

EMBEDDED VISION DESIGN 3

FEATURE EXPLORATION HANDS-ON

JEROEN VEEN



HAN_UNIVERSITY
OF APPLIED SCIENCES

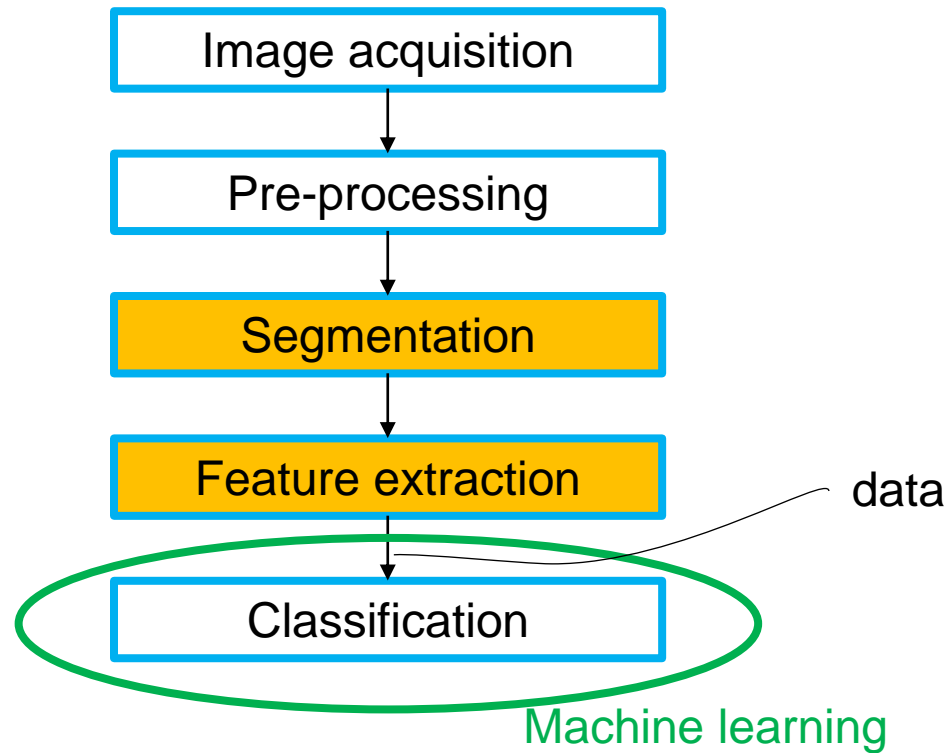
QUIZ TIME

- Individual, multiple-choice questions
- Online: <http://www.socrative.com> room **1PTGB6PY**
- Open book quiz, so books and slides can be consulted
- **HAN student number**, so NOT your name, nickname or anything else.
- Quiz starts exactly at class hour and takes 10 minutes.
- Be on time and have your equipment prepared.

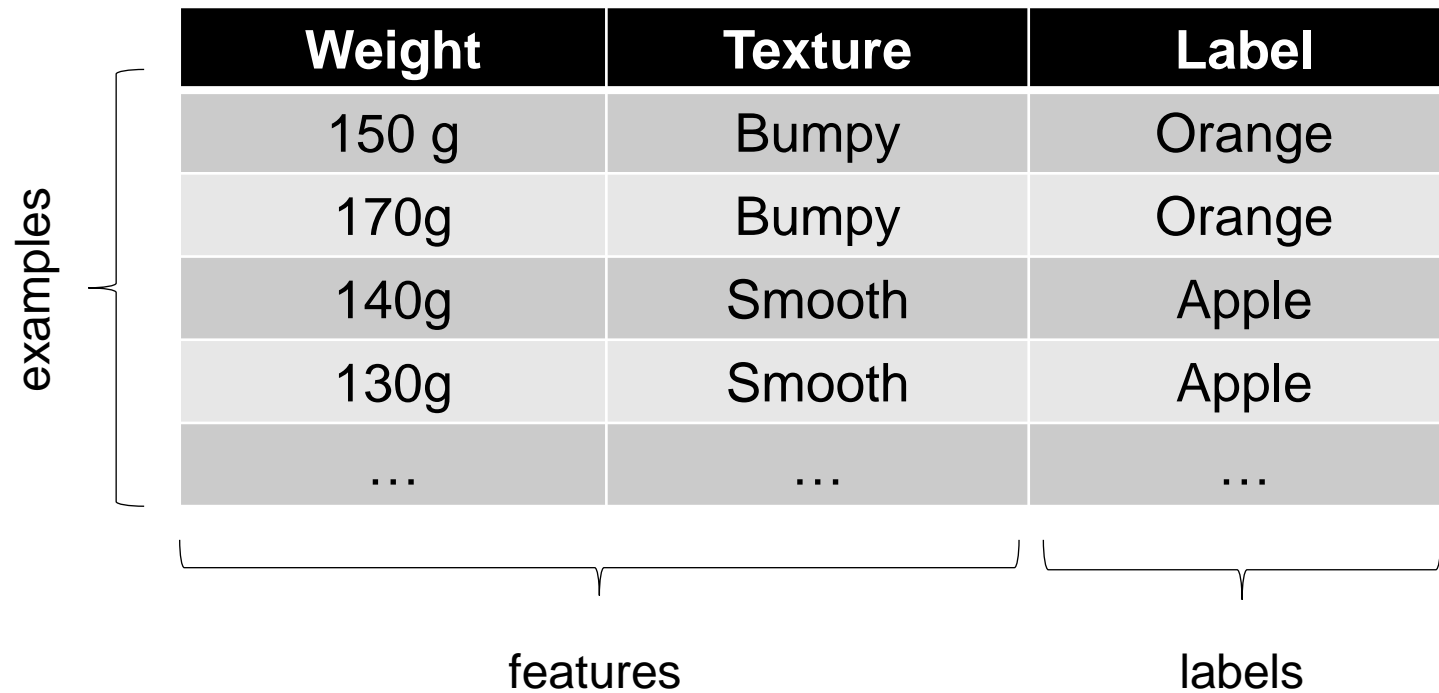
CONTENTS

- Basic segmentation and feature extraction
- Splitting your data
- Exploratory data analysis
- Feature engineering
- Data preparation

