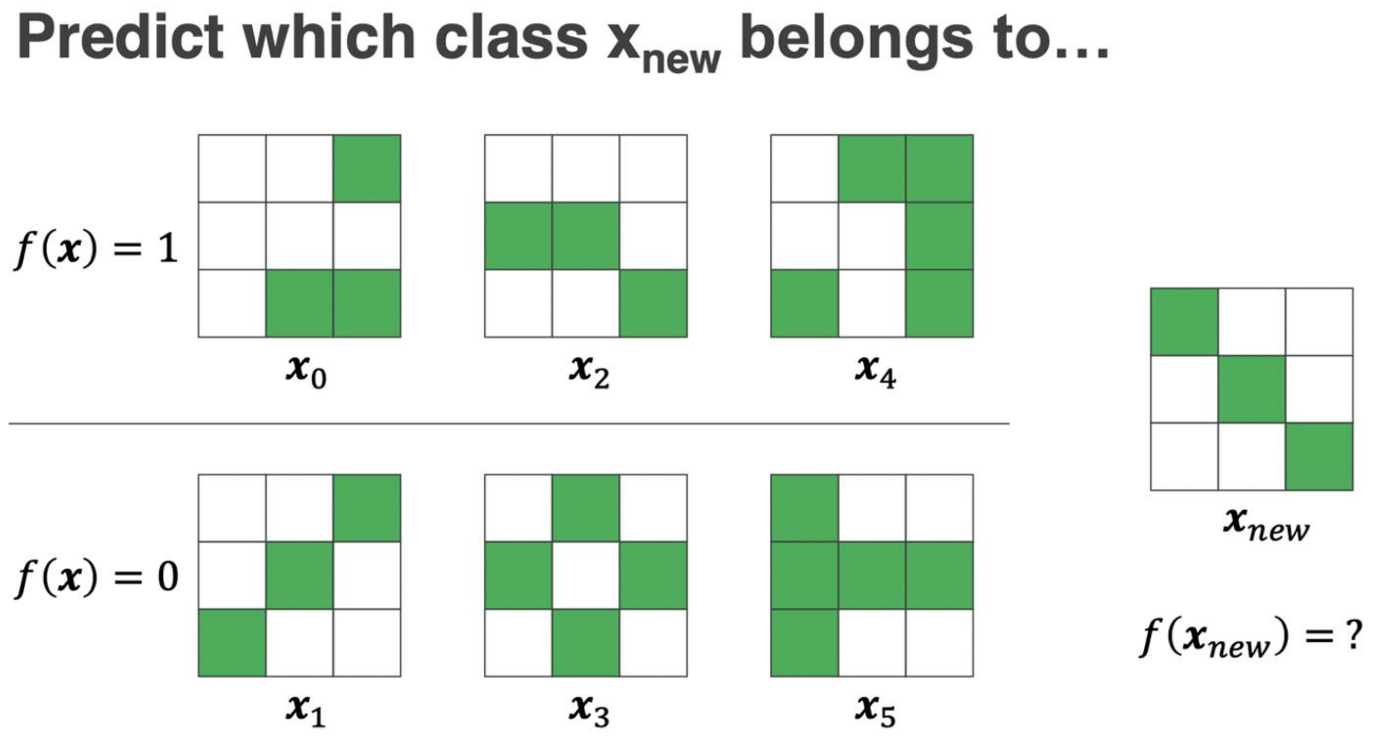
End-to-end machine learning

7. Recall this example from Lecture 1. All of the rules below correctly classify the training data (x_0 through x_5). If I were creating a rule for classifying the new example (x_{new}), which rule would be preferable?



Example credit: Yaser Abu-Mostafa, 2012

a. $f(x) = 1$ if there is NOT an axis of symmetry; else $f(x) = 0$
b. $f(x) = 1$ if the bottom-right pixel is green; else $f(x) = 0$
c. $f(x) = 1$ if the bottom-right pixel is green AND the top-right pixel is white; else $f(x) = 0$
d. $f(x) = 1$ if x EXACTLY matches either x_0, x_2, or x_4; else $f(x) = 0$

8. Why would the rule for the previous question be preferable?
    a. It will outperform the other rules on the training data
    b. It is the parsimonious choice: the simplest rule that achieves the best performance
    c. It is the most complex rule, allowing for greater flexibility and greater generalization performance

9. What type of a machine learning would the problem presented in Question 7 be?
    a. Unsupervised learning
    b. Supervised learning
    c. Reinforcement learning

MODULE 1

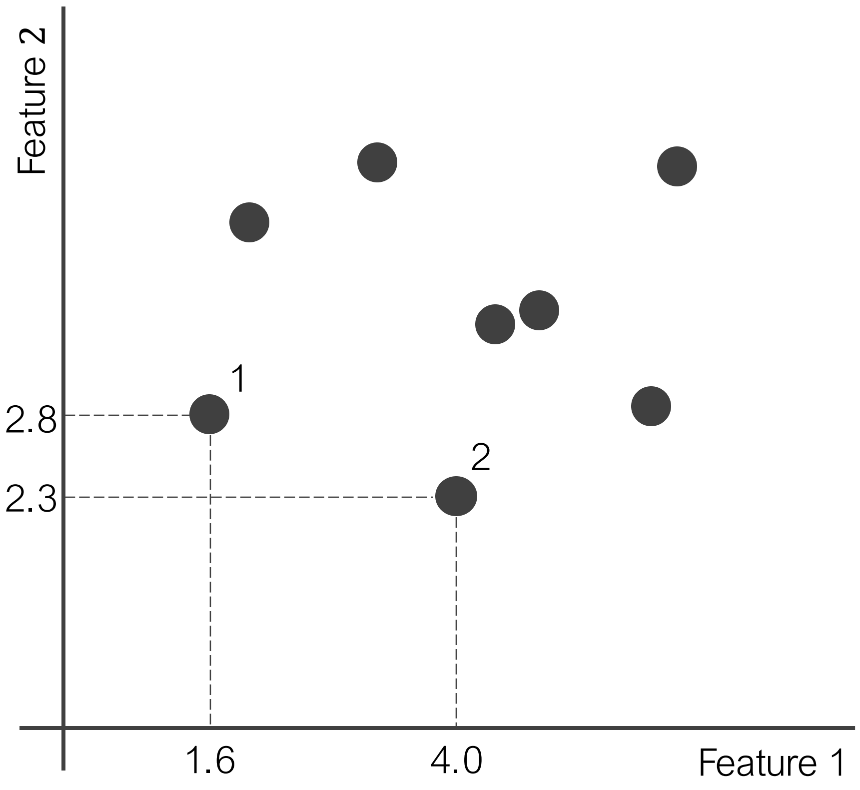
Supervised Machine Learning

Types of machine learning

	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Goal	**Predict** …from examples	**Describe** ...structure in data	**Strategize** learn by trial and error
Data	(x, y)	x	delayed feedback
Types	• Classification • Regression	• Density estimation • Clustering • Dimensionality reduction • Anomaly detection	• Model-free learning • Model-based learning

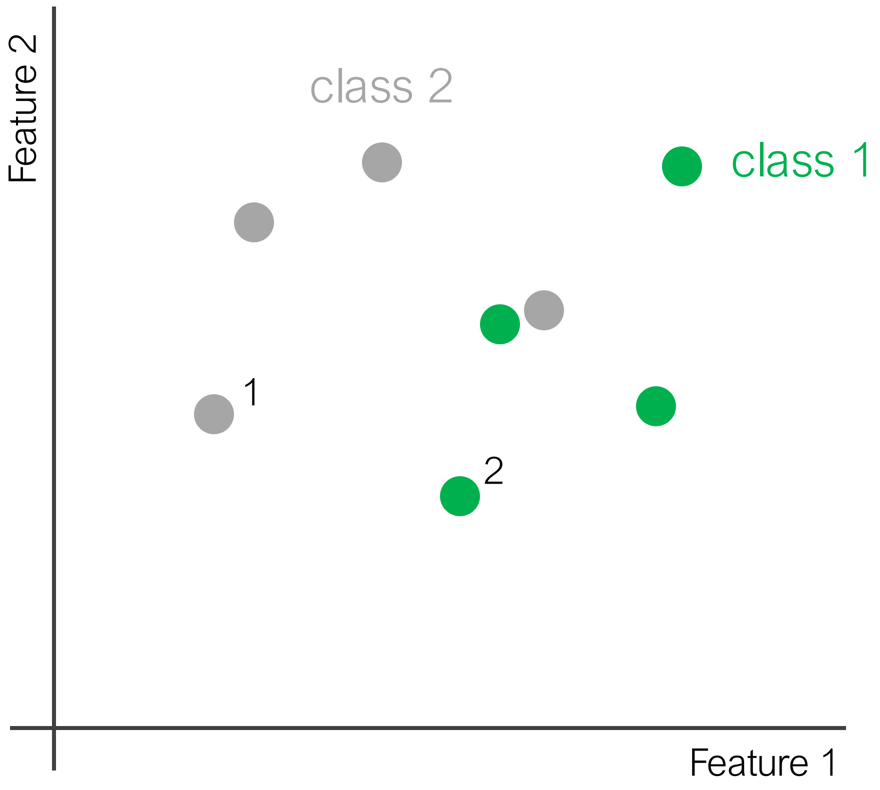
Components of supervised learning

Data



	Feature 1	Feature 2
Data Sample 1	1.6	2.8
Data Sample 2	4.0	2.3
…	###	###

Labels



	Labels
Data Sample 1	Class 1
Data Sample 2	Class 2
…	###

Performance Metric

% of samples correctly classified
(classification)

Mean squared error
(regression)

Supervised machine learning model

We search for the model that best fits our data

$$y = f(x, \boldsymbol{w}) + \epsilon$$

Labels Model Input Data Parameter(s) Random Noise

Hypothesis

Typically prevents perfe
performance

Components of supervised learning

Input	$\boldsymbol{x}$	
Output	y	
Training Data	$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)$	
Target function	$f(\boldsymbol{x}) \rightarrow y$	This is unknown, but the best you could ever do
Hypothesis set	$f_i(\boldsymbol{x}) \rightarrow \hat{y}$	Functions to consider in trying to approximate $f(x)$
Learning algorithm	Optimization technique that searches the hypothesis set for the function f_i that best approximates f (typically by choosing parameters in a model)	

Supervised Learning

Unobservable

Data Generating Process
$p(X,Y)$

Target Function
The best function predicting y from x
$f(x) \rightarrow y$

Observable

Training Data
$(x_1, y_1), \ldots, (x_N, y_N)$

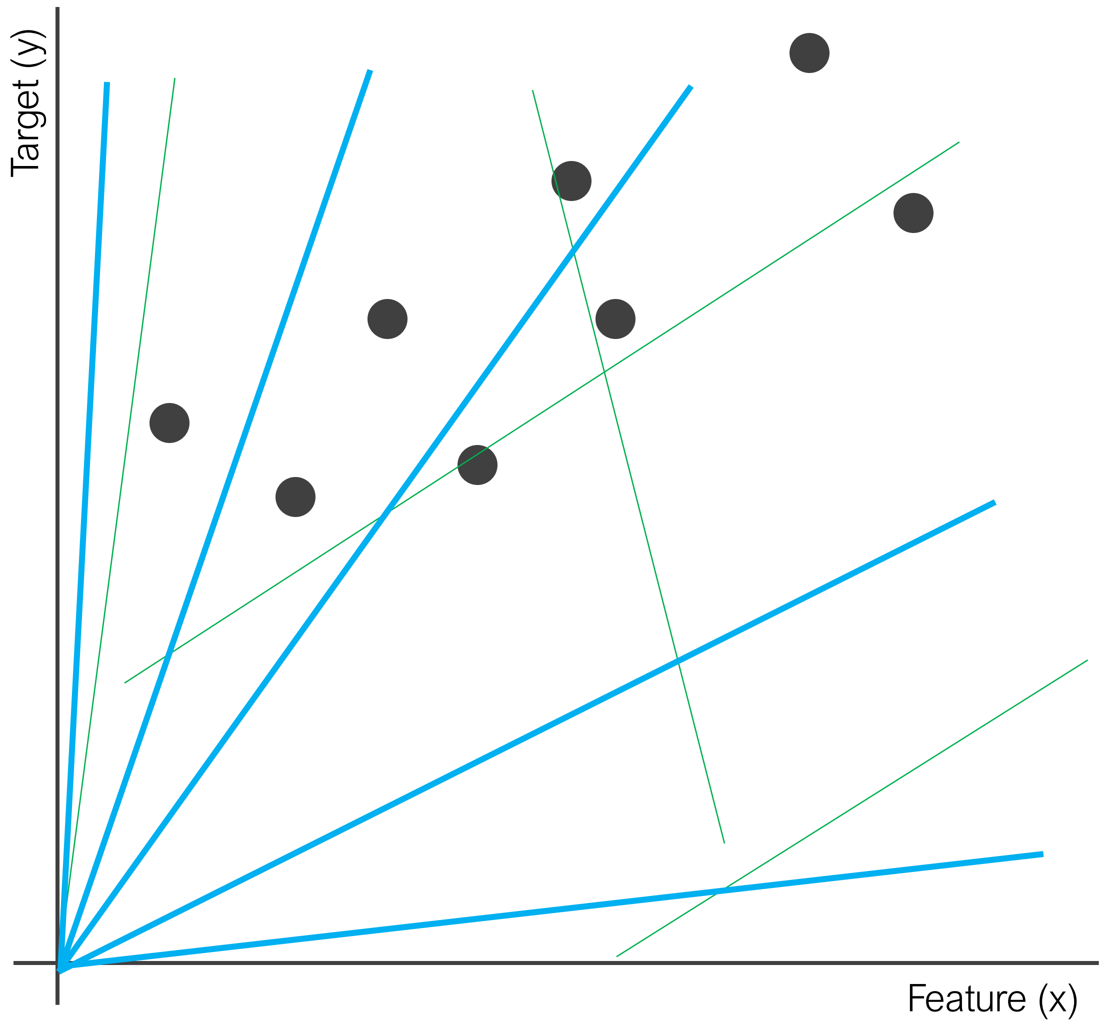
Learning Algorithm
Chooses a hypothesis, $\hat{f} = f_i$ based on the training data such that $\hat{f}(x) \approx f(x)$

Hypothesis Functions Set
$f_1, f_2, f_3, \ldots$

Final Hypothesis
$\hat{f}(x) \rightarrow \hat{y}$
predictions

- Need to select the hypothesis functions (models to train)
- Need to select the learning algorithm (for fitting the models to the data)

Example: linear regression



Using any line of the form $y = w_0 + w_1 x$ as the set of hypothesis functions, how many possible hypothesis functions are in the set?

Infinitely many

Using any line of the form $y = wx$ as the set of hypothesis functions, how many possible hypothesis functions are in this set?

Infinitely many

Which set contains the better hypothesis?

Which set has more options to consider?

What is our learning algorithm?

Next class

Model flexibility and the bias variance tradeoff

End-to-end machine learning

Define the problem

How do we predict the median price of a house in a region?

- What data do we need?
- How do we measure success?

Supervised Machine Learning Process

1. Define your problem, set your goal, and how you will measure success
2. Get the data
3. Explore and prepare the data
4. Propose one or more hypotheses: prospective models
5. Evaluate model performance and iteratively fine tune
6. Deploy your trained model

Deep dive here

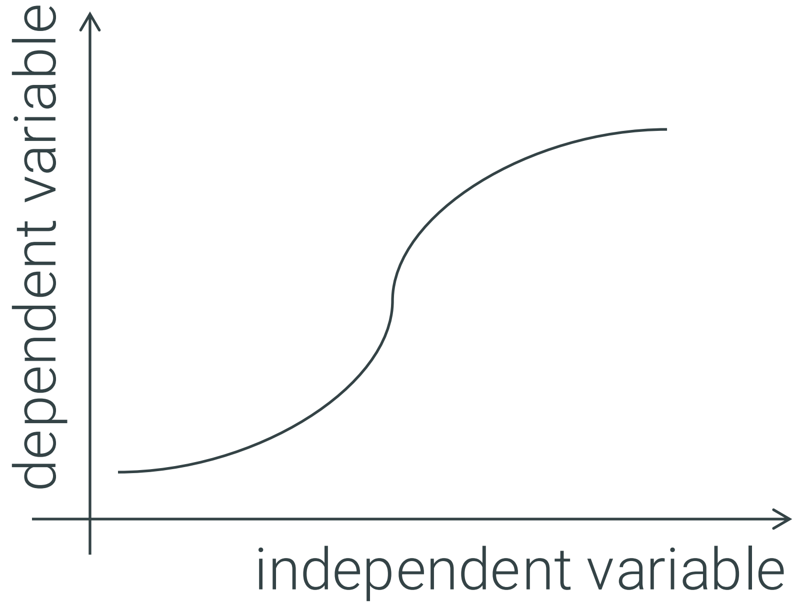
Create training / validation / test data split

Ensure your training data are representative of your test data
(sometimes need to use stratified sampling to avoid sampling bias)

Train	Validation	Test
Used for model training / fitting	Used for model comparison and development	**DO NOT TOUCH** Used to evaluate generalization performance of the final model

Technical note: don't create a DIFFERENT random sample of the dataset each time you run your code – this will expose your modeling to more of the data and contaminate your train/test split

Quick aside:
Common language on variables



independent variable	**dependent** variable
input	output
predictor	response
feature	target
x	y

Supervised learning in practice

Preprocessing

Explore & prepare data

Data Visualization and Exploration

Identify patterns that can be leveraged for learning and issues with data

Data Cleaning

- Missing data
- Noisy data
- Erroneous data

Scaling (Standardization)

Prepare data for use in scale-dependent algorithms.

Feature Extraction

Dimensionality reduction eliminates redundant information

Model training

Select models (hypotheses)

Select model options that may fit the data well. We'll call them "hypotheses".

Fit the model to training data

Pick the "best" hypothesis function of the options by choosing model parameters

Iteratively fine tune the model

Performance evaluation

Make a prediction on validation data

Metrics

Classification
Precision, Recall, F_1, ROC Curves (Binary), Confusion Matrices (Multiclass)

Regression
MSE, explained variance, R^2

Supervised learning in practice

Preprocessing
Explore & prepare data

A Data Visualization and Exploration

Identify patterns that can be leveraged for learning and issues with data

B Data Cleaning

- Missing data
- Noisy data
- Erroneous data

C Scaling (Standardization)

Prepare data for use in scale-dependent algorithms.

