Machine Learning 1

Regression & Classification

Agenda

- **Missing Values**
- Introduction to Machine Learning
- Scikit-Learn
- Regression
- Classification

Missing Values: NaN

```
string_data = Series(['aardvark', 'artichoke', np.nan, 'avocado'])
string_data
```

```
0     aardvark
1    artichoke
2          NaN
3      avocado
dtype: object
```

```
string_data.isnull()
```

```
0    False
1    False
2     True
3    False
dtype: bool
```

Strategy 1: filtering out missing values

```
from numpy import nan as NA
data = Series([1, NA, 3.5, NA, 7])
data.dropna()
```

```
0    1.0
2    3.5
4    7.0
dtype: float64
```

```
data[data.notnull()]
```

```
0    1.0
2    3.5
4    7.0
dtype: float64
```

Strategy 1: filtering out missing values

```
data = DataFrame([[1., 6.5, 3.], [1., NA, NA],
                  [NA, NA, NA], [NA, 6.5, 3.]])
cleaned = data.dropna()
data
```

	0	1	2
0	1.0	6.5	3.0
1	1.0	NaN	NaN
2	NaN	NaN	NaN
3	NaN	6.5	3.0

```
cleaned
```

	0	1	2
0	1.0	6.5	3.0

Strategy 2: filtering out some missing values

```
data = DataFrame([[1., 6.5, 3.], [1., NA, NA],
                  [NA, NA, NA], [NA, 6.5, 3.]])
cleaned = data.dropna()
data
```

	0	1	2
0	1.0	6.5	3.0
1	1.0	NaN	NaN
2	NaN	NaN	NaN
3	NaN	6.5	3.0

```
data.dropna(how='all')
```

	0	1	2
0	1.0	6.5	3.0
1	1.0	NaN	NaN
3	NaN	6.5	3.0

Strategy 2: filtering out some missing values

```
data[4] = NA
data
```

	0	1	2	4
0	1.0	6.5	3.0	NaN
1	1.0	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	6.5	3.0	NaN

```
data.dropna(axis=1, how='all')
```

	0	1	2
0	1.0	6.5	3.0
1	1.0	NaN	NaN
2	NaN	NaN	NaN
3	NaN	6.5	3.0

Strategy 3.1: filling in missing values

```
df.fillna(0)
```

	0	1	2
0	-0.204708	0.000000	0.000000
1	-0.555730	0.000000	0.000000
2	0.092908	0.000000	0.000000
3	1.246435	0.000000	-1.296221
4	0.274992	0.000000	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

Strategy 3.2: filling in missing values

```
df.fillna({1: 0.5, 2: -1})
```

	0	1	2
0	-0.577087	0.500000	-1.000000
1	0.523772	0.500000	-1.000000
2	-0.713544	0.500000	-1.000000
3	-1.860761	0.500000	0.560145
4	-1.265934	0.500000	-1.063512
5	0.332883	-2.359419	-0.199543
6	-1.541996	-0.970736	-1.307030

Strategy 3.3 forward filling missing values

```
df = DataFrame(np.random.randn(6, 3))
df.ix[2:, 1] = NA; df.ix[4:, 2] = NA
df
```

	0	1	2
0	-0.831154	-2.370232	-1.860761
1	-0.860757	0.560145	-1.265934
2	0.119827	NaN	0.332883
3	-2.359419	NaN	-1.541996
4	-0.970736	NaN	NaN
5	0.377984	NaN	NaN

Strategy 3.3 forward filling missing values

```
df.fillna(method='ffill')
```

	0	1	2
0	-0.831154	-2.370232	-1.860761
1	-0.860757	0.560145	-1.265934
2	0.119827	0.560145	0.332883
3	-2.359419	0.560145	-1.541996
4	-0.970736	0.560145	-1.541996
5	0.377984	0.560145	-1.541996

Strategy 3.3 forward filling missing values

```
df.fillna(method='ffill', limit=2)
```

	0	1	2
0	-0.831154	-2.370232	-1.860761
1	-0.860757	0.560145	-1.265934
2	0.119827	0.560145	0.332883
3	-2.359419	0.560145	-1.541996
4	-0.970736	NaN	-1.541996
5	0.377984	NaN	-1.541996

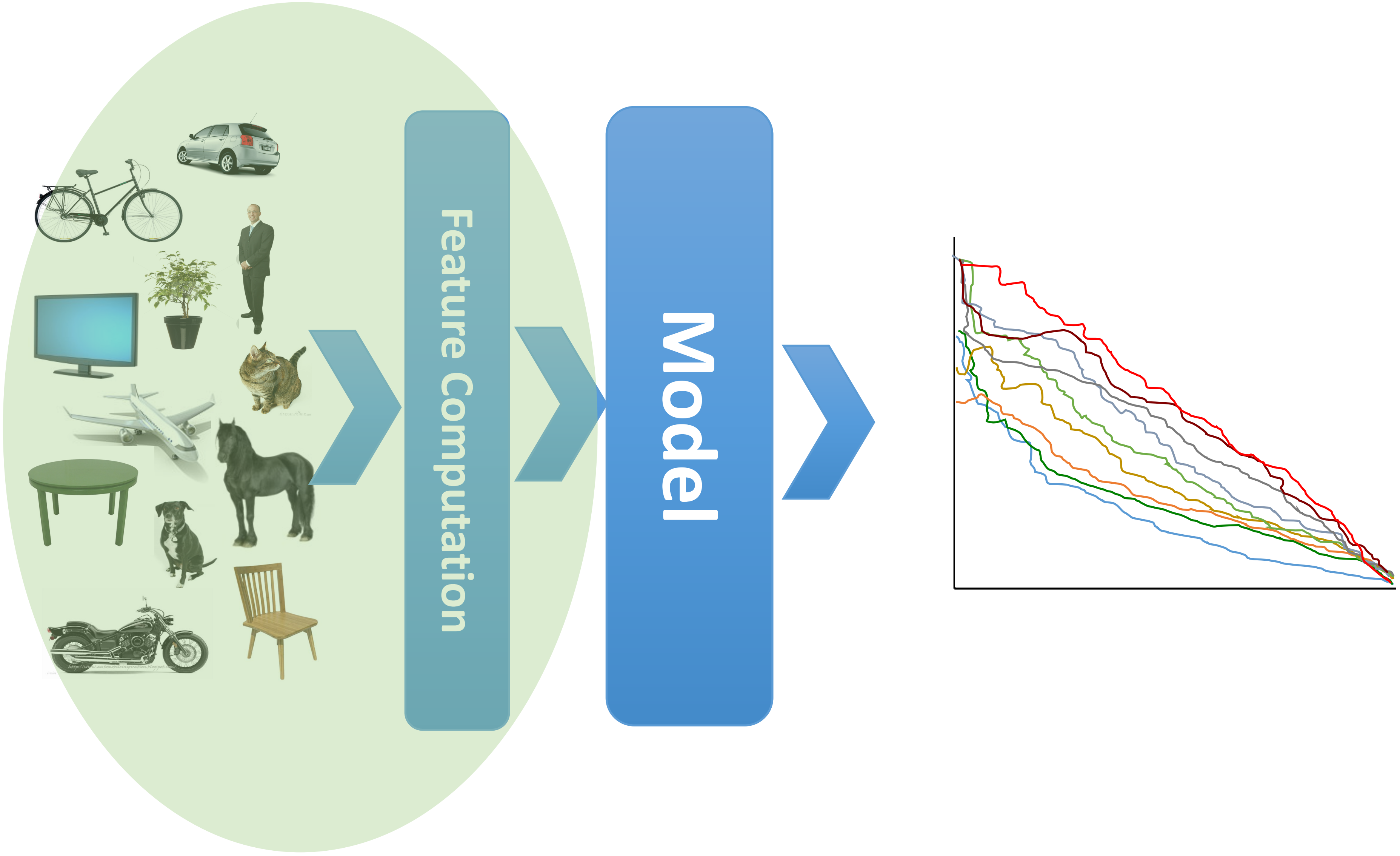
Agenda

- Missing Values
- **Introduction to Machine Learning**
- Scikit-Learn
- Regression
- Classification

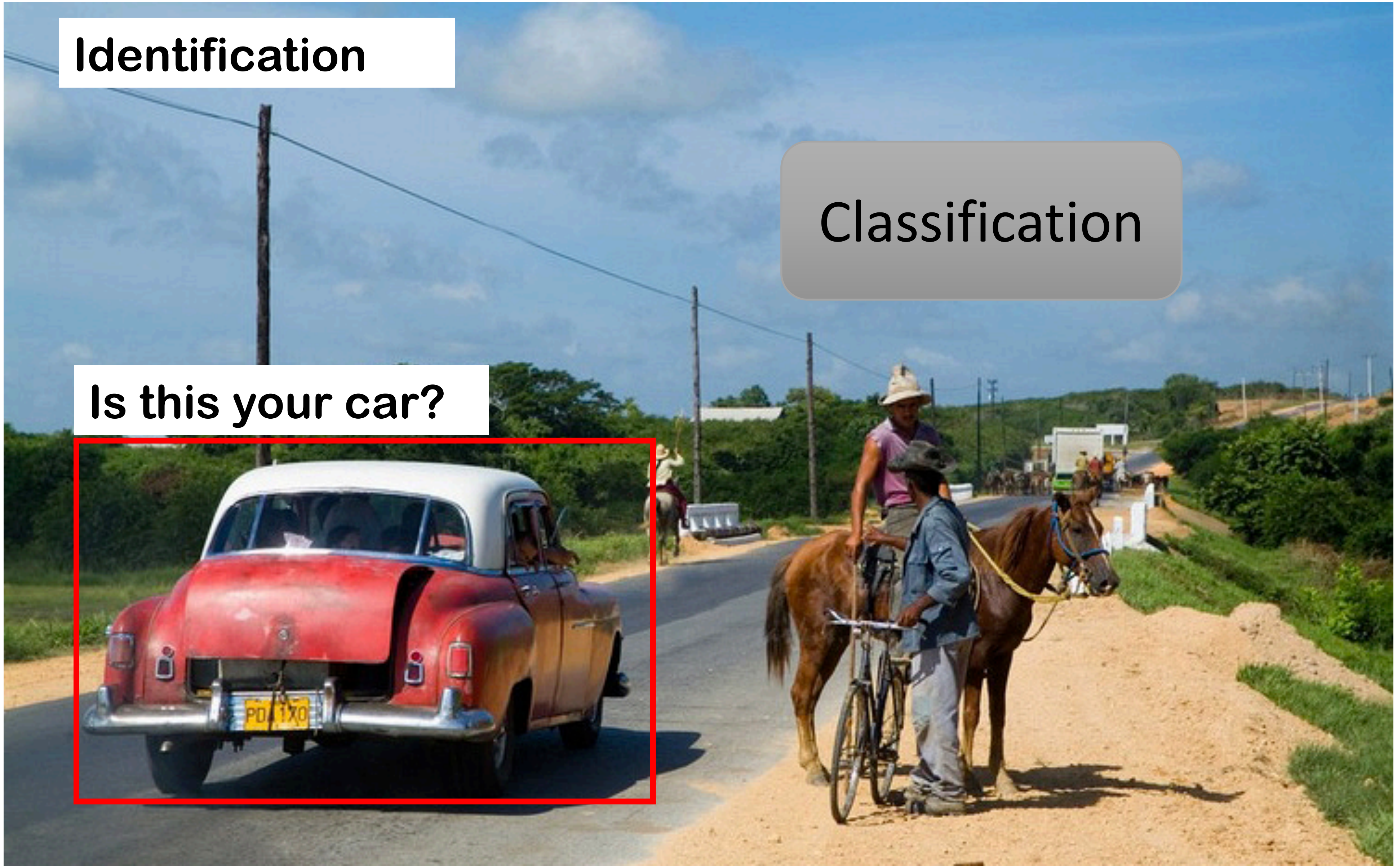
Machine learning

- T = tasks to be performed
- E = experience (usually in the form of history datasets)
- P = performance for an algorithm for T using E

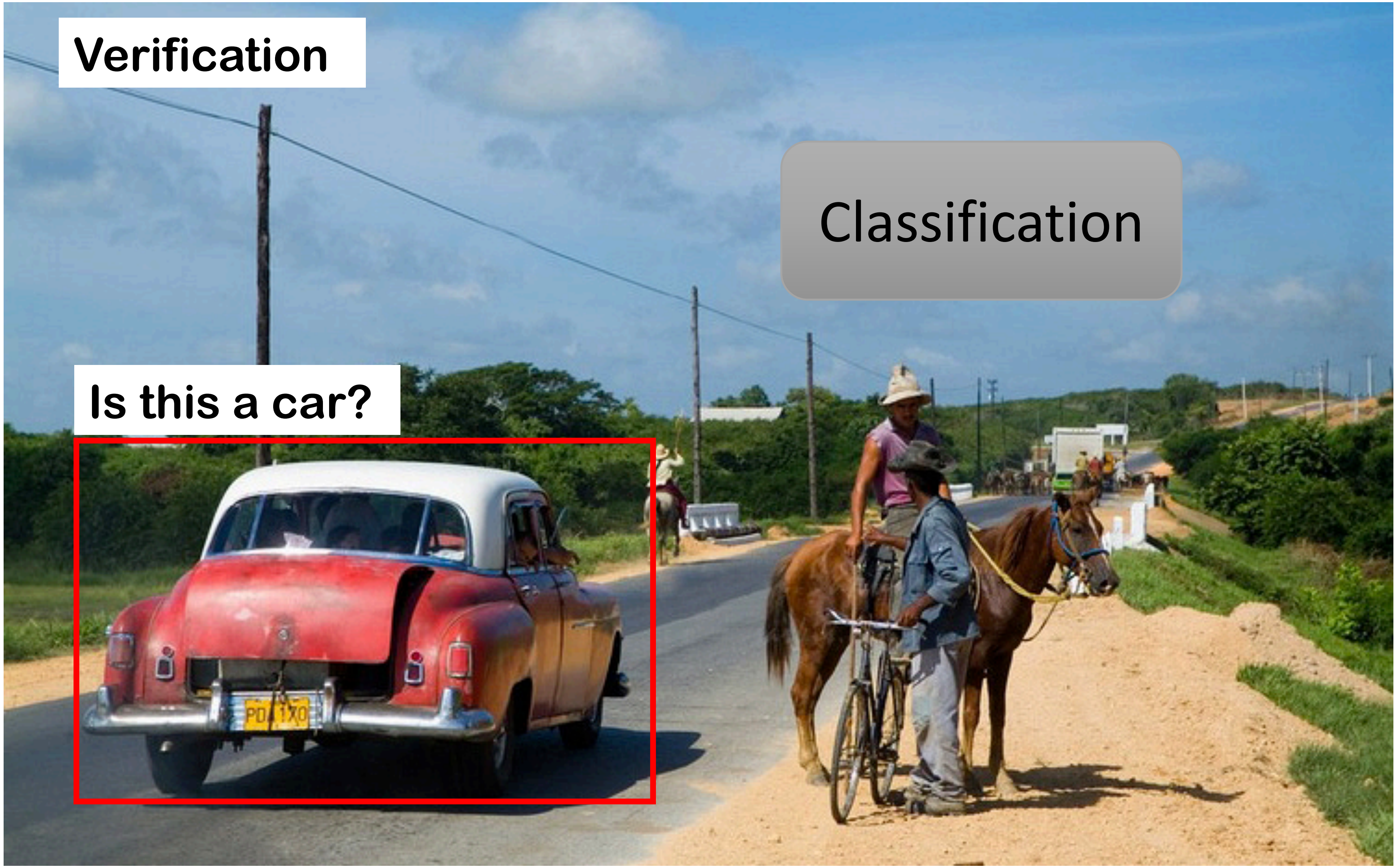
Typical Paradigms of Recognition



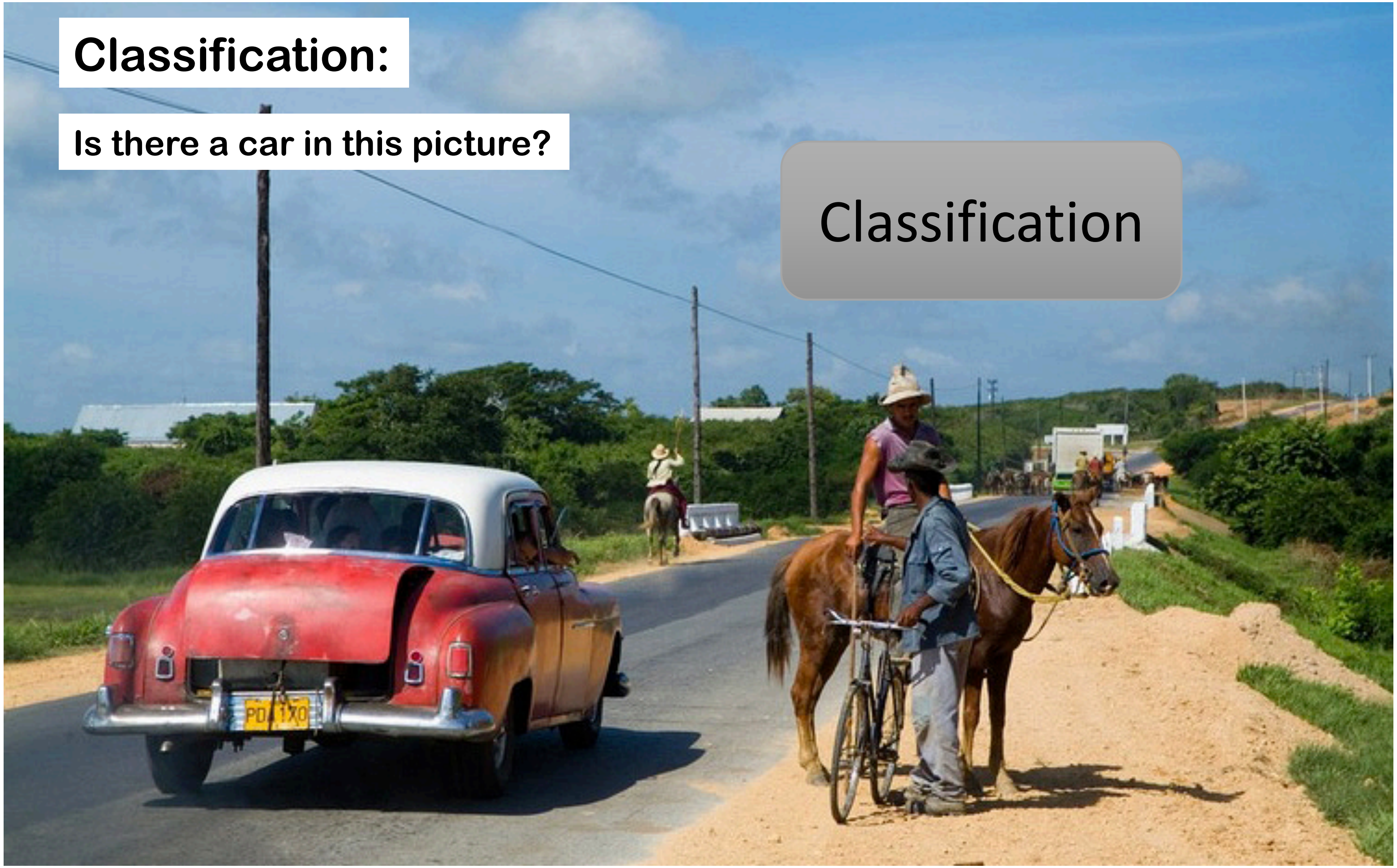
Visual Recognition



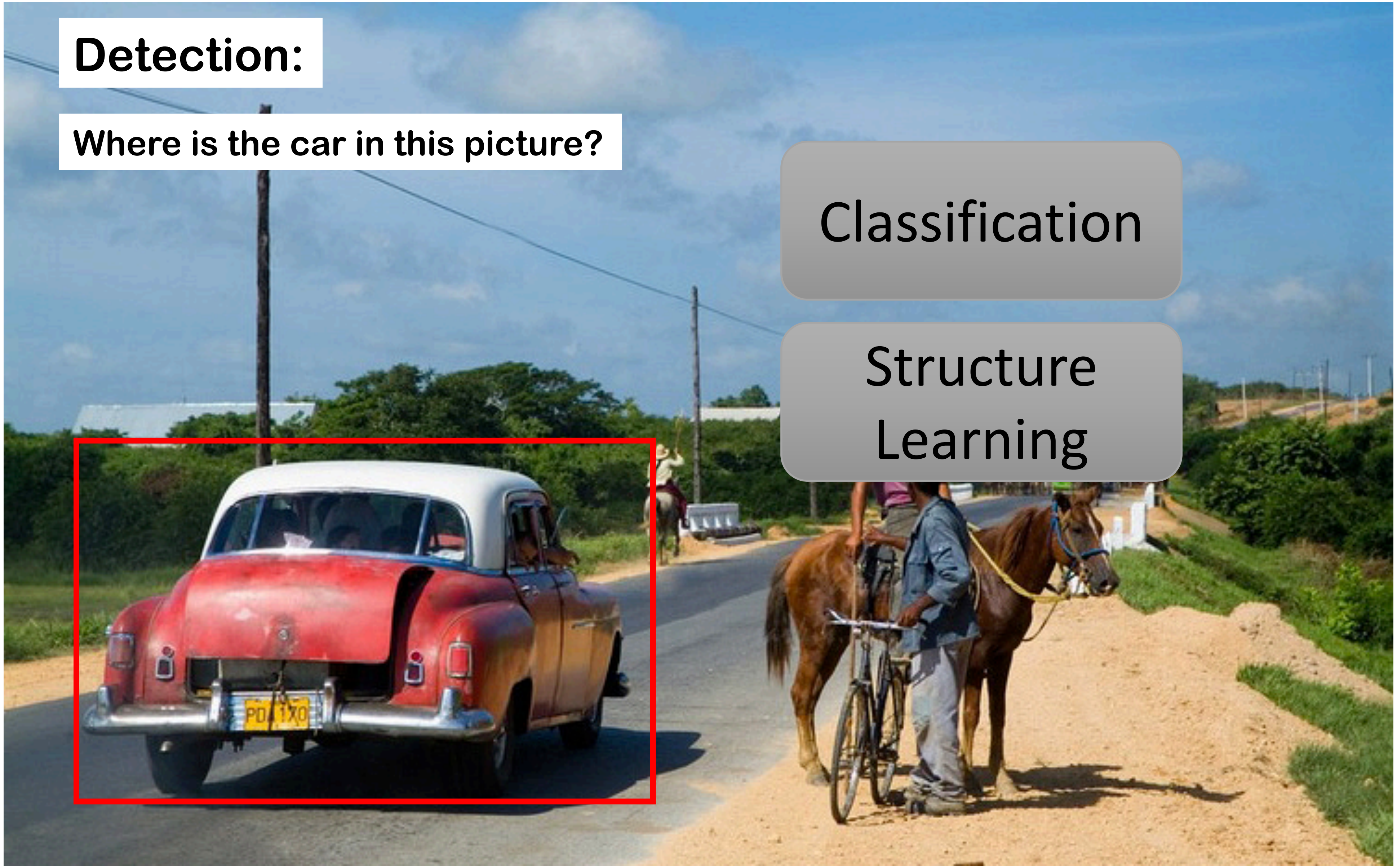
Visual Recognition



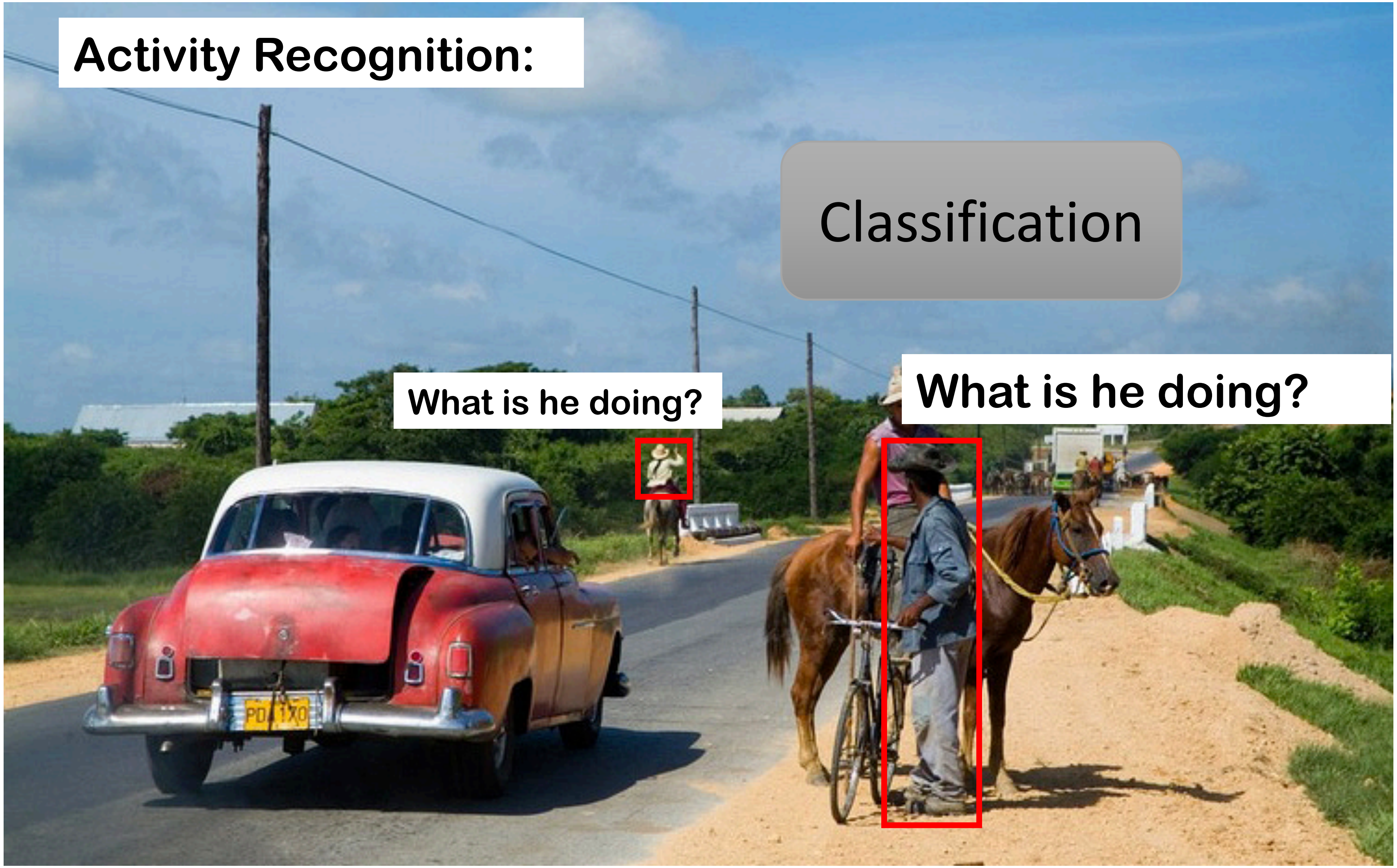
Visual Recognition



Visual Recognition

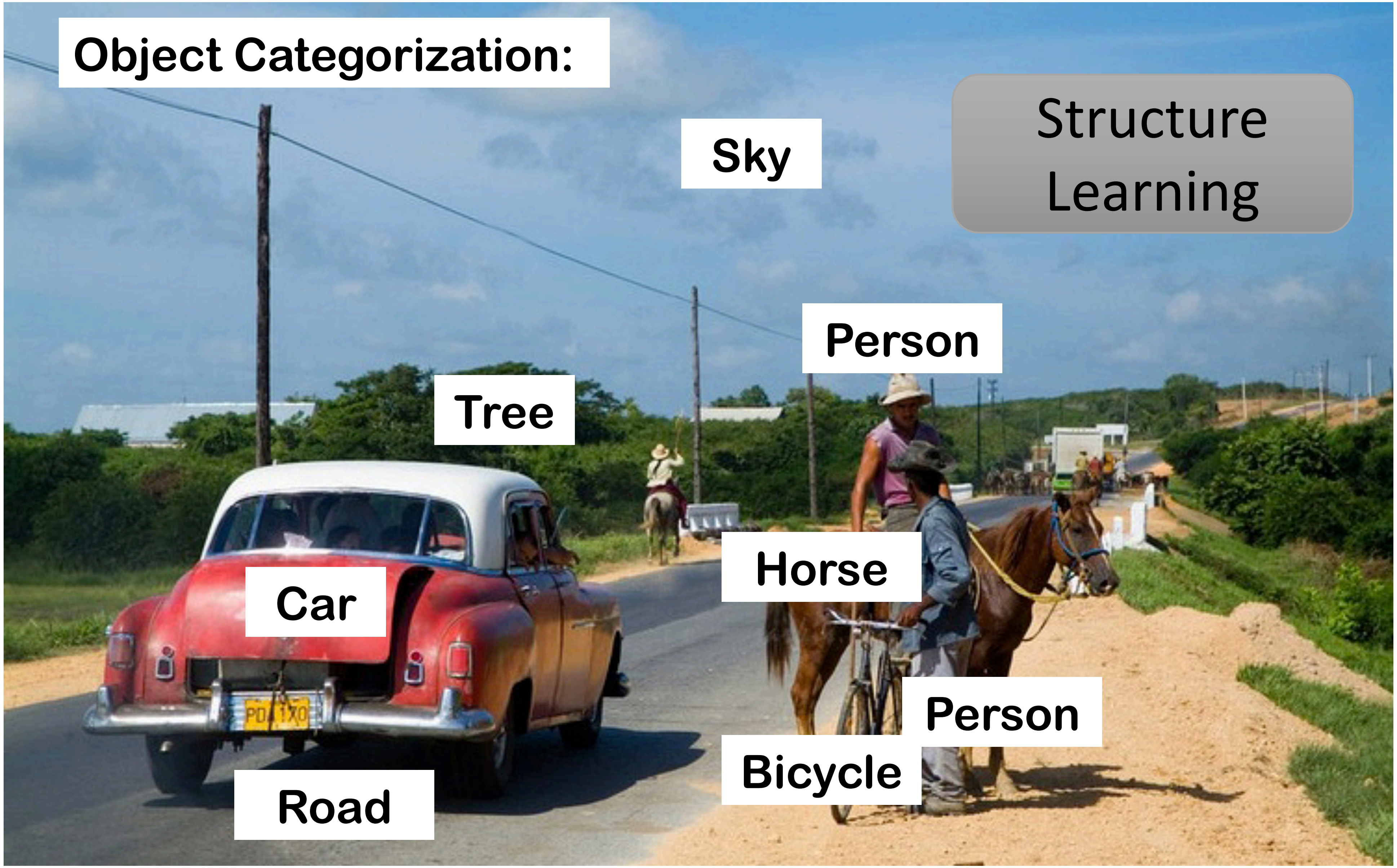


Visual Recognition

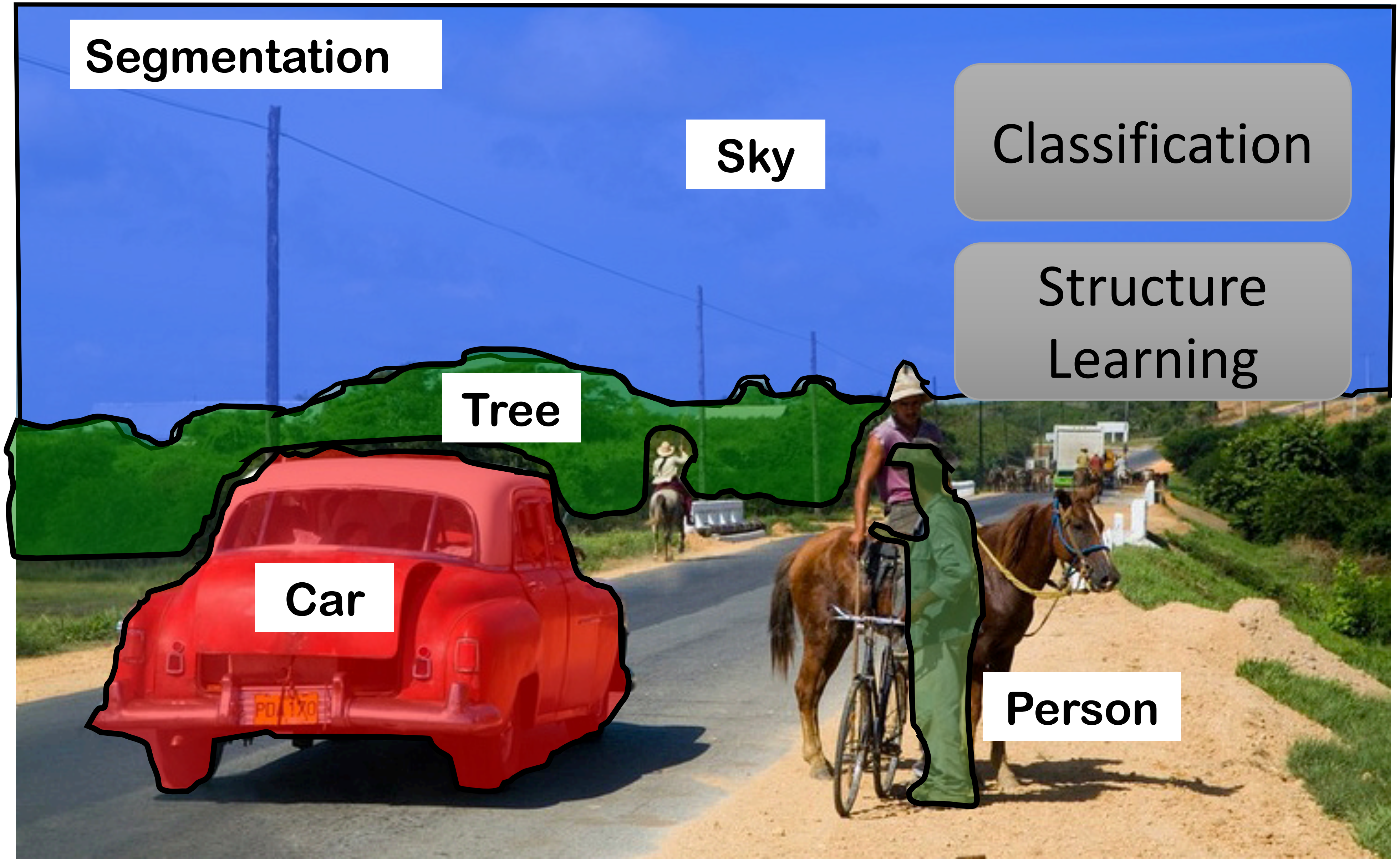


Visual Recognition

Visual Recognition



Visual Recognition



Types of ML approaches

- Regression
  - Linear regression
  - Structured output regression

- Classification
  - Generative vs. Discriminative
  - Supervised, unsupervised, semi-supervised, weakly supervised
  - Linear, nonlinear
  - Ensemble methods
  - Probabilistic

- Structure Learning
  - Graphical Models
  - Margin based approaches

Agenda

- Missing Values
- Introduction to Machine Learning
- **Scikit-Learn**
- Regression
- Classification

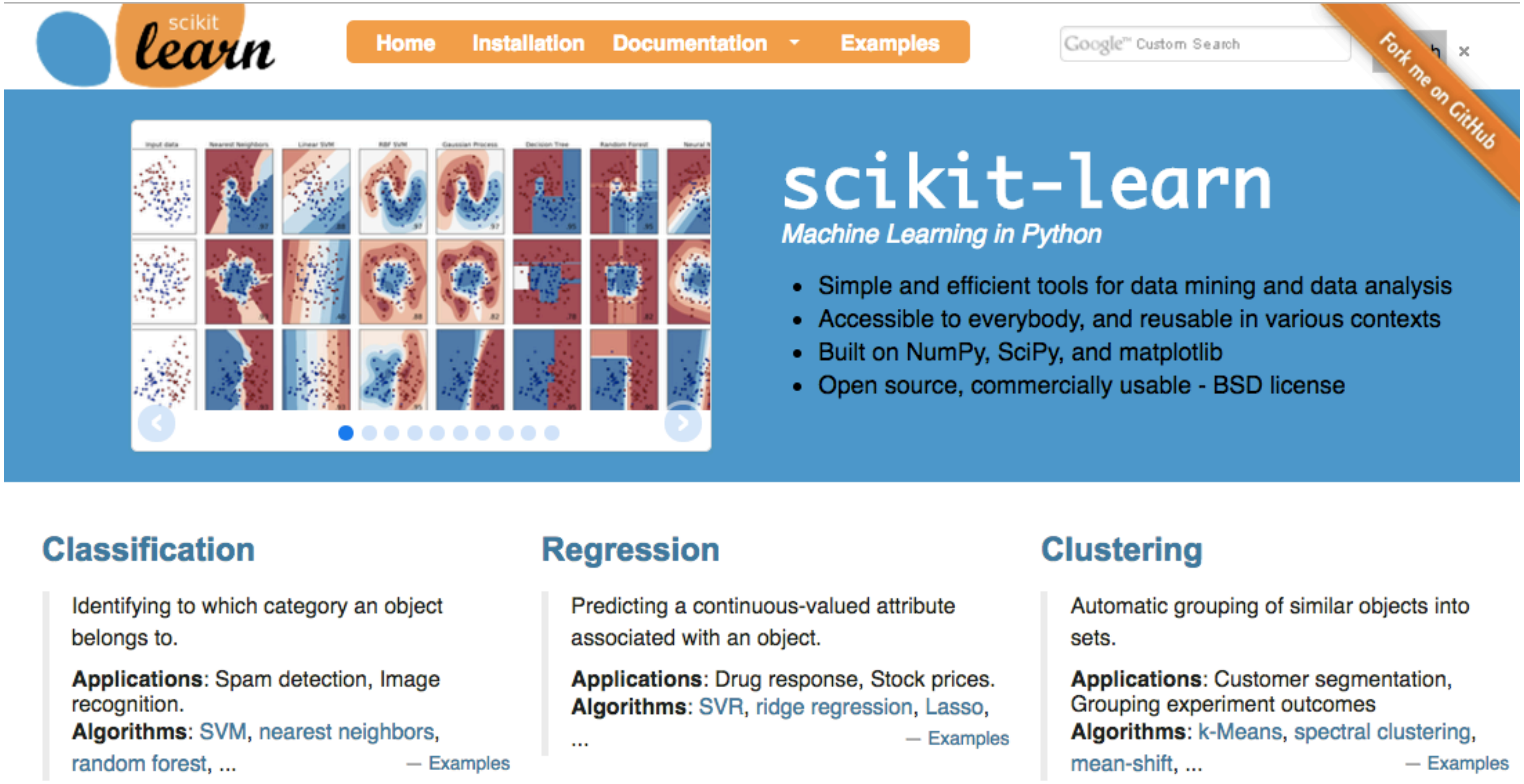
Scikit-learn : ML in Python

scikit-learn is a Python module integrating classic machine learning algorithms in the tightly-knit world of scientific Python packages (numpy, scipy, matplotlib)

Tools for:

- Regression
- Classification
- Clustering
- Dimensionality Reduction
- Model Selection

Scikit-learn : http://scikit-learn.org/



Fit and predict

- All models (Classification and regression) implement at least two functions:
  - fit(x, y) - Fit the model to the given dataset
  - predict(x) - Predict the y values associated with the x values

```
>>> clf = linear_model.Lasso(alpha = 0.1)
>>> clf.fit([[0, 0], [1, 1]], [0, 1])
Lasso(alpha=0.1, copy_X=True, fit_intercept=True, max_iter=1000,
   normalize=False, positive=False, precompute=False, random_state=None,
   selection='cyclic', tol=0.0001, warm_start=False)
>>> clf.predict([[1, 1]])
array([ 0.8])
```

Agenda

- Missing Values
- Introduction to Machine Learning
- Scikit-Learn
- **Regression**
- Classification

Regression: fitting ”lines”

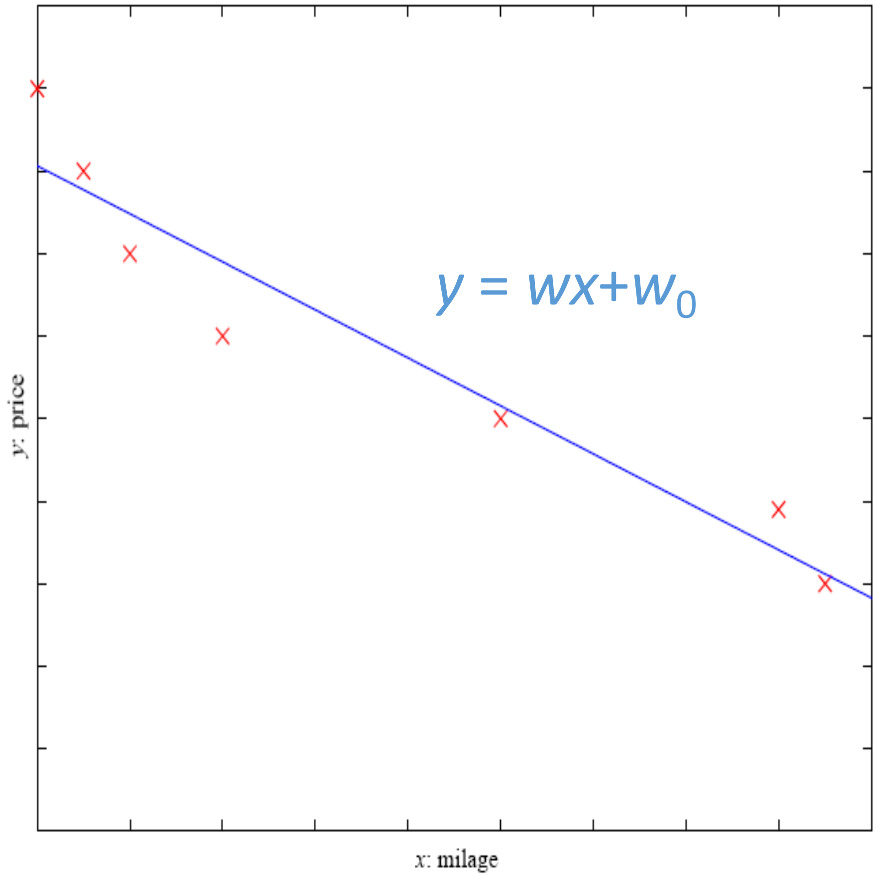
- Example: Price of a used car
- x : car attributes

  y : price

  $$y = g(x \mid \theta)$$

  $g(\ )$ model,

  θ parameters



Supervised Learning: Regression

- Regression: tries to fit a mathematical function that describes the learning data set E to minimize some error function
- **Example**: House price prediction
  - E = house prices (“targets”) with characteristics on # rooms, location, sq footage, etc.
  - T = predict the house sale price
  - P = how well the estimator can accurately predict the actual sale price

Linear Regression : House Price Prediction

- UC Irvine House Dataset : Boston House Prices for 506 homes
  - http://archive.ics.uci.edu/ml/datasets/Housing

```
from sklearn.datasets import load_boston
```

```
boston = load_boston()
```

```
print boston.DESCR
```

Boston Housing Dataset: 13 Attributes

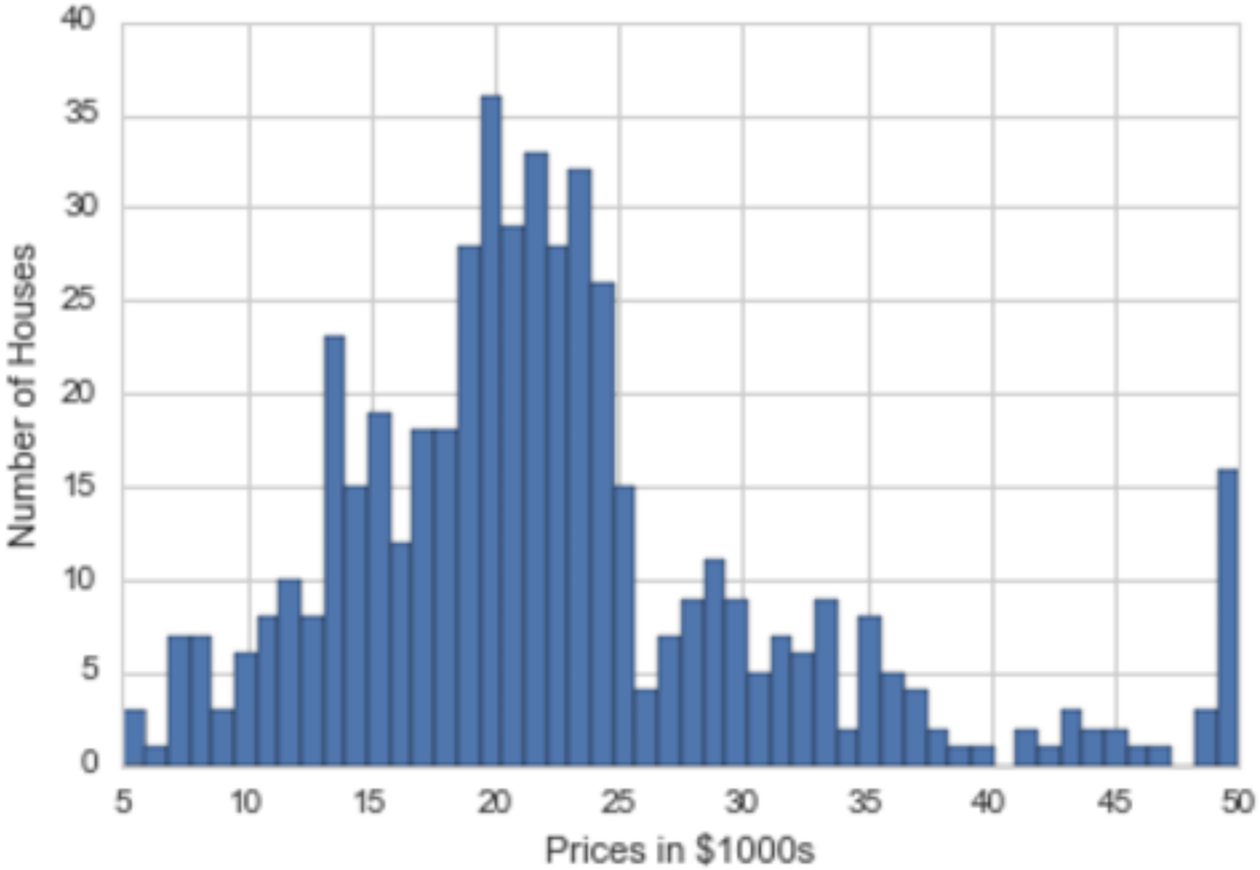
CRIM	per capita crime rate by town
ZN	proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS	proportion of non-retail business acres per town
CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX	nitric oxides concentration (parts per 10 million)
RM	average number of rooms per dwelling
AGE	proportion of owner-occupied units built prior to 1940
DIS	weighted distances to five Boston employment centres
RAD	index of accessibility to radial highways
TAX	full-value property-tax rate per $10,000
PTRATIO	pupil-teacher ratio by town
Bk	$1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
LSTAT	% lower status of the population

“boston.target”

MEDV	Median value of owner-occupied homes in $1000's

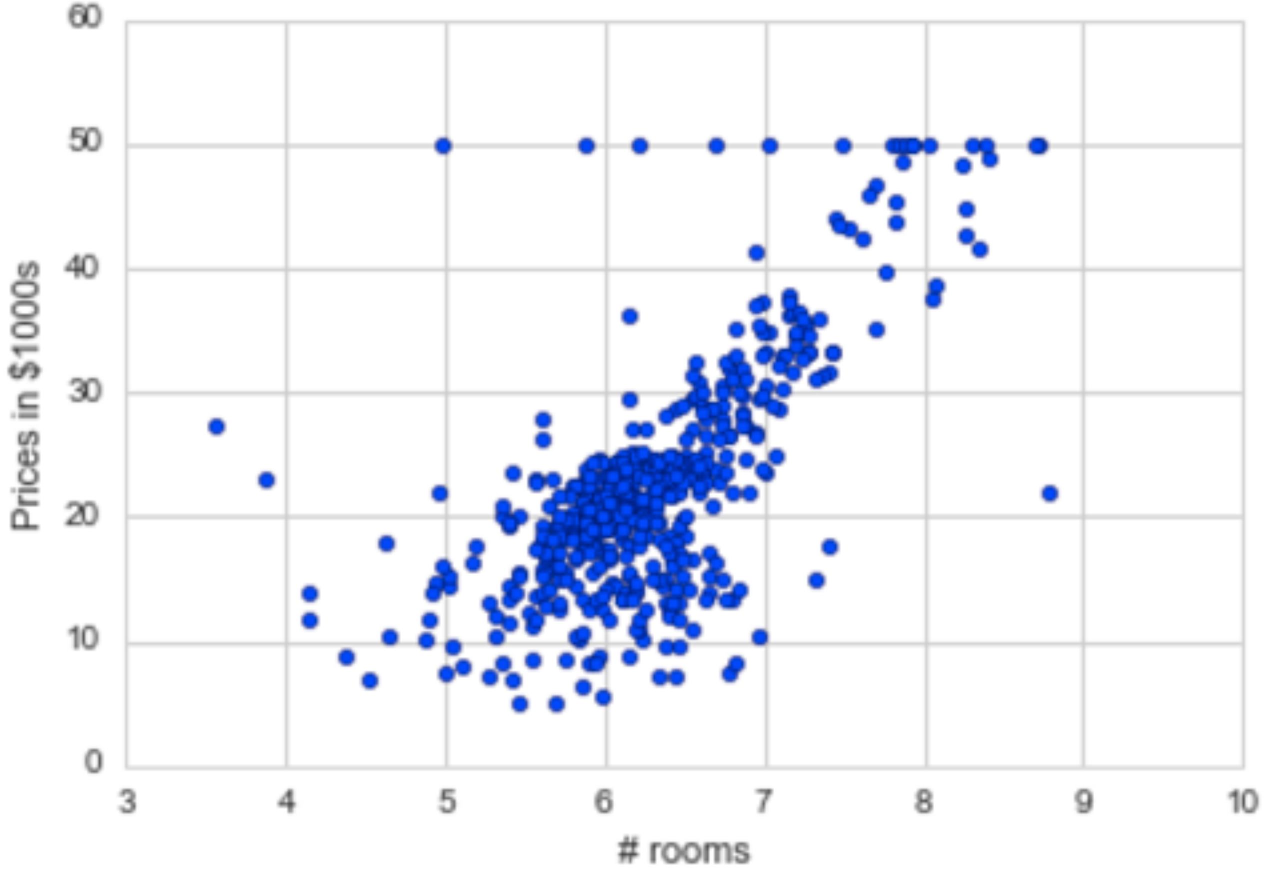
Visualizing Boston House Prices

```
plt.hist(boston.target, bins=50)
plt.xlabel("Prices in $1000s")
plt.ylabel("Number of Houses")
```



Visualizing Influence of # Rooms

```
# the 5th column in "boston" dataset is "RM" (# rooms)
plt.scatter(boston.data[:,5], boston.target)
plt.ylabel("Prices in $1000s")
plt.xlabel("# rooms")
```



Visualizing Influence of # Rooms with fit

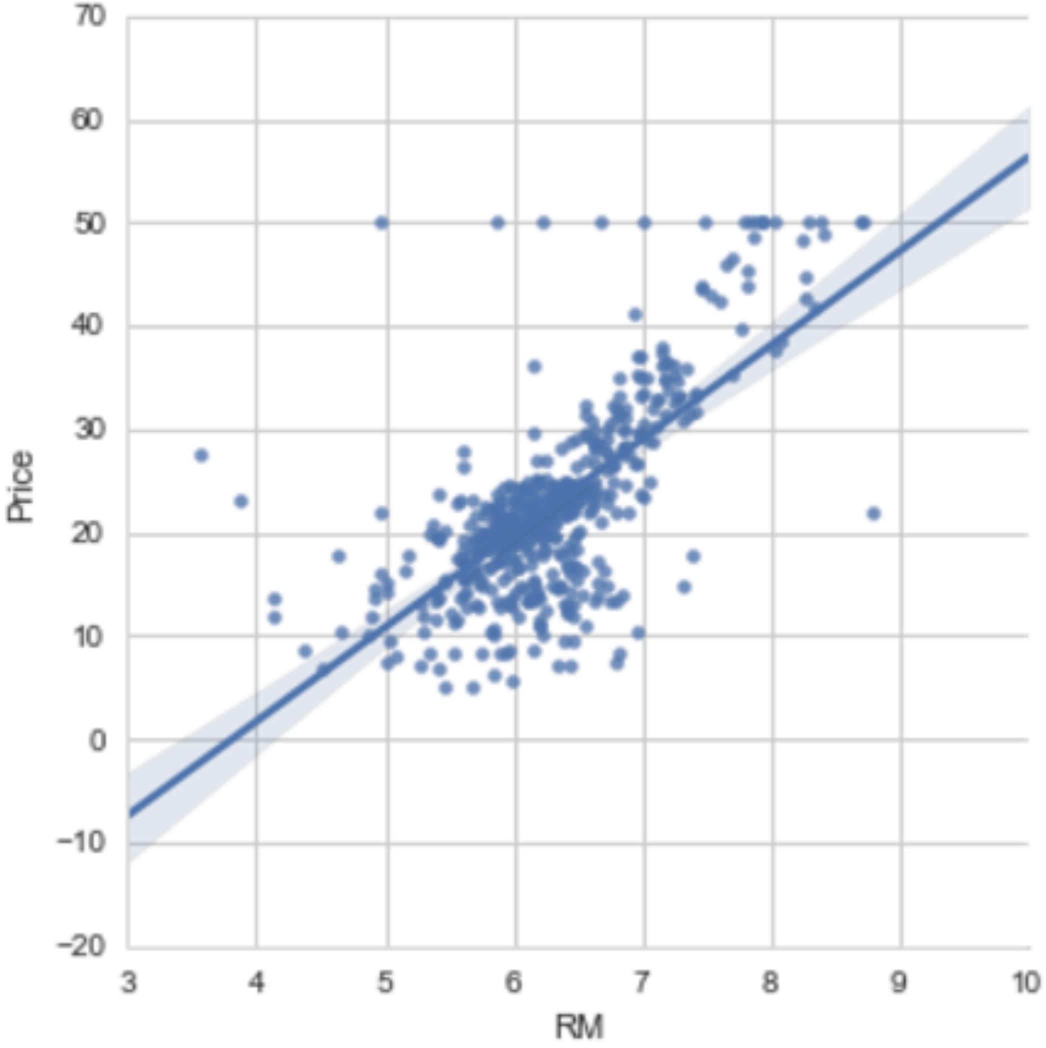
- Start by creating a DataFrame that can be used by Seaborn

```
boston_df = DataFrame(boston.data)
boston_df.columns = boston.feature_names
boston_df['Price'] = boston.target
boston_df.head(5)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

Visualizing Influence of # Rooms with fit

```
sns.lmplot('RM', 'Price', data=boston_df)
```



Ridge regression

Linear prediction: $\widehat{y}_i = \mathbf{w} \cdot \mathbf{x}_i$

Loss function:

$$L = \underbrace{\sum_i \frac{1}{2}(\hat{y}_i - y_i)^2}_{\text{Fit quality}} + \underbrace{\frac{1}{2}\lambda|\mathbf{w}|^2}_{\text{Penalty}}$$

Both the fit quality and the penalty can be changed.

Changing the penalty

- $|\mathbf{w}| = \sqrt{\sum_i w_i^2}$ is called the "L_2 norm"
- $|\mathbf{w}|_1 \ = \sum_i |w_i|$ is called the "L_1 norm"
- In general $|\mathbf{w}|_p \ = \sqrt[p]{\sum_i |w_i|^p}$ is called the "L_p norm"

The LASSO

Loss function:

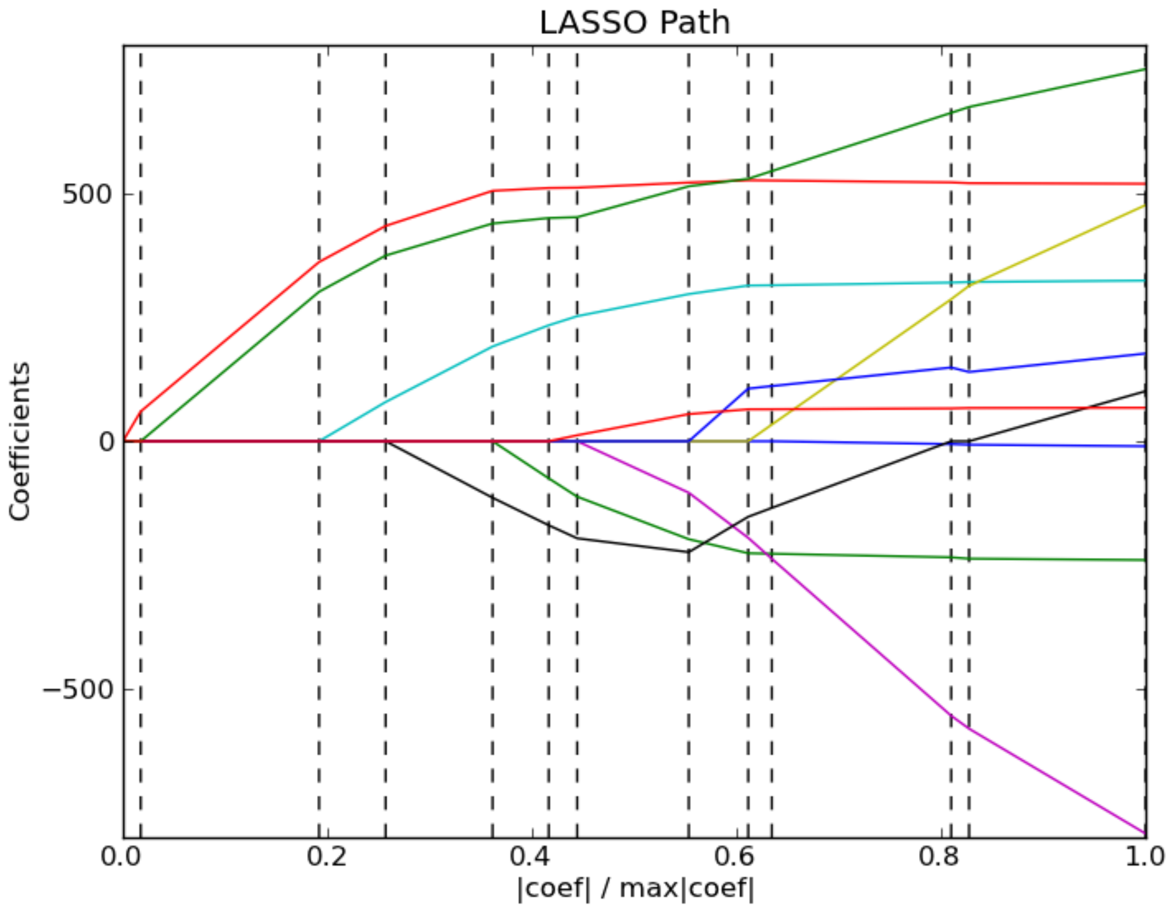
$$L = \sum_i \frac{1}{2}(\hat{y}_i - y_i)^2 \qquad + \qquad \frac{1}{2}\lambda|\mathbf{w}|_1$$

Fit quality

Penalty

LASSO regularization path

- Most weights are exactly zero
- "sparse solution", selects a small number of explanatory variables
- This can help avoid **overfitting** when p>>N
- Models are easier to interpret – but remember there is no proof of causation.
- Path is piecewise-linear



http://scikit-learn.org/0.11/auto_examples/linear_model/plot_lasso_lars.html

Agenda

- Missing Values
- Introduction to Machine Learning
- Scikit-Learn
- Regression
- **Classification**

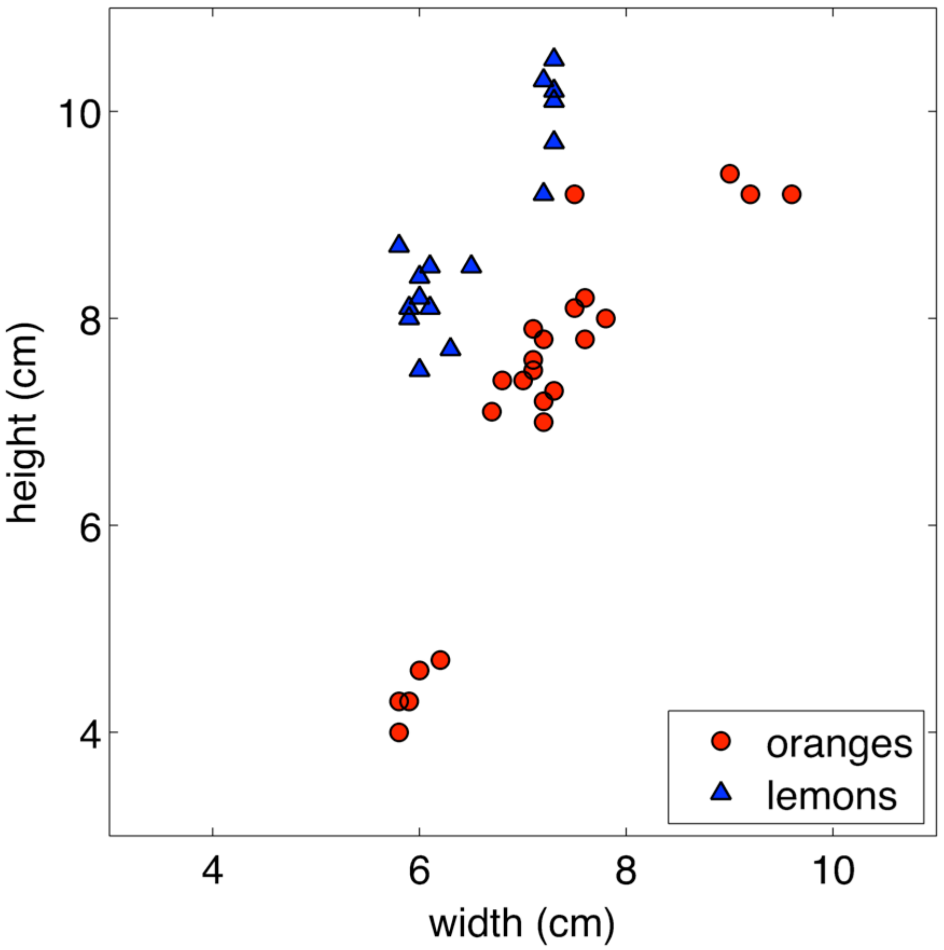
Supervised Learning: Classification

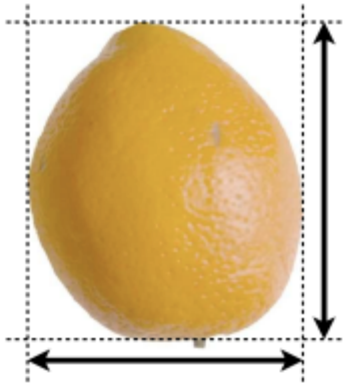
- Classification: assign discrete labels to input data
- **Example**: Medical image diagnosis
  - E = CAT medical scans with labels (“targets”) on (1) tumor, or (2) no tumor
  - T = predict tumor or not tumor for new images
  - P = how well the estimator can accurately predict tumor or not

Supervised Learning: Use Cases

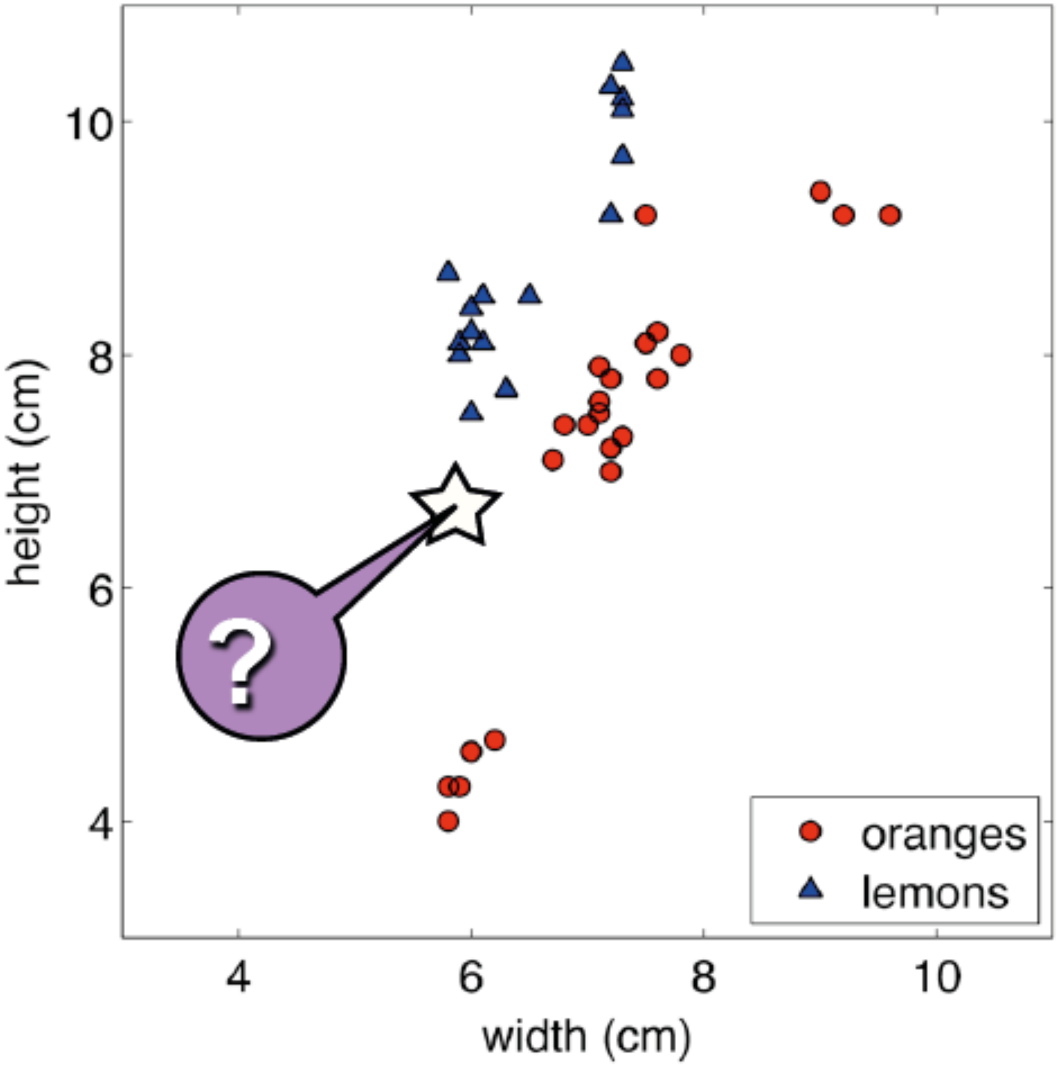
- Prediction of future cases: Use the rule to predict the output for future inputs
- Knowledge extraction: The rule is easy to understand
- Compression: The rule is simpler than the data it explains
- Outlier detection: Exceptions that are not covered by the rule, e.g., fraud

Classification: Oranges and Lemons





Classification: Oranges and Lemons



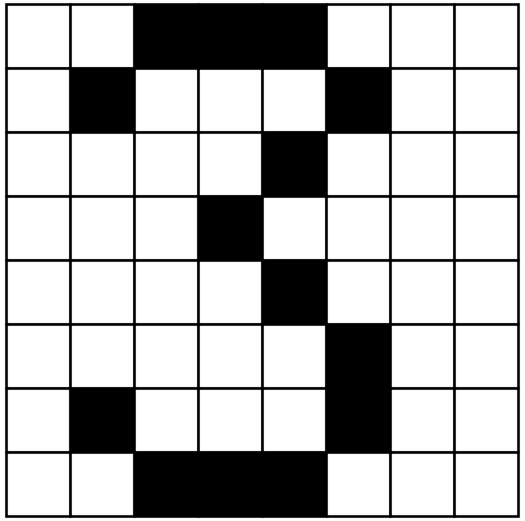
Classification problem

- Given: Training set
  - labeled set of N input-output pairs
    - $D = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
    - $y = \{1, \ldots, K\}$
- Goal: Given an input $\boldsymbol{x}$, assign it to one of K classes

- Examples:
  - Spam filter
  - Handwritten digit recognition

Digit Recognition

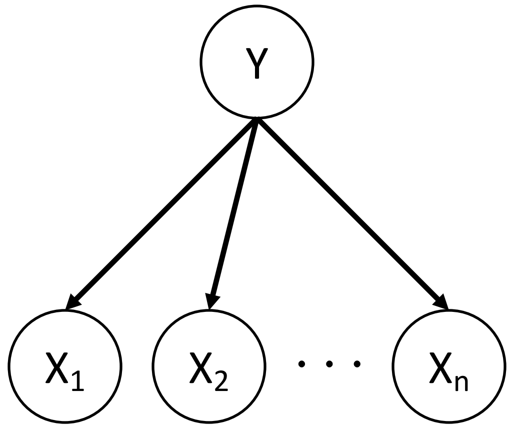
- Input: pixel grids



- Output: a digit 0-9

Naïve Bayes Classifier

- Given:
  - Prior P(Y)
  - *n* conditionally independent features **X** given the class Y
  - For each X_i, we have likelihood $P(X_i|Y)$



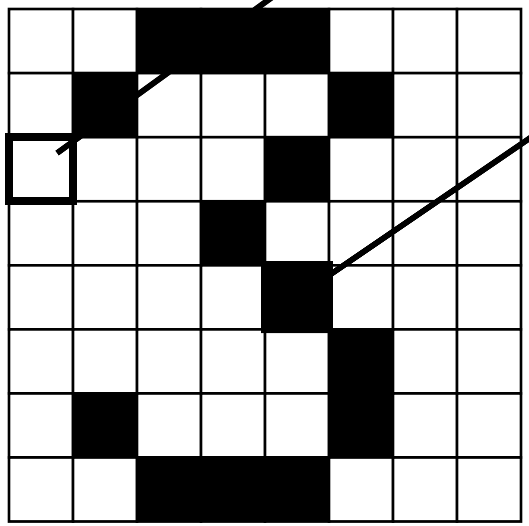
- Decision rule:

$$\begin{aligned} y^* = h_{NB}(\mathbf{x}) &= \arg\max_y P(y)P(x_1,\ldots,x_n \mid y) \\ &= \arg\max_y P(y)\prod_i P(x_i|y) \end{aligned}$$

Example Distribution

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|Y)$

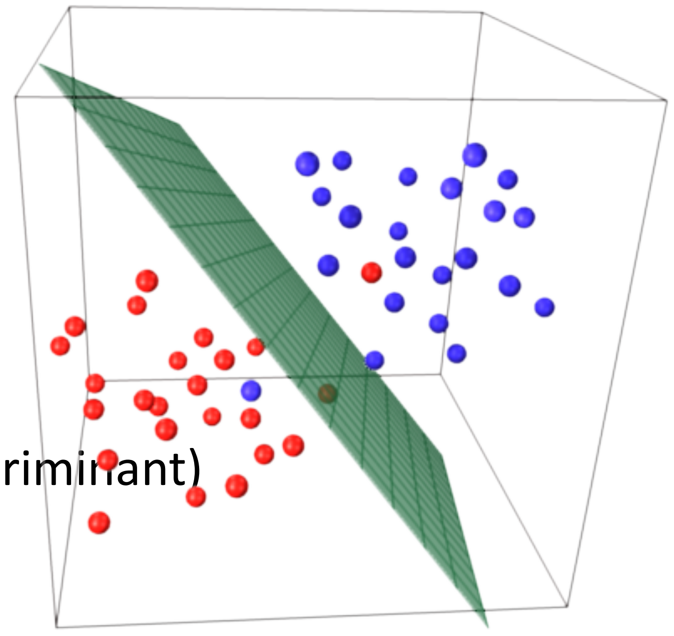
1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

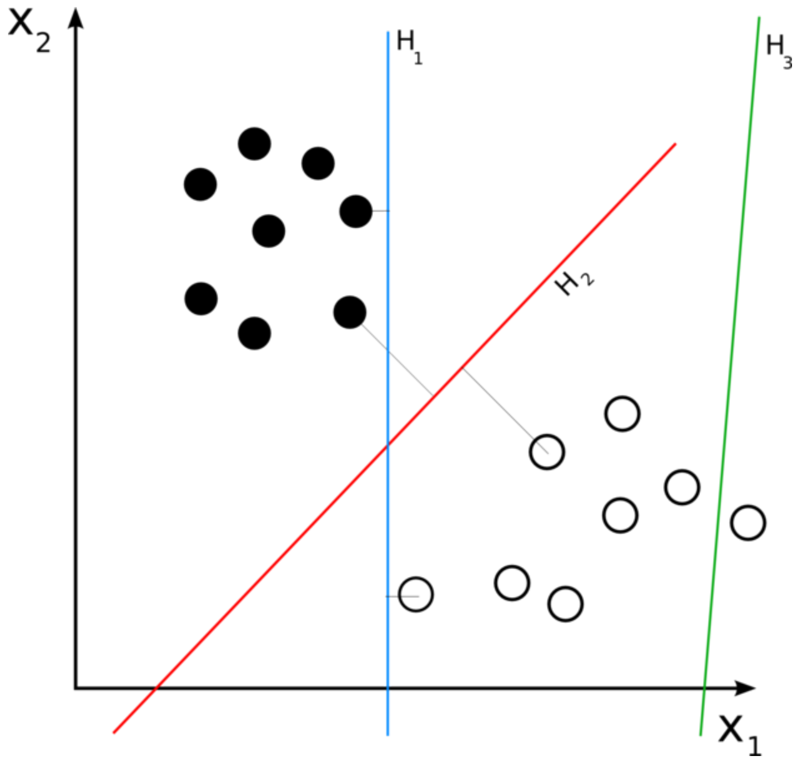
Linear classifiers

- Linear classifiers:
  - Decision boundaries are linear functions
    - $d-1$ dimensional hyper-plane within the d dimensional input space.
  - Examples
    - Perceptron
    - Support vector machine
    - Decision Tree
    - KNN
    - Naive Bayes classifier
    - Linear Discriminant Analysis (or Fisher's linear discriminant)



Linear classifiers

- Linearly separable
  - Data points can be exactly classified by a linear decision surface.
- Binary classification
  - Target variable
    - $y \in \{0,1\}$
    - $y \in \{-1,1\}$



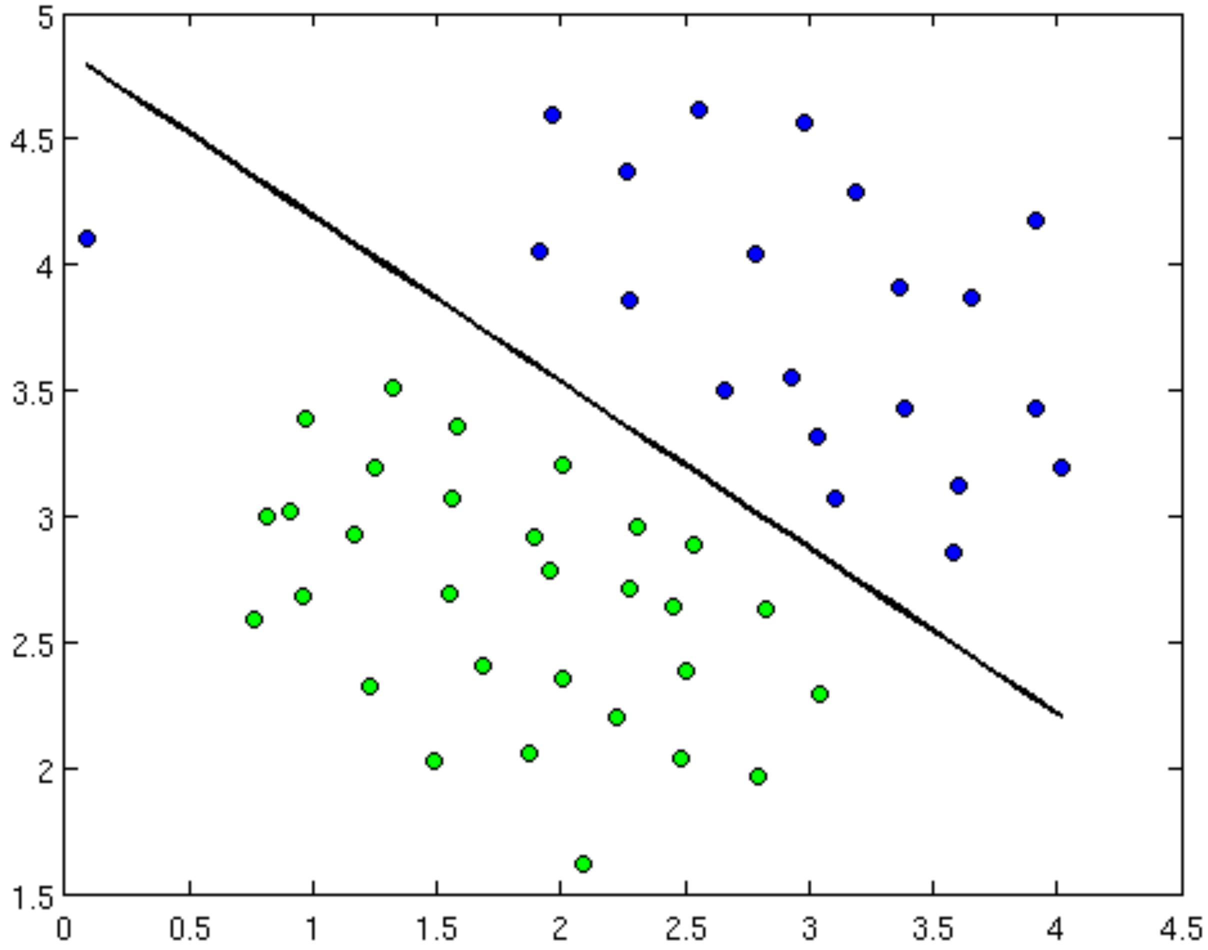
Decision boundary

- Discriminant function : $f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{x}$
  - $\boldsymbol{x} = [1\ x_1\ x_2\ \ldots x_d]$
  - $\boldsymbol{w} = [w_0\ w_1\ w_2\ \ldots w_d]$
  - w_0: bias
- if $f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{x} \geq 0$ then $C1$

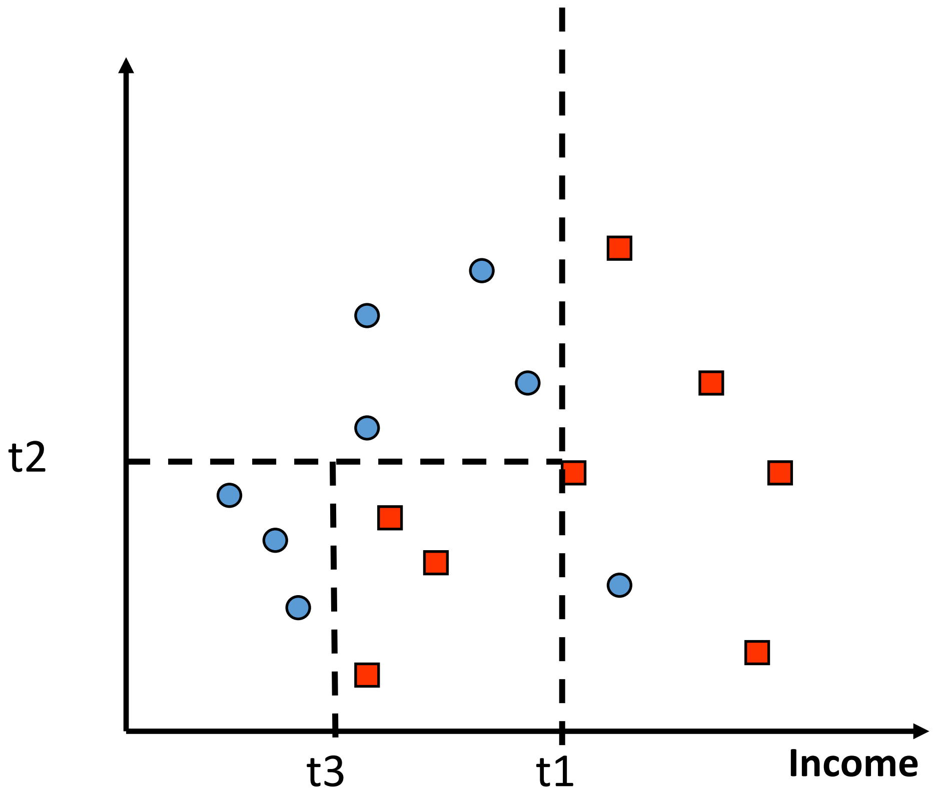
  else $C2$

- Decision boundary: $f(\boldsymbol{x};\boldsymbol{w}) = 0$
  - The sign of $f(\boldsymbol{x};\boldsymbol{w})$ predicts binary class labels

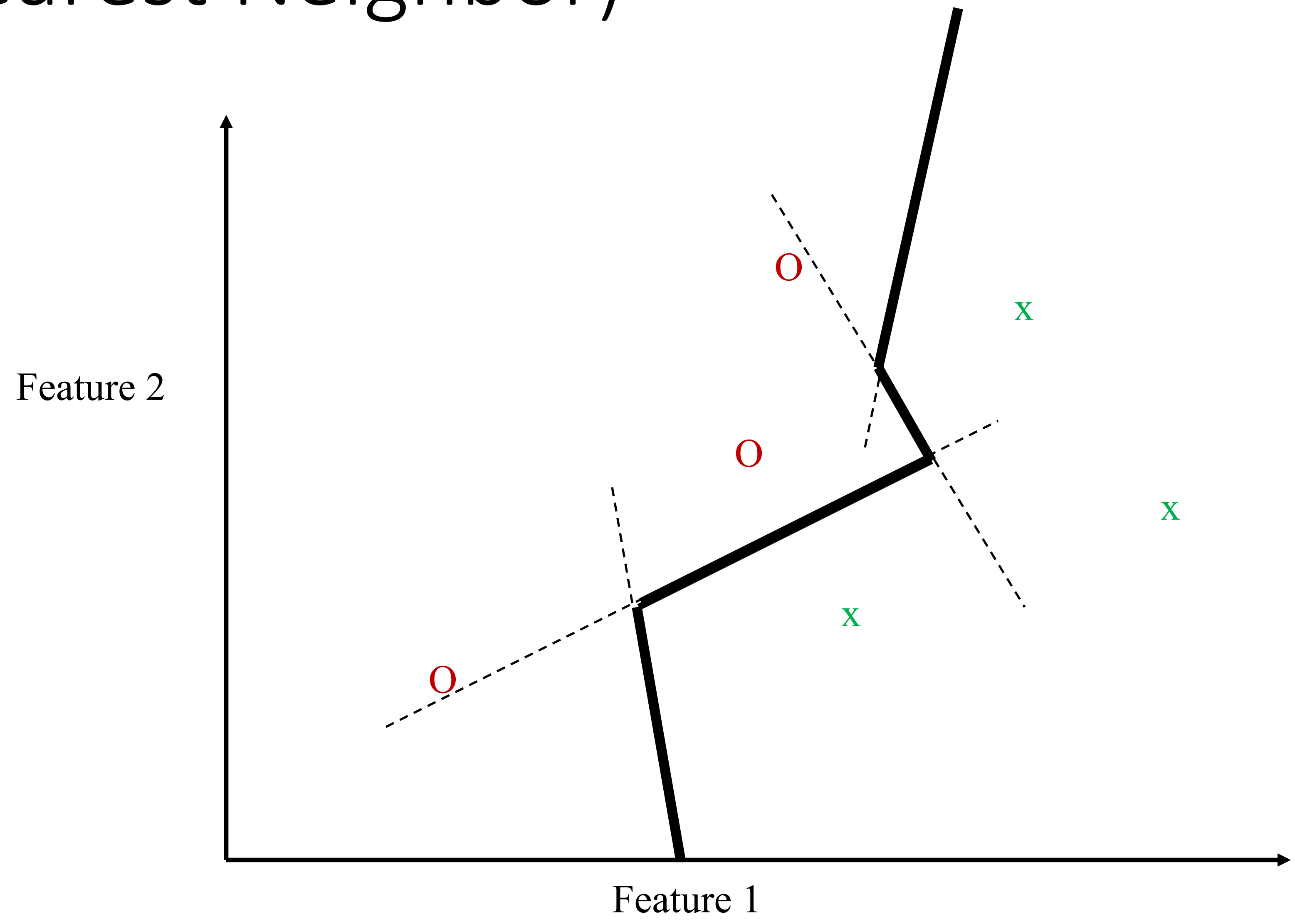
Linear Decision boundary (Perceptron)



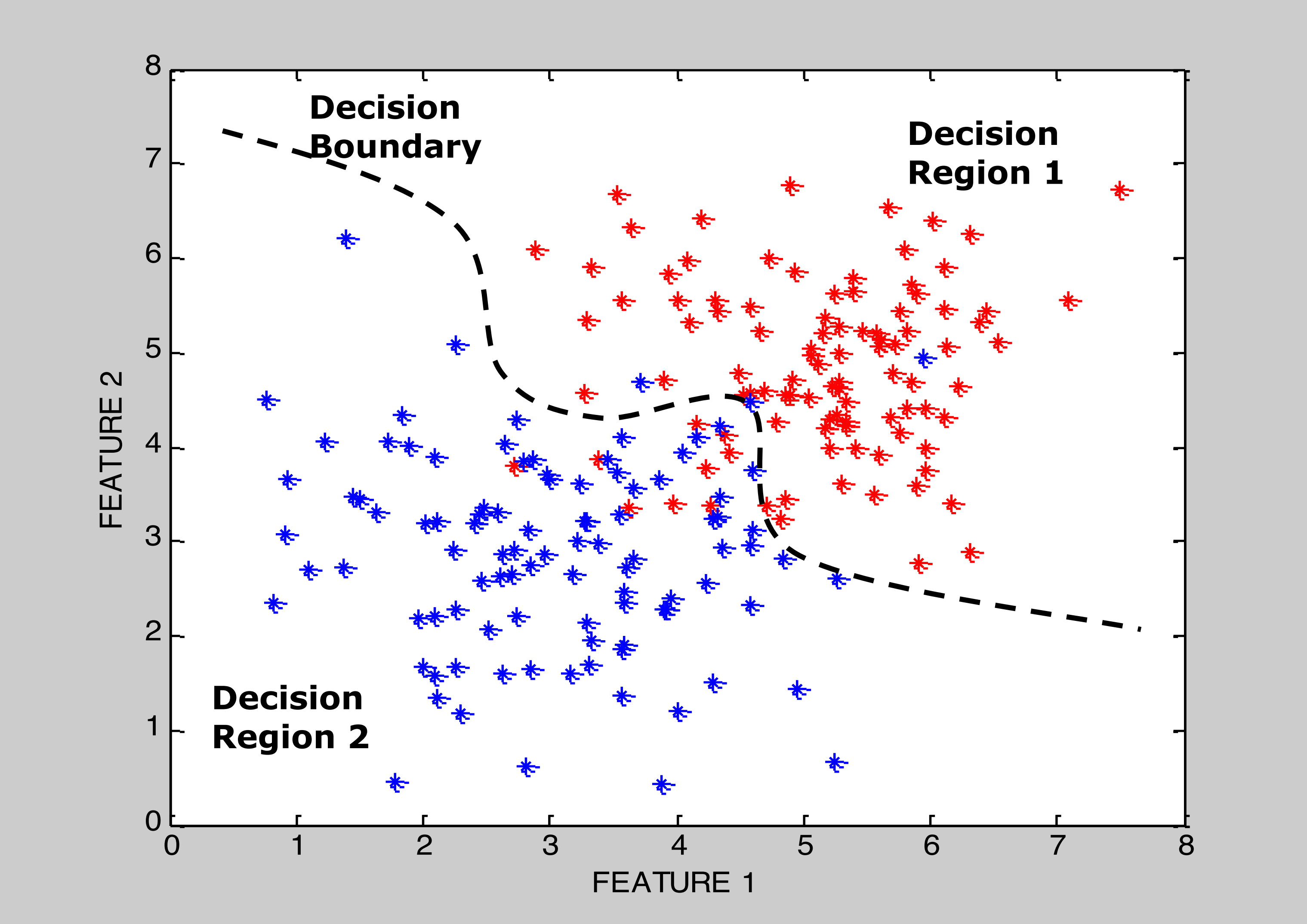
Linear Decision boundary (Decision Tree)



Linear Decision boundary (K Nearest Neighbor)

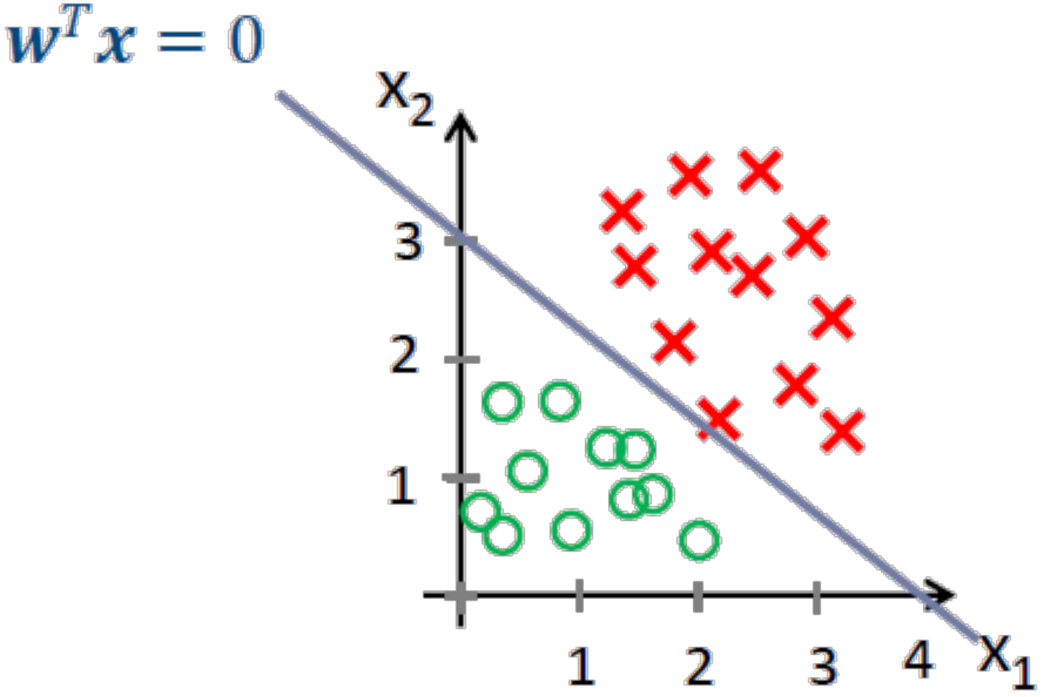


Non-Linear Decision boundary



Decision boundary

- Linear classifier



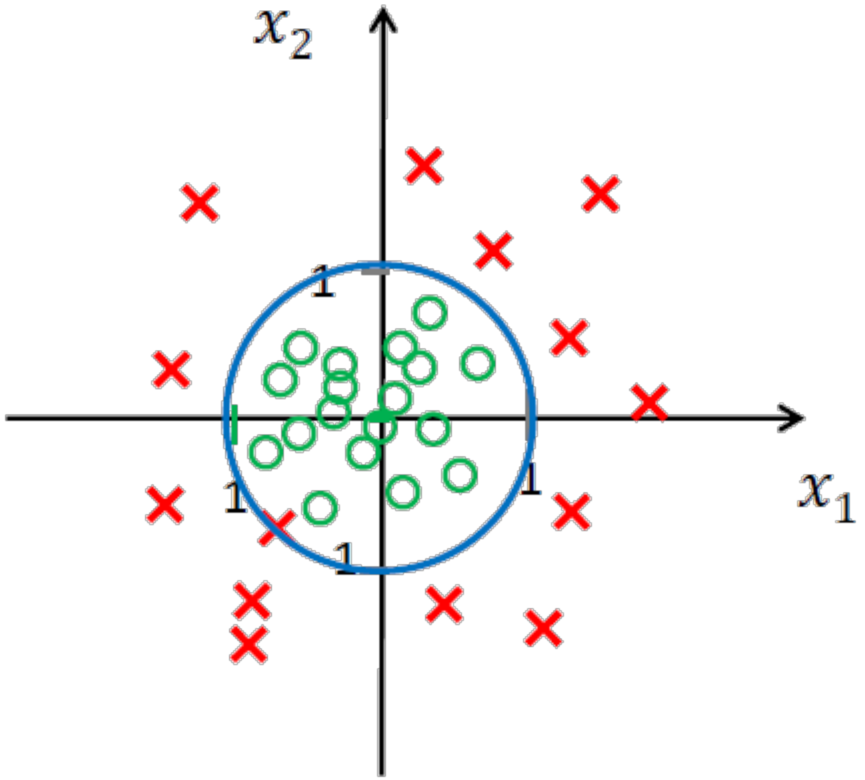
$$3 + \frac{3}{4}x_1 + x_2 = 0$$

if $w^T x \geq 0$ then $y = 1$
else $y = -1$

$$w = [3, 0.75, 1]$$

Non-linear decision boundary

- Choose non-linear features
  - Classifier still linear in parameters $\boldsymbol{w}$



$\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2]$

$$-1 + x_1^2 + x_2^2 = 0$$

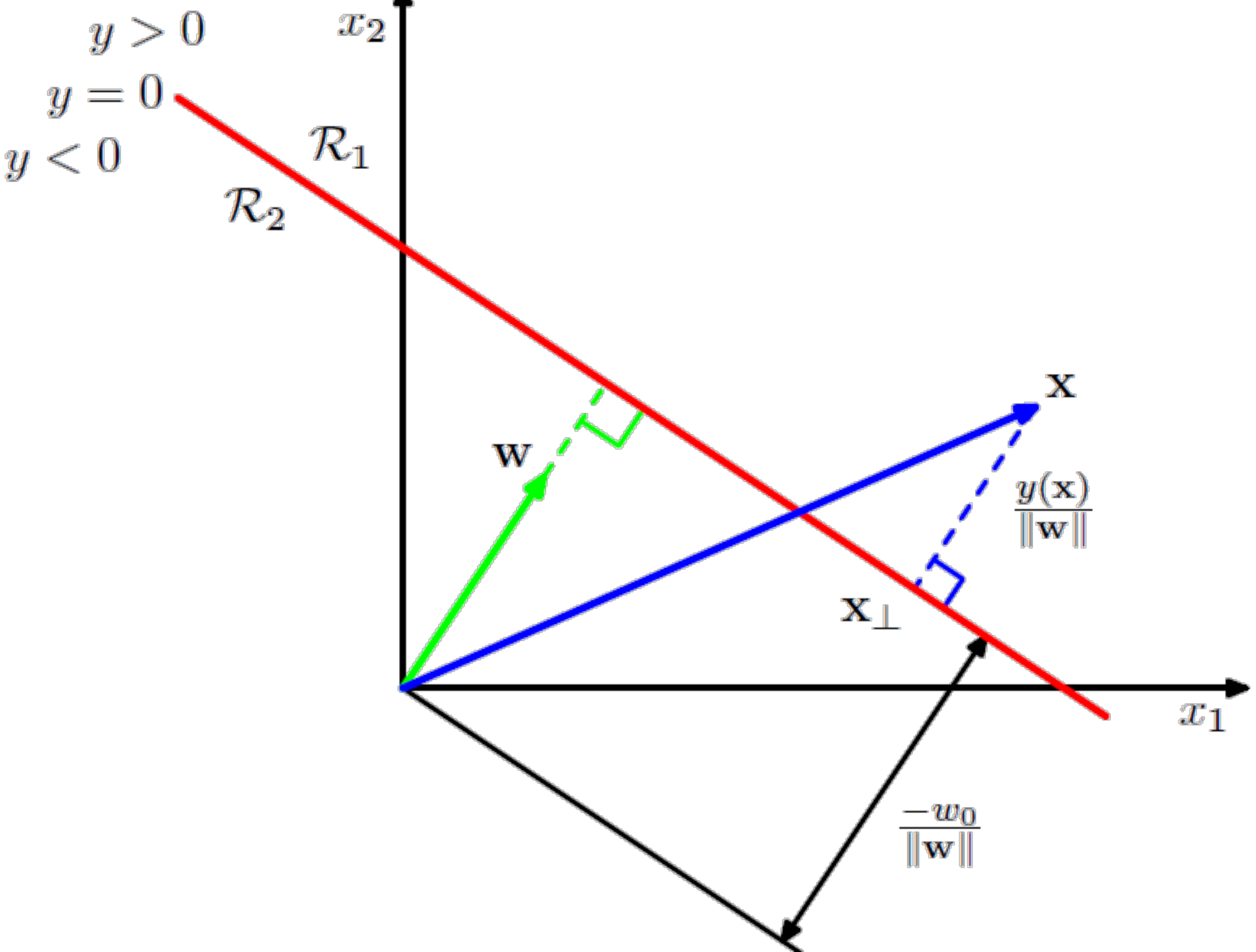
$$\boldsymbol{\phi}(\boldsymbol{x}) = [1, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_1^2, \boldsymbol{x}_2^2, \boldsymbol{x}_1 \boldsymbol{x}_2]$$

$$\boldsymbol{w} = [-1, 0, 0, 1, 1, 0]$$

if $\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) \geq 0$ then $y = 1$
else $y = -1$

Linear boundary: geometry

- In this Slide: $y = \boldsymbol{w}^T \boldsymbol{x} + w_0; \boldsymbol{w} = [\mathrm{w}_1 \ldots \mathrm{w}_\mathrm{d}]$



Unsupervised Learning

- Data has no labels (no “outputs”)
- Clustering: Grouping similar instances
- Goal is to find similarity among the data – to “discover” labels from the data itself
- **Examples**:
  - Customer segmentation: given purchase behaviors and demographics, classify type of consumers for different marketing campaigns.
  - Image compression: Color quantization
  - Bioinformatics: Learning motifs

Reinforcement Learning

- Learning a policy: A sequence of outputs
- No supervised output but delayed reward
- Credit assignment problem
- Game playing
- Robot in a maze
- Multiple agents, partial observability, ...

APPENDIX

Learning Associations

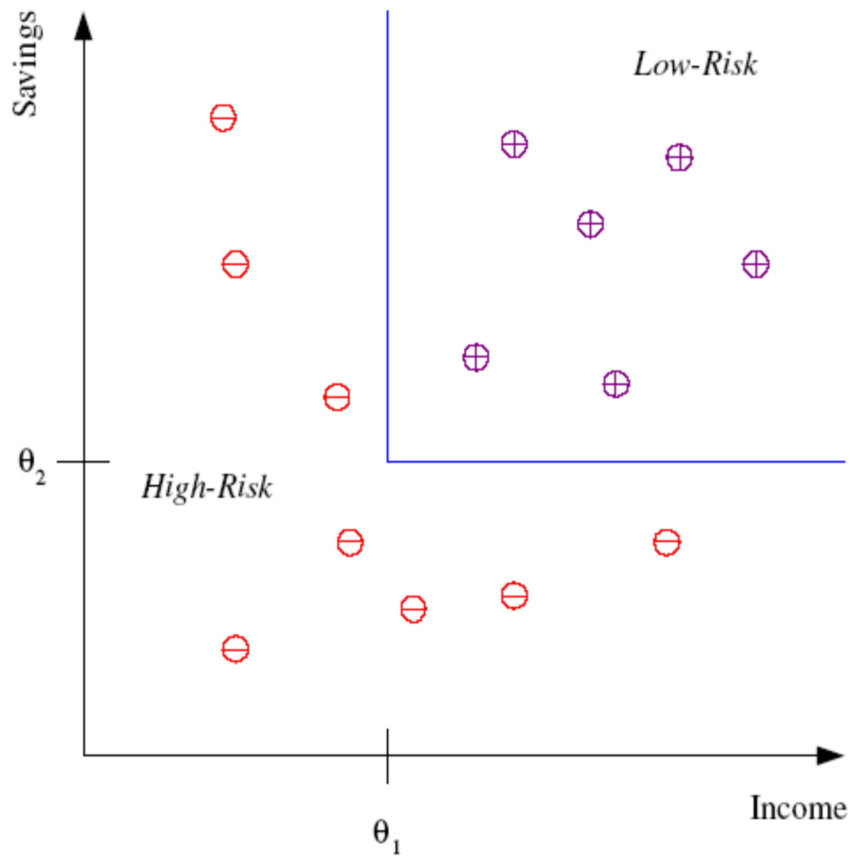
- Basket analysis:

  $P\ (Y \mid X\ )$ probability that somebody who buys X also buys Y where X and Y are products/services.

  Example: P (chips | beer) = 0.7

Classification

- Example: Credit scoring
- Differentiating between low-risk and high-risk customers from their *income* and *savings*



Discriminant: IF *income* > θ_1 AND *savings* > θ_2

THEN low-risk ELSE high-risk

Classification: Applications

- Aka Pattern recognition
- Face recognition: Pose, lighting, occlusion (glasses, beard), make-up, hair style
- Character recognition: Different handwriting styles.
- Speech recognition: Temporal dependency.
- Medical diagnosis: From symptoms to illnesses
- Biometrics: Recognition/authentication using physical and/or behavioral characteristics: Face, iris, signature, etc
- ...

Face Recognition

Training examples of a person

Test images