A JUMP-START TO DATA 2



TRAINING DATA



The diagram illustrates the structure of training data. A table with three columns (Weight, Texture, Label) and five rows of data is shown. A vertical bracket on the left labeled 'examples' spans all rows. A horizontal bracket below the first two columns labeled 'features' spans the Weight and Texture columns. Another horizontal bracket below the last column labeled 'labels' spans the Label column.

Weight	Texture	Label
150 g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
...

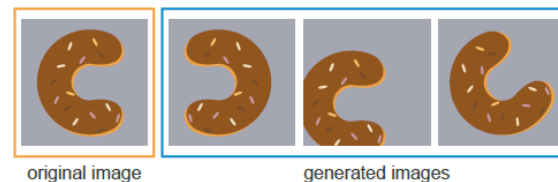
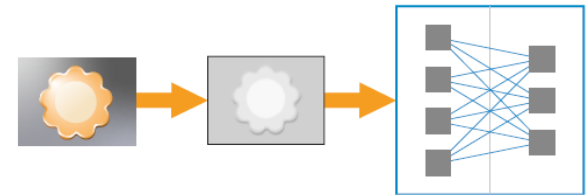
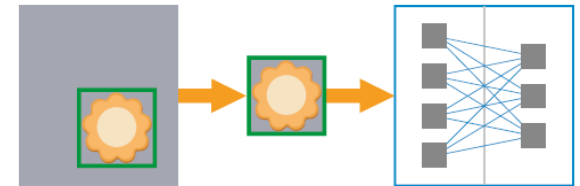
GENERAL TIPS ON DATA

- Reduction of the image size
- Minimize variance

(If the avoidable differences to the inspected images are reduced, less image data is needed to train an algorithm)

- Static ambient conditions such as stable lighting, a consistent monochrome background, fixed positioning of the inspected objects and unchanging orientation

- Increase in the number and variance of the training data, e.g. by generating additional training data, or so-called augmentation.



BASIC SEGMENTATION EXAMPLE

- segment.py
create segmentation function in a module

```
6  def maskBlueBG(img):
7      """ Asssuming the background is blue, segment the image and return a
8          |   BW image with foreground (white) and background (black)
9          |   """
10         # Change image color space
11         img_hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)
12         # Note that 0≤V≤1, 0≤S≤1, 0≤H≤360 and if H<0 then H←H+360
13         # 8-bit images: V←255V,S←255S,H←H/2(to fit to 0 to 255)
14         # see https://docs.opencv.org/4.5.3/de/d25/imgproc\_color\_conversions.html#color\_convert\_rgb\_hsv
15
16         # Define background color range in HSV space
17         light_blue = (75,125,0) # converted from HSV value obtained with colorpicker (150,50,0)
18         dark_blue  = (140,255,255) # converted from HSV value obtained with colorpicker (250,100,100)
19
20         light_blue = (0,0,0) # converted from HSV value obtained with colorpicker (150,50,0)
21         dark_blue  = (255,50,255) # converted from HSV value obtained with colorpicker (250,100,100)
22
23         # Mark pixels outside background color range
24         mask = ~cv.inRange(img_hsv, light_blue, dark_blue)
25         return mask
```

BASIC SEGMENTATION EXAMPLE

- segment.py

```
# mask background  
mask = maskBlueBG(img)  
masked_img = cv.bitwise_and(img, img, mask=mask)
```



BASIC FEATURE EXTRACTION EXAMPLE

- extract.py or fetch_data.py

import our segmentation function from module

```
5 from segment import maskBlueBG
```

segment the image and do a bit of denoising

```
# mask background
img_BW = maskBlueBG(img)

# perform a series of erosions and dilations to remove any small regions of noise
img_BW = cv.erode(img_BW, None, iterations=2)
img_BW = cv.dilate(img_BW, None, iterations=2)
```

BASIC FEATURE EXTRACTION EXAMPLE

- extract.py or fetch_data.py

get some features from contour

```
# find largest contour
contour = getLargestContour(img_Bw)

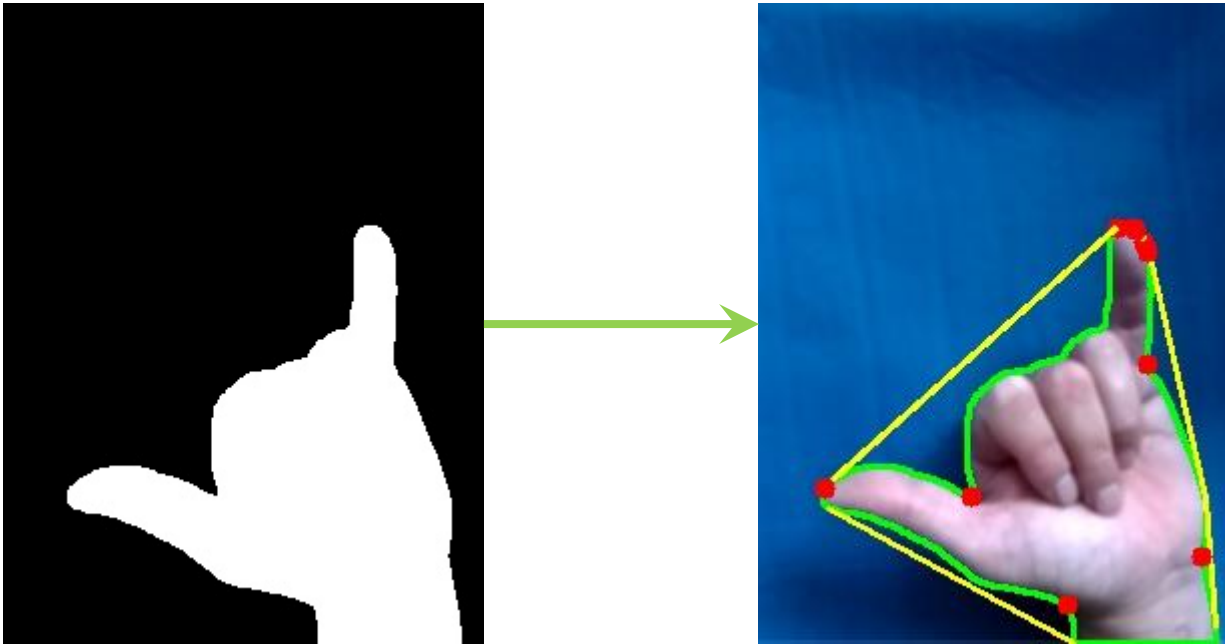
# extract features
features, defects = getContourFeatures(contour)
print("[INFO] contour features: {}".format(features))
```

using standard OpenCV functions to find perimeter, area, etc.

- You can do much better than this!!

BASIC FEATURE EXTRACTION EXAMPLE

- 03_extract.py or 04_fetch_data.py



BASIC FEATURE EXTRACTION EXAMPLE

- `extract.py` or `fetch_data.py`

```
>>> gestures.feature_names
['area', 'perimeter', 'aspect ratio', 'extent']
```

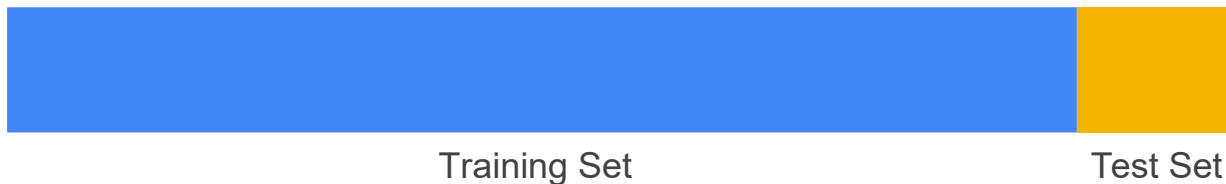
```
>>> gestures.data  
array([[2.04375000e+04, 8.50541191e+02, 8.36206897e-01, 4.54085940e-01],  
       [2.02550000e+04, 8.37068102e+02, 7.81115880e-01, 4.77644673e-01],  
       [2.08760000e+04, 8.42582821e+02, 8.70370370e-01, 5.14085894e-01],  
       ...])
```

```
>>> gestures.unique_targets
array(['hang loose', 'ignore', 'paper', 'rock', 'scissors'], dtype='<U10')
```

[illegible]

TRAINING AND TEST SETS: SPLITTING DATA

- **training set**—a subset to train a model.
- **test set**—a subset to test the trained model.
- You could imagine slicing the single data set as follows:

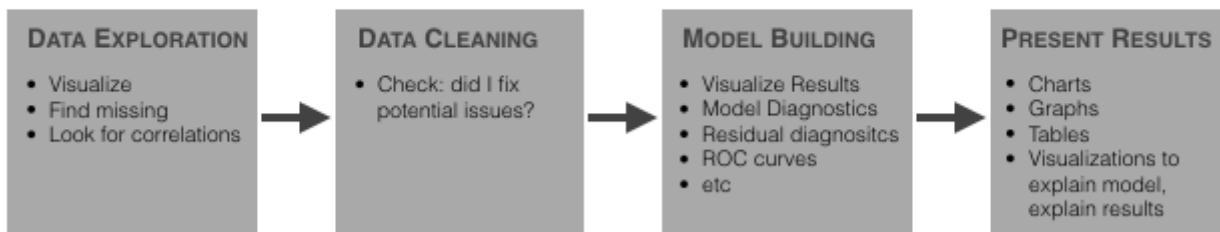


- Make sure that your test set meets the following two conditions:
 - Is large enough to yield statistically meaningful results.
 - Is representative of the data set as a whole. In other words, don't pick a test set with different characteristics than the training set.

EXPLORATORY DATA ANALYSIS

- Initial investigations on data to discover patterns
- Spot anomalies, and to check assumptions
- Summary statistics and graphical representations.

WE USE DATA ANALYSIS AND VISUALIZATION AT EVERY STEP OF THE MACHINE LEARNING PROCESS



Source: Stanford: Statistical reasoning MOOC

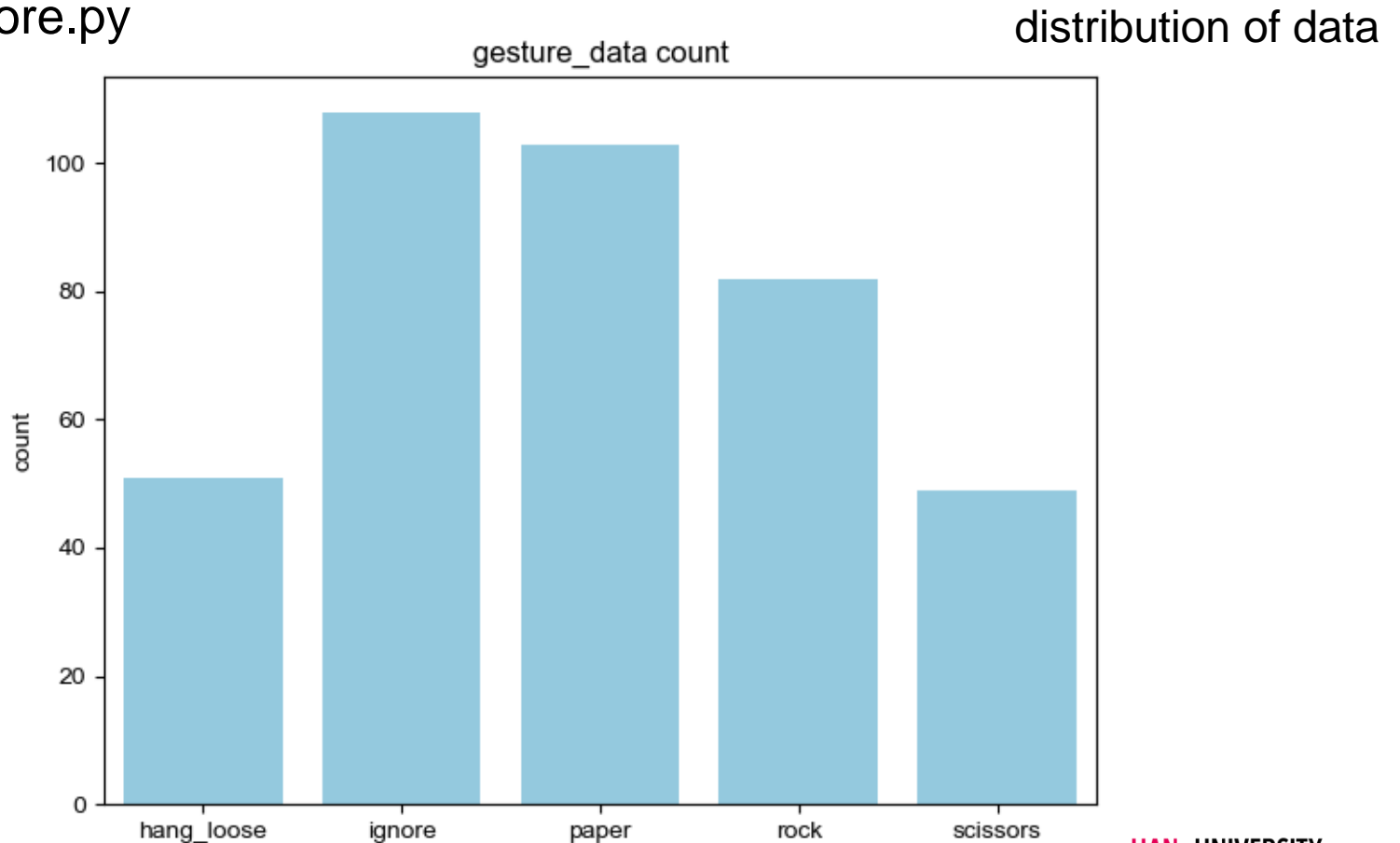
DATA EXPLORATION EXAMPLE

- explore.py

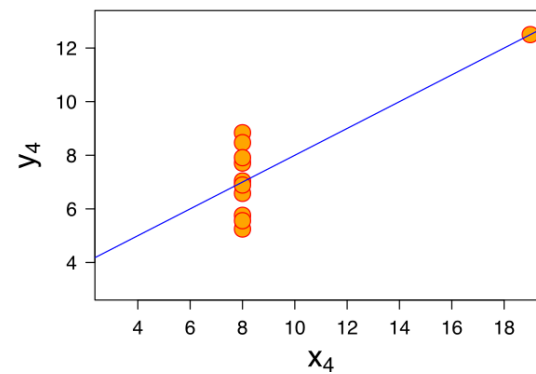
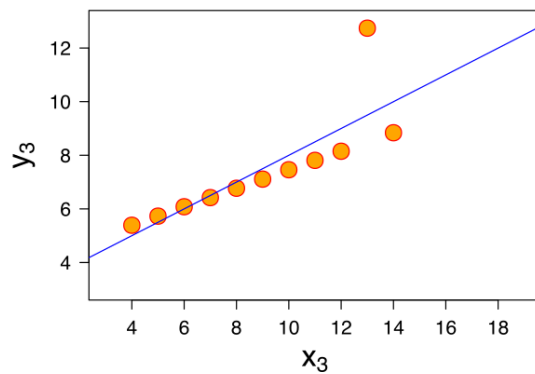
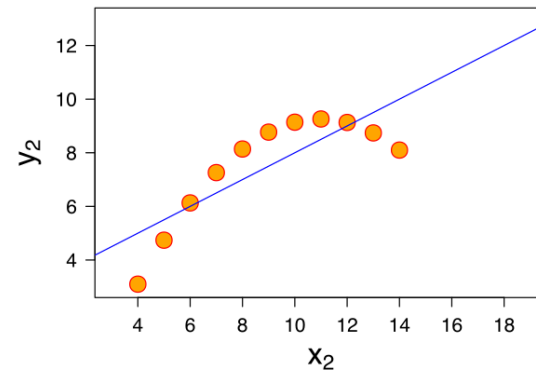
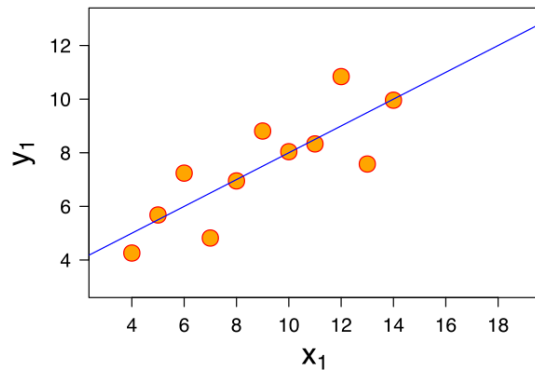
```
1  import numpy as np
2  from fetch_data import fetch_data
3  from sklearn.preprocessing import LabelEncoder, StandardScaler
4  from sklearn.model_selection import train_test_split
5  import matplotlib.pyplot as plt
6  import seaborn as sns
7
8
9  if __name__ == "__main__":
10     """feature exploration"""
11     data_path = r'G:\My Drive\data\gesture_data'
12
13     # fetch the data
14     gestures = fetch_data(data_path)
15
16     # encode the categorical labels
17     le = LabelEncoder()
18     coded_labels = le.fit_transform(gestures.target)
19
20     # partition the data into training and testing splits using 75% of
21     # the data for training and the remaining 25% for testing
22     (trainX, testX, trainY, testY) = train_test_split(gestures.data, coded_labels,
23     test_size=0.25, stratify=gestures.target)#, random_state=42)
24
```

