

Engineering Analytics (ESE1007)

Seminar 6



Machine Learning Linear Regression

Machine Learning

Herbert Alexander Simon:

- “Learning is any process by which a system improves performance from experience.”
- “Machine Learning is concerned with computer programs that automatically improve their performance through experience.”



Herbert Simon
[Turing Award](#) 1975
[Nobel Prize in Economics](#) 1978

Machine Learning Main Tasks

- Develop systems that can automatically adapt and customize themselves to individual users.
 - Personalized news or mail filter
- Discover new knowledge from large databases
 - Market basket analysis (e.g. diapers and beer)

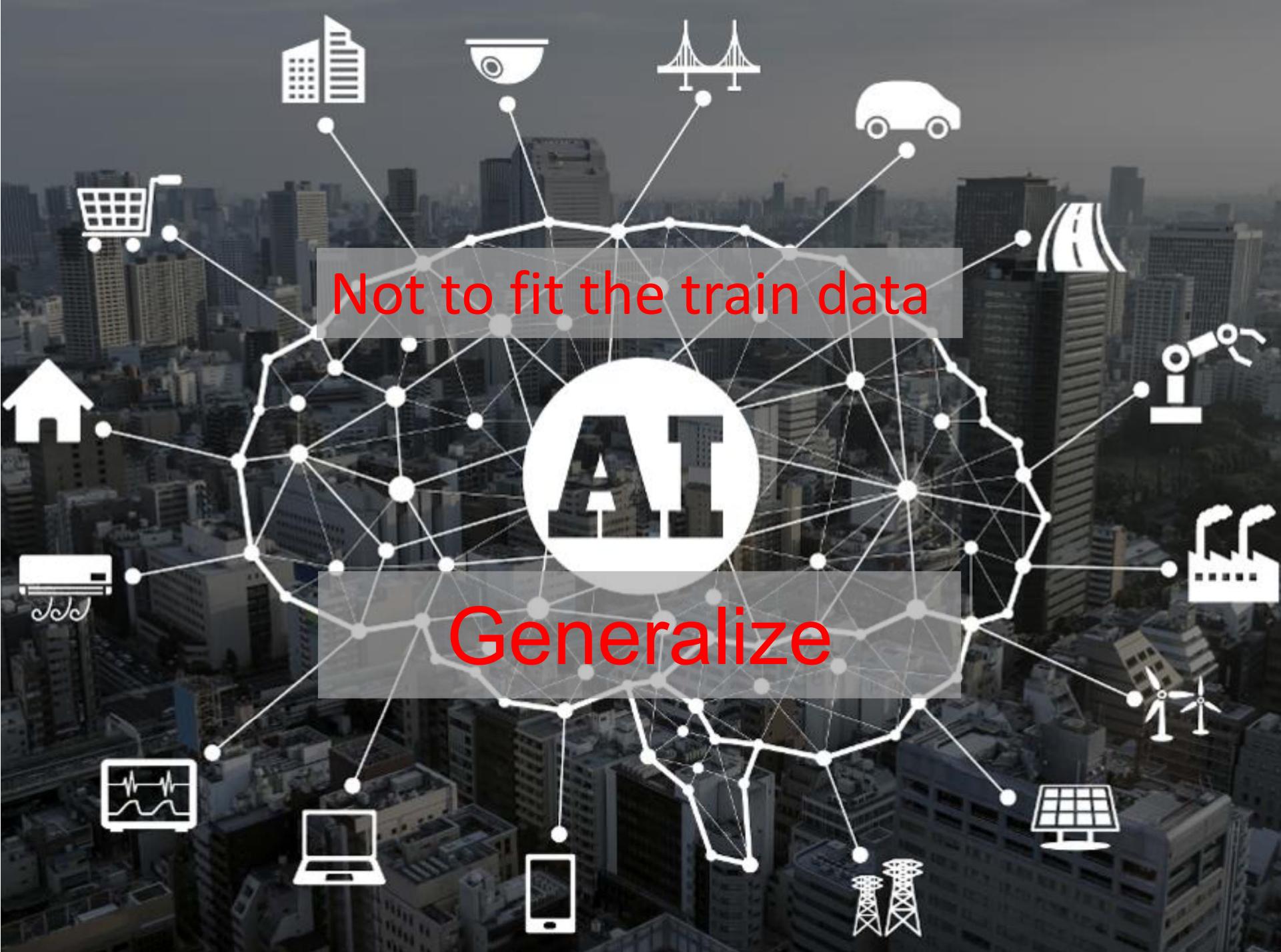
Machine Learning Main Tasks

- Ability to mimic human and replace certain monotonous tasks which require some intelligence.
- like recognizing handwritten characters
- Develop systems that are too difficult/expensive to construct manually because they require specific detailed skills or knowledge tuned to a specific task (knowledge engineering bottleneck).

Human Learning



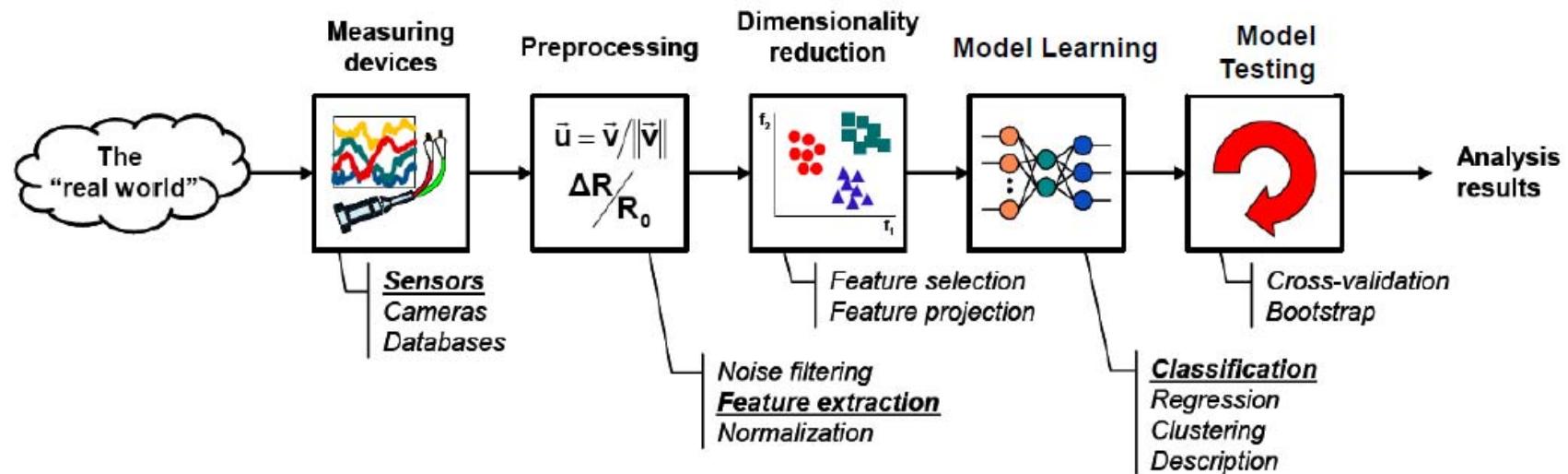
- Show sample of examples for learning
 - Not for the kid to be fixed to the material
 - Able to apply
 - Able to extrapolate
 - Able to adapt to similar cases
 - Able to innovate
-



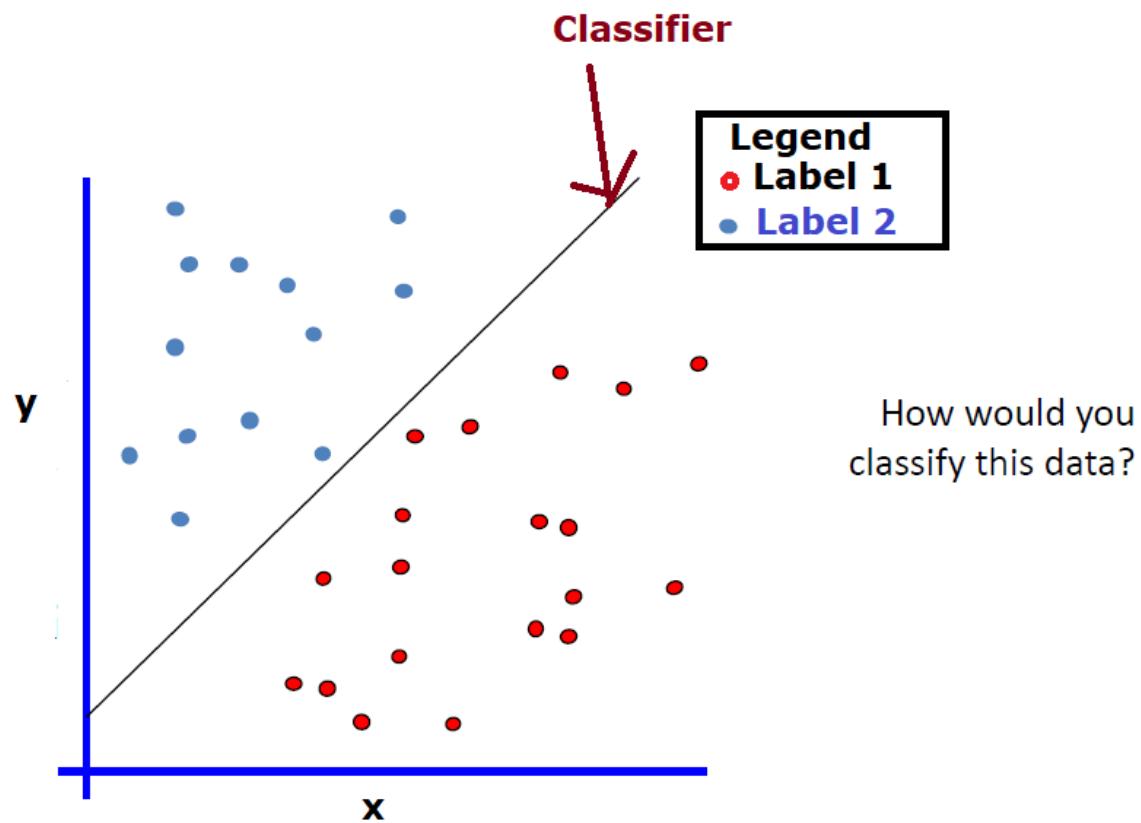
Not to fit the train data

Generalize

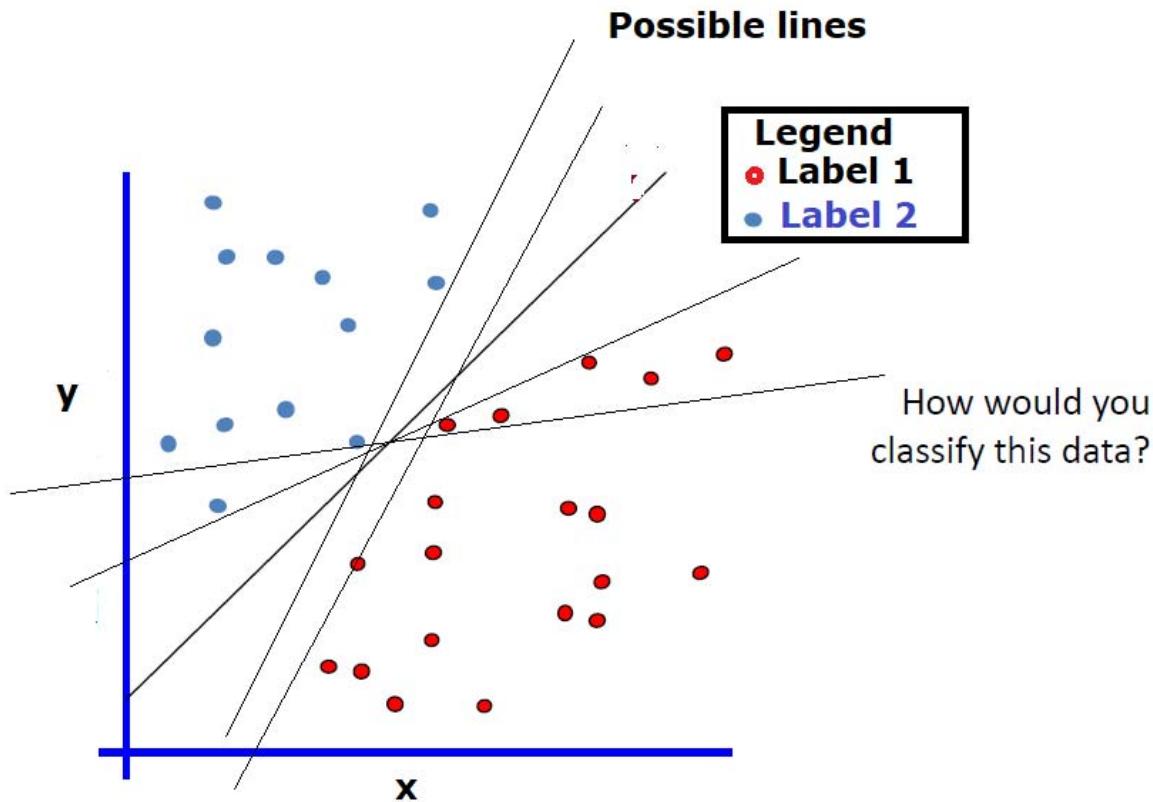
The Learning Process



Linear Classifiers



Linear Regression $Y=ax+b$ where a is the slope and b is the y-intercept

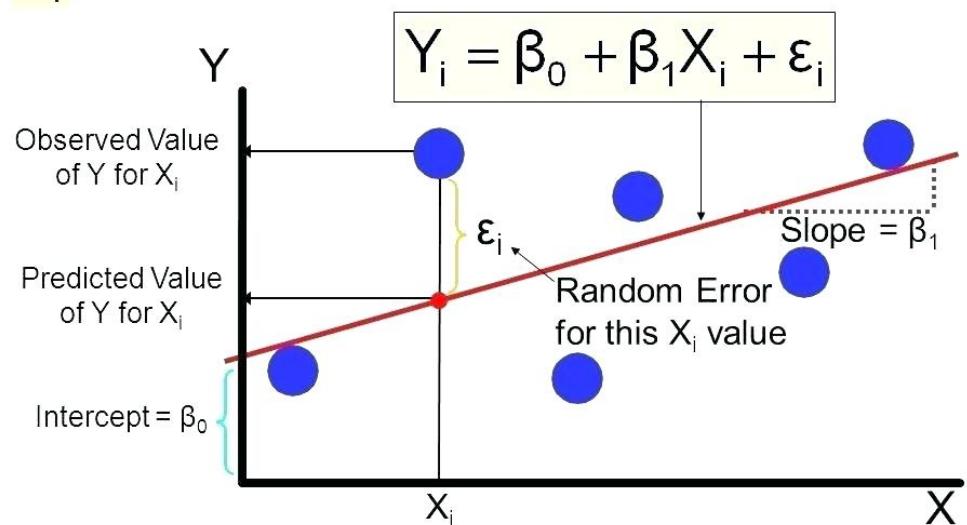


By changing the value of **a** and **b**, we could have many possible lines as classifier, which one could achieve the best performance?

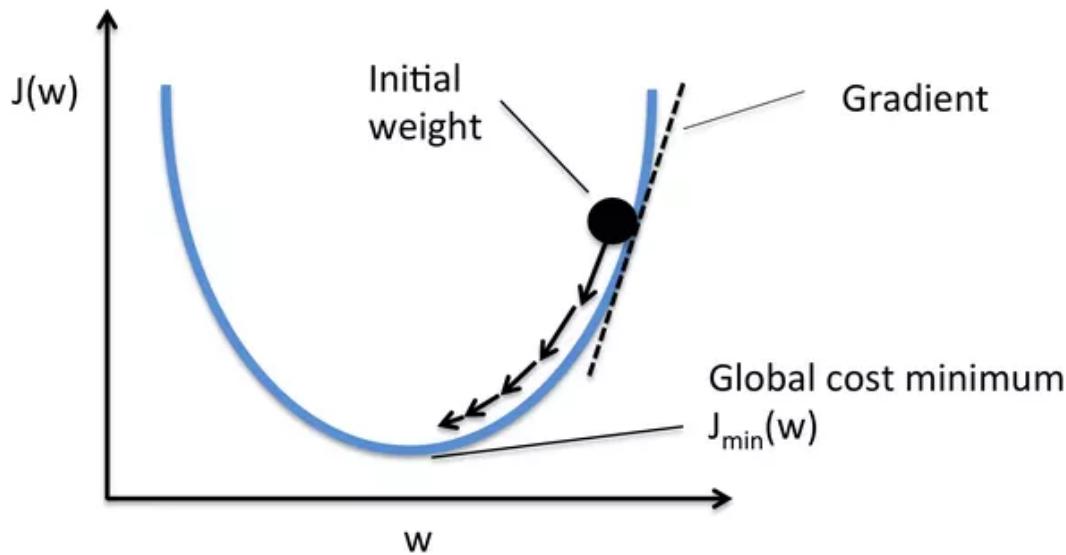
Cost Function

- Linear regression tries to fit points to a line generated by an algorithm. This optimized line (the model) is capable of predicting values for certain input values and can be plotted. The square error is the commonly cost function to optimize and find the best line that have produce the lowest square error.

$$J(w) = \frac{1}{2} \sum_i (Target^i - Actual^i)^2$$



Gradient Descent (GD)



$$J(w) = \frac{1}{2} \sum_i (Target^i - Actual^i)^2$$

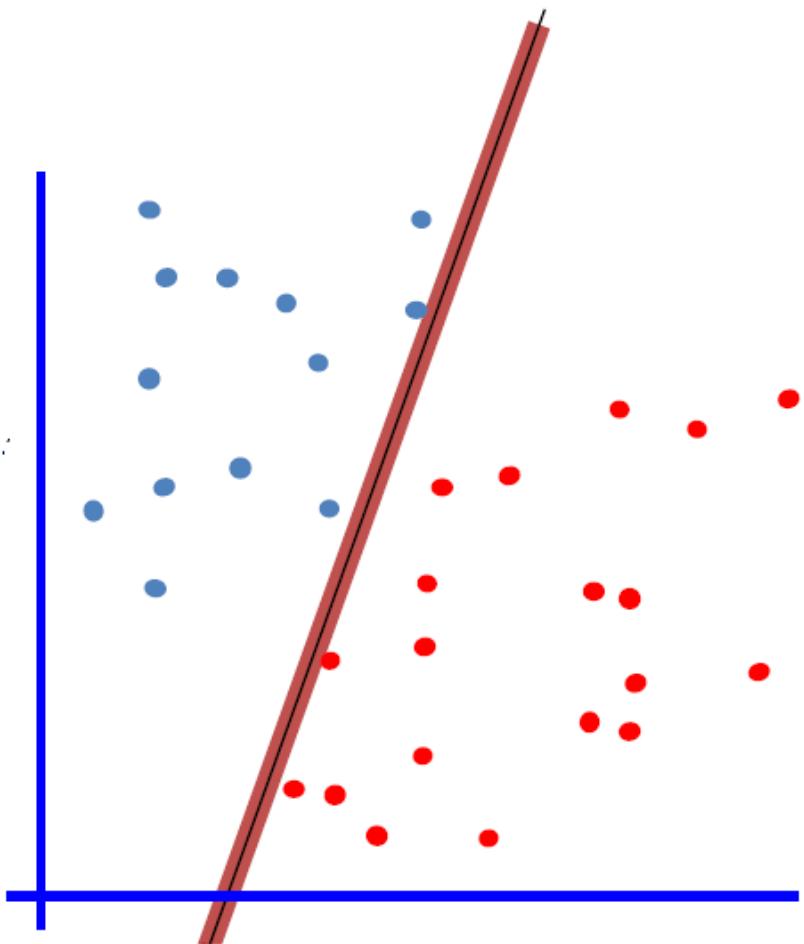
$$\Delta w_j = -\mu \frac{\partial J}{\partial w_j}$$

$$w_j = w_j + \Delta w_j$$

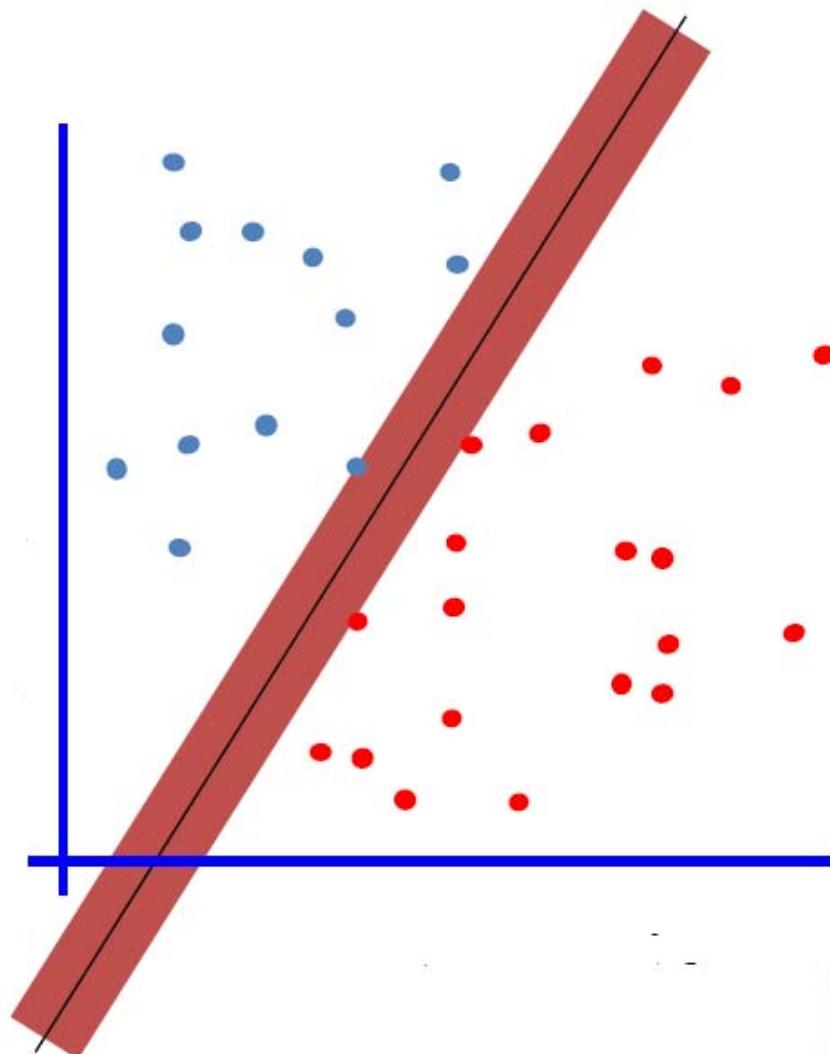
Where μ is the learning rate

Source: <https://www.quora.com/Whats-the-difference-between-gradient-descent-and-stochastic-gradient-descent>

- Gradient Descent is an effective yet simple optimization method commonly adapted in machine learning.

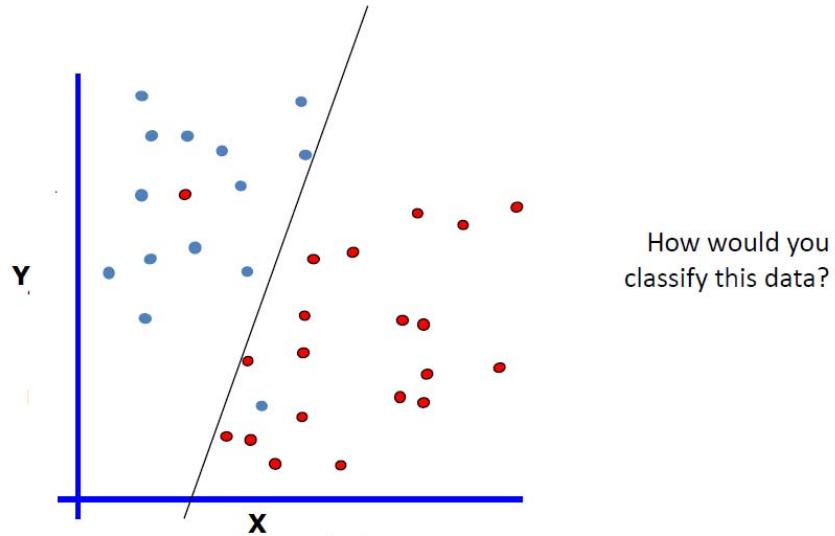


Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.



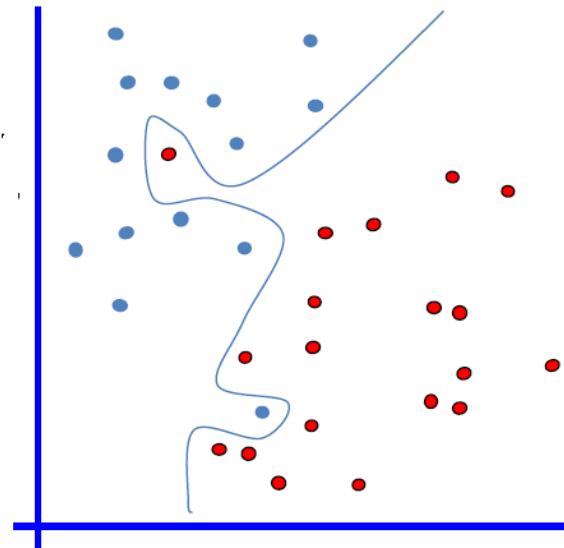
The maximum margin linear classifier is the linear classifier with the maximum margin. This is the simplest kind of SVM (Called an LSVM)

Linear SVM



No linear
classifier can
cover all
instances.

Non-Linear
classifier is
required

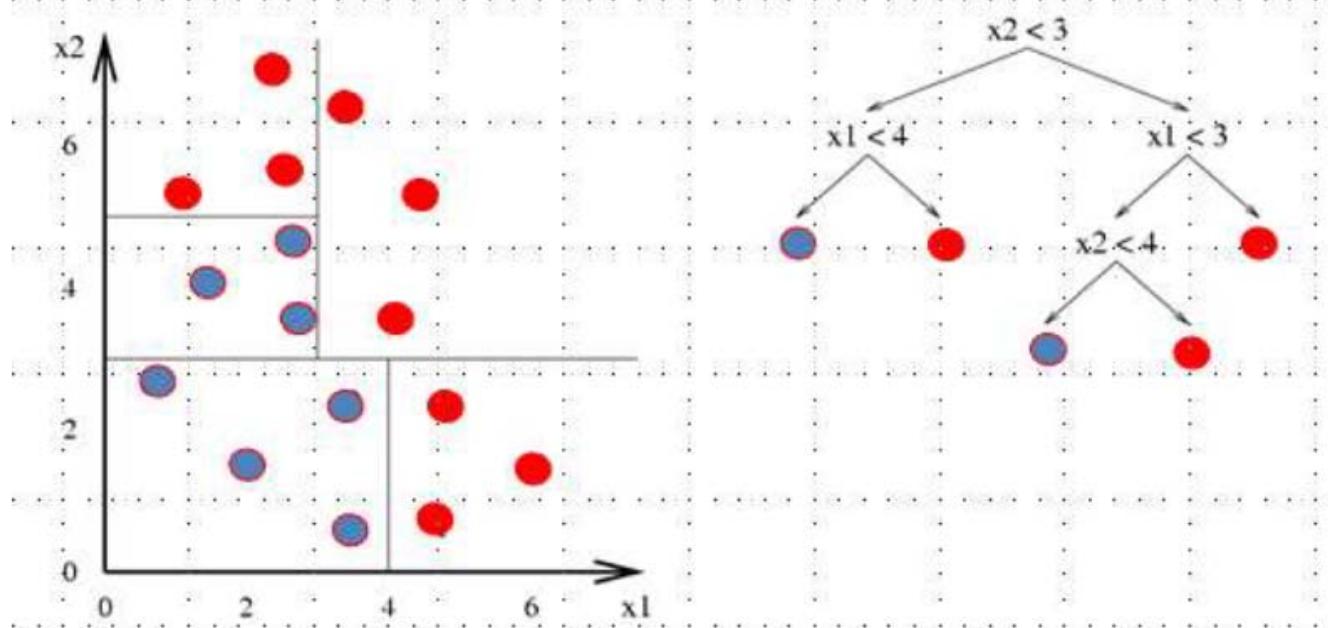


Linear Regression



<https://www.youtube.com/watch?v=CtKeHnfK5uA>

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.



- A flow-chart like tree structure
 - Internal node denotes a test on an attribute
 - Branch represents an outcome of the test
 - Leaf nodes represents an outcome of the test
 - Leaf nodes represent class labels or class distribution
-

Data Split

Idea #1: Choose hyperparameters that work best on the data

BAD: always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

train

test

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

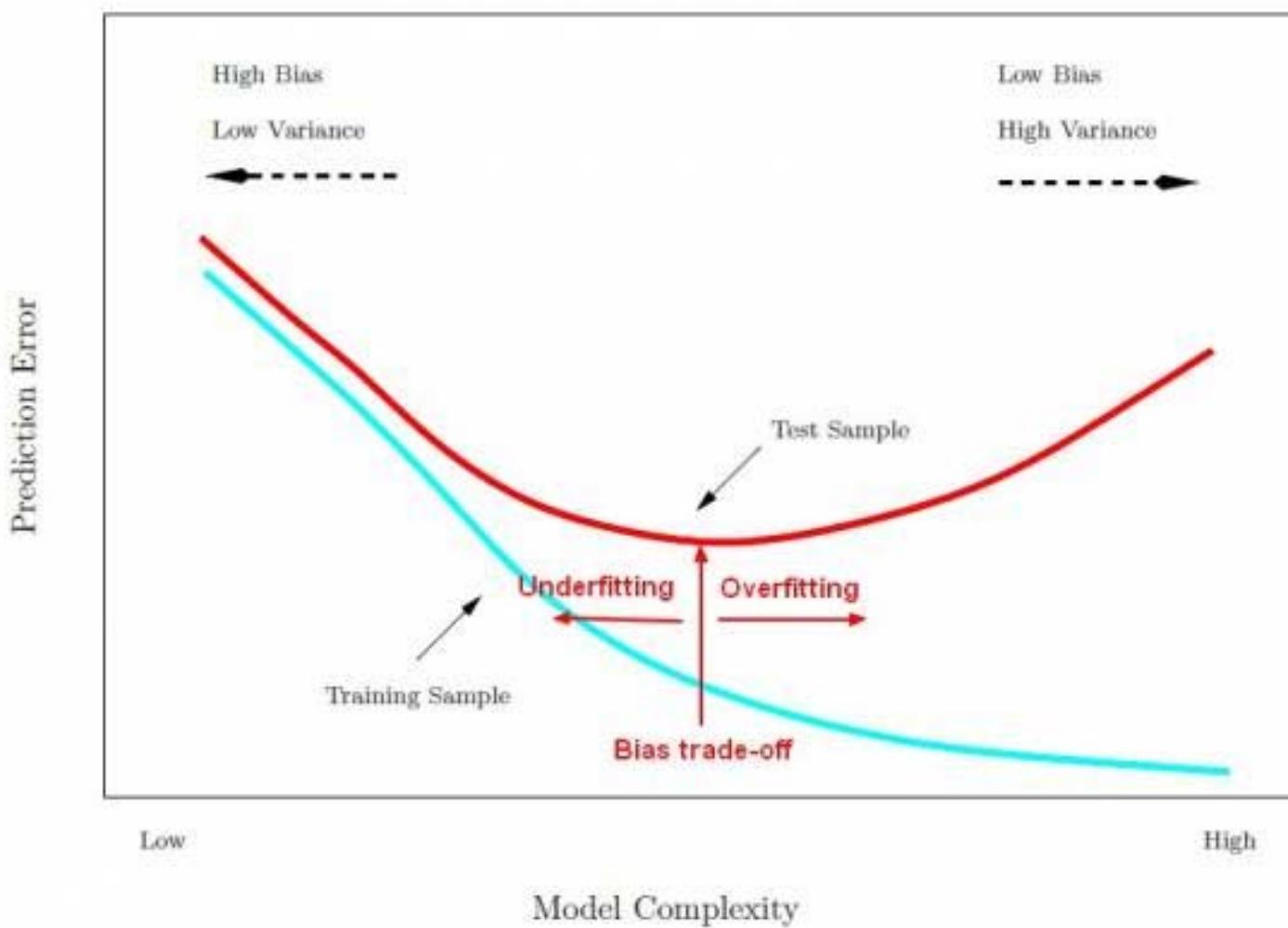
Better!

train

validation

test

Under fitting and Overfitting



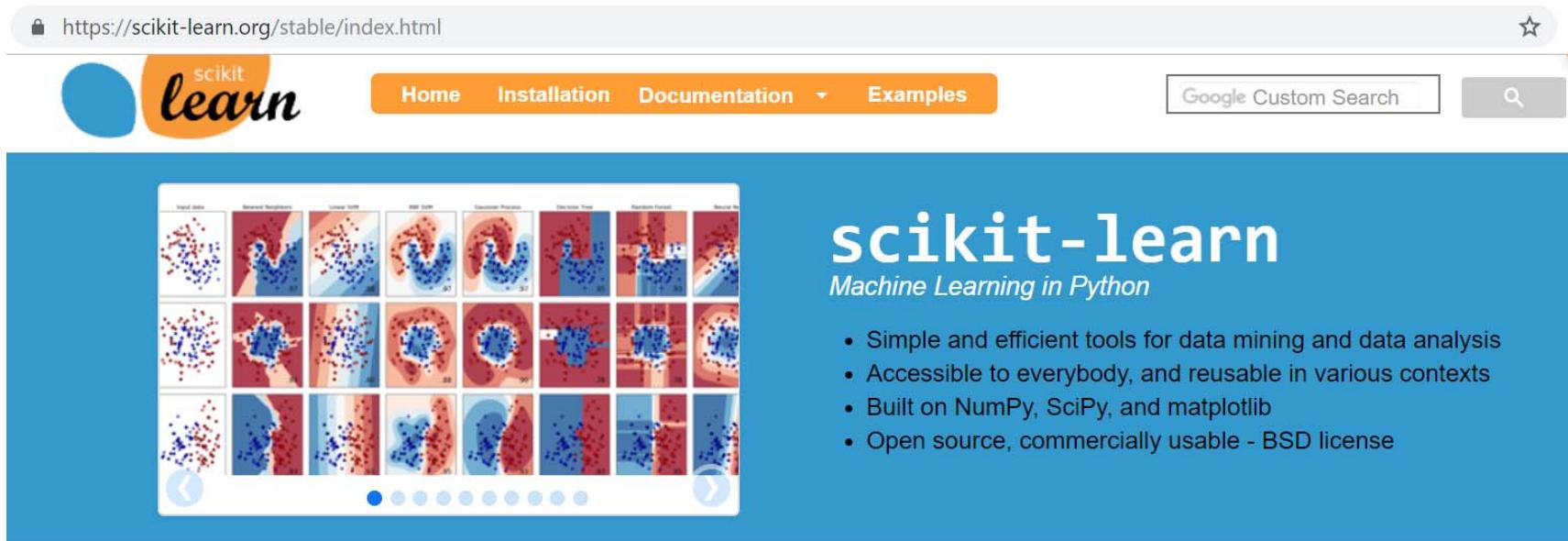
Type I & II Errors

Figure 1

		Reality	
		H_0 Is True	H_1 Is True
Conclusion	Do Not Reject H_0	Correct Conclusion	Type II Error
	Reject H_0	Type I Error	Correct Conclusion

scikit-learn

Machine Learning in Python



<https://scikit-learn.org/stable/index.html>

scikit-learn commonly used steps

Commonly steps in using the Scikit-Learn estimator API:

- Choose a class of model by importing the appropriate estimator class from Scikit-Learn
- Choose model hyperparameters by instantiating this class with desired values
- Arrange data into a feature matrix and target vector
- Fit the model to data by call the fit() method of the model instance.
- Apply the model to new data:
 - For supervised learning, we predict labels for unknown data using the predict() methhod
 - For unsupervised learning, we often transform or infer properties of the data using the transform or predict() methods

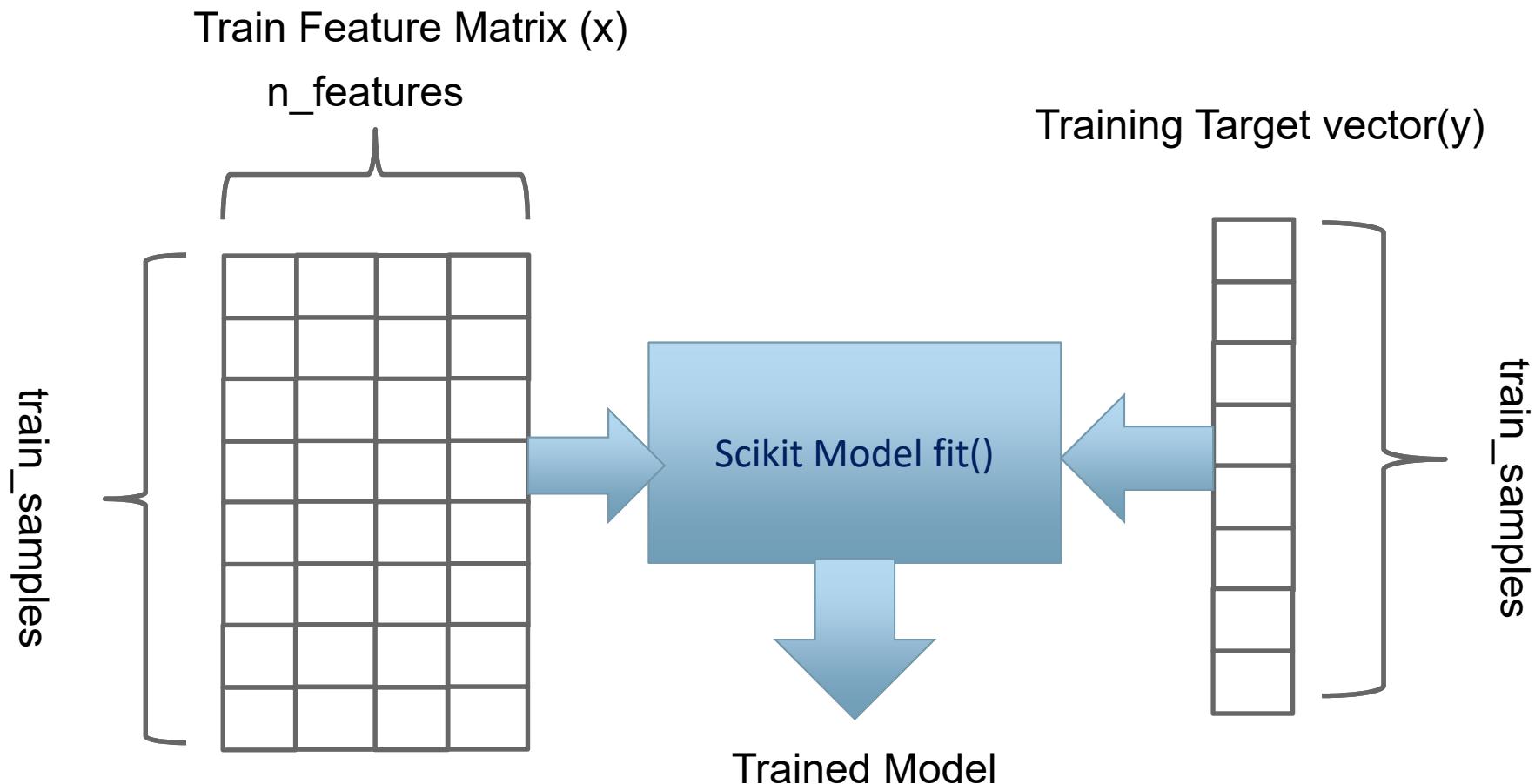
Choosing a class of model

The screenshot shows the scikit-learn documentation page for supervised learning. The URL in the address bar is https://scikit-learn.org/stable/supervised_learning.html. The page features the scikit-learn logo at the top left. A navigation bar with links for Home, Installation, Documentation (with a dropdown menu), Examples, and a Google Custom Search bar. On the left, there's a sidebar with links for Previous User Guide, Next 1.1. Generali..., Up User Guide, and a version dropdown showing 'scikit-learn v0.20.0' with 'Other versions' below it. A note encourages users to cite the software. The main content area has a light blue header '1. Supervised learning'. Below it, '1.1. Generalized Linear Models' is the current section. A detailed list of sub-topics follows:

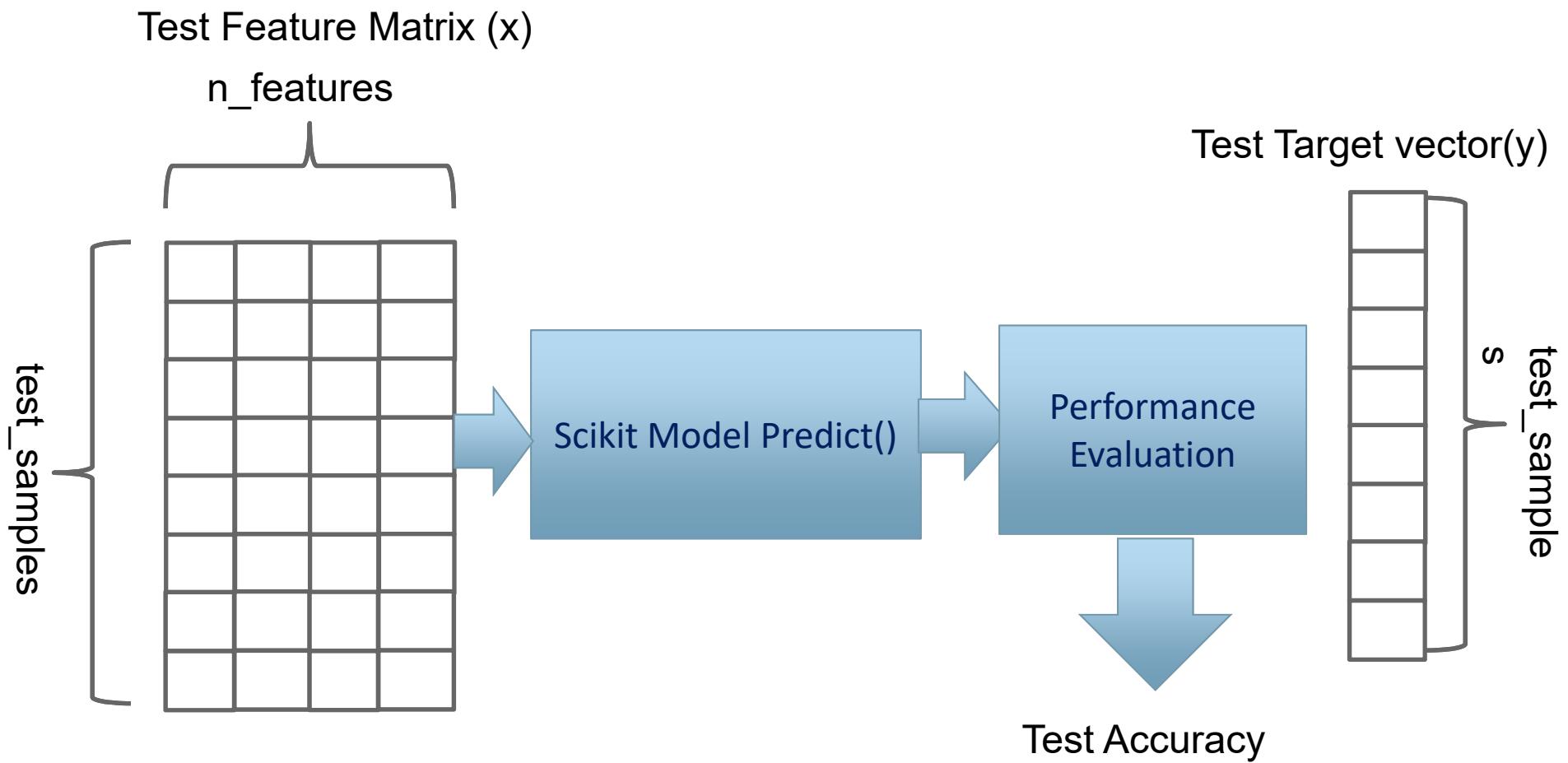
- 1.1.1. Ordinary Least Squares
 - 1.1.1.1. Ordinary Least Squares Complexity
- 1.1.2. Ridge Regression
 - 1.1.2.1. Ridge Complexity
 - 1.1.2.2. Setting the regularization parameter: generalized Cross-Validation
- 1.1.3. Lasso
 - 1.1.3.1. Setting regularization parameter
 - 1.1.3.1.1. Using cross-validation
 - 1.1.3.1.2. Information-criteria based model selection
 - 1.1.3.1.3. Comparison with the regularization parameter of SVM
- 1.1.4. Multi-task Lasso

https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

SCKIT Model Fit



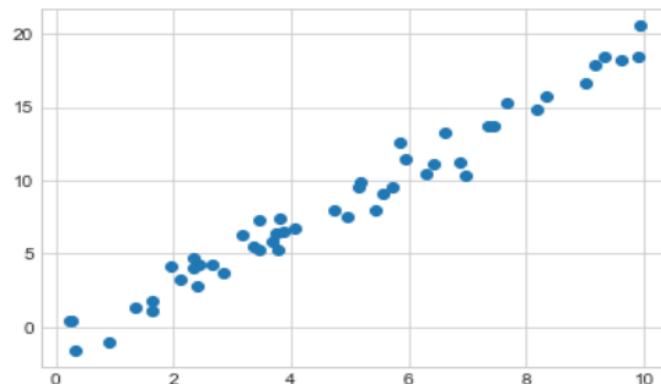
SCKIT Model Predict



Simple Linear Example

```
In [1]: 1 %matplotlib inline
2 #import library required
3 import matplotlib.pyplot as plt
4 plt.style.use('seaborn-whitegrid') #set the style to use
5 import numpy as np
6
7 rn=np.random.RandomState(30) #set the seed for reproducibility
8 x=10*rn.rand(50) #produce 10 random numbers
9 a=2 # the slope of the line
10 c=5 #y intersection of the line
11
12 y=a*x+-1+rn.randn(50)
13 ax=plt.axes()
14 ax.scatter(x,y)
```

Out[1]: <matplotlib.collections.PathCollection at 0x1609dc83f28>



- Generate a 2-dimensional data x and y
- The relationship is linear $y=ax+b$
- X is randomly generated

Simple Linear Example

```
1 #use a sklearn linear model to fit the data
2 from sklearn.linear_model import LinearRegression
3 model=LinearRegression(fit_intercept=True)
4 print(x.shape)
5 X=x[:,np.newaxis]
6 print(X.shape)
7 model.fit(X,y)
8 print(model.coef_)      #print the coeffficients
9 print(model.intercept_) #print the incept value
10 xfit=np.linspace(-1,11)
11 xfit=xfit[:,np.newaxis]
12 yfit=model.predict(xfit)
13 plt.scatter(x,y)
14 plt.plot(xfit,yfit,c='r')

(50,)
(50, 1)
[2.02341135]
-1.2628188060722358
```

Import chosen Linear Regression model

#step 1

#step 2

#step 3

#step 4

#step 5

initialize the

class with
desired
hyperparameter

Arrange data
into a feature
matrix

Fit the model
to data by call
the fit()
method

Apply the model for evaluation

MNIST Handwritten Dataset

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4

5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5

6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6

7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7

8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8

9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

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



MNIST Handwritten Dataset

- The MNIST database (**Modified National Institute of Standards and Technology database**) is a large database of handwritten digits that is commonly used for training various image processing systems.
- The database is also widely used for training and testing in the field of machine learning.
- It was created by "re-mixing" the samples from NIST's original datasets. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset.
- The MNIST database contains 60,000 training images and 10,000 testing images.

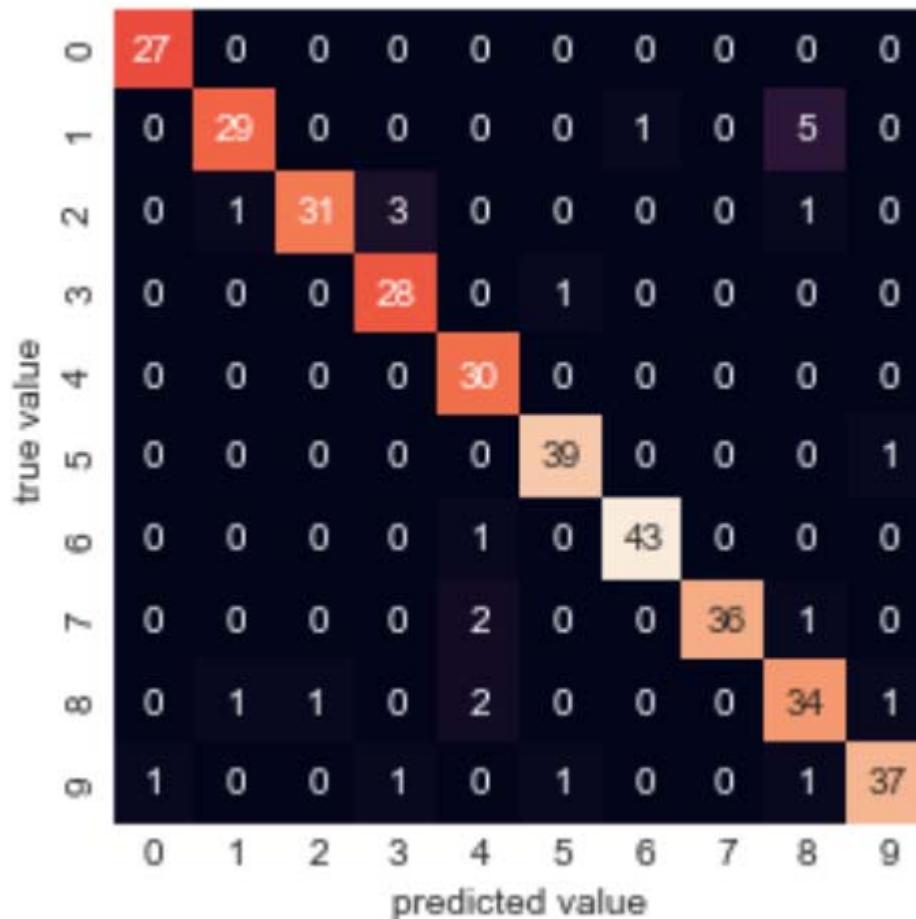
More information on the digit dataset from Scikit:

http://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html

Detail information on the digit dataset from UCI:

<http://archive.ics.uci.edu/ml/datasets/PenBased+Recognition+of+Handwritten+Digits>

Confusion Matrix for Classification



Confusion Matrix plotted as heat-map for SGDClassifier for MNIST Handwritten Dataset with classification accuracy of 92%