D Feature Extraction

Dimensionality reduction eliminates redundant information

Model training

Select models (hypotheses)

Select model options that may fit the data well. We'll call them "hypotheses".

Fit the model to training data

Pick the "best" hypothesis function of the options by choosing model parameters

Iteratively fine tune the model

Performance evaluation

Make a prediction on validation data

Metrics

Classification
Precision, Recall, F_1, ROC Curves (Binary), Confusion Matrices (Multiclass)

Regression
MSE, explained variance, R^2

Ⓐ Always check your data

Data Visualization and Exploration

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY
5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	269700.0	NEAR BAY
6	-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591	299200.0	NEAR BAY
7	-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.1200	241400.0	NEAR BAY
8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	226700.0	NEAR BAY
9	-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.6912	261100.0	NEAR BAY

The data have been scaled (potentially for anonymization purposes)

These data are categorical
Categories/counts below:

```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
```

A cautionary tale

Adapted from from Hands-On Machine Learning with Scikit-Learn & TensorFlow by Aurélion Géron

A Summary info on the data

Data Visualization and Exploration

```
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
```

We're missing data from total_bedrooms

ocean_proximity is not numerical data

Adapted from from Hands-On Machine Learning with Scikit-Learn & TensorFlow by Aurélion Géron

Ⓐ Overall statistics of the data

Data Visualization and Exploration

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

Notice the data seem to be on wildly different scales

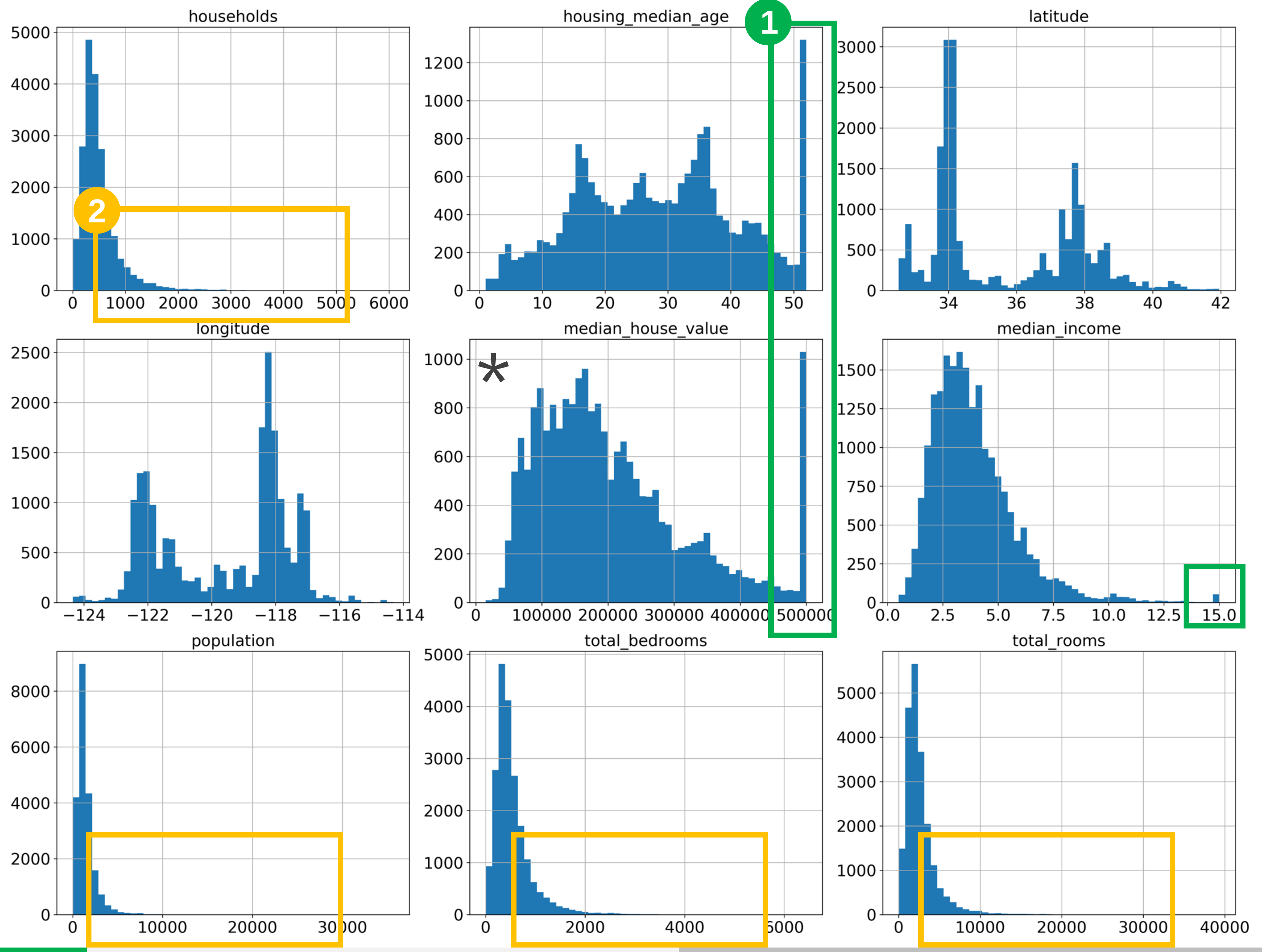
Adapted from from Hands-On Machine Learning with Scikit-Learn & TensorFlow by Aurélion Géron

A View feature distributions

Data Visualization and Exploration

1. **Values are clipped**
   Prevents us from making accurate predictions in those cases
2. **Some features are heavy-tailed**
   Some ML techniques assume normally-distributed data
3. **Scale of values**
   Feature data are on vastly different scales