DATA EXPLORATION EXAMPLE

- explore.py



ANSCOMBE'S QUARTET

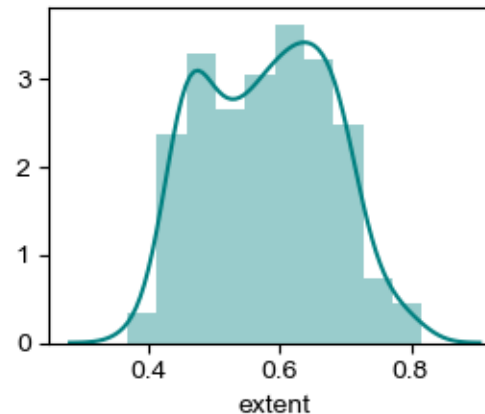
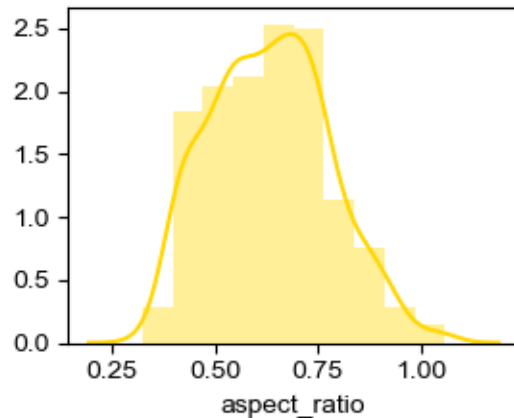
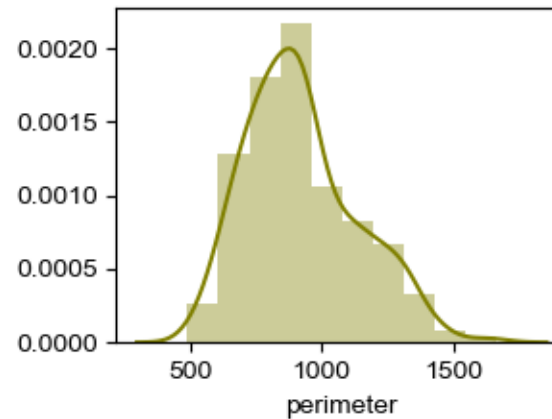
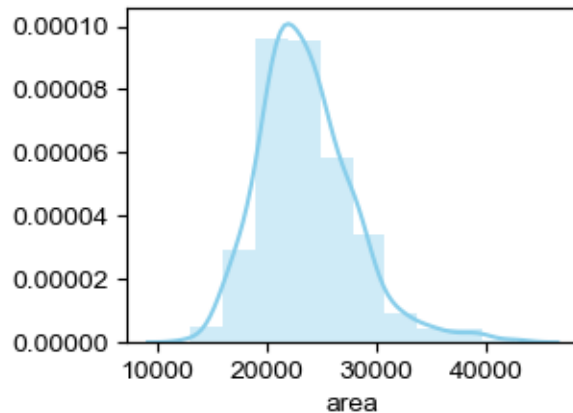


Source: By Anscombe.svg: Schutz(label using subscripts): Avenue - Anscombe.svg, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=9838454>

DATA EXPLORATION EXAMPLE

- explore.py

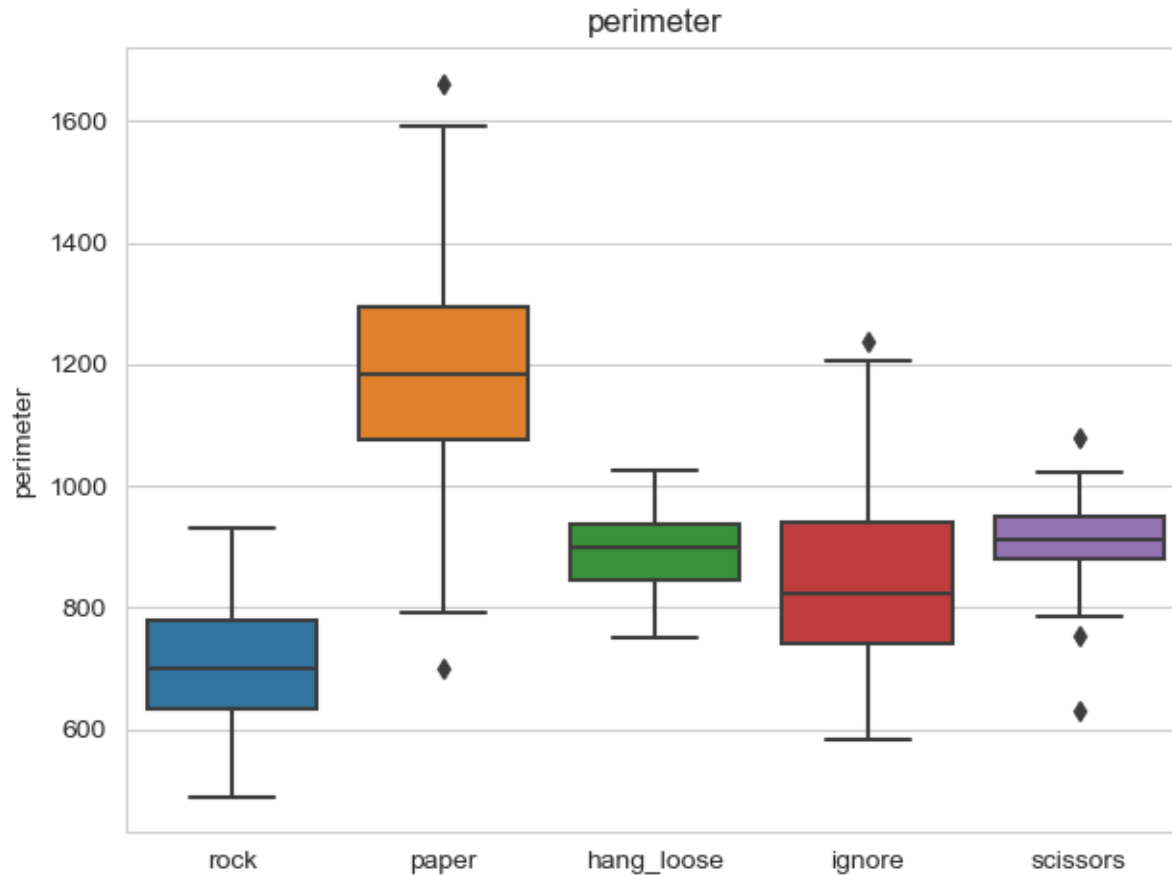
Total feature histograms



DATA EXPLORATION EXAMPLE

- explore.py

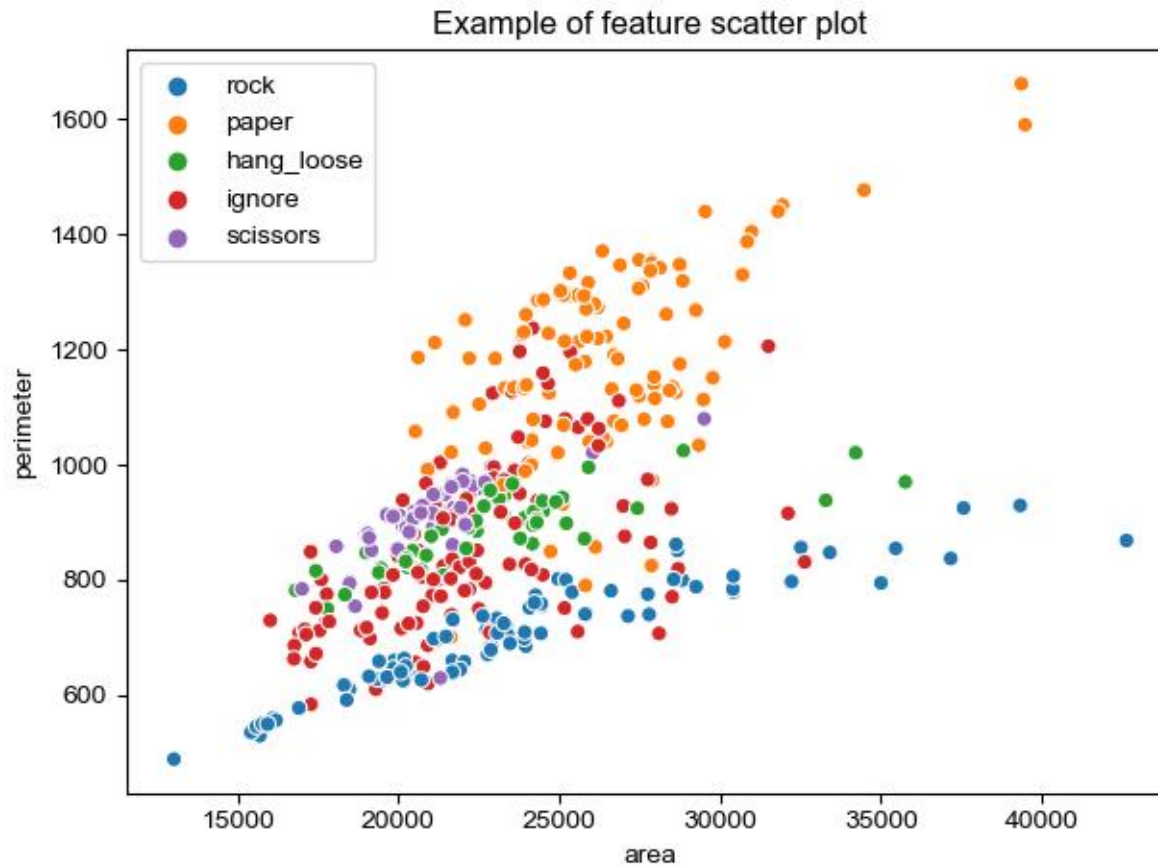
Boxplot of a single feature



DATA EXPLORATION EXAMPLE

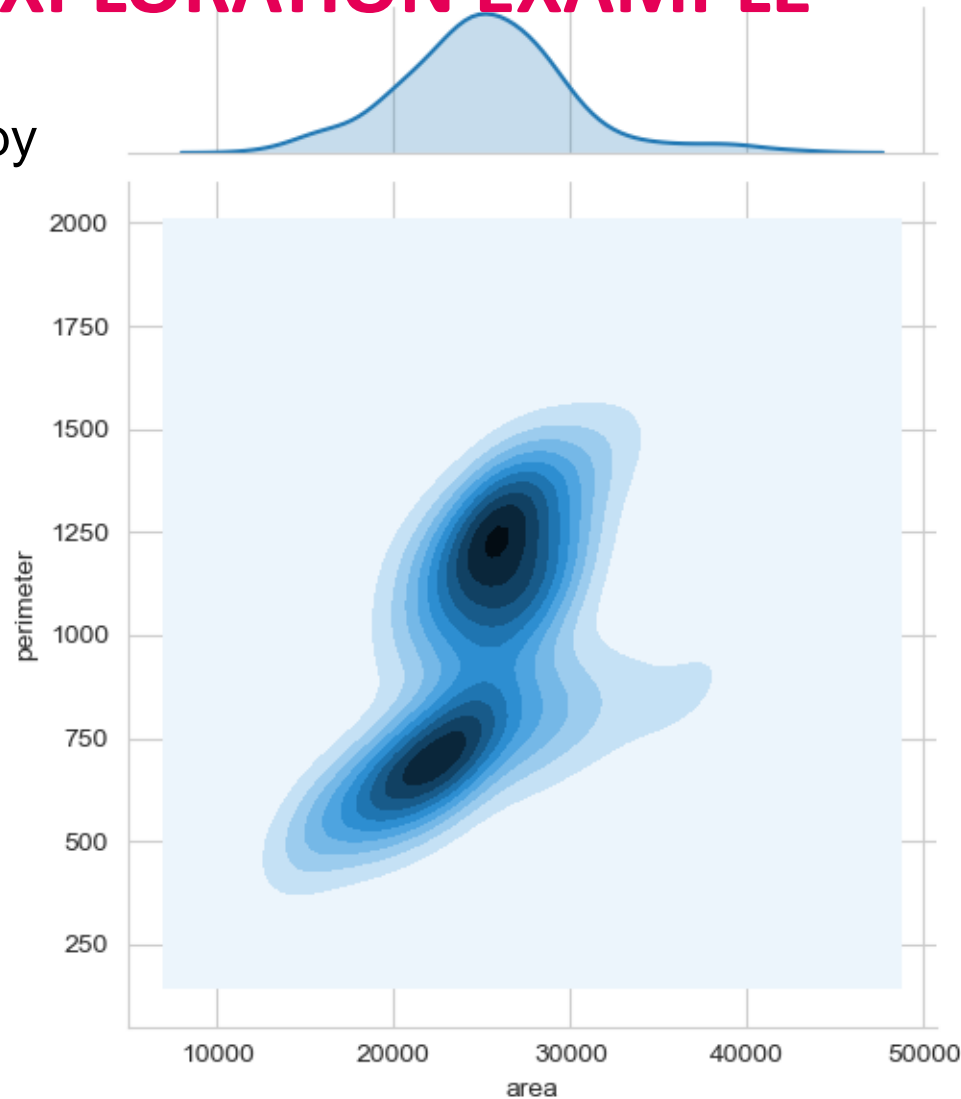
- explore.py

Scatterplot of 2 features



DATA EXPLORATION EXAMPLE

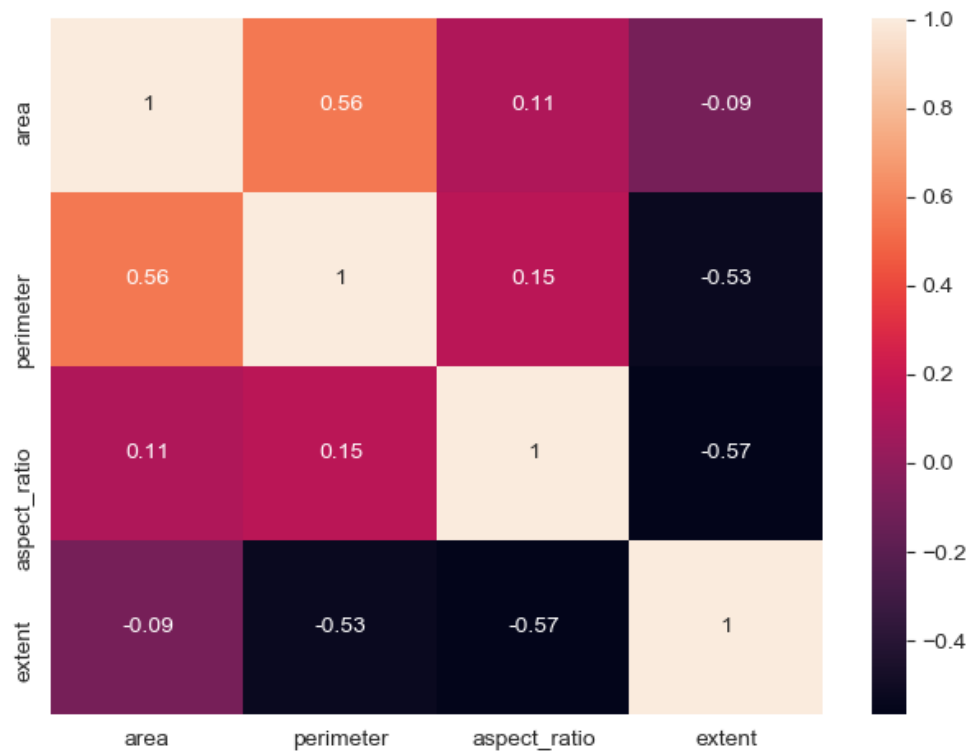
- explore.py



Joint distribution plot
of 2 features
for 2 classes

DATA EXPLORATION EXAMPLE

- explore.py

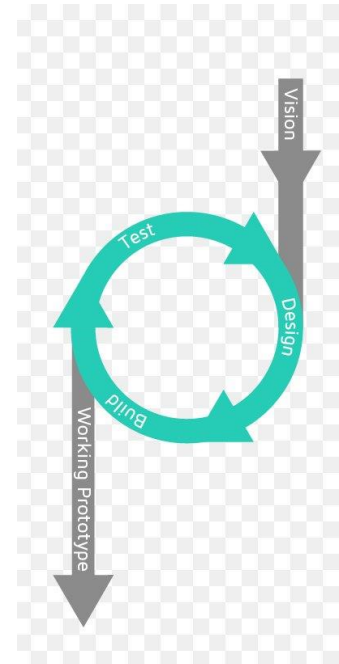


Feature correlation
heatmap

See: <https://www.statology.org/how-to-read-a-correlation-matrix/>

FEATURE ENGINEERING

- Based on your explorations:
 - Select features
 - Decompose features (e.g. area -> length, width)
