MATM063: Principles of Data Science, Python LAB

Worksheet 3 (Week 3)

MATM063: Pandas basics and exploring the data

Werner Bauer
Department of Mathematics

Key Learning: More on Pandas, loading data files, exploring and visualizing the data

1 USUAL WORK FLOW IN DATA SCIENCE (PART 1)

Here, in short, an outline of the steps that have to be done in order to prepare data such that they are suitable to train a machine learning algorithm.

1. Download (load) the data into a Pandas dataframe! Check and correct types.
2. Split full dataset into train and test sets.
3. Explore the data and find correlations between attributes and the quantity of interest (e.g. between predictors and target values).
4. Explore attribute combinations.

2 IMPORT EXTERNAL DATASETS

Pandas provides many functions to load a large variety of different file types:

```python
import pandas as pd

data = pd.read_csv('file_name.csv') # csv (comma-separated values)

data = pd.read_csv('file_name.txt',sep='\s+') # txt with unknown nmb of whitespaces between entries

data = pd.read_excel('file_name.xlsx', sheet_name = 'sheet1') # reads in sheet 1 of excel file
```

```python
import os
import tarfile
import urllib
import pandas as pd

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml2/master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"

def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    os.makedirs(housing_path, exist_ok=True)
    tgz_path=os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()

def load_housing_data(housing_path=HOUSING_PATH):
    csv_path=os.path.join(housing_path,"housing.csv")
    return pd.read_csv(csv_path)

# execute these functions:
fetch_housing_data() # fetch the data
housing = load_housing_data() # loading data into workspace
```

Listing 1. Load external datasets

3 VISUALIZING DATA/ADAPTING ATTRIBUTE TYPES

Types are not away correctly identified. To correct them, you can do:

```
print(data.describe()) # prints an overview of each numerical attribute (mean, std, min/max, ...)

# print attribute types with:
print(data.info()) # OR
print(data.dtypes)

# types can be modified with and written as a copy into data with:
data = data.astype({'Attribute' : 'NewType'}, copy=True) # DEFAULT: copy=True
```

4 CREATING TEST AND TRAIN DATA SETS

To split (randomly, with random seed) the data into test and train set, simple do:

```
from sklearn.model_selection import train_test_split
# purely random sampling method:
train_set, test_set = train_test_split(housing,test_size=0.2,random_state=42)
```

For smaller datasets, to create a test set that represents properly e.g. `median_income`, one way could be:

```
from sklearn.model_selection import StratifiedShuffleSplit

housing["income_cat"]=pd.cut(housing["median_income"],
                 bins=[0., 1.5, 3.0, 4.5, 6., np.inf],
                 labels=[1, 2, 3, 4, 5])

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(housing, housing["income_cat"]):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]

# remove income_cut to return to original data:
for set_ in (strat_train_set, strat_test_set):
    set_.drop("income_cat", axis=1, inplace=True)
```

Listing 2. Stratified shuffle split method

5 VISUALIZATION OF DATA

If you want to output all columns of a dataframe, use the command:

```
pd.set_option('display.max_columns', None)
```

Here some commands to plot e.g. histogrammes and correlations

```
import pandas as pd
import matplotlib.pyplot as plt

# plotting histogrammes
data.hist(bins=50,figsize=(20,15))

# plotting a scatter plot (for e.g. housing.csv):
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4, s=housing["population"]/100, label="
    population", figsize=(10,7), c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True)
plt.legend()

# Plotting correlations:
corr_matrix = housing.corr()
print(corr_matrix) # for all attributes:

# OR relative to median_income:
print(corr_matrix['median_income'].sort_values(ascending=False))
```

6 EXERCISES

Question 1 (Step 1 of workflow):

Loading the data and preliminary visualisations.

1. Read in the data ’housing’ using the code in Listing 1. Where do you find the data file? In what format is it?
2. Display the head of the file! Is the header included (i.e. are the column names correctly read in)?
3. Display the types of each attribute! Permanently set the `ocean_proximity attribute` as type ’category’.
4. Display if there are any missing values in some (or all) of the attributes!
5. Plot the histogramme (with 100 bins) of all numerical attributes and discuss shortly what they tell us! Are there some obvious data flaws visible in the plots? In which attributes would you expect them?

Question 2 (Step 2 of workflow):

Splitting the data set into train and test sets!

1. Split the housing dataset into a test and train set. First, use random picking of instances (with fixed random seed) and create a test set that is 15% in size of the original set. Display the shape of both test and train sets to see if the ratio of splitting is indeed 15%.
2. Create a stratified shuffle split (20% test set size) where the quantity that should correctly be represented is `median_house_value`. Plot a histogramme that shows this distribution.
3. Print the distribution of the `median_house_value` attribute for the overall dataset with the random split and the stratified split test set (use the formula from the lecture) `SET['median_house_value'] / len(SET)` where SET represents the full, the random split, and the stratified split datasets.

Question 3 (Step 3 of workflow):

Explore the training dataset!

1. Calculate the standard correlation coefficient matrix of the housing data set.
2. Assume your quantity of interest is the `median_income`. Provide the correlation coefficient with respect to this attribute and sort them in ascending order.

Question 4 (Step 4 of workflow):

In the lecture notes, we defined the following attribute combinations:

```
housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
housing["bedrooms_per_room"] = housing["total_bedrooms"]/housing["total_rooms"]
housing["population_per_household"] = housing["population"]/housing["households"]
```

1. Plot the scatter matrix using the following attributes:
   `rooms_per_household`, `bedrooms_per_room`,
   `population_per_household`, `median_income`
   `median_house_value`.
   Plot also a scatter plot for the attributes `median_house_value` over `rooms_per_household`.
2. As comparison, print out the standard correlation coefficients relative to `median_house_value` (sort them as you like). Discuss: (i) is the negative correlation of `bedrooms_per_room` to `median_house_value` visible in the scatter matrix; and (ii) do we see a positive correlation of `median_house_value` and `rooms_per_household`? What problem could impact our capability to infer from the picture such correlation?