Machine Learning The Big Picture and Data

Michael Claudius, Associate Professor, Roskilde

29.08.2021

Machine Learning Project

- ***Machine Learning has a number of phases***
- ***The phases can be overlapping and/or iterative***

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 solution.**
8. **Launch, monitor, and maintain your system.**

- **A detailed checklist is given on** ML Management Checklist (PDF)

- **Remember always adapt the order and the checklist to your needs**

Machine Learning: The big picture and data

- ***It is about understanding business and the data!***

1. ***The context***
2. ***Frame the problem***
3. ***Select performance measure***
4. ***Setup workspace***
5. ***Get the data in hand***
6. ***Explore the data tables***
7. ***Create a test set***
8. ***Visual graphs and correlations***
9. ***Experiment with attribute combinations***

The context: Housing prices

- **California median housing price for a block group**
- **Block group: unit population of 600-3.000 people**
- **Data size app. 20.000**

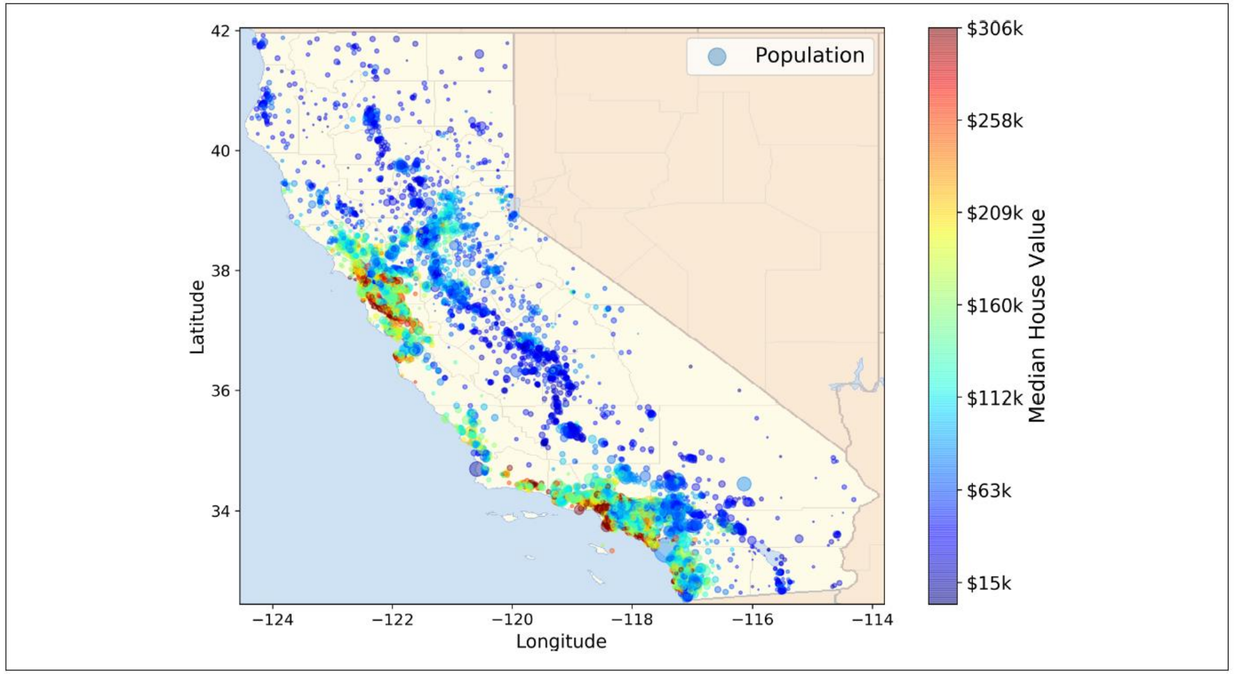

Figure 2-1. California housing prices

Frame the problem

- **Purpose of the output. Interview the stakeholders**
  - **Predict housing price to be used by investment company**

- **ML Types (Student discussion)**
  - **Supervised or unsupervised**
  - **Regression (values) or classification (category)**
  - **Multiple features or not**
  - **Univariant (one value) or multivariate (predict several values)**
  - **Online or batch**

Performance measure

- **Root Mean Square Error (RMSE) or Mean Absolute Error (MAE)**
- **RMSE: higher weight to outliners (normal choice)**
- **MAE: to be used if many outliners**

Equation 2-1. Root Mean Square Error (RMSE)

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

Equation 2-2. Mean absolute error (MAE)

$$\mathrm{MAE}(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}\left|h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right|$$

RMSE formula explained

Notations

This equation introduces several very common Machine Learning notations that we will use throughout this book:

- m is the number of instances in the dataset you are measuring the RMSE on.
  — For example, if you are evaluating the RMSE on a validation set of 2,000 districts, then $m = 2{,}000$.
- $\mathbf{x}^{(i)}$ is a vector of all the feature values (excluding the label) of the i^{th} instance in the dataset, and $y^{(i)}$ is its label (the desired output value for that instance).
  — For example, if the first district in the dataset is located at longitude –118.29°, latitude 33.91°, and it has 1,416 inhabitants with a median income of $38,372, and the median house value is $156,400 (ignoring the other features for now), then:

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

and:

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

RMSE formula explained

- *X is the matrix with all features of all instances*
- *The label y not included !*

- $\mathbf{X}$ is a matrix containing all the feature values (excluding labels) of all instances in the dataset. There is one row per instance, and the i^{th} row is equal to the transpose of $\mathbf{x}^{(i)}$, noted $(\mathbf{x}^{(i)})^\intercal$.[4]
  - For example, if the first district is as just described, then the matrix $\mathbf{X}$ looks like this:

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

RMSE formula explained

- ***h is the hypothesis function e.g. a linear regression***
- ***h(X) =*** $\theta_0 + \theta_1 X_1 + \theta_2 X_2 + \ldots\ldots + \theta_n X_n$
- ***RMSE(X,h) is the cost function i.e. the performance measure***

- h is your system's prediction function, also called a *hypothesis*. When your system is given an instance's feature vector $\mathbf{x}^{(i)}$, it outputs a predicted value $\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$ for that instance ($\hat{y}$ is pronounced "y-hat").
  — For example, if your system predicts that the median housing price in the first district is \$158,400, then $\hat{y}^{(1)} = h(\mathbf{x}^{(1)}) = 158{,}400$. The prediction error for this district is $\hat{y}^{(1)} - y^{(1)} = 2{,}000$.
- RMSE($\mathbf{X}$,h) is the cost function measured on the set of examples using your hypothesis h.

We use lowercase italic font for scalar values (such as m or $y^{(i)}$) and function names (such as h), lowercase bold font for vectors (such as $\mathbf{x}^{(i)}$), and uppercase bold font for matrices (such as $\mathbf{X}$).

Get the data

- ***Download the data***
- ***Housing data already installed in the Github project***
- ***Always create a local copy to work on***

- ***Explained later in an exercise***

- ***Take a Look at the HousingTest Code***

Set up workspace

- ***Anaconda with Jupyter***
- ***Explained in an exercise***

Explore the data attributes

- ***Check the first 5 rows using head() method***

In [5]:
```
housing = load_housing_data()
housing.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populatio
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0

Figure 2-5. Top five rows in the dataset

Explore the data types

- ***Look at the data type using the info() method***

```
In [6]: housing.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

Figure 2-6. Housing info

Explore the data statistics

- ***Make a summary of numerical attributes using the describe() method***

```
In [8]: housing.describe()
```

Out[8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedr(
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.0000
mean	-119.569704	35.631861	28.639486	2635.763081	537.87055
std	2.003532	2.135952	12.585558	2181.615252	421.38507
min	-124.350000	32.540000	1.000000	2.000000	1.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.00000
50%	-118.490000	34.260000	29.000000	2127.000000	435.00000
75%	-118.010000	37.710000	37.000000	3148.000000	647.00000
max	-114.310000	41.950000	52.000000	39320.000000	6445.0000

Figure 2-7. Summary of each numerical attribute

Explore the data distribution

- ***Make a summary of numerical attributes using the describe() method***

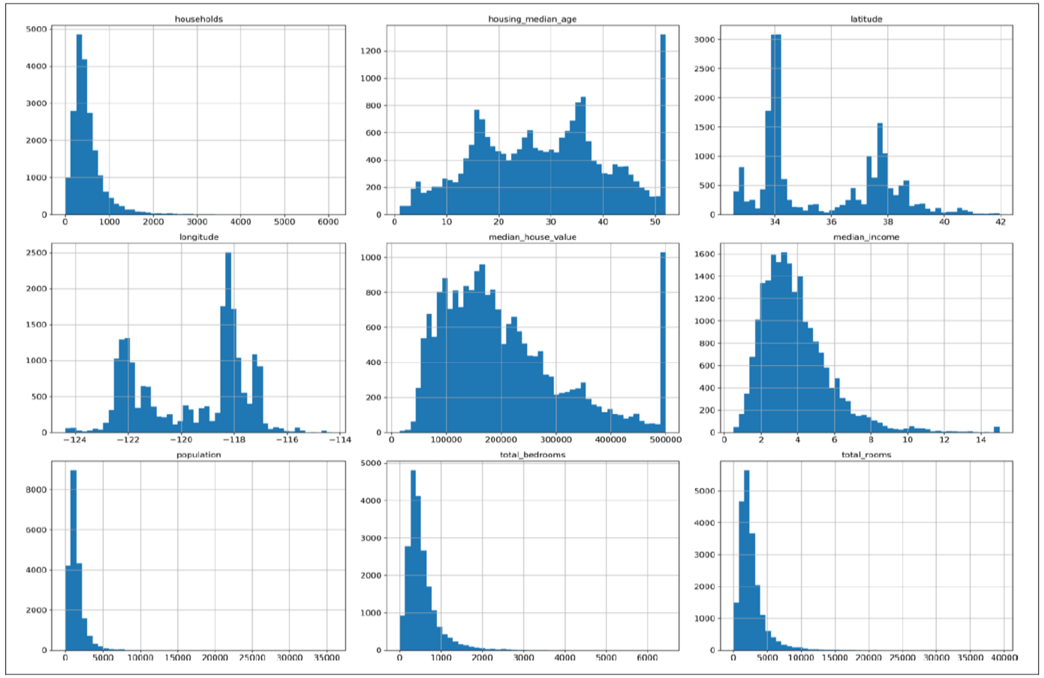

Figure 2-8. A histogram for each numerical attribute

Explore the data anomalies

- ***Notice anomalies like:***
  - ***Capped values like median house income, house age and house value***
  - ***Heavy tail distribution***
  - ***Very different scale a summary of numerical attributes using the describe() method***

- ***These issues might cause problems for some ML algorithms***
- ***To be discussed later***

Create a test set by random sampling

- ***Normally training set is 80%. Test set is 20% of the total data set***

- ***Test set a can be sampled random***
  - ***If you want the same test set each time use seed(42) function or***

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

  - ***Disadvantage of random: what about if new data coming later***
  - ***Solution: take them into the data set and the training set using a special identifier***
    - ***Speciel Identifier: Unique ID, Row no, Latitude & longitude***

  - ***Disadvantage of random: might lead to a skewed test data set not representing original data distribution***
    - ***E.g. male/female distribution 49%/51%***
  - ***Solution: Test set can be sampled stratified***

Create a test set by stratified sampling

- ***Assume median income must be representative in test set***
- ***Create an income category 1, 2 3 4 5 and use it for the data split***

```
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
```

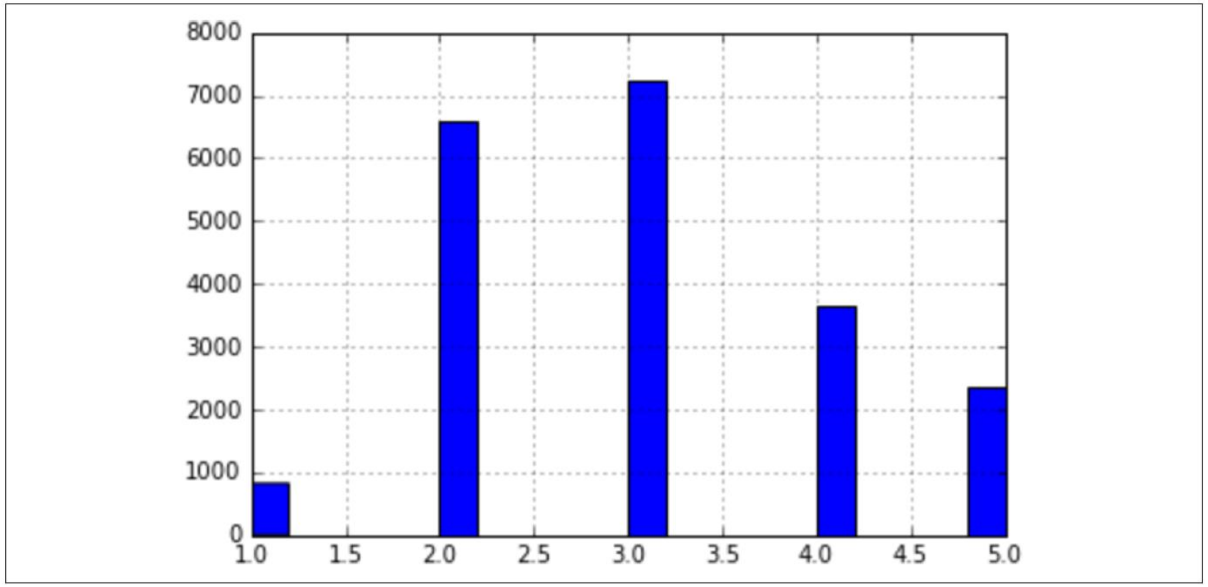

Figure 2-9. Histogram of income categories

Sampling bias comparison

- ***Random versus stratified***
- ***Stratified represents better !***

	Overall	Stratified	Random	Rand. %error	Strat. %error
1	0.039826	0.039729	0.040213	0.973236	-0.243309
2	0.318847	0.318798	0.324370	1.732260	-0.015195
3	0.350581	0.350533	0.358527	2.266446	-0.013820
4	0.176308	0.176357	0.167393	-5.056334	0.027480
5	0.114438	0.114583	0.109496	-4.318374	0.127011

Figure 2-10. Sampling bias comparison of stratified versus purely random sampling

Exercises

- **It is time for looking at Housing code, discussion, coding your own linear regression**
- **Finally, you will make a housing project !!**

- Python Basic No. 2
- Linear Regression
- Housing Ch. 2 No. 1