 - Extract features (e.g. aggregate, combinations)
 - Add promising transformations of features (e.g., $\log(x)$, \sqrt{x} , x^2 , etc.).
- Propose new features?
- Adjust data acquisition?
- Remember that machine learning is an iterative process!



MORE ON FEATURE SELECTION

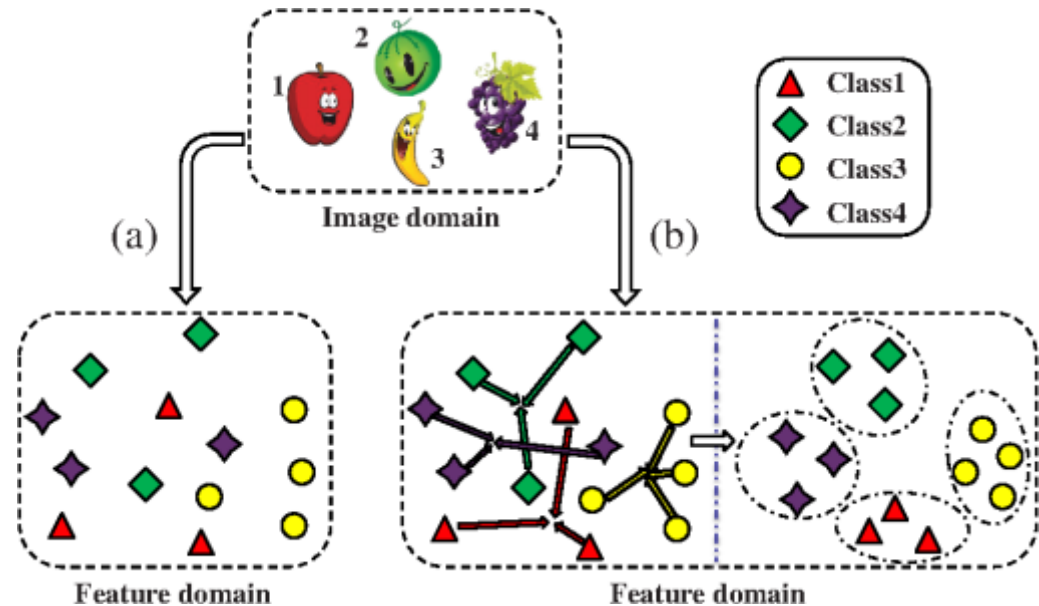
- Possible strategy
 - Create many, many features,
 - Use automated process to select best
 - E.g. using scikit learn feature selection module

https://scikit-learn.org/stable/modules/feature_selection.html

- Alternatively, use regularization techniques, we'll get to that in another lesson

QUALITIES OF GOOD FEATURES

- Informative
- Discriminating
- Independent
- Nearly unique



Source: <https://www.spiedigitallibrary.org/ContentImages/Journals/JEIME5/26/1/013023>

- NB feature scaling may be required

DATA PREPARATION

- Data cleaning:
 - Fix or remove outliers (optional)
 - Fill in missing values (e.g., with zero, mean, median...) or drop their rows (or columns).
- Feature computation:
 - Selection
 - Transformation
- Feature scaling:
 - Standardize or normalize features.

EXAMPLE OF FEATURE SCALING

- explore.py

```
from fetch_data import fetch_data  
from sklearn.preprocessing import LabelEncoder, StandardScaler  
from sklearn.model_selection import train_test_split
```

```
# Data preparation (note that a pipeline would help here)|  
trainX = StandardScaler().fit_transform(trainX)
```

- See https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html for more inspiration

EXAMPLE OF FEATURE SCALING

- explore.py

