
- This round of exploration does not have to be thorough
- The point is to start off on the right foot and gain some quick insights to help get a good first prototype – but ***this is an iterative process***

In-class activity

- I’ve posted 3 different datasets with the lecture notes for today
- Make a copy of the chapter 2 load data part 2 – call this new Python file whatever your want
- Put your selected dataset into the same folder as your new Python file
- Modify your new Python script to do the same things we did with chapter 2 load data part 2 with your new dataset
- If you have time, try the same thing with part 2b