



Penerapan **Data Science** Dalam Kehidupan **Nyata**

Achmad Benny Mutiara

Dekan Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma

SEKJEN-APTIKOM

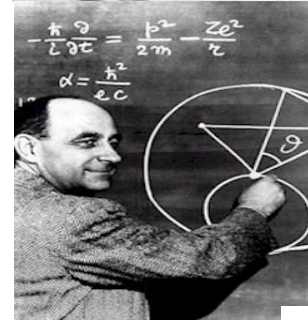
2020

Definisi **Data Science** dari **NIST** (2018).

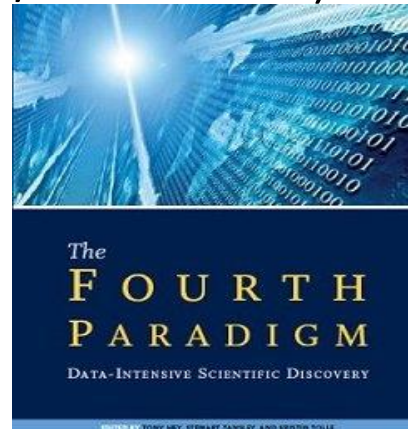
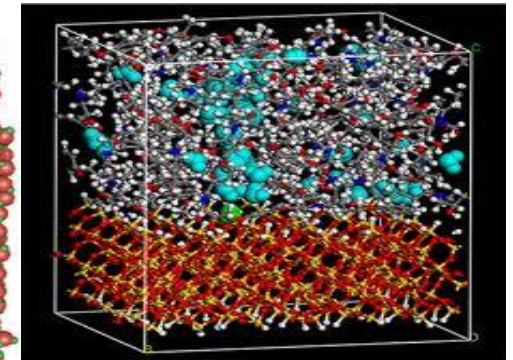
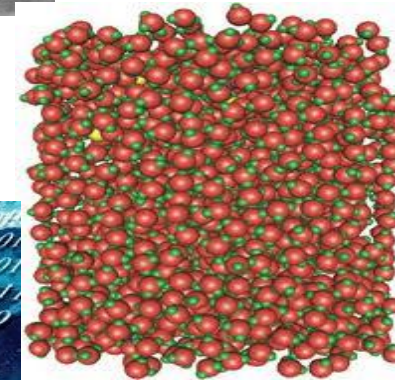
*Data science is the **extraction of useful knowledge** directly from data through a **process of discovery**, or of hypothesis formulation and hypothesis testing.*

Data Science : Evolusi Penemuan Saintifik Ke -4

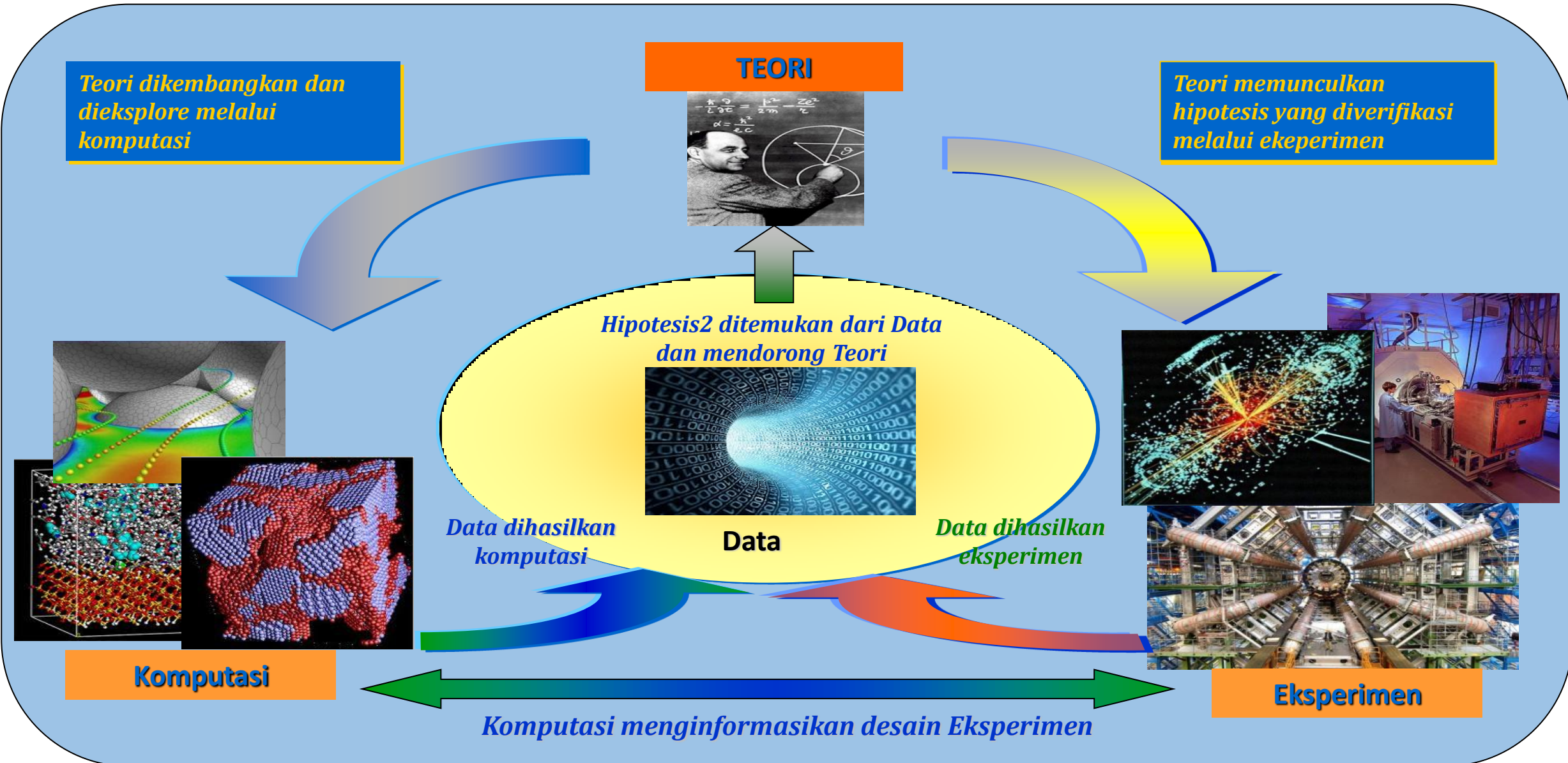
- Ribuan Tahun yang lalu:
 - Sains Empiris (eksperimen/observasi)
 - u/ mendeskripsikan fenomena alam
- Ratusan Tahun yang lalu:
 - Sains Teoritis
 - Mengembangkan model dan generalisasi
- Puluhan tahun yang lalu:
 - Sains Komputasi
 - Simulasi fenomena kompleks
- Saat ini:
 - Sains Data-intensive (Data Intensive Science/Data Science)
 - *Sintesis teori, eksperimen dan komputasi dengan manajemen dan statistik data “**advanced**” → **new algorithms***



$$H(t) |\psi(t)\rangle = i\hbar \frac{d}{dt} |\psi(t)\rangle$$



Metode Saintifik Abad 21



Sains Data Intensive (SDI)/Sains Data

Masalah-masalah dimana data menjadi faktor yang dominan

Laju Akuisisi
Volume
Kompleksitas
Ketidakpastian

Sains Komputasi Tradisional

- Komputasi memiliki lokalitas spasial dan temporal
- Masalah dimuat ke memori
- Metode memerlukan aritmatika presisi tinggi
- Datanya statis

Pemodelan dan Simulasi

Sains Data Intensive/Sains Data

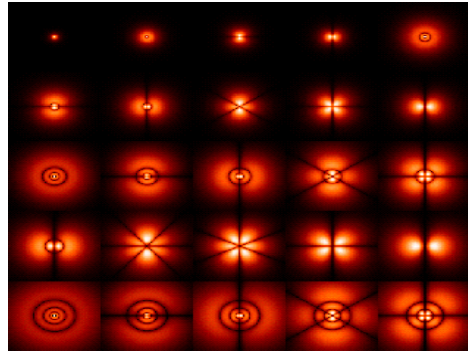
- Komputasi *tidak* atau *sedikit* memiliki lokalitas
- Masalah *tidak* dimuat ke memori
- Presisi atau pembulatan variabel berbasis aritmatika
- Datanya dinamis

Analisis

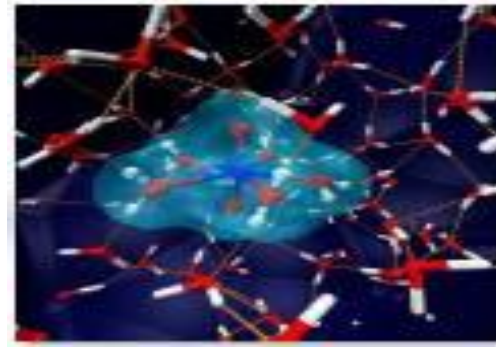


Pemodelan & Simulasi Data Intensive

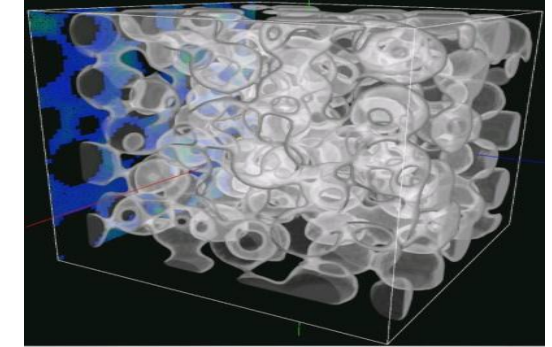
Hasil2 digali utk menemukan parameter2 bagi simlasi skala yg lebih tinggi



Kuantum



Molekular

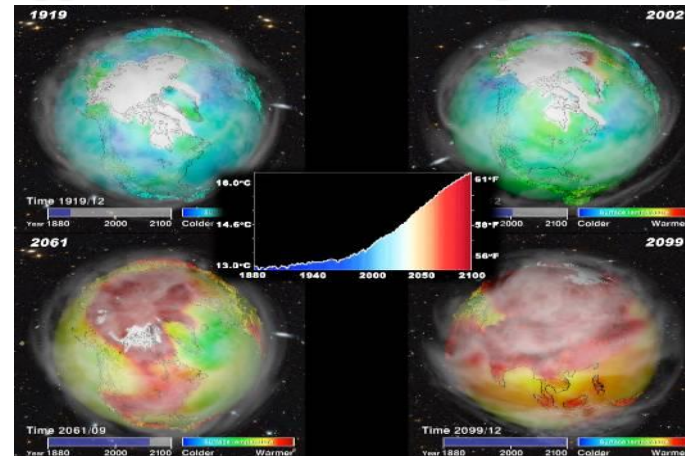


Kontinum

Data Instrument menggerakan/mendorong simulasi



Sensor2 Ruang Angkasa



Simulasi Iklim



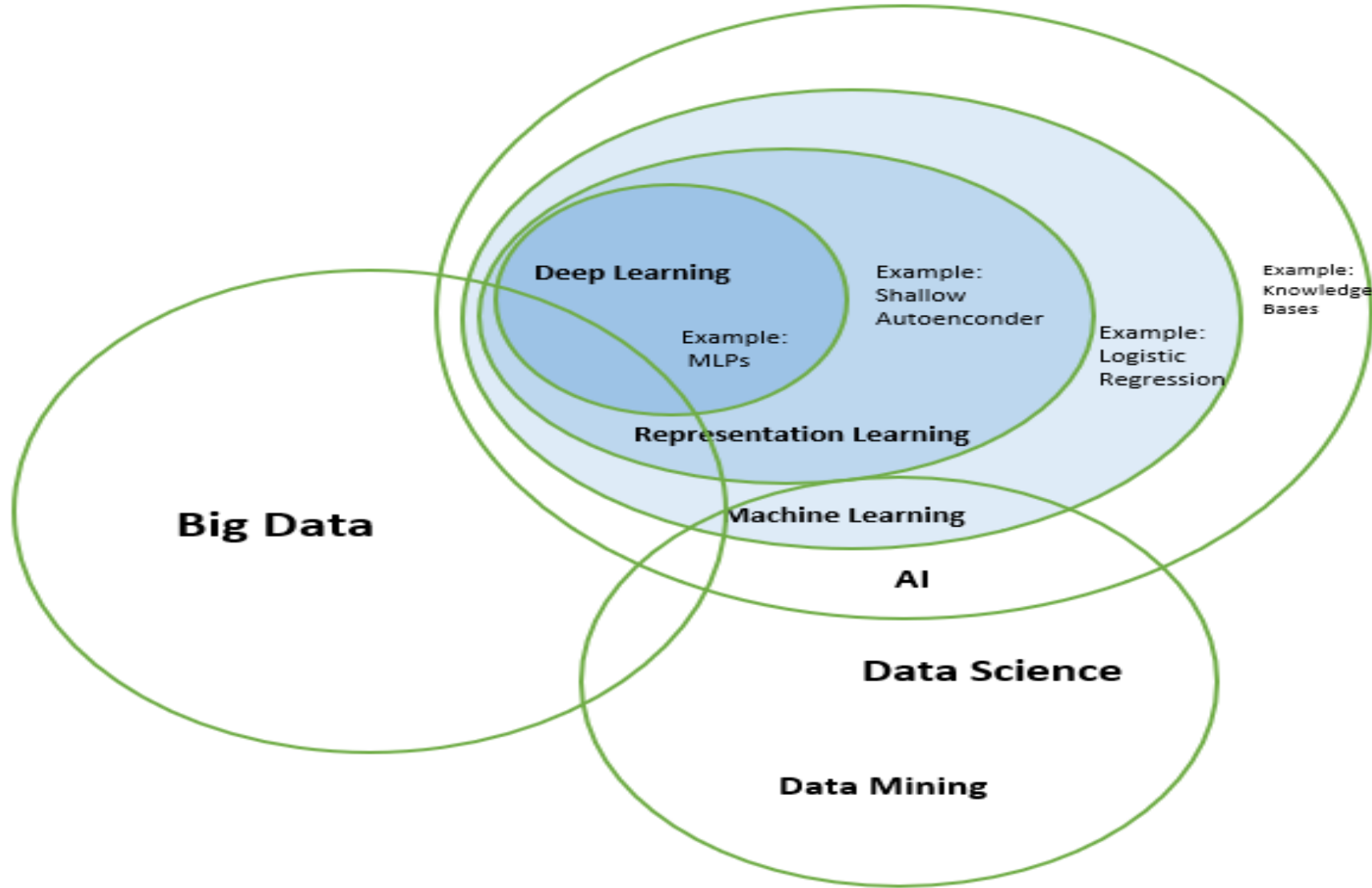
Sensor2 Bumi

Teknik dan Teknologi Pada SDI/SD

- SDI/SD memanfaatkan teknik ilmu komputer dan TIK
 - Sensor2 dan jaringan sensor
 - Jaringan Backbone
 - Databases
 - Data mining
 - Machine learning
 - Data visualization
 - Cluster/grid computing pada skala besar



Hubungan DS-BD-AI Dewasa ini



Source: adaptation from Ian Goodfellow, et.al 2016 & and Matthew Mayo, 2016

Contoh **Penerapan** Data Science Dalam Berbagai Bidang **Kehidupan**

Contoh Penerapan Data Science dalam Berbagai Bidang Kehidupan



e-Commerce



Manufacturing



Banking



Healthcare



Transport



Finance

Contoh Penerapan Data Science dalam Berbagai Bidang Kehidupan



E-Commerce

- Identifying Consumers
- Recommending Products
- Analyzing Review



MANUFACTURING

- Predicting Potential Problems
- Monitoring Systems
- Automating Manufacturing Units
- Maintenance Scheduling
- Anomaly Detection

Contoh Penerapan Data Science dalam Berbagai Bidang Kehidupan



Banking

- Fraud Detection
- Credit Risk Modeling
- Customer Lifetime Value



Healthcare

- Medical Image Analysis
- Drug Discovery
- Bioinformatics
- Virtual Assistants

Contoh Penerapan Data Science dalam Berbagai Bidang Kehidupan



Transport

- Self Driving Cars
- Enhanced Driving Experience
- Car Monitoring System
- Enhancing the Safety of Passengers

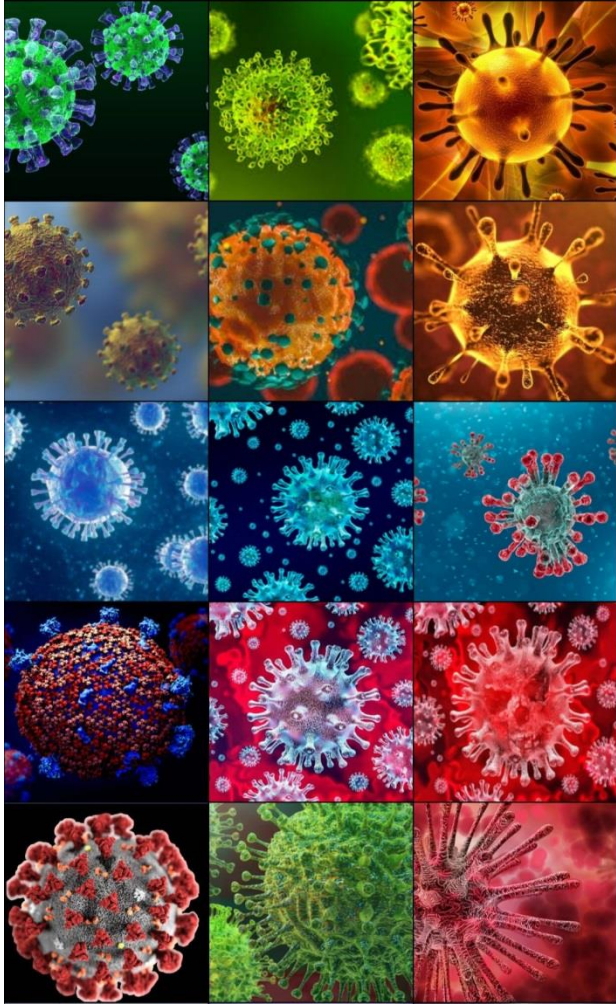


Finance

- Customer Segmentation
- Strategic Decision Making
- Algorithmic Trading
- Risk Analytics

Contoh **Penerapan** Data Science pada Penelitian- **COVID-19**

Data Science Pada Penelitian Pandemi Covid-19



From [kaggle.com](https://www.kaggle.com):

- COVID-19 data with SIR model; COVID Global Forecast: SIR model + ML regressions; COVID19 Global Forecasting: Forecast daily COVID-19 spread in regions around world
- Indonesia-Coronavirus: Data Science for COVID-19 Indonesia (DSCI) Initiative
- Logistic Model for Indonesia COVID-19
- CoronaHack -Chest X-Ray-Dataset Classify the X Ray image which is having Corona; COVID-19 Lung CT Scans: A CT Scan Dataset about COVID-19
- Covid-19 Predictions, Growth Factor, and Calculus
- Covid19 Population Feature Engineering
- Global Forecasting Covid-19 Random Forest
- Covid-19 Time Series Starter Code
- Bayesian Model for COVID-19 Spread Prediction
- CNN Model on COVID-19 images

From <https://ddi.sutd.edu.sg/>:

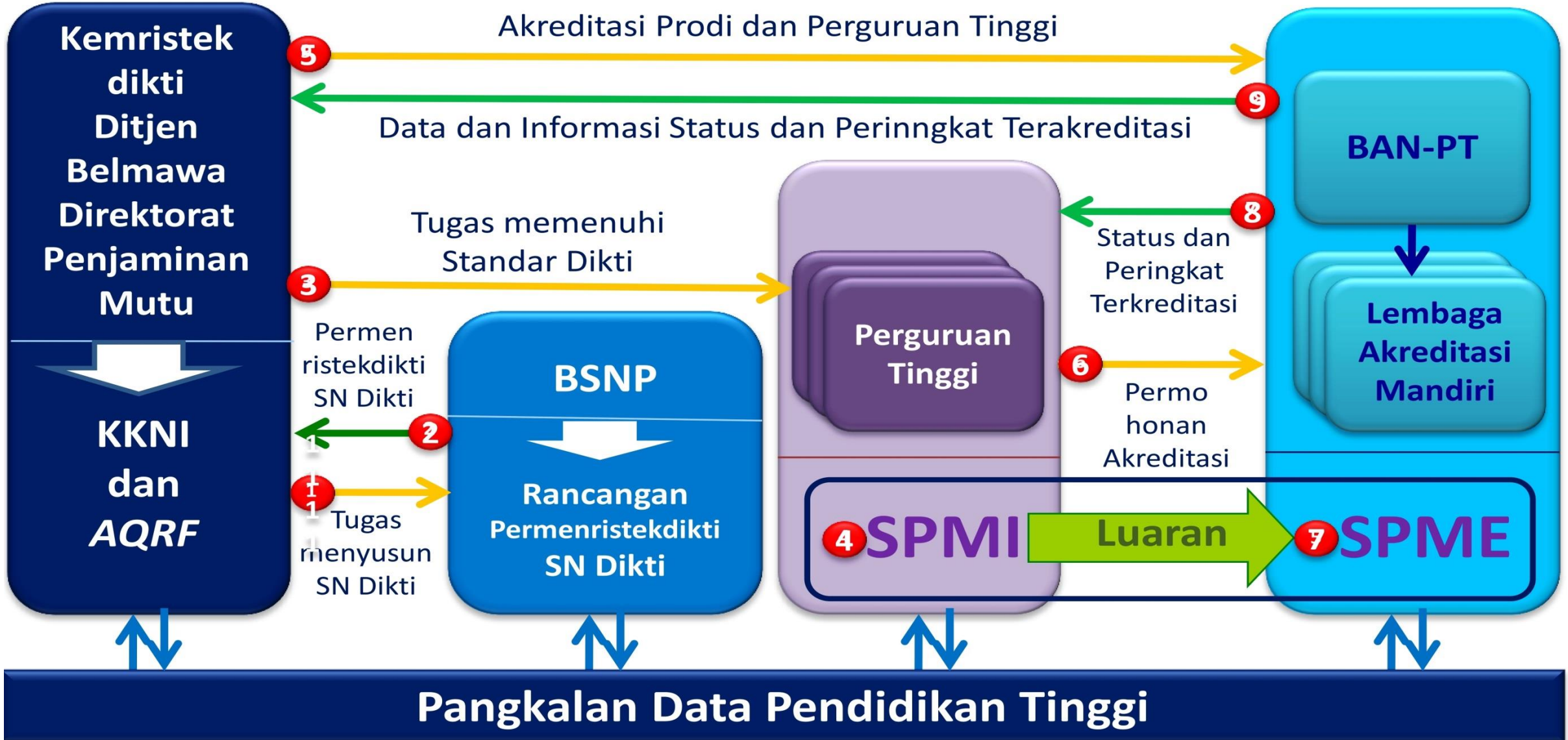
- **Predictive Monitoring of COVID-19**

Link-Link Data Science dalam Penelitian Pandemi Covid-19

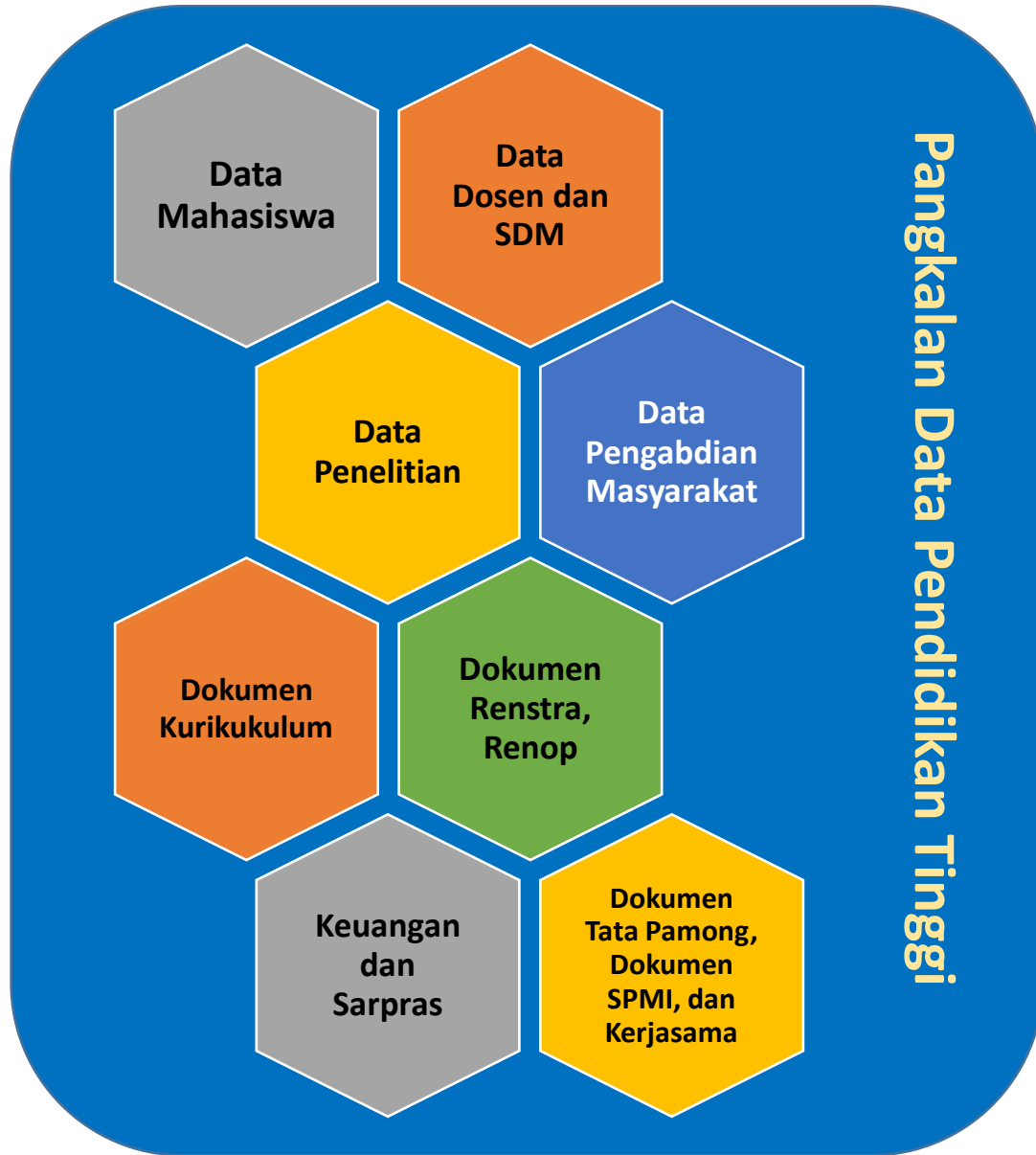
- **Penelitian Covid-19**, kunjungi
 - arXiv.org <http://arxiv.org/>
 - COVID-19 SARS-CoV-2 preprints from medRxiv and bioRxiv (<https://connect.biorxiv.org/relate/content/181>)
- **Metode/Ide Solusi**
 - KDnuggets (Machine Learning, Data Science, Big Data, Analytics, AI): <https://www.kdnuggets.com/>
 - Kaggle: Your Machine Learning and Data Science Community: <https://www.kaggle.com/>
 - The world's leading software development platform · GitHub: <https://github.com/>
 - Google AI: <https://ai.google/>
 - Facebook AI: <https://ai.facebook.com/>
- **Datasets**, kunjungi
 - Google Datasets search: <https://datasetsearch.research.google.com/>
 - Open Datasets : <https://pathmind.com/wiki/open-datasets>
 - Kaggle Datasets: <https://www.kaggle.com/datasets>

Model Penerapan Data Science dan Big Data Dalam Sistem Penjaminan Mutu PS

Proses Implementasi SPM Dikti



Tantangan Pengembangan Sistem Asesmen Prodi Otomatis

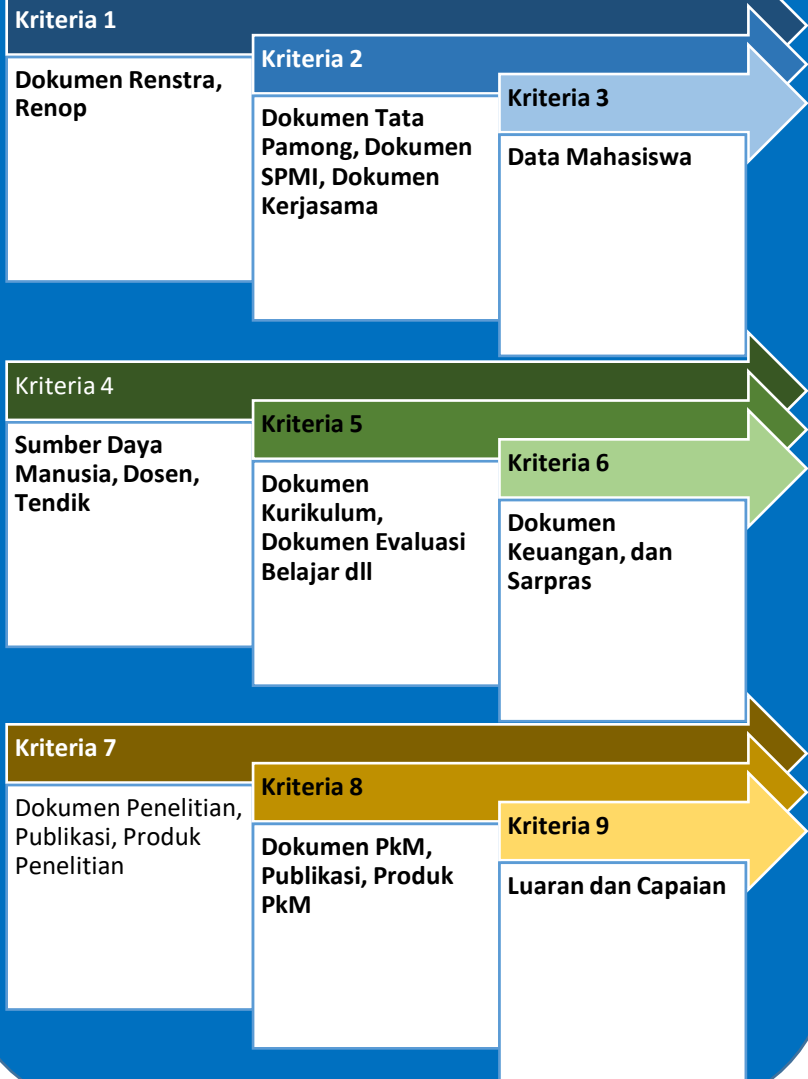


Pengembangan Sistem Asesmen Prodi Otomatis

- Prodi **tidak perlu menulis** Borang
- Mempercepat Proses Akreditasi
- Obyektifitas tinggi
- Mudah memonitor perkembangan Prodi dll.

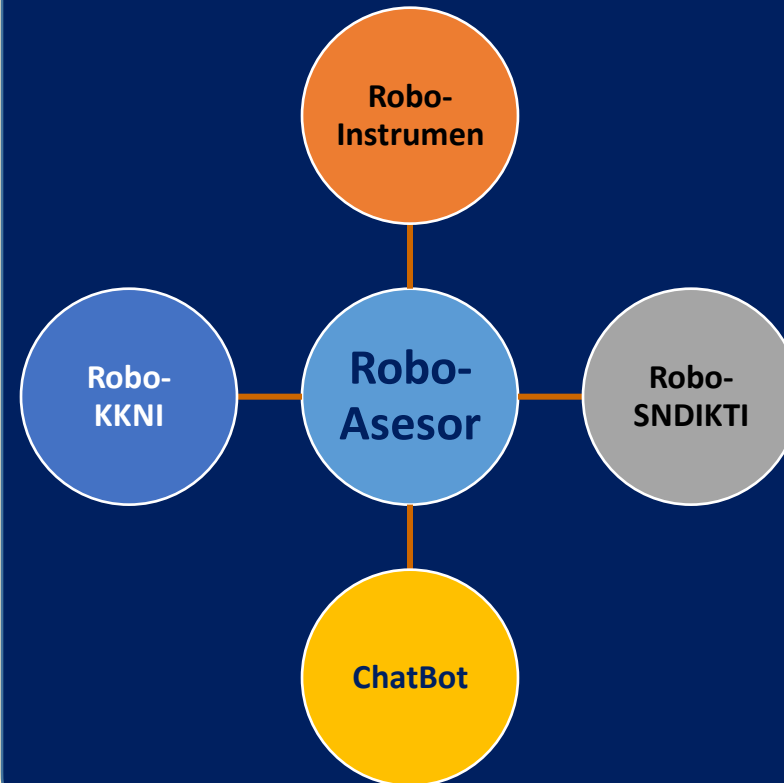
Model Sistem Asesmen Prodi INFOKOM Otomatis berbasis DS-AI-BD

Program Studi



PANGKALAN DATA PENDIDIKAN TINGGI

LAM-INFOKOM SISTEM ASESMEN OTOMATIS



LAM-INFOKOM

Hasil Asesmen Otomatis

- Peringkat Sementara Akreditasi
- Memuat Deskripsi Prodi
- Memuat Prediktif Prodi
- Memuat Preskripsi Prodi
- Saran Perbaikan

Asesmen Lapangan

- Asesor LAM

Develop Machine Learning using Python

Machine Learning

Machine Learning

- ▶ a machine learning-based system is not explicitly programmed but learnt from data.
- ▶ a machine learning algorithm infers patterns and relationships between different variables in a dataset then uses that knowledge to generalize beyond the training dataset.

Terms used in the context of machine learning

1. Data mining
2. Features
3. Labels
4. Models
5. Accuracy, and Precision

Data Mining

- Definition: "the application of specific algorithms for extracting patterns from data."
- **data mining** is a **process**, during which **machine learning** algorithms are utilized as **tools** to extract potentially-valuable patterns held within datasets.
- Data mining is exploratory analysis with unsupervised machine learning algorithms

The main application is text mining in a big data environment:

1. Text parsing
2. Sentiment analysis
3. Opinion mining
4. Natural language processing using *deep learning algorithm

* deep learning is the process of applying deep neural network technologies - that is, neural network architectures with multiple hidden layers - to solve problems.

Deep learning is a process, like data mining, which employs deep neural network architectures, which are particular types of machine learning algorithms.

Data Mining algorithms:

1. C4.5
2. k-means
3. Support vector machines
4. Apriori
5. EM
6. PageRank
7. AdaBoost
8. kNN
9. Naive Bayes
10. CART

Features

- a feature represents an independent variable
- In a tabular dataset, a row represents an observation and column represents a feature.
- Features are also collectively referred to as dimensions.

Categorical Features

- It can take on one of a fixed number of discrete values with name or a label.
- The values of a categorical feature have no ordering.
- gender is a categorical feature. It can take on only one of two values.

Numerical Features

- It can take on any numerical value
- numerical feature have mathematical ordering
- discrete numerical feature : number of persons, number of rooms
- continuous numerical feature: temperature value

Labels

- label is a variable that a machine learning learns to predict
- categorical label: ex, category of a news article is a categorical label
- numerical label: ex, price is a numerical label

← Features →					Label
Position	Experience	Skill	Country	City	Salary (\$)
Developer	0	1	USA	New York	103100
Developer	1	1	USA	New York	104900
Developer	2	1	USA	New York	106800
Developer	3	1	USA	New York	108700
Developer	4	1	USA	New York	110400
Developer	5	1	USA	New York	112300
Developer	6	1	USA	New York	114200
Developer	7	1	USA	New York	116100
Developer	8	1	USA	New York	117800
Developer	9	1	USA	New York	119700
Developer	10	1	USA	New York	121600

Models

- a mathematical construct for capturing patterns within a dataset and estimates the relationship between the dependent and independent variables and has predictive capability. and can calculate or predict the value for the dependent variable when get the values of the independent variables.
- Training a model is a compute intensive task, while using it is not as compute intensive.
- A model is generally saved to disk, so that it can be used without go to training step again.

Training Data (80% of data)

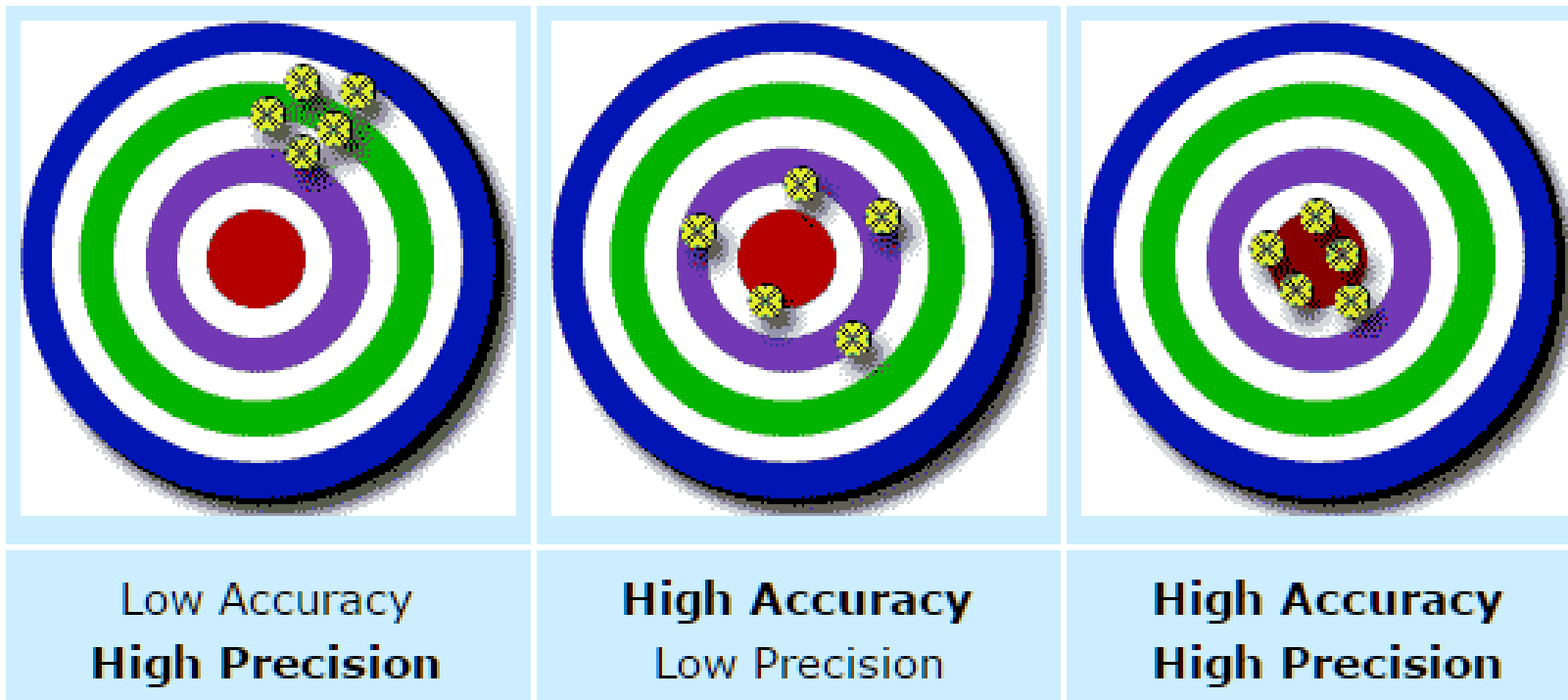
- The data used by a machine learning algorithm to train a model

Test Data (20% of data)

- The data used for evaluating the predictive performance of a model

Accuracy vs Precision

