Class 03: models

蕭佳明
遠東科技大學 助理教授

Machine Learning

- **Everything Old Is New Again**
  - Computer capacity
  - Amount of data available
- **You Are Already Reaping the Benefits of Machine Learning**
  - Netflix account
  - Amazon
- **It’s All About the Data**

https://dzone.com/articles/how-will-deep-learning-change-the-way-we-design-ou

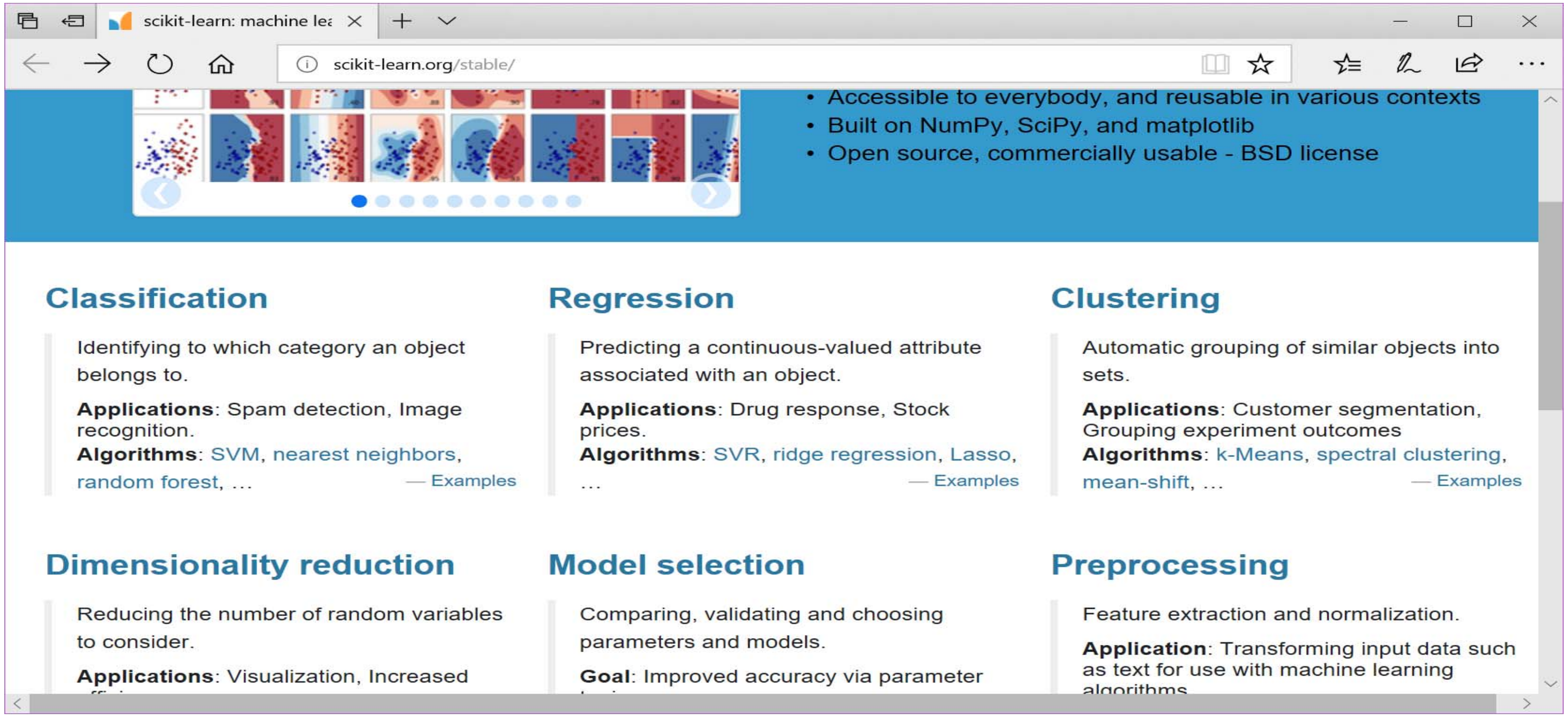
It’s All About the Data

- Given the importance of the data to the success of any machine learning implementation.
  - Data Quality
  - Data Volume
  - Data Timeliness
  - Data Pedigree

https://dzone.com/articles/how-will-deep-learning-change-the-way-we-design-ou

Machine Learning

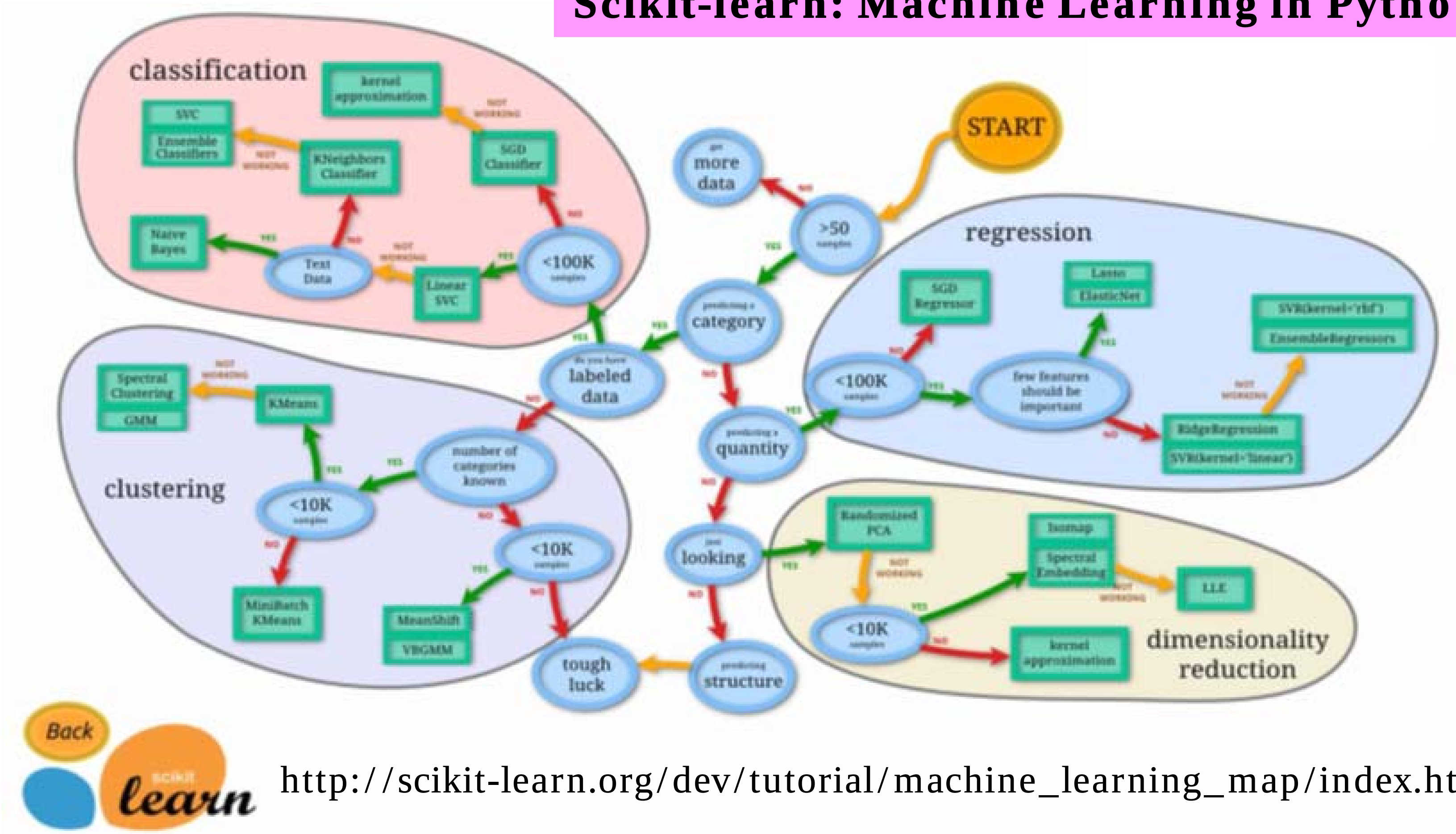
- Scikit-Learn



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

Scikit-Learn Packages

Scikit-learn: Machine Learning in Python



http://scikit-learn.org/dev/tutorial/machine_learning_map/index.html

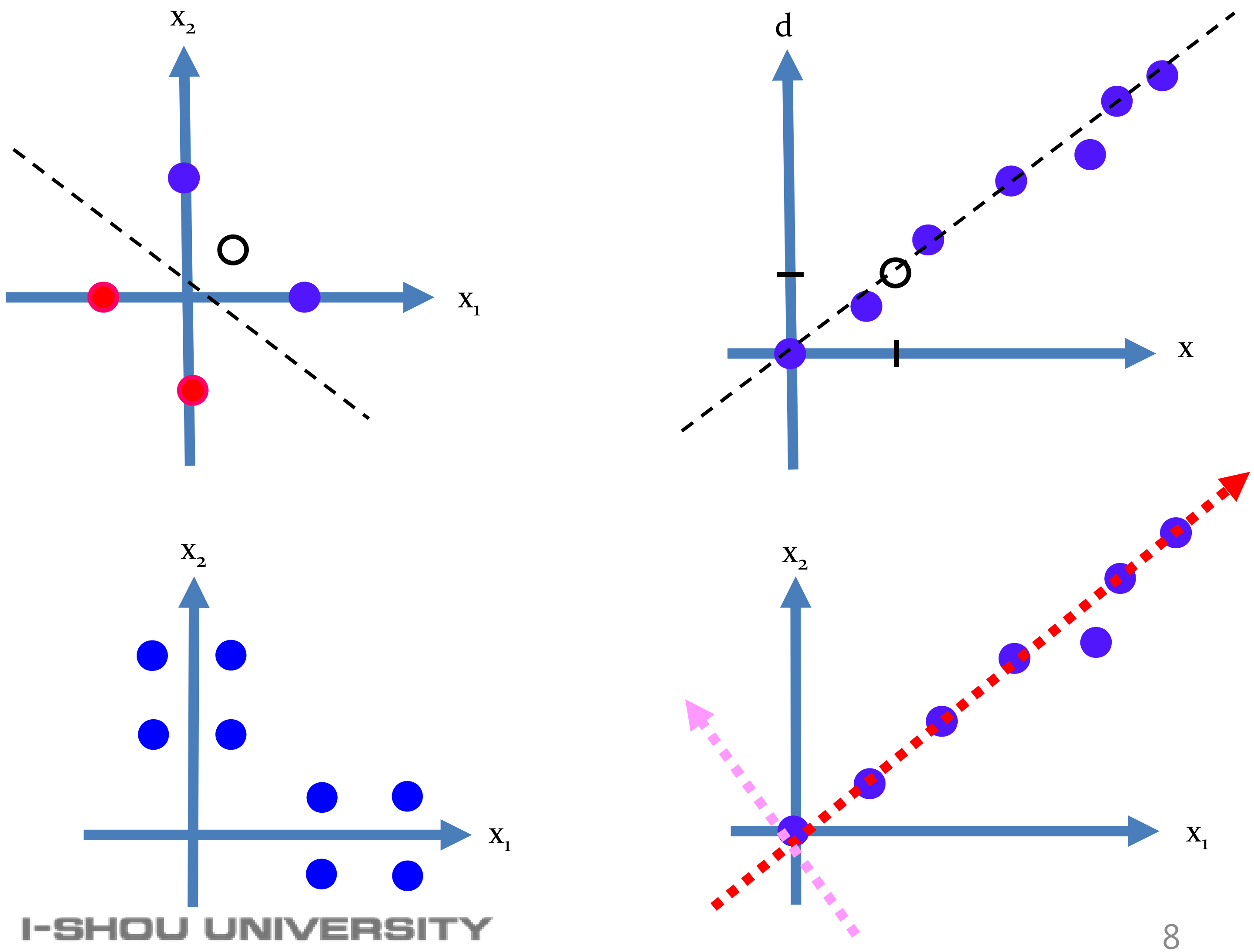
Supervised Learning

- Model-based learning
  - Linear regression
  - Regression with regularization
  - Logistic regression
  - Support vector machine
  - Decision Tree
  - Random Forests
- Instance-based learning
  - Naive Bayesian model
  - K-nearest neighbor(KNN)

Unsupervised Learning

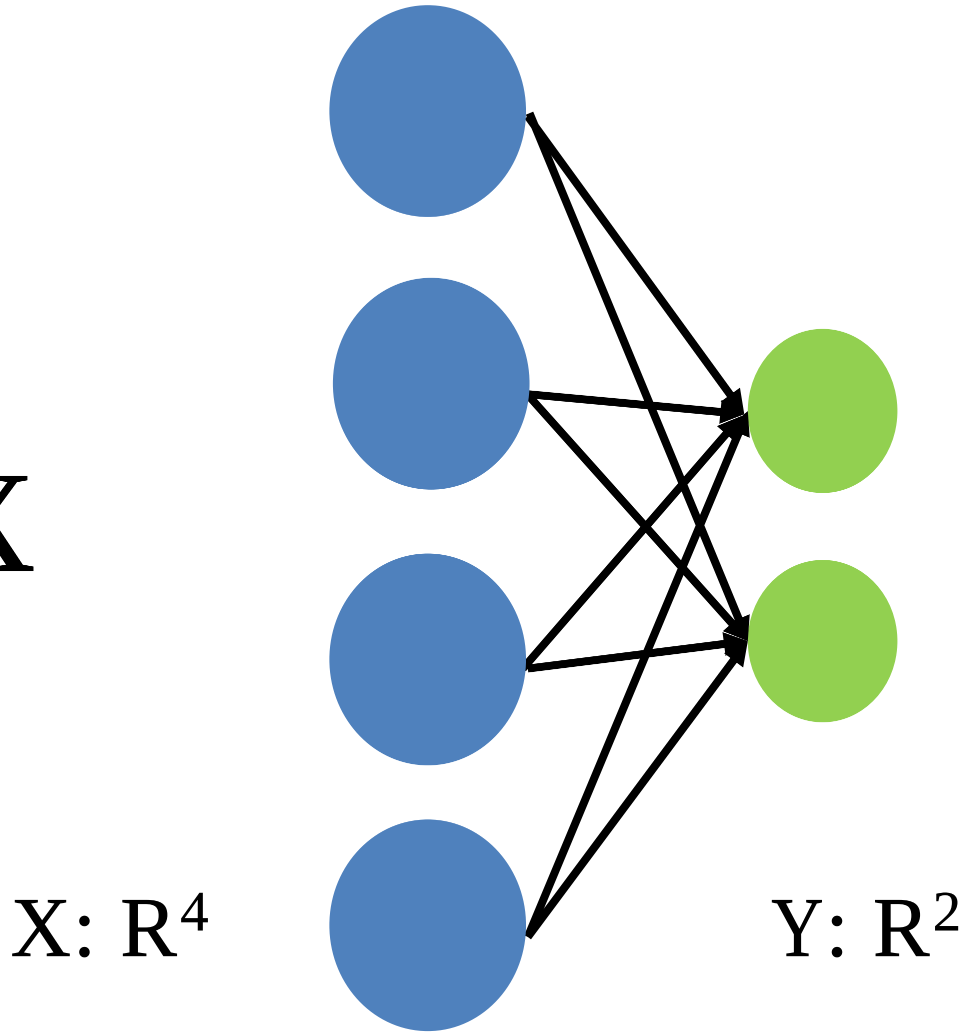
- Principal Component Analysis
- K-mean

Model Diagnostic



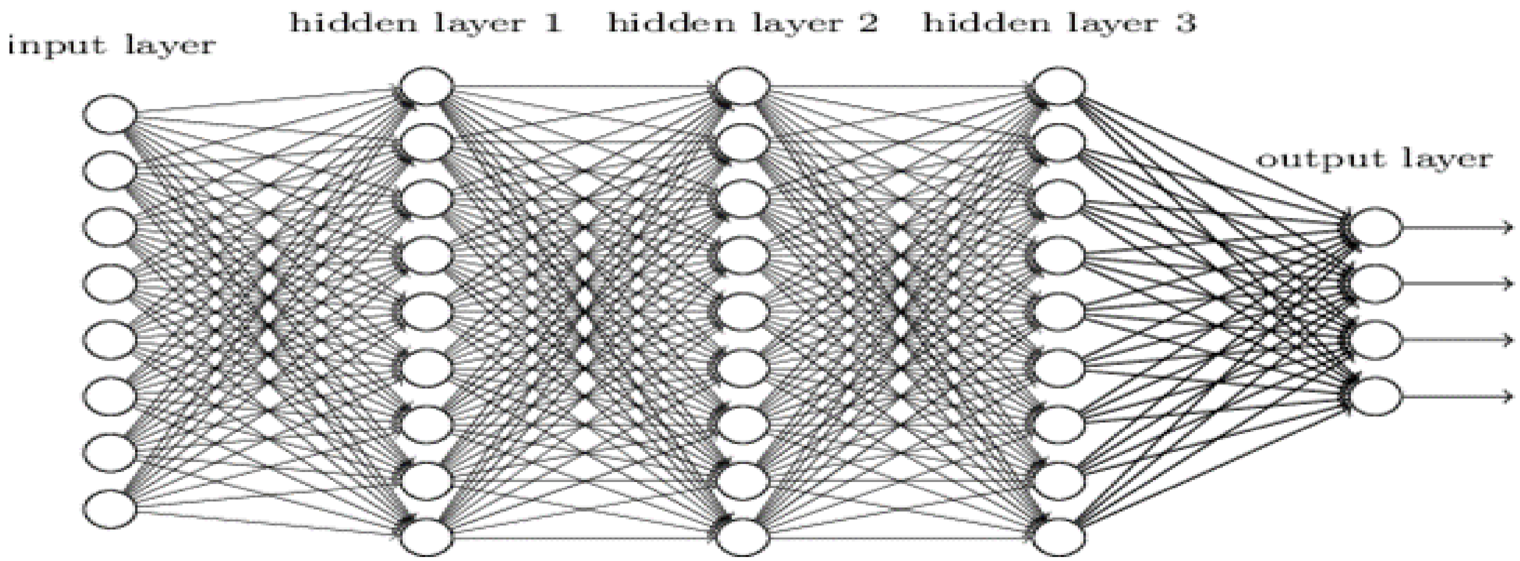
Mapping

$$Y=AX$$



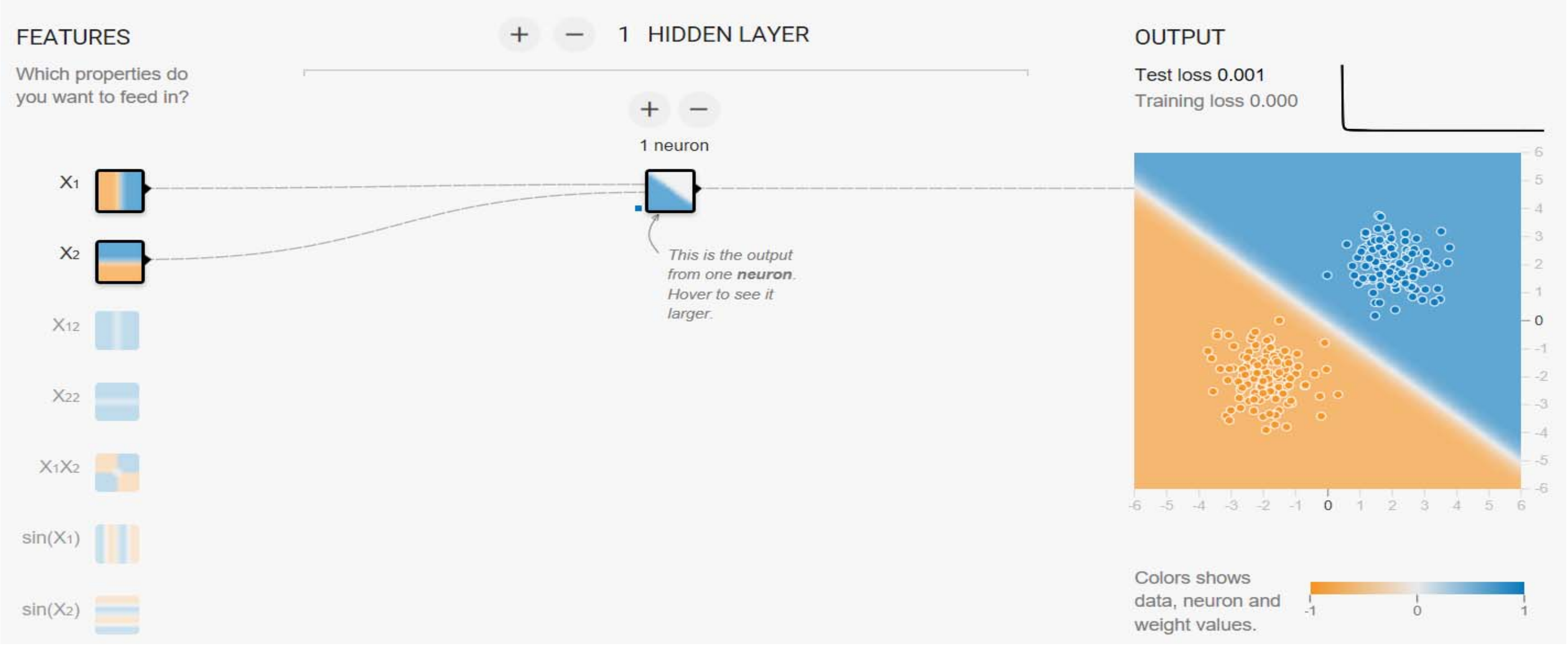
Deep Feature Map

- 深度學習模型 (Model)



Neural Network

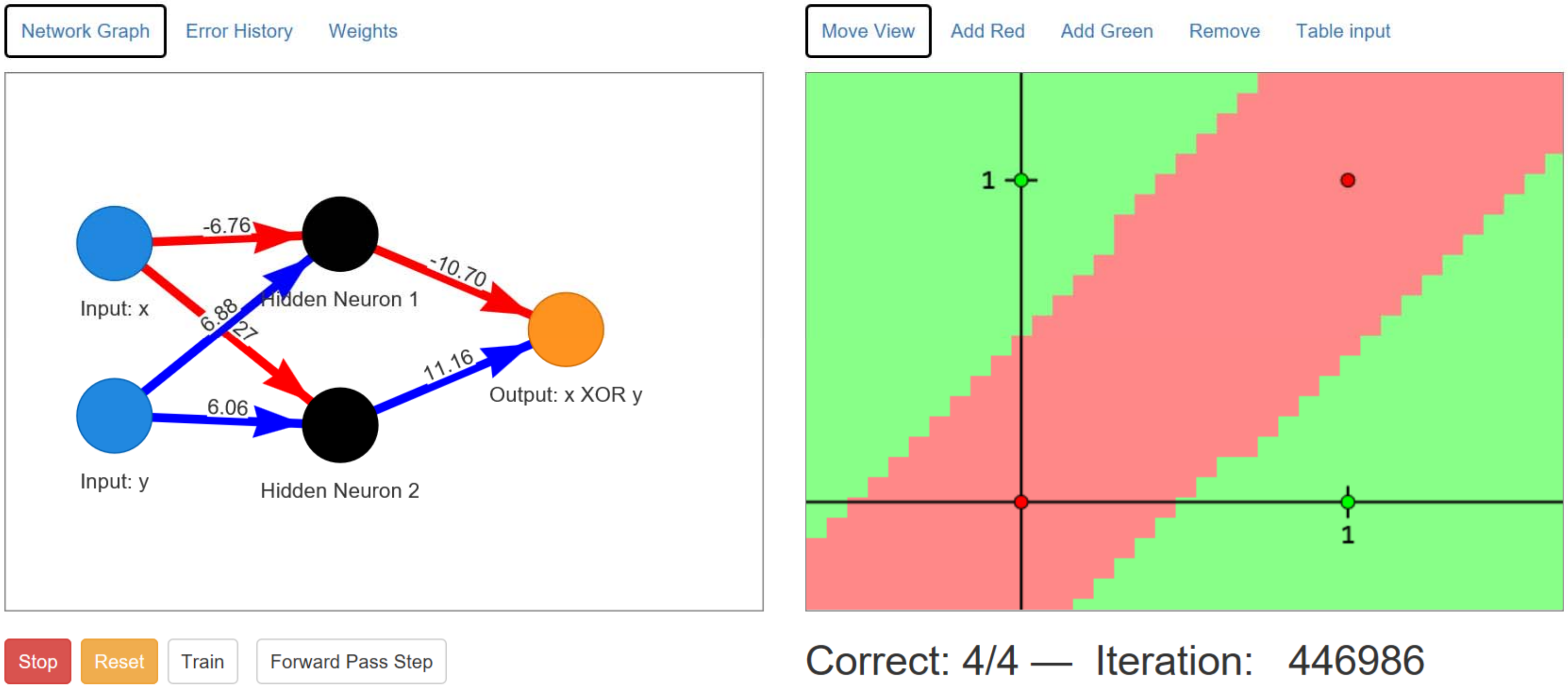
- **Linearly Problem**



https://playground.tensorflow.org/

Neural Network demo

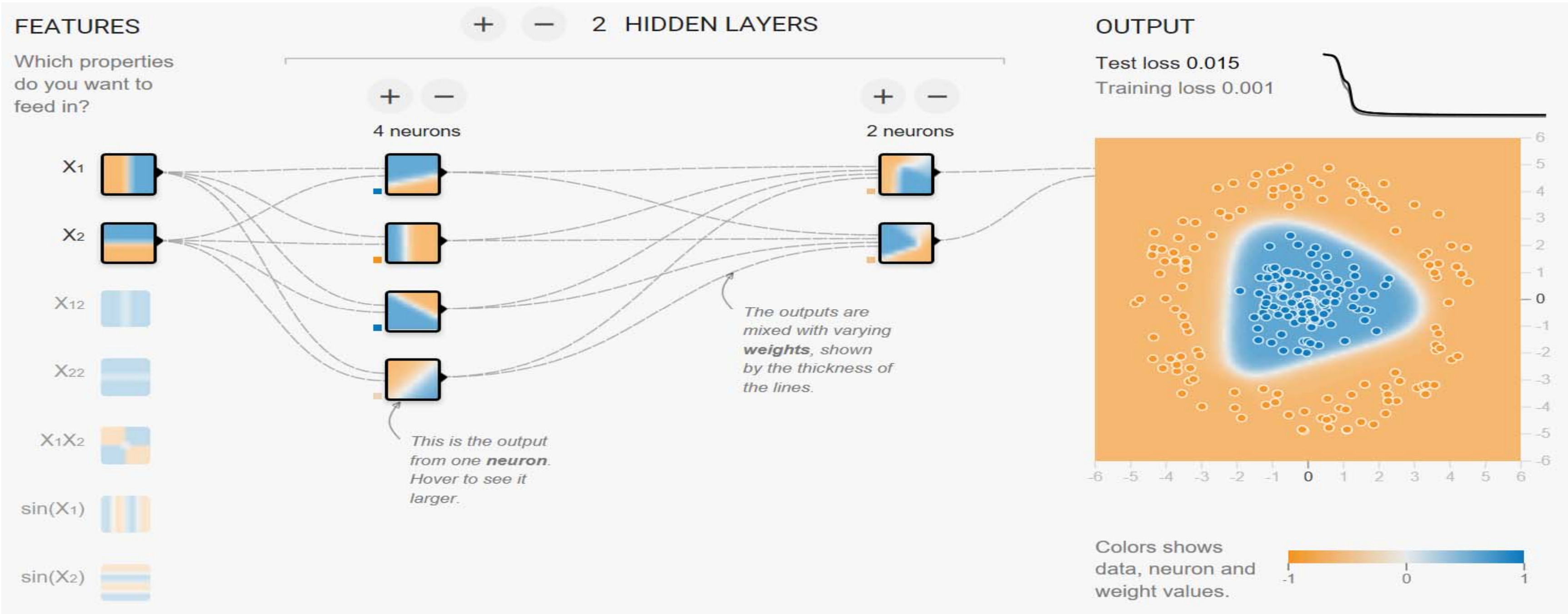
- **Nonlinear model**



https://lecture-demo.ira.uka.de/neural-network-demo/

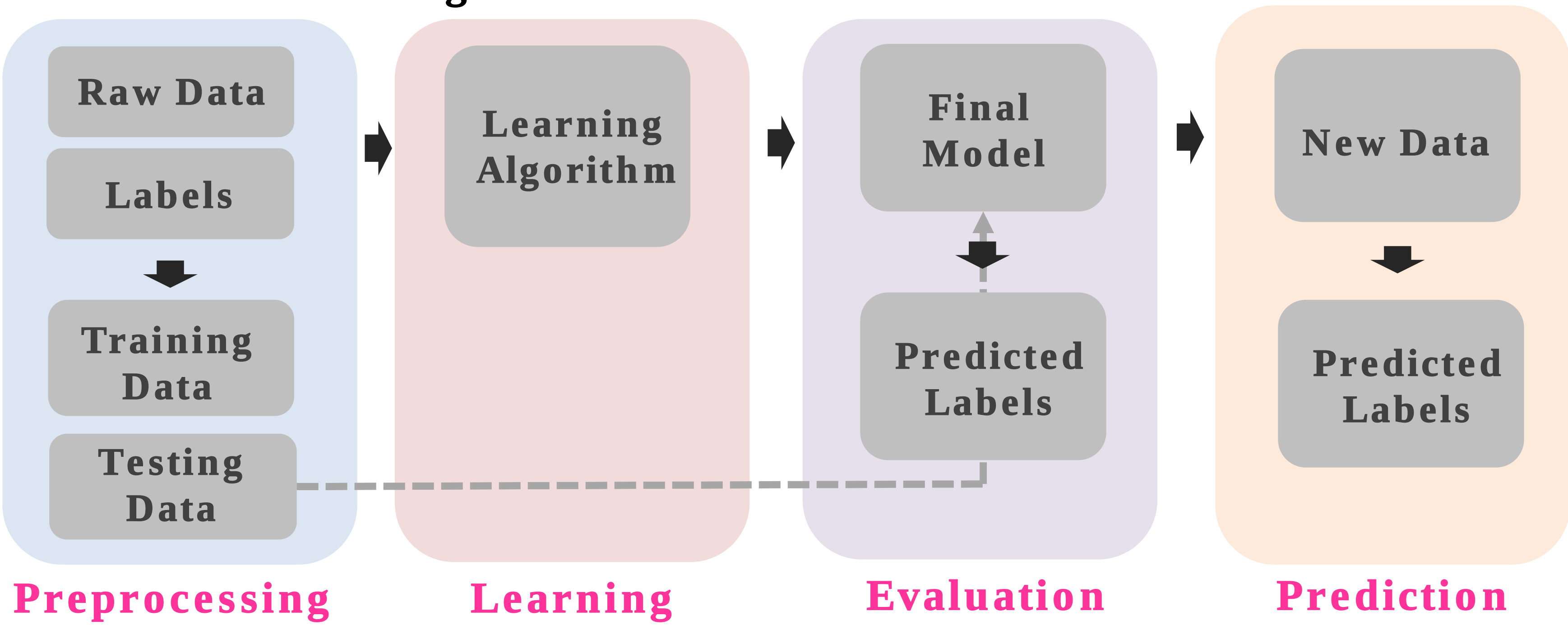
Neural Network

- **Nonlinear model**



Supervised Learning

Predictive Modeling

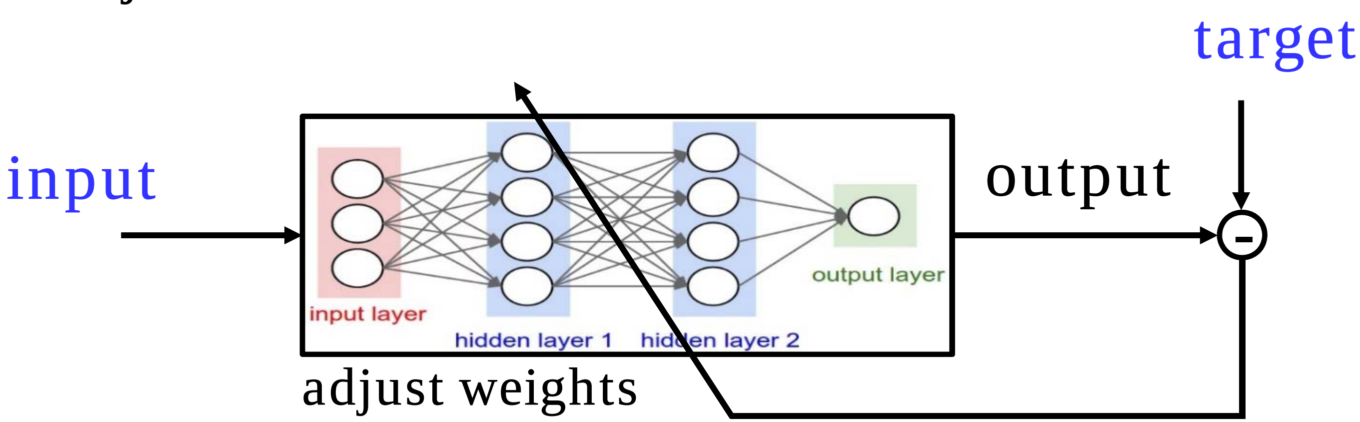


- Feature extraction -normalization
- Feature selection
- Dimension reduction
- Sampling

- Model selection
- Cross-validation
- Metric
- Hyperparameter optimization

Supervised Learning

- Supervised Learning
  - Model Selection
    - Parameters of Model
    - Parameters of Learning
    - Learning Algorithm
    - Object Function



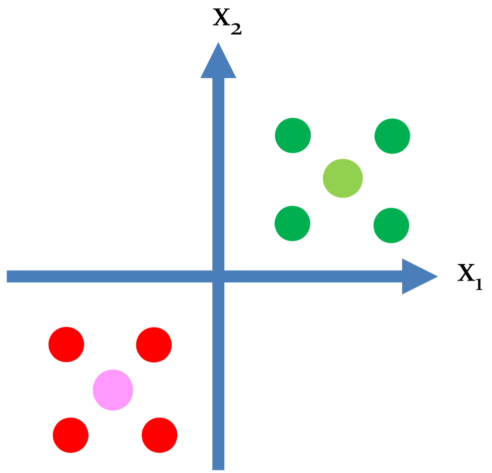
Toy dataset

- Clustering (data reduction/compression)
- Density estimation

input (x1)	input (x2)
-1	-1
-2	-1
-1	-2
-2	-2
1	1
2	1
1	2
2	2

input (x1)	input (x2)
-1	-1
-2	-1
-1	-2
-2	-2
1	1
2	1
1	2
2	2

input (x1)	input (x2)
-1.5	-1.5
1.5	1.5



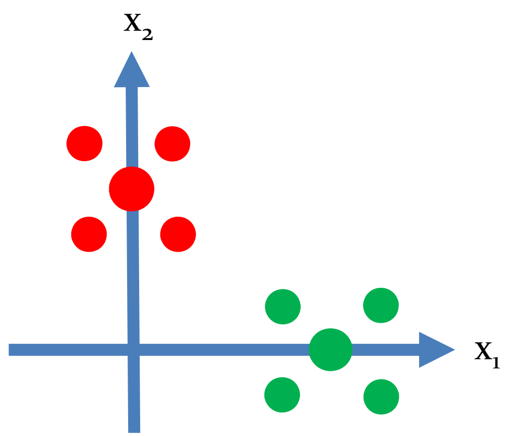
Toy dataset

- Data mining

input (x1)	input (x2)
3	1
3	-1
5	-1
5	1
1	1
1	3
-1	1
-1	3

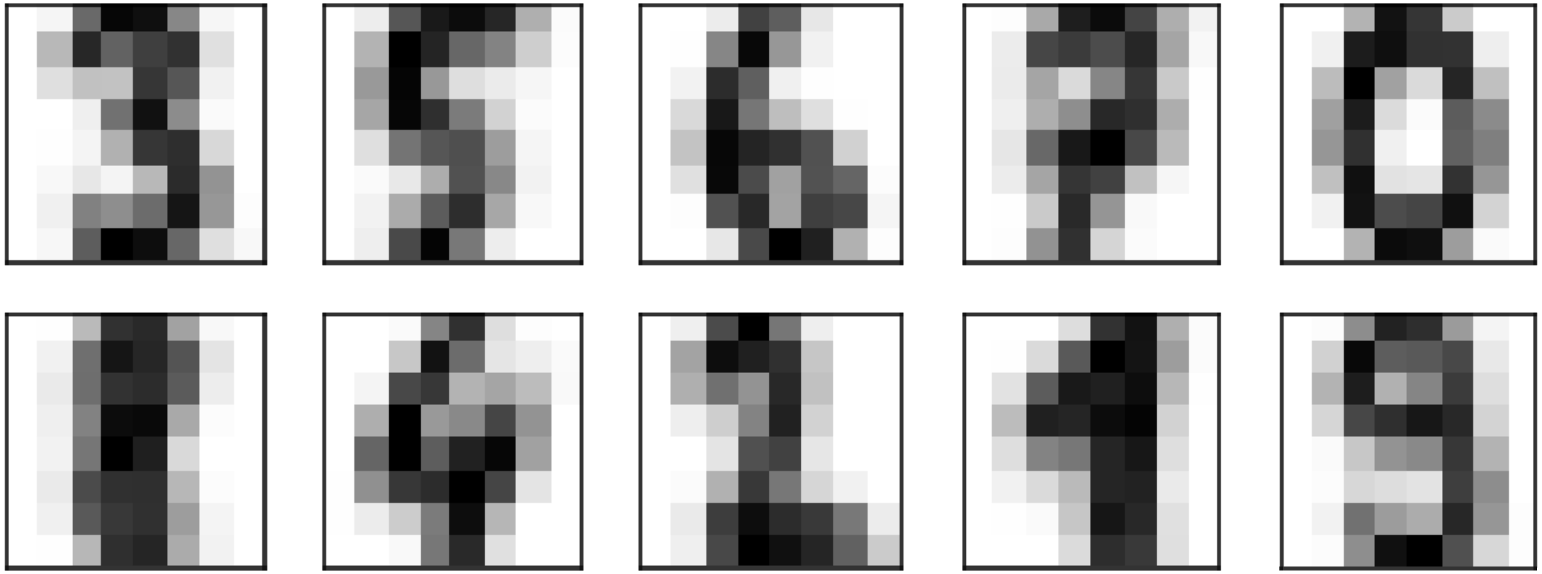
input (x1)	input (x2)
3	1
3	-1
5	-1
5	1
1	1
1	3
-1	1
-1	3

input (x1)	input (x2)
4	~~0~~
~~0~~	4



Unsupervised Learning

- K-means clustering for data compression



digits.images.shape = (1797, 8, 8)

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

06_kmeans_clustering.ipynb

PCA

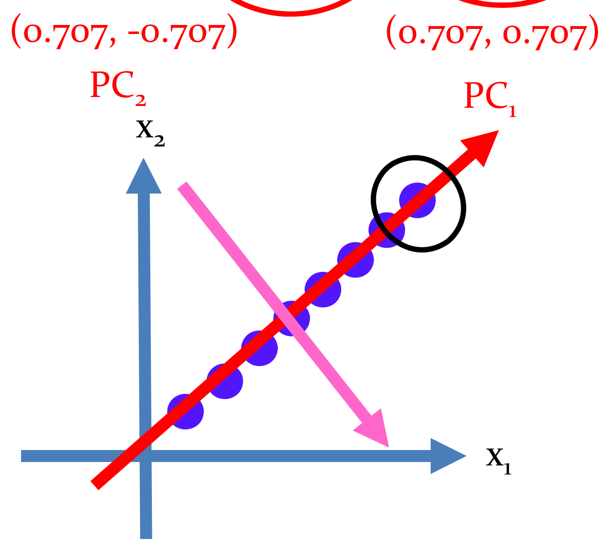
- Dimensionality reduction
- Data mining

```
                            PC1       PC2
Standard deviation        1.414 1.115e-16
```

```
Rotation:
        PC1        PC2
x 0.7071068  0.7071068
y 0.7071068 -0.7071068
```

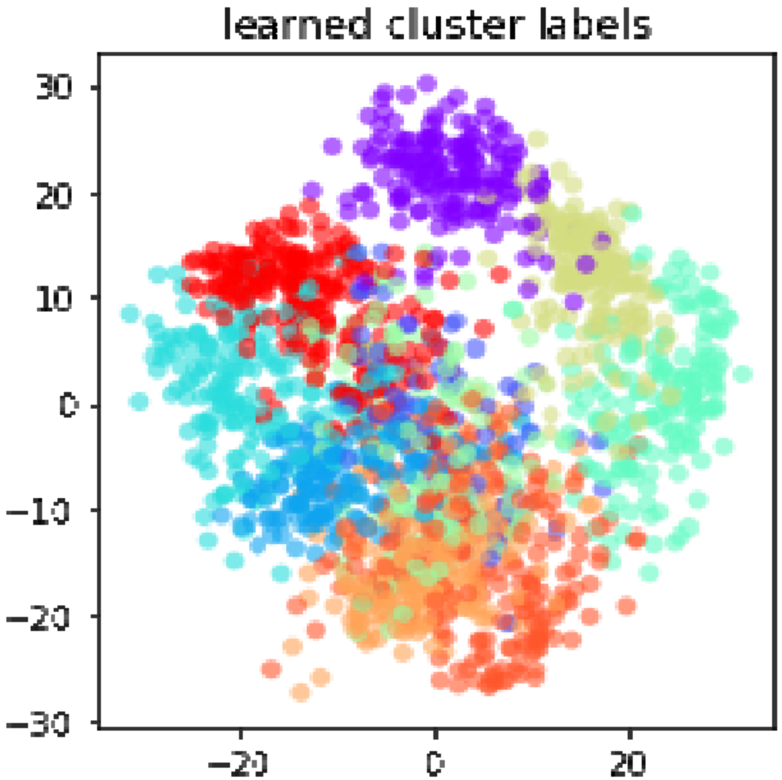
x1	x2
1	3
2	5
3	7
4	9
5	11
6	13
7	15
8	17

PC_1	PC_2
-2.02	0.00
-1.44	0.00
-0.86	0.00
-0.28	0.00
0.28	0.00
0.86	0.00
1.44	0.00
2.02	0.00

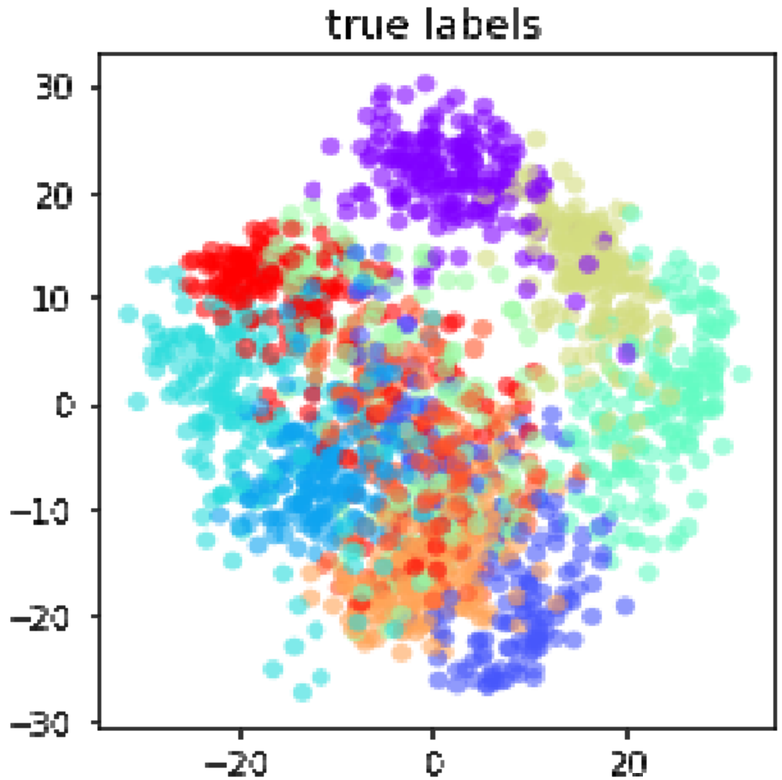


Unsupervised Learning

- PCA Visualization



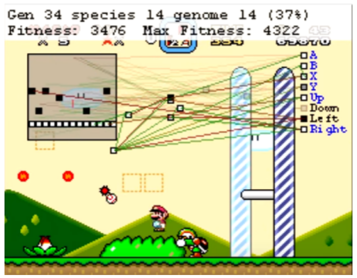
digits.images.shape = (1797, 2)

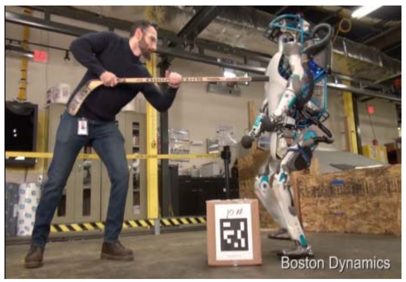


digits.images.shape = (1797, 2)

Reinforcement Learning

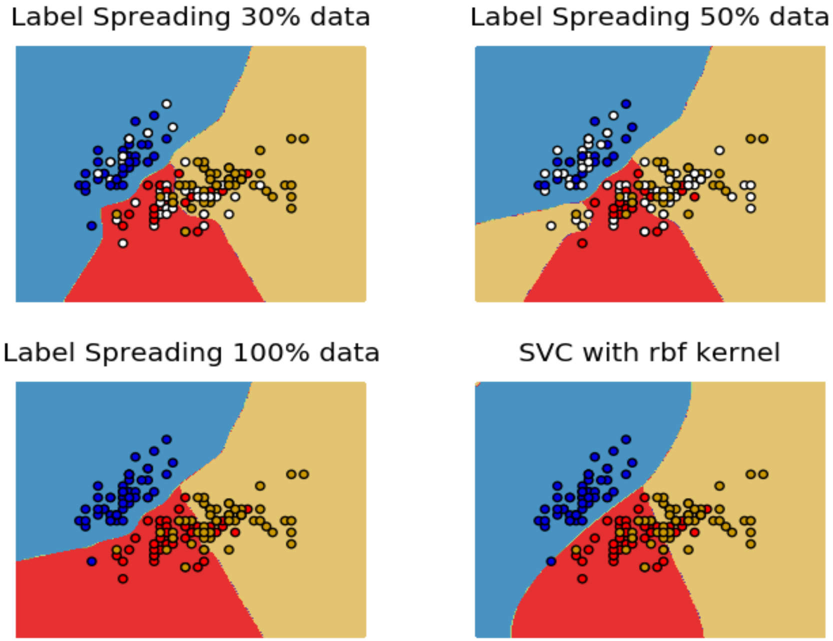
- Training data: (S, A, R). (State-Action-Reward)
- Goal: Develop an optimal policy (sequence of decision rules) for the learner so as to **maximize its long-term reward**.

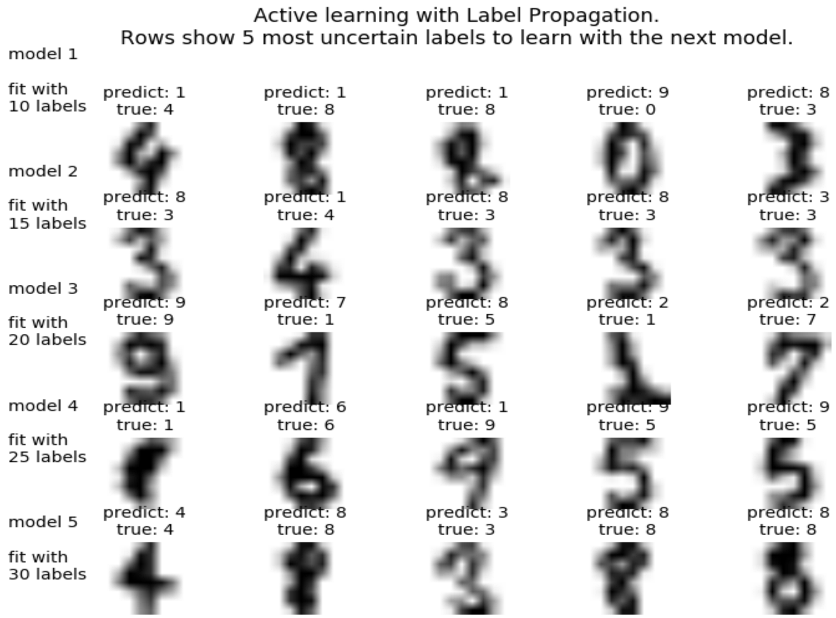




Semi-supervised Learning

- Semi-supervised learning is about using these unlabeled examples to improve supervised learning methods, which generally require labeled examples for training.





Machine Learning

- **Supervised learning**
- **Unsupervised learning**
- Reinforcement learning
- Semi-supervised learning

Scikit-learn's Estimator

- Available in **all estimators**
  - **model.fit()**: fit training data, X: data, Y: label
  - Supervised: **model.fit(X,Y)** ⟺ Unsupervised:**model.fit(X)**
- Available in **supervised estimators**
  - **model.predict()** : predict the label of a new set of data by model
  - **model.predict_proba()** : for classification problems, some estimators provides this method to return probability of each class
- Available in **unsupervised estimators**
  - **model.transform()** : transform new data into new basis by model
  - **model.fit_transform() :** some estimators implement this method to efficiently perform a fit and transform on the same input data

Supervised Learning

- In supervised learning, we have a dataset consisting of both **features** (input variables) and **labels** (output variables)
- The task is to construct an **estimator**(model) which enables to predict the labels of an instance given the set of features
- Two categories: **Classification** and **Regression**
  - Classification: the label is **discrete**
  - Regression: the label is **continuous**
- Split into **training** and **testing** datasets

Supervised Learning

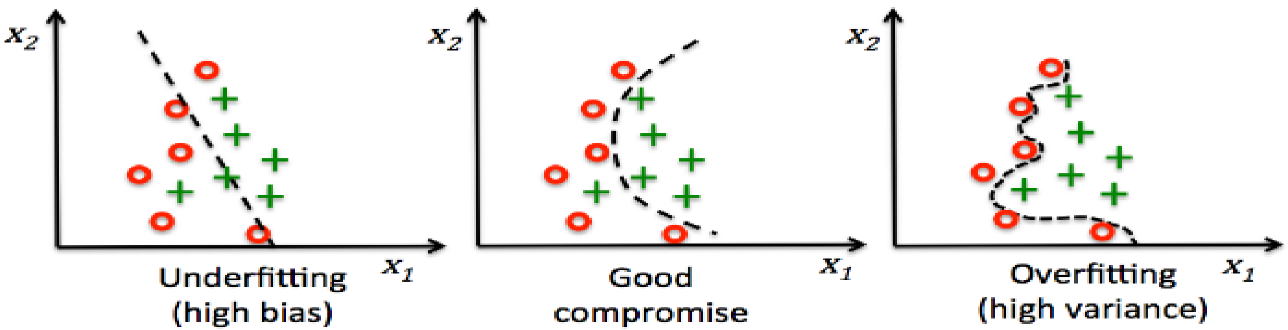
- Model-based learning
  - Linear regression
  - Regression with regularization
  - Logistic regression
  - Support vector machine
  - Decision Tree
  - Random Forests
  - Neural Networks
- Instance-based learning
  - Naive Bayesian model
  - K-nearest neighbor(KNN)

Training and Testing Dataset

- To validate the generalization of a trained model, splitting a whole dataset into **training** and **testing** datasets
- We fit and optimize predictive models using the training dataset
- Use testing datasets to evaluate the performance of trained models to ensure the model **generalization**

Generalization

- We hope that a model trained on the basis of a training dataset can seamlessly apply to **unseen testing** dataset
- If the model over fits the training dataset, its performance on testing dataset will be worse
- Higher model complexity, easier to overfitting



Hyperparameters

- Learning the parameters of a prediction function and testing it on **the same data** is a methodological mistake
- There are **Hyperparameters**: choices about the algorithm that we **set** rather than learn

Your Dataset: Images / Corpus

Hyperparameters

- **idea 1:** Choose **hyperparameters** that work best on the data Your Dataset

Your Dataset

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

Train	Testing

- **idea 3:** Split data into train, val., and test; choose **hyperparameters** on val. and evaluate on test

Train	Validation	Testing

Hyperparameters

- **idea 4: Cross-Validation**: Split data into folds, try each fold as validation and **average** the results

fold1	**fold2**	**fold3**	**Testing**
fold1	**fold2**	**fold3**	**Testing**
fold1	**fold2**	**fold3**	**Testing**

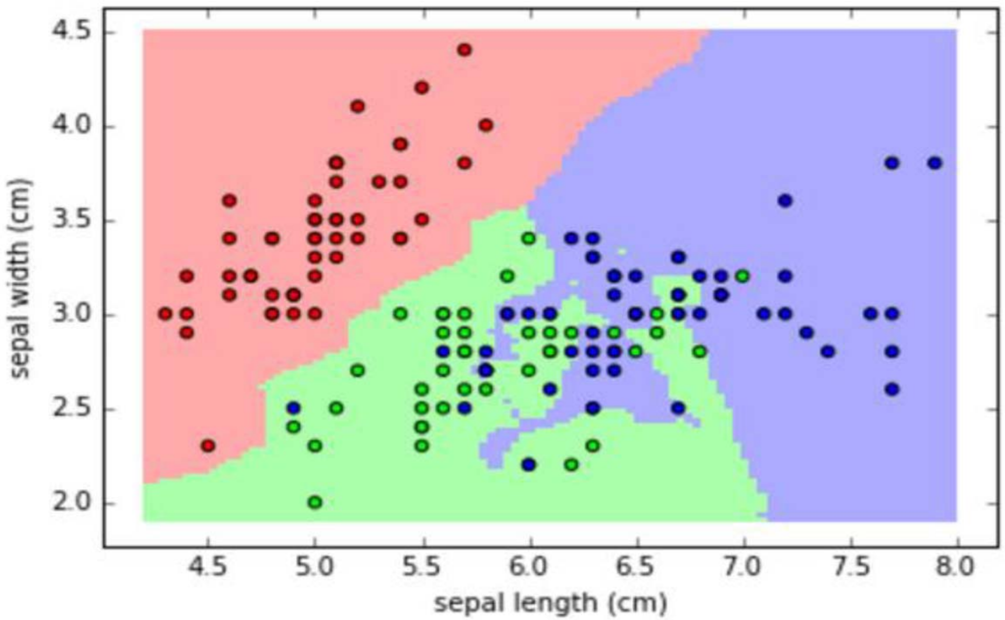
Useful for small datasets, but not used too frequently in deep learning

Classification

- **k-Nearest Neighbor** for iris

Fisher's *Iris* Data

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.8	1.9	0.4	*I. setosa*
4.8	3.0	1.4	0.3	*I. setosa*
5.1	3.8	1.6	0.2	*I. setosa*
4.6	3.2	1.4	0.2	*I. setosa*
5.3	3.7	1.5	0.2	*I. setosa*
5.0	3.3	1.4	0.2	*I. setosa*
7.0	3.2	4.7	1.4	*I. versicolor*
6.4	3.2	4.5	1.5	*I. versicolor*
6.9	3.1	4.9	1.5	*I. versicolor*
5.5	2.3	4.0	1.3	*I. versicolor*
6.5	2.8	4.6	1.5	*I. versicolor*
5.1	2.5	3.0	1.1	*I. versicolor*
5.7	2.8	4.1	1.3	*I. versicolor*
6.3	3.3	6.0	2.5	*I. virginica*
5.8	2.7	5.1	1.9	*I. virginica*
7.1	3.0	5.9	2.1	*I. virginica*
6.3	2.9	5.6	1.8	*I. virginica*
6.5	3.0	5.8	2.2	*I. virginica*
7.6	3.0	6.6	2.1	*I. virginica*
4.9	2.5	4.5	1.7	*I. virginica*



http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html#sklearn.neighbors.NearestNeighbors

Confusion matrix

	Predicted 0	Predicted 1
Actual 0	True Negative (TN)	False Positive (FP)
Actual 1	False Negative (FN)	True Positive (TP)

Recall = TP/ (FN+TP)

Precision = TP / (FP+TP)

F1 score = 2*(Recall∗Precision)/(Recall+Precision)

Confusion matrix

	Predicted 0	Predicted 1
Actual 0	True Negative (TN)	False Positive (FP)
Actual 1	False Negative (FN)	True Positive (TP)

Accuracy = (TN+TP)/ (TN+FP+FN+TP)

Metrics in Classification

```
from sklearn import metrics
```

```
metrics.confusion_matrix(y_test,y_pred)
```

```
array([[ 7,  0,  0],
       [ 0,  8,  4],
       [ 0,  1, 10]])
```

```
metrics.accuracy_score(y_test,y_pred)
```

```
0.83333333333333337
```

```
metrics.precision_score(y_test,y_pred,average="weighted")
```

```
0.85079365079365077
```

```
metrics.recall_score(y_test,y_pred,average="weighted")
```

```
0.83333333333333337
```

```
metrics.f1_score(y_pred=y_pred,y_true=y_test,average="weighted")
```

```
0.83142857142857152
```

```
print(metrics.classification_report(y_test,y_pred))
```

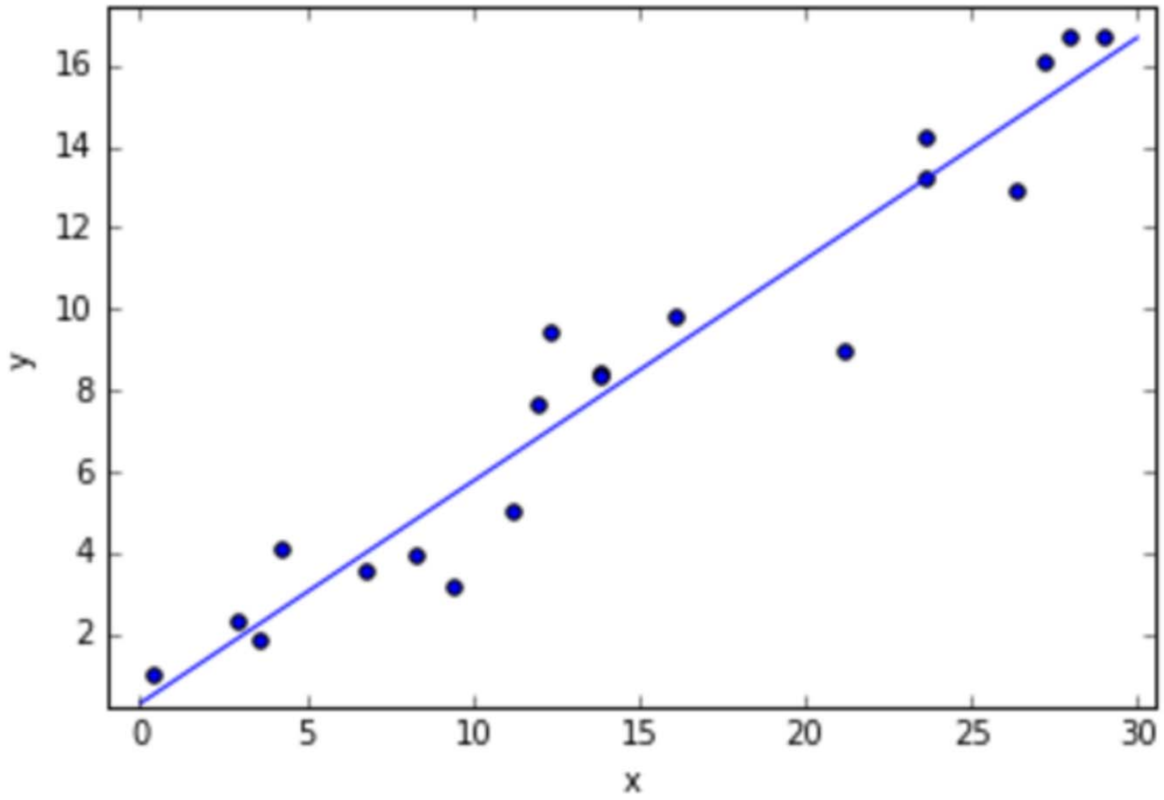
```
             precision    recall  f1-score   support

          0       1.00      1.00      1.00         7
          1       0.89      0.67      0.76        12
          2       0.71      0.91      0.80        11

avg / total       0.85      0.83      0.83        30
```

Regression

- Fit a line to the data



Metrics in Regression

- Mean Absolute Error (MAE)
- Mean Square Error (MSE)
- R Squared
- **Peak Signal to Noise Ratio (PNSR)**
- **Structural Similarity Index Measure (SSIM)**

```
metrics.r2_score(y_pred=y_pred,y_true=y_test)
```

0.8203173595813334

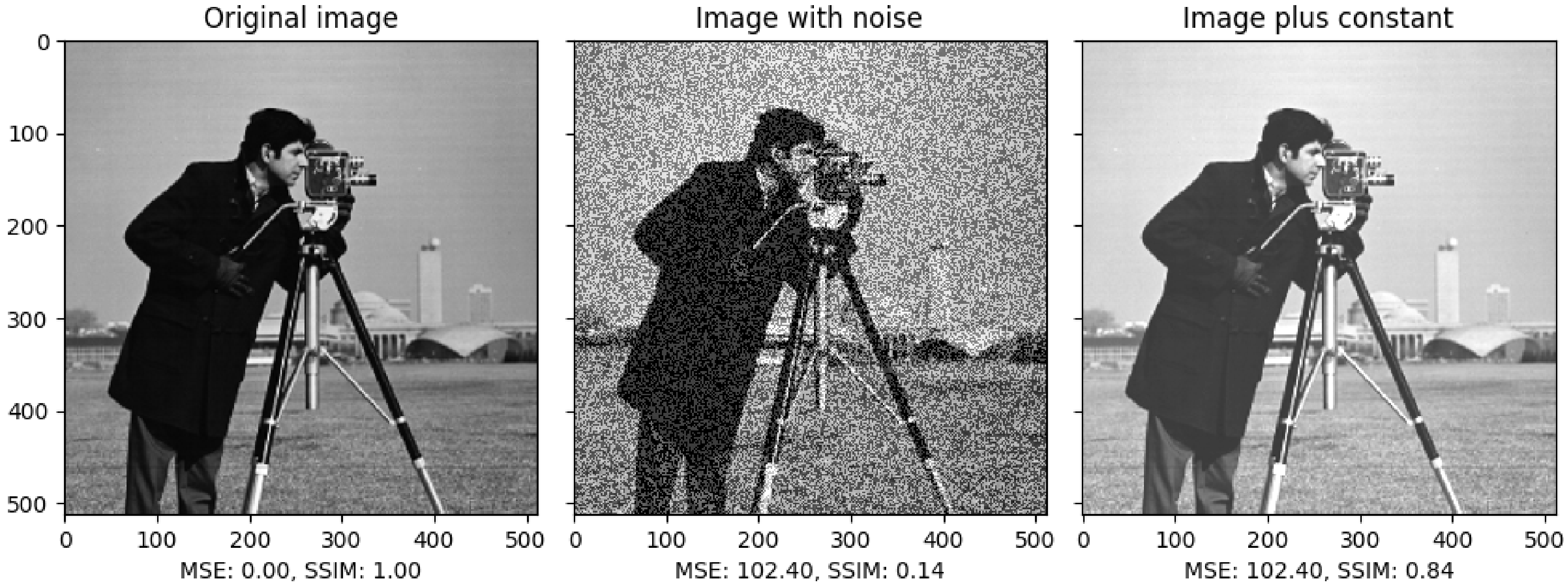
```
metrics.mean_absolute_error(y_pred=y_pred,y_true=y_test)
```

0.67860317120632041

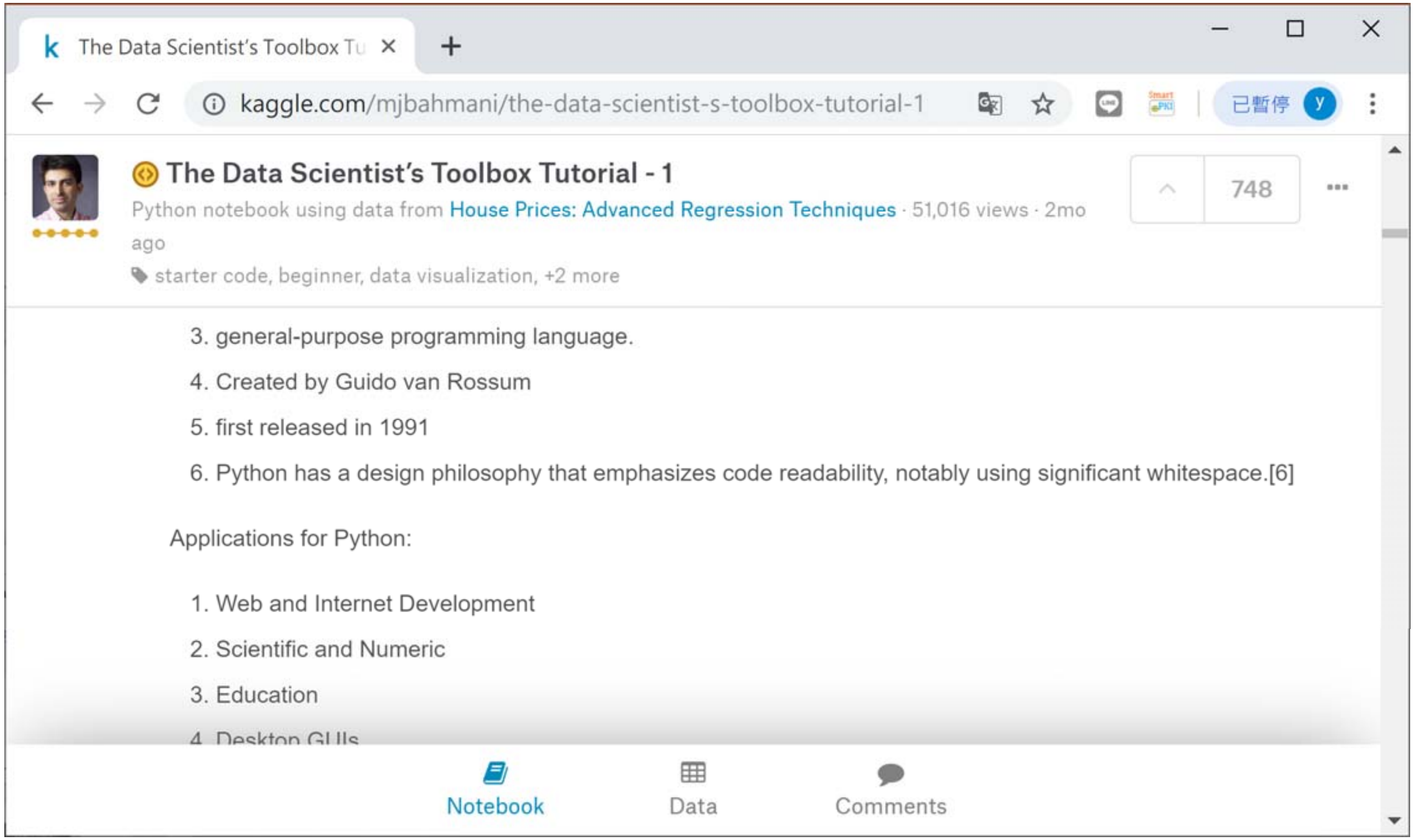
```
metrics.mean_squared_error(y_pred=y_pred,y_true=y_test)
```

0.62402155001086912

SSIM

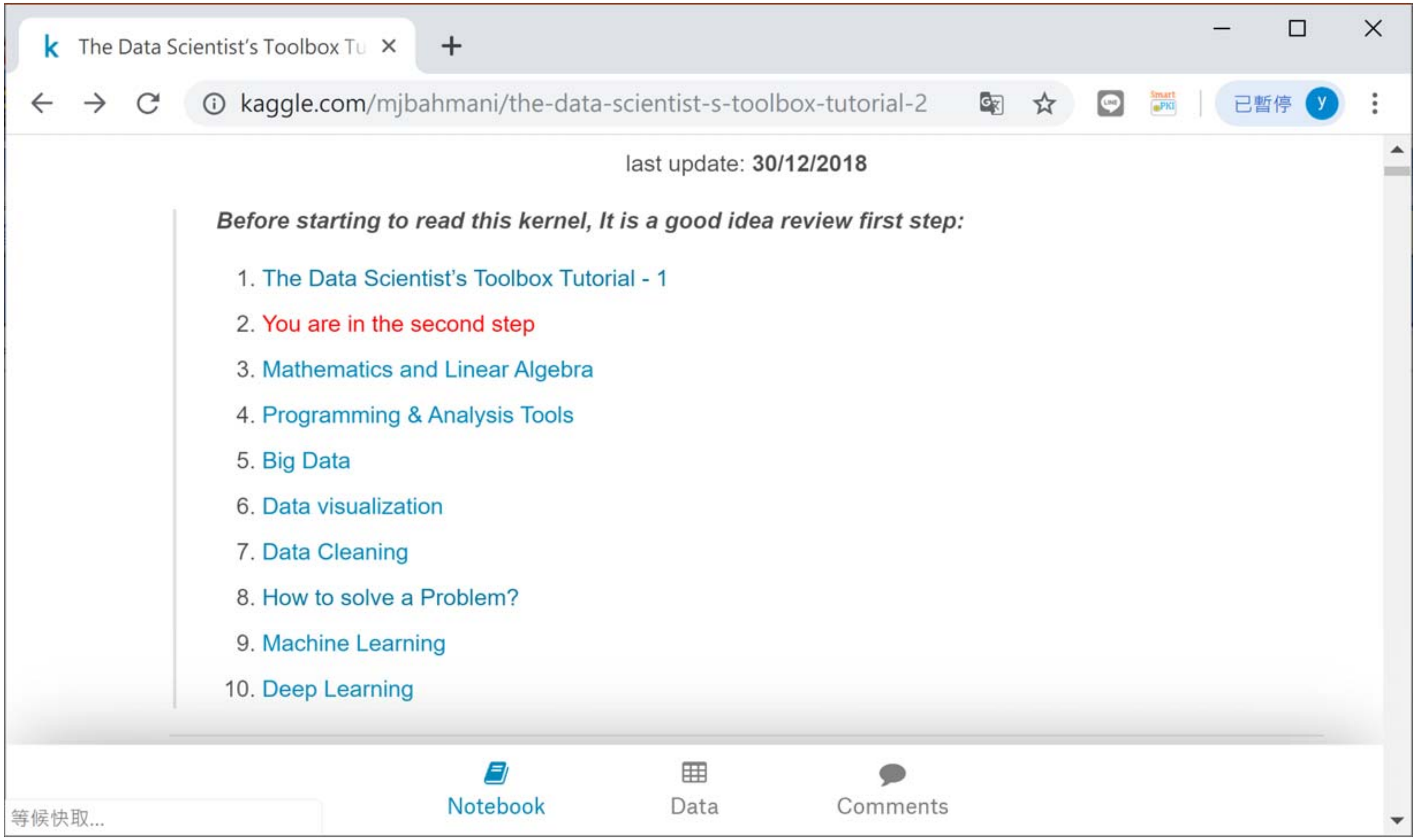

Original image
Image with noise
Image plus constant
0
100
200
300
400
500
MSE: 0.00, SSIM: 1.00
MSE: 102.40, SSIM: 0.14
MSE: 102.40, SSIM: 0.84

Data Scientist's Toolbox



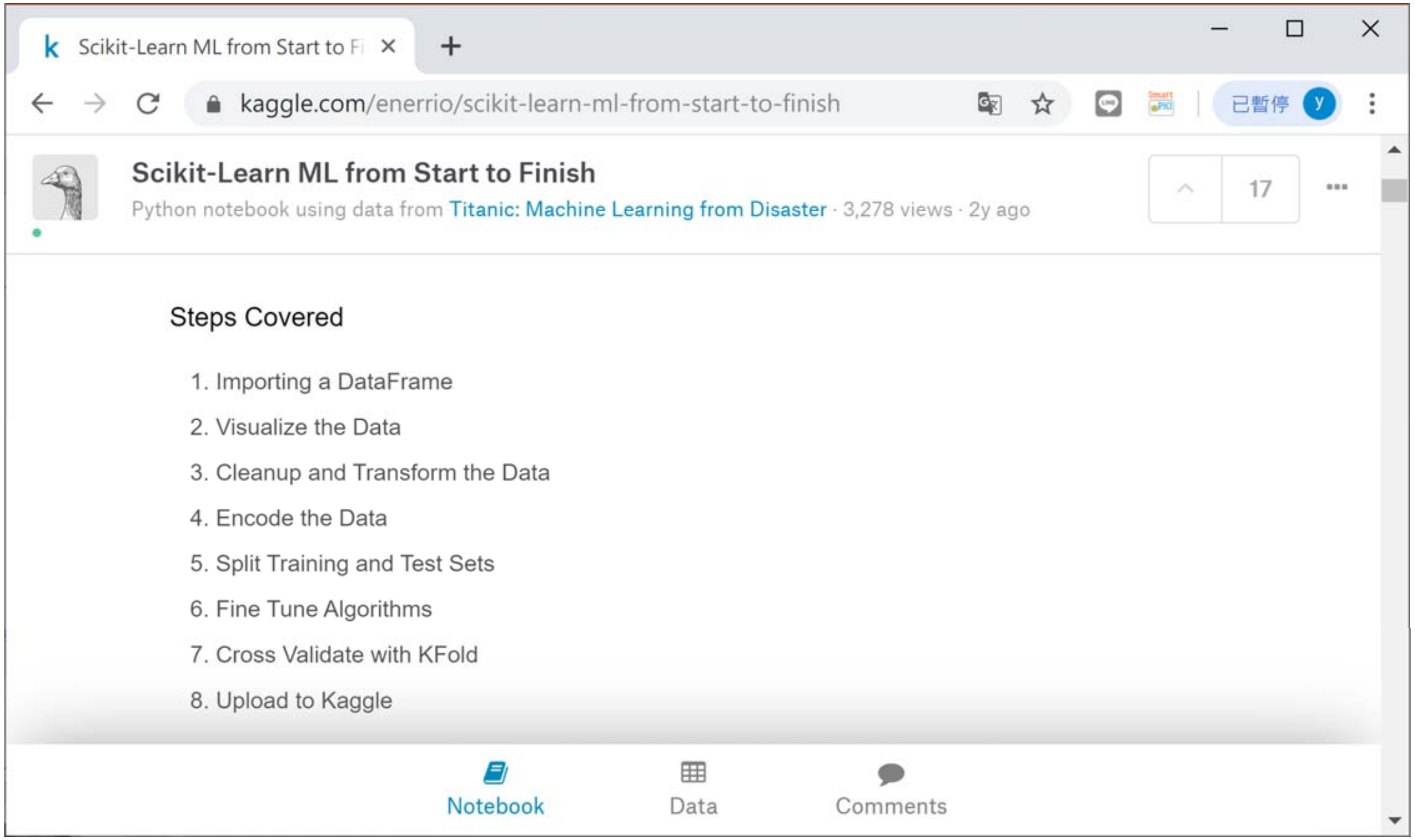
https://www.kaggle.com/mjbahmani/the-data-scientist-s-toolbox-tutorial-1

Data Scientist's Toolbox



https://www.kaggle.com/mjbahmani/the-data-scientist-s-toolbox-tutorial-2

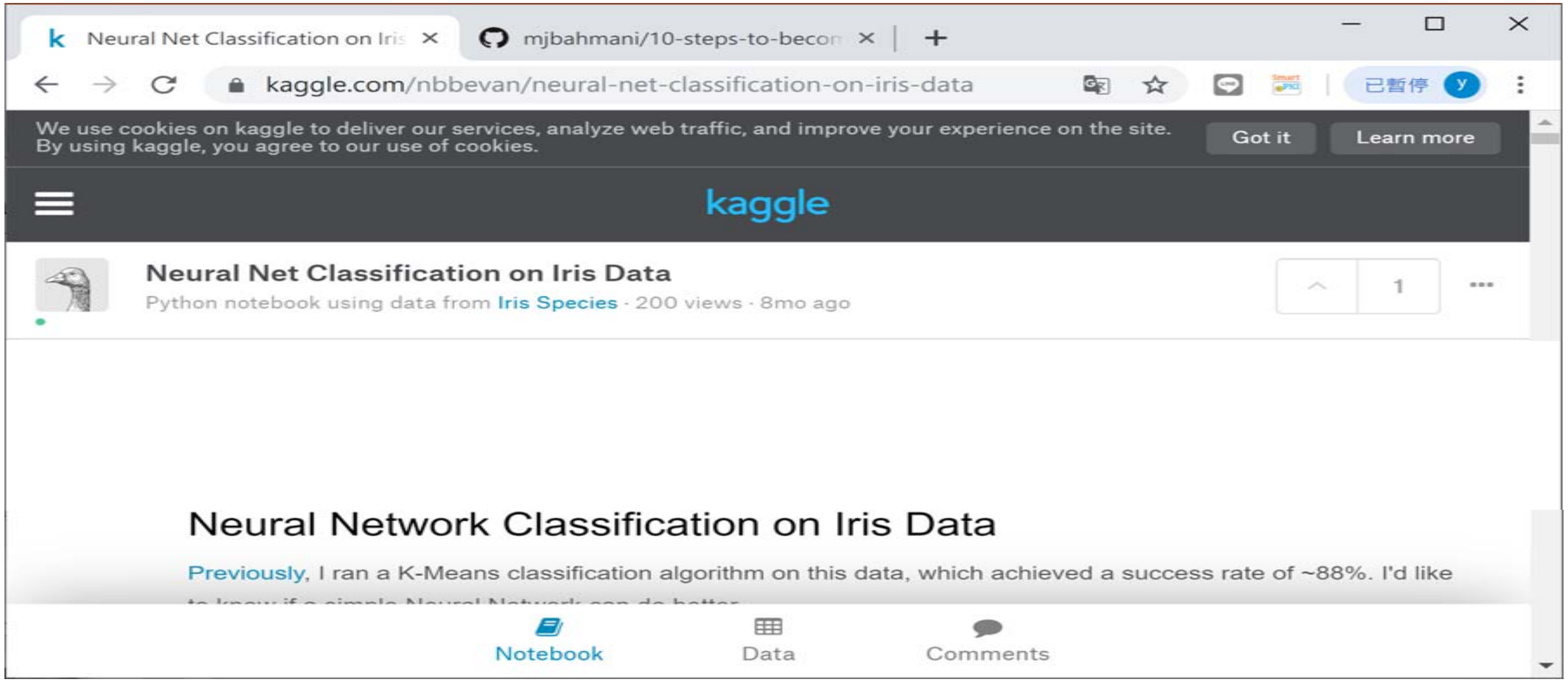
Scikit-Learn ML



https://www.kaggle.com/enerrio/scikit-learn-ml-from-start-to-finish

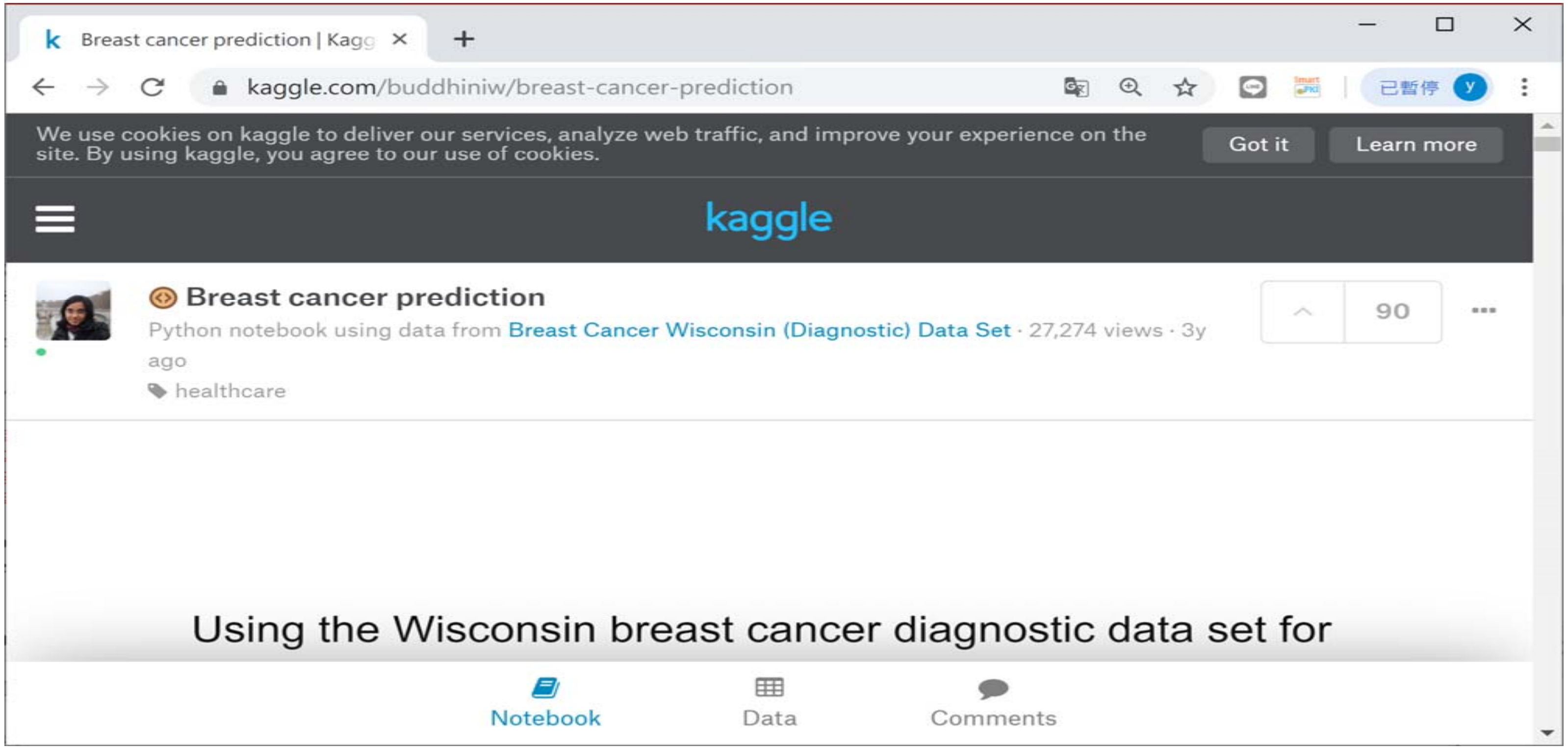
IRIS Classification

- **Neural Network**



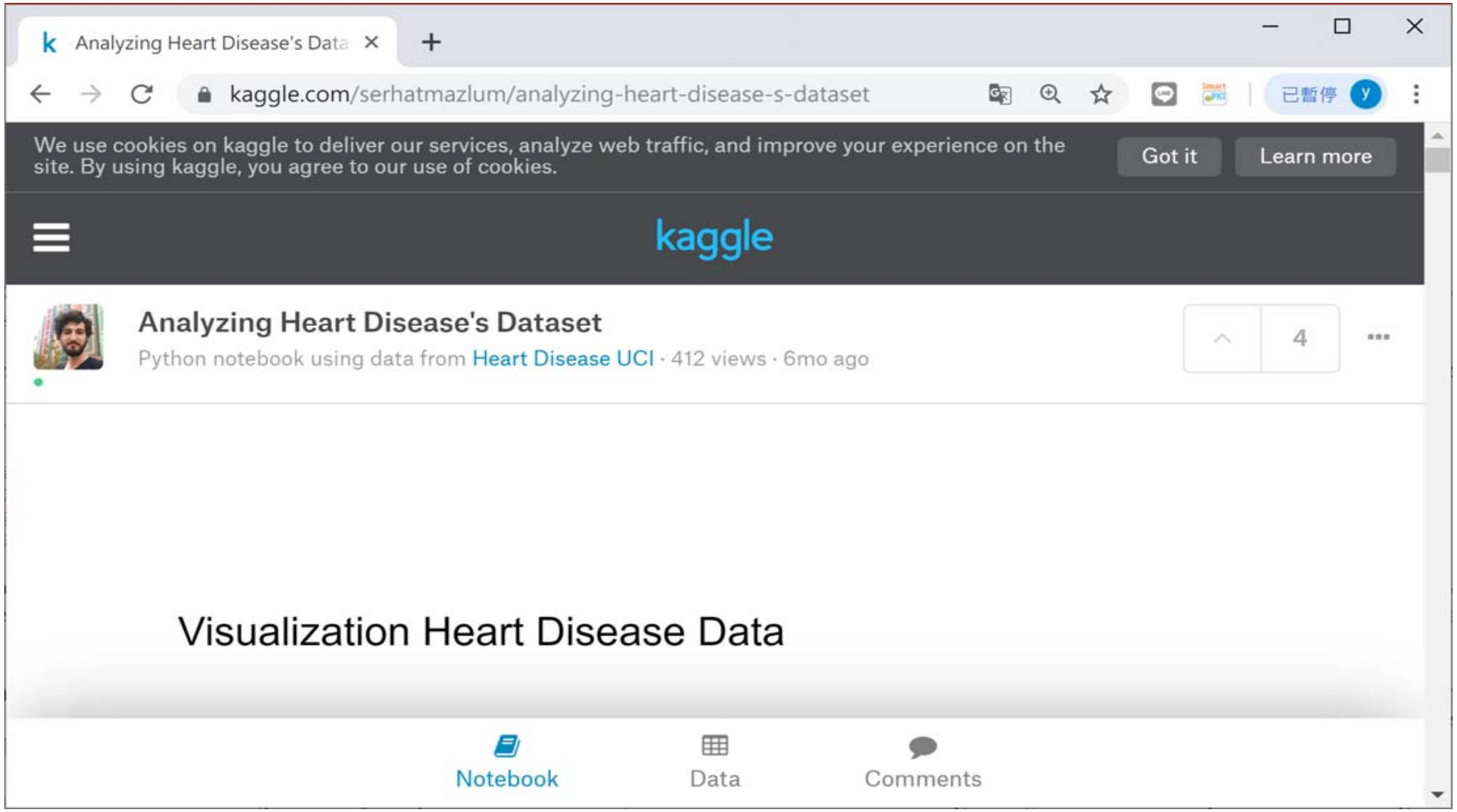
https://www.kaggle.com/anthonyhills/classifying-species-of-iris-flowers
https://www.kaggle.com/mjbahmani/20-ml-algorithms-15-plot-for-beginners
https://www.kaggle.com/nbbevan/neural-net-classification-on-iris-data

Breast Cancer Dataset



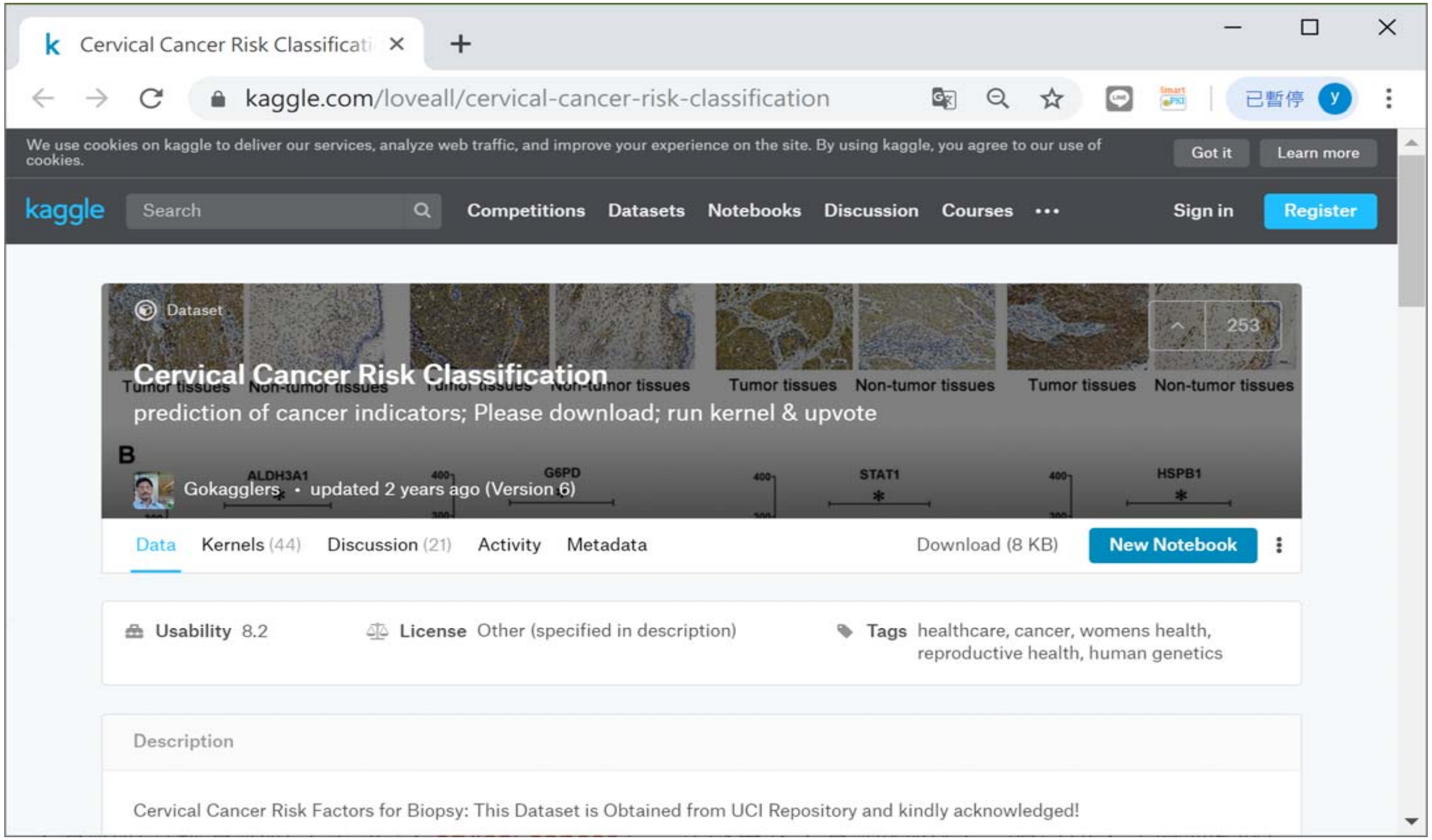
https://www.kaggle.com/buddhiniw/breast-cancer-prediction

Heart Disease Dataset



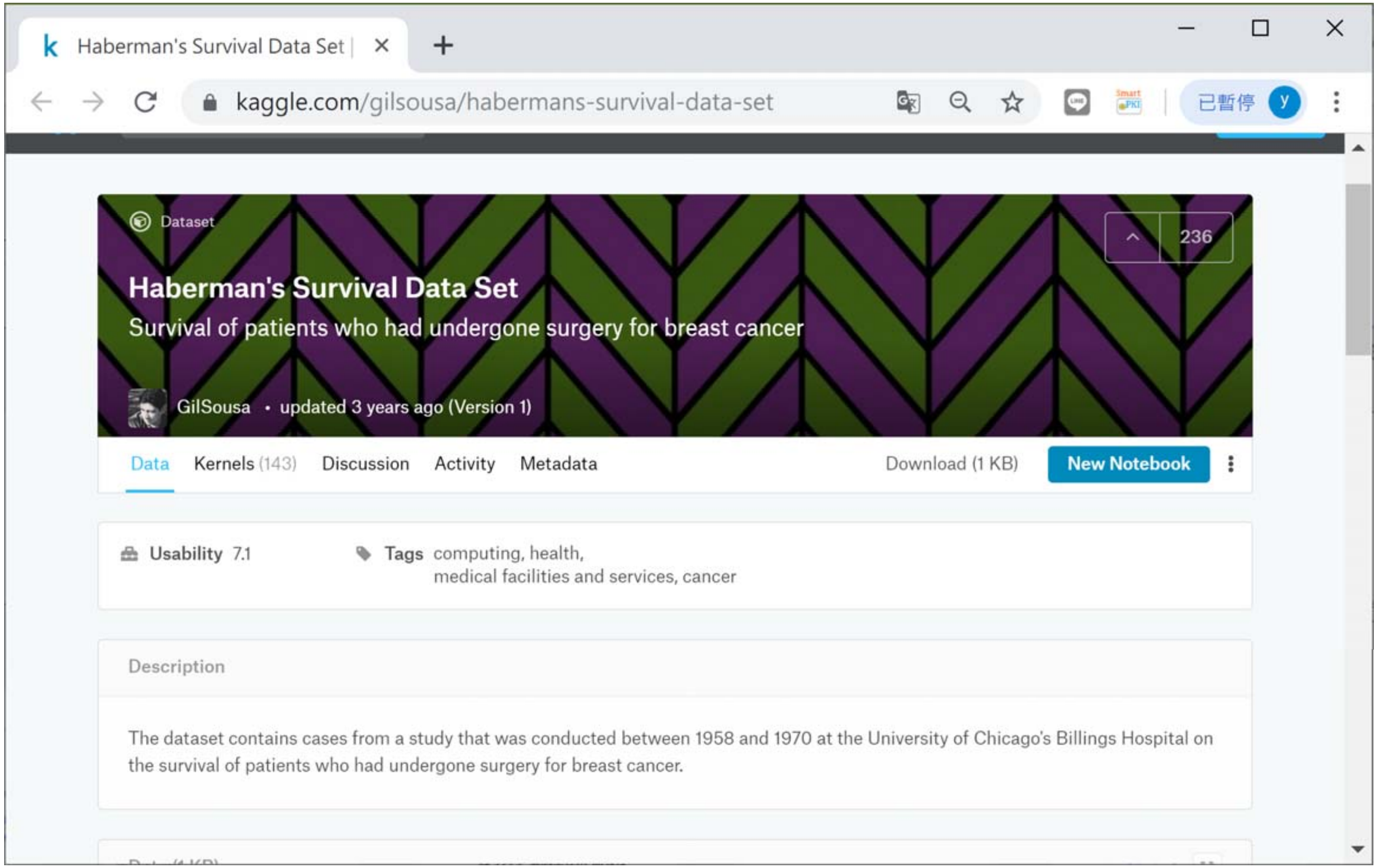
https://www.kaggle.com/serhatmazlum/analyzing-heart-disease-s-dataset

Cervical Cancer Risk Dataset



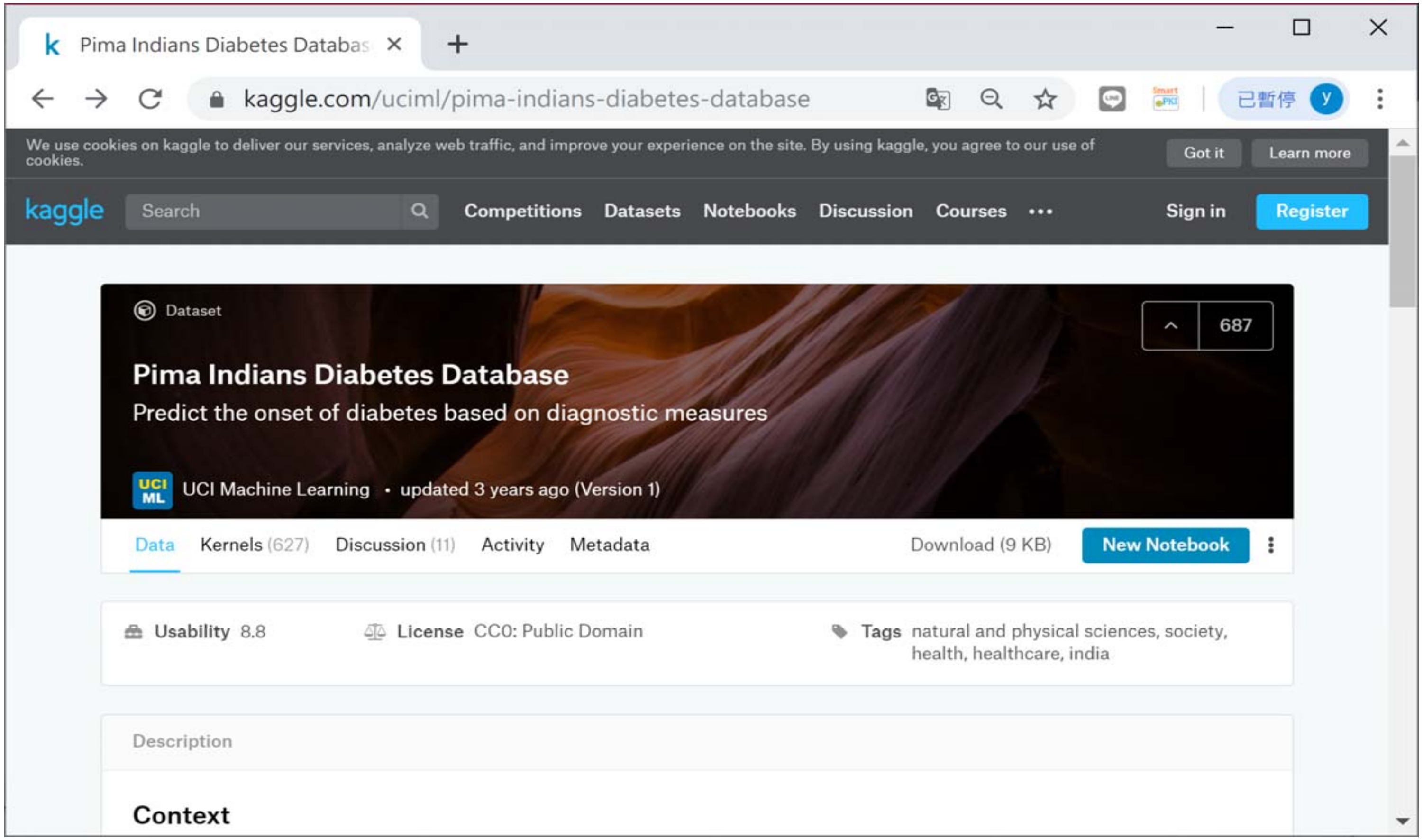
https://www.kaggle.com/loveall/cervical-cancer-risk-classification

Survival Dataset



https://www.kaggle.com/gilsousa/habermans-survival-data-set

Diabetes Database



https://www.kaggle.com/buddhiniw/breast-cancer-prediction

課後練習

- **IRIS Classification**
  - 微調模型與優化

作業二

- 使用 Breast Cancer Dataset 或 自行檢索的資料庫，實現Neural Networks。