*= target variable

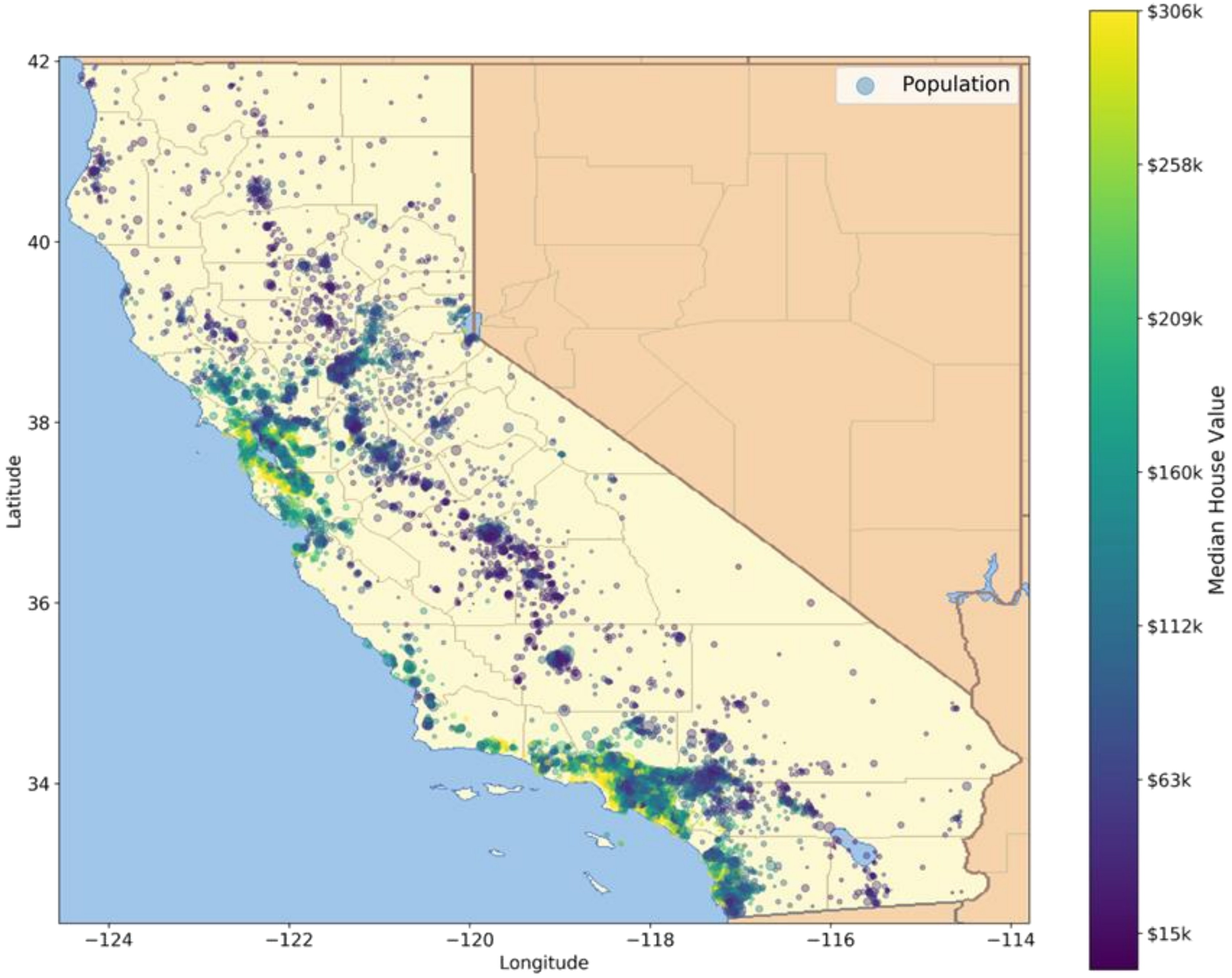


Adapted from from Hands-On Machine Learning with Scikit-Learn & TensorFlow by Aurélion Géron

A

Data Visualization and Exploration

View the data spatially for further insights



Adapted from from Hands-On Machine Learning with Scikit-Learn & TensorFlow by Aurélion Géron

Ⓑ Handling Categorical data

Data Cleaning

Recall ocean_proximity has the following categories:

<1H OCEAN
INLAND
NEAR OCEAN
NEAR BAY
ISLAND

We need to convert this into numerical data to process it

1

Assign numbers to each class

Original value	New feature value
<1H OCEAN	0
INLAND	1
NEAR OCEAN	2
NEAR BAY	3
ISLAND	4

What do these numbers mean?

2

Create one binary feature for each category

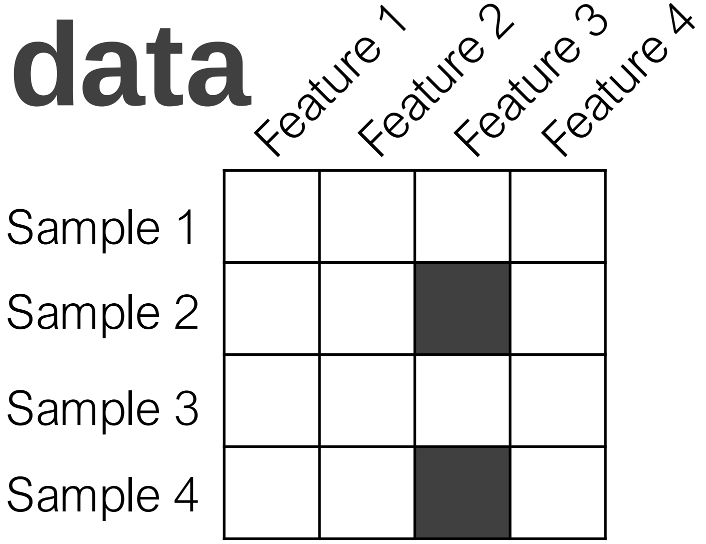
Original value	F_1	F_2	F_3	F_4	F_5
<1H OCEAN	1	0	0	0	0
INLAND	0	1	0	0	0
NEAR OCEAN	0	0	1	0	0
NEAR BAY	0	0	0	1	0
ISLAND	0	0	0	0	1

One-hot-encoding: create a new feature for each category

Ⓑ Handling missing data

Data Cleaning

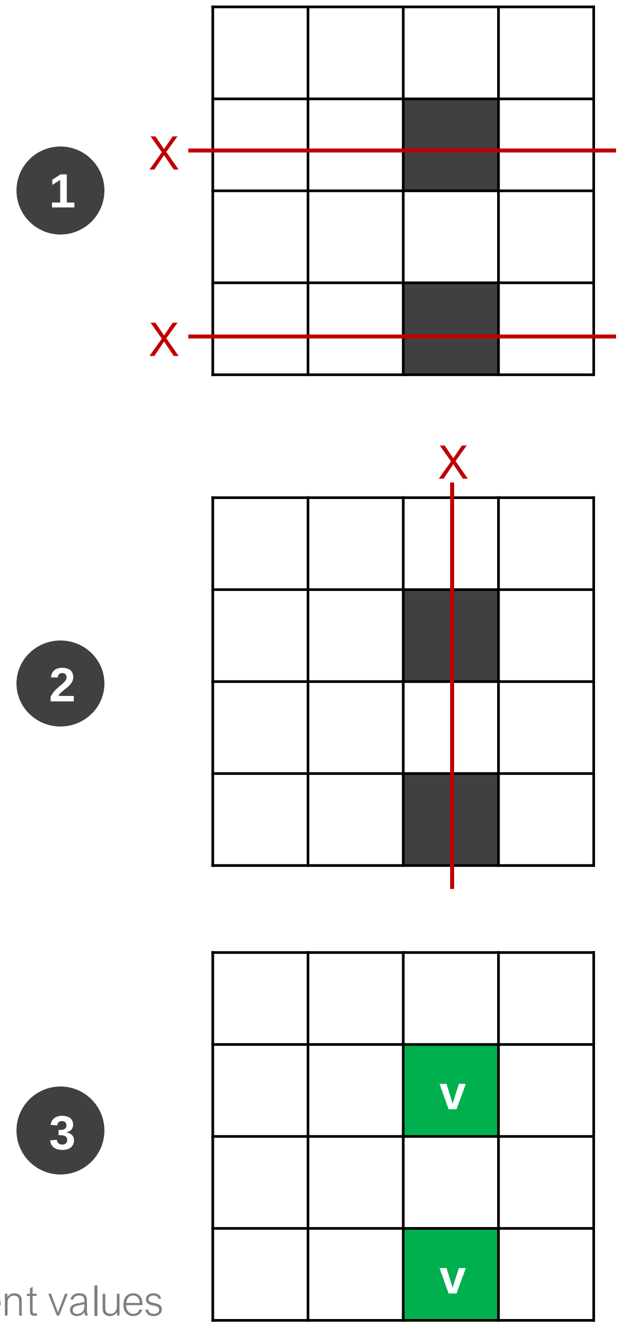
total_bedrooms contains missing values



Feature 3 has 2 missing values

Options:

1. Remove samples that have missing values
2. Remove features that have missing values
3. Fill in (impute) the missing values
   - Fill with average or median
   - Compute a value based on other features



v = replacement values

C

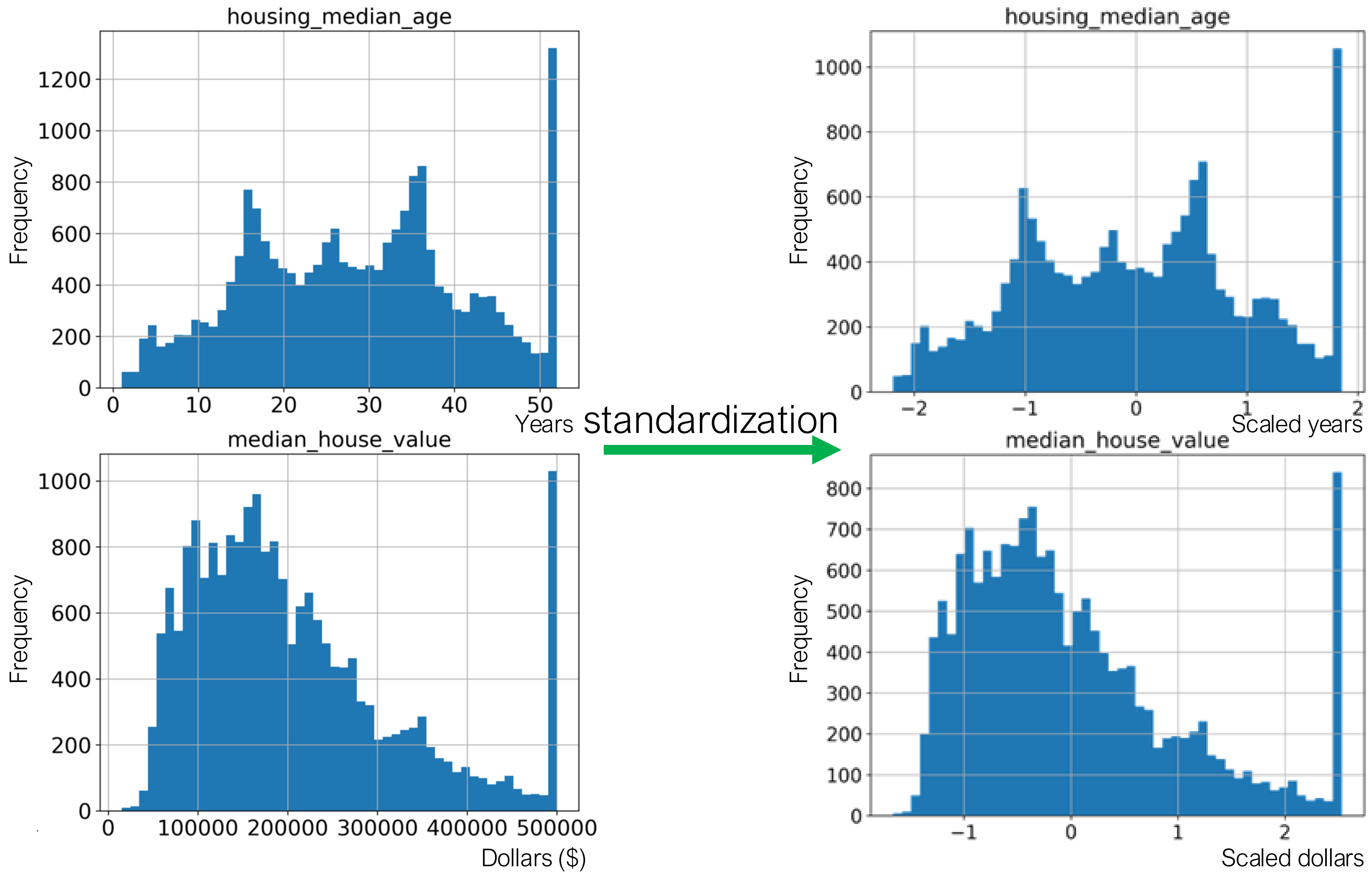
Data Scaling

Scaling features

Standardization

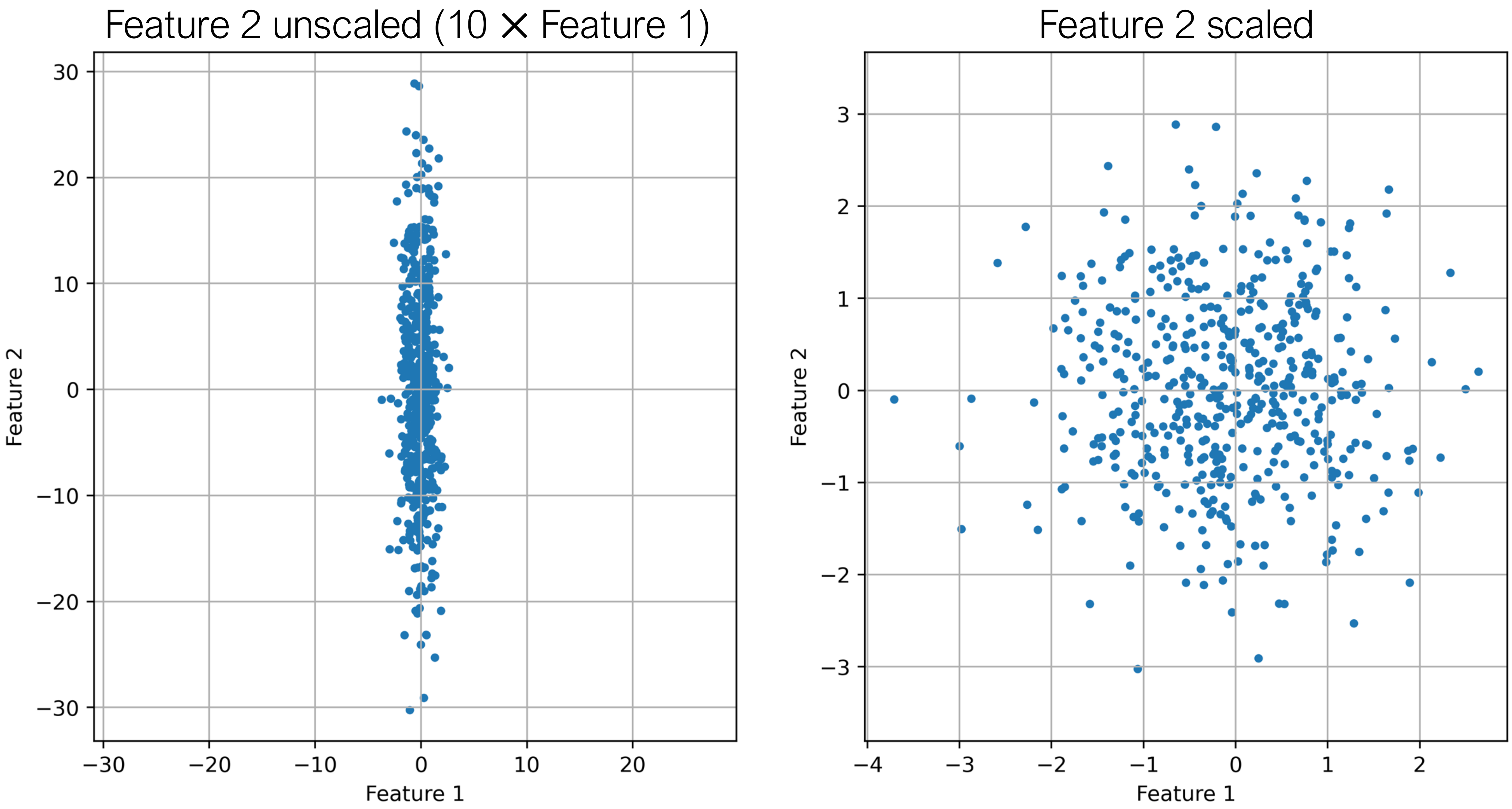
$$x^{new} = \frac{x - \bar{x}}{\sigma(x)}$$

Subtract the mean, divide by the standard deviation



Why do we care about scaling?

Data
Scaling



Feature scaling is critical for algorithms that rely on distances between data points

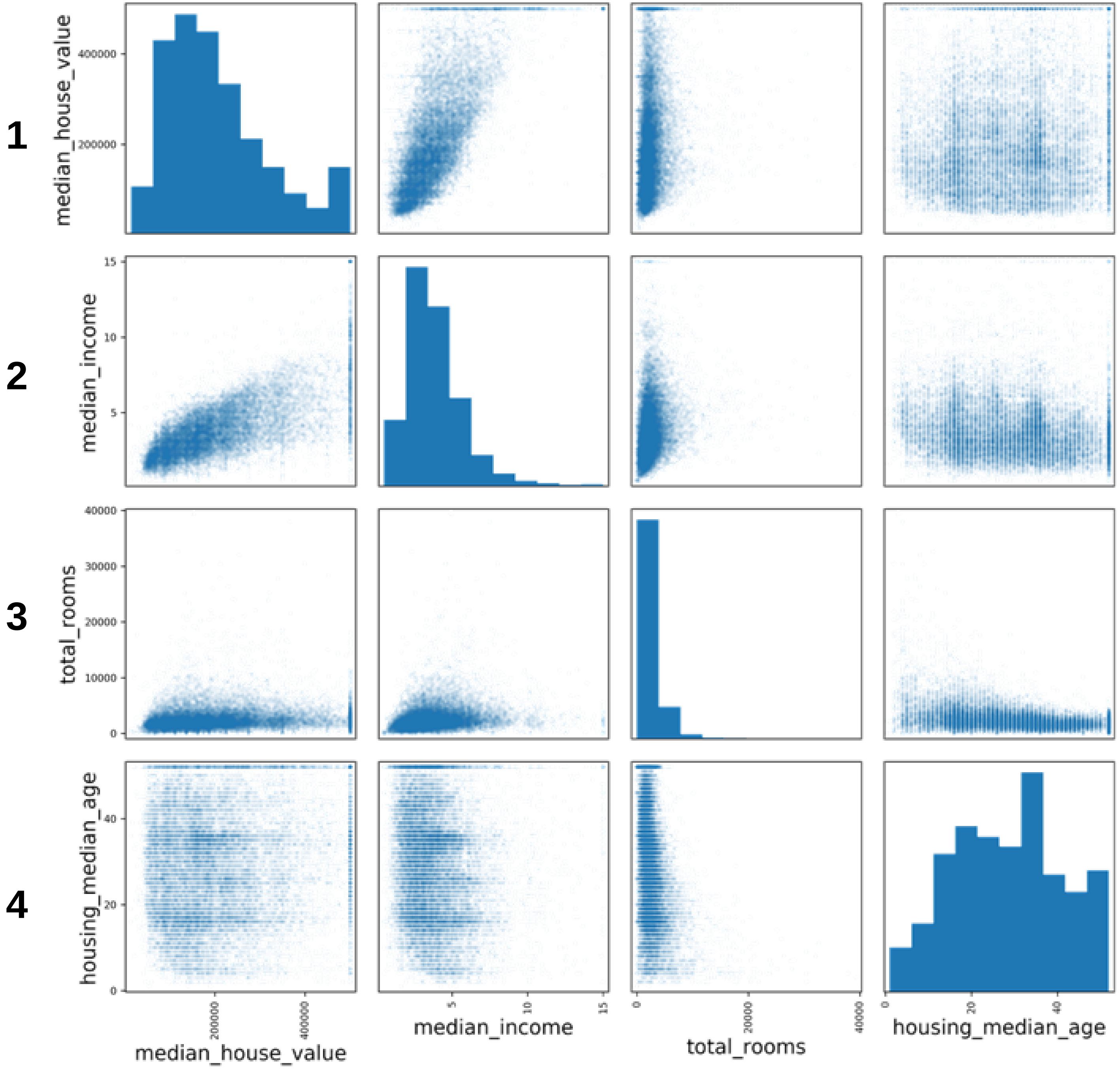
D

Feature Extraction

Explore correlations in the data to begin identifying important variables

Correlation with our response variable, median_house_value:

1	median_house_value	1.000000
2	median_income	0.690647
3	total_rooms	0.133989
4	housing_median_age	0.103706
	households	0.063714
	total_bedrooms	0.047980
	population	-0.026032
	longitude	-0.046349
	latitude	-0.142983



Adapted from from Hands-On Machine Learning with Scikit-Learn & TensorFlow by Aurélion Géron

Transform variables (feature engineering)

D
Feature Extraction

```
median_house_value
median_income
total_rooms
housing_median_age
households
total_bedrooms
population
longitude
latitude
```

rooms_per_household = total_rooms / households

bedrooms_per_room = total_bedrooms / total_rooms

population_per_household = population / households

Resulting correlations:

```
median_house_value          1.000000
median_income               0.690647
rooms_per_household         0.158485
total_rooms                 0.133989
housing_median_age          0.103706
households                  0.063714
total_bedrooms              0.047980
population_per_household   -0.022030
population                 -0.026032
longitude                  -0.046349
latitude                   -0.142983
bedrooms_per_room          -0.257419
```

Adapted from from Hands-On Machine Learning with Scikit-Learn & TensorFlow by Aurélion Géron

Preprocessed data

- Divided our data into training and testing sets
- Viewed the data and looked for problems
- Engineered new features that have real-world meaning
- Categorical data transformed into binary features (1-hot-encoding) enabling ML techniques
- Missing values replaced (imputed)
- Features standardized (now have zero mean and std of 1)

We're ready to train a machine learning model and evaluate performance

Supervised learning in practice

Preprocessing
Explore & prepare data

Data Visualization and Exploration

Identify patterns that can be leveraged for learning and issues with data

Data Cleaning

- Missing data
- Noisy data
- Erroneous data

Scaling (Standardization)

Prepare data for use in scale-dependent algorithms.

Feature Extraction

Dimensionality reduction eliminates redundant information

Model training

Select models (hypotheses)

Select model options that may fit the data well. We'll call them "hypotheses".

Fit the model to training data

Pick the "best" hypothesis function of the options by choosing model parameters

Iteratively fine tune the model

Performance evaluation

Make a prediction on validation data

Metrics

Classification
Precision, Recall, F_1, ROC Curves (Binary), Confusion Matrices (Multiclass)

Regression
MSE, explained variance, R^2

Model Training Considerations

Model Selection

K-Nearest Neighbors
Linear regression
Logistic regression
Linear Discriminant Analysis
Naïve Bayes
Classification and Regression Trees
Random Forests
Support Vector Machines
Neural Networks

We will spend the first half of the course on these pieces

Other Considerations

Combine models through ensembles (bagging, boosting, stacking)

Selecting cost functions

Regularizing our models to avoid overfit

Selecting model hyperparameters through grid search, random search, or Bayesian optimization

Experiment with three models

Validation data performance

Model	Root Mean Square Error RMSE ($)	RMSE / Median Home Price * 100 (%)
Linear regression	68,628	38.1
Random forest	52,564	29.2
Random forest with feature selection	49,694	27.6

Once we have a model we are confident in, we can evaluate our generalization performance on our **test set**:

Test set performance	47,766	26.5

Operationalizing the solution

Now the code needs to be run at scale
(production-grade code, production environment)

The ML solution will need to be maintained and updated
(Update the codebase, update model with new data)

Continued monitoring of accuracy will be required
(check for model drift – are distributions changing?)

How fast does it need to run?
(i.e. in real-time)

Supervised Machine Learning Process

1. Define your problem, set your goal, and how you will measure success
2. Get the data
3. Explore and prepare the data
4. Propose one or more hypotheses: prospective models
5. Evaluate model performance and iteratively fine tune
6. Deploy your trained model

References

Abu-Mostafa, Yaser S., Malik Magdon-Ismail, and Hsuan-Tien Lin. Learning from data. Vol. 4. New York, NY, USA:: AMLBook, 2012.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. New York: Springer series in statistics, 2001.

Géron, Aurélion. Hands-On Machine Learning with Scikit-Learn & TensorFlow, 2017.

Moore, Cristopher, and Stephan Mertens. The nature of computation. OUP Oxford, 2011.