Week 1 Get familier with Jupyter Notebook

Learning outcome:

1. Get familiar with Jupyter Notebook
2. Try some small exercises on the pre-work of data analysis, such as data loading, exploratory data analysis, visualization, etc.

1 Load the housing dataset using pandas

Download housing.tgz from LMS and save it in the datasets/housing directory in your workspace

[ ]:
```
import pandas as pd
import os
import tarfile
```

[ ]:
```
HOUSING_PATH = os.path.join("datasets", "housing")
```

[ ]:
```
tgz_path = os.path.join(HOUSING_PATH, "housing.tgz")
housing_tgz = tarfile.open(tgz_path)
housing_tgz.extractall(path=HOUSING_PATH)
housing_tgz.close()
```

Write a small function to load the data:

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

Check the top five rows using the head() method.

Pay attention to the attributes of a new dataset. How many attributes? What are they?

[ ]:
```
housing = load_housing_data()
housing.head()
```

Alternatively, you can use info() method to get a quick description of the data.

What is each attribute's data type?

[ ]:
```
housing.info()
```

Find out what categories exist in 'ocean_proximity', and how many districts belong to each category using value_count() method.

[ ]:
```
housing["ocean_proximity"].value_counts()
```

Next, describe() method shows a summary of the numerical attributes.

[ ]:
```
housing.describe()
```

total_bedrooms -> 20,433, not 20,640. This is because the null values are ignored.

Let's plot a histogram for each numerical attribute to get a feel of the data we are dealing with. A histogram shows the number of instances (on the vertical axis) that have a given value range (on the horizontal axis).

Before we can plot anything, we need to specify which backend Matplotlib should use.

We will use Jupyter's magic command %matplotlib inline -> This tells Jupyter to set up Matplotlib, so it uses Jupyter's own backend. Plots are then rendered within the notebook itself.

[ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

[ ]:
```
housing.hist(bins=50, figsize=(20,15))
plt.show()
```

Do you observe any problems with these histograms?

2 Create a Test Set

The idea of creating a test set is simple: pick some instances randomly, typically 20% of the dataset (the ratio may vary), and set them aside.

[ ]:
```
import numpy as np
```

[ ]:
```
from sklearn.model_selection import train_test_split
```

2.1 Random sampling

[ ]:
```
train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
```

2.2 Stratified sampling

We're told that the median income is essential in predicting median housing prices. So we want to ensure that the test set is representative of the various categories of incomes in the whole dataset.

The following code uses the pd.cut() function to create an income category attribute with five categories:

category 1 0-1.5

category 2 1.3-3,

and so on

```
[ ]: 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])
```

```
[ ]: housing["income_cat"].hist()
```

```
[ ]: from sklearn.model_selection import StratifiedShuffleSplit
```

```
[ ]: 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]
```

Check the income category proportions in the test set

```
[ ]: strat_test_set["income_cat"].value_counts()/len(strat_test_set)
```

Now we need to delete the 'income_cat' attribute, so the data is back to its original state.

```
[ ]: for set_ in (strat_train_set, strat_test_set):
         set_.drop("income_cat", axis=1, inplace=True)
```

3 Discover the dataset - visualization

Next we will only explore the trainig data set and put the testing set aside. Let's create a copy so that the following procedures will not harm the training set:

```
[ ]: housing = strat_train_set.copy()
```

Let's first create a scatterplot of all districts to visualize the data.

```
[ ]: housing.plot(kind="scatter", x="longitude", y="latitude")
```

We can observe an overplotting issue, making it difficult to see individual data points in a data visualization.

We can adjust the alpha option to make the visualization better reflect the high density of data points.

```
[ ]: housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.1)
```

4 Correlations between attributes

We can easily compute the standard correlation coefficient between every pair of attributes

For example, let's check how much each attribute correlates with the median house value:

[ ]:
```
corr_matrix = housing.corr()
corr_matrix["median_house_value"].sort_values(ascending=False)
```

How to interpret the results?

Alternatively, we can check for correlations between attributes using the pandas scatter_matrix() function.

Let's focus on a few promising attributes that seem most correlated with the median housing value.

[ ]:
```
from pandas.plotting import scatter_matrix
```

[ ]:
```
attributes = ["median_house_value", "median_income", "total_rooms",␣
 ↪"housing_median_age"]
scatter_matrix(housing[attributes], figsize=(12, 8))
```

Which attributes seem to be more predictable of the median house value?

** Disclaim: The above code is modified from the textbook "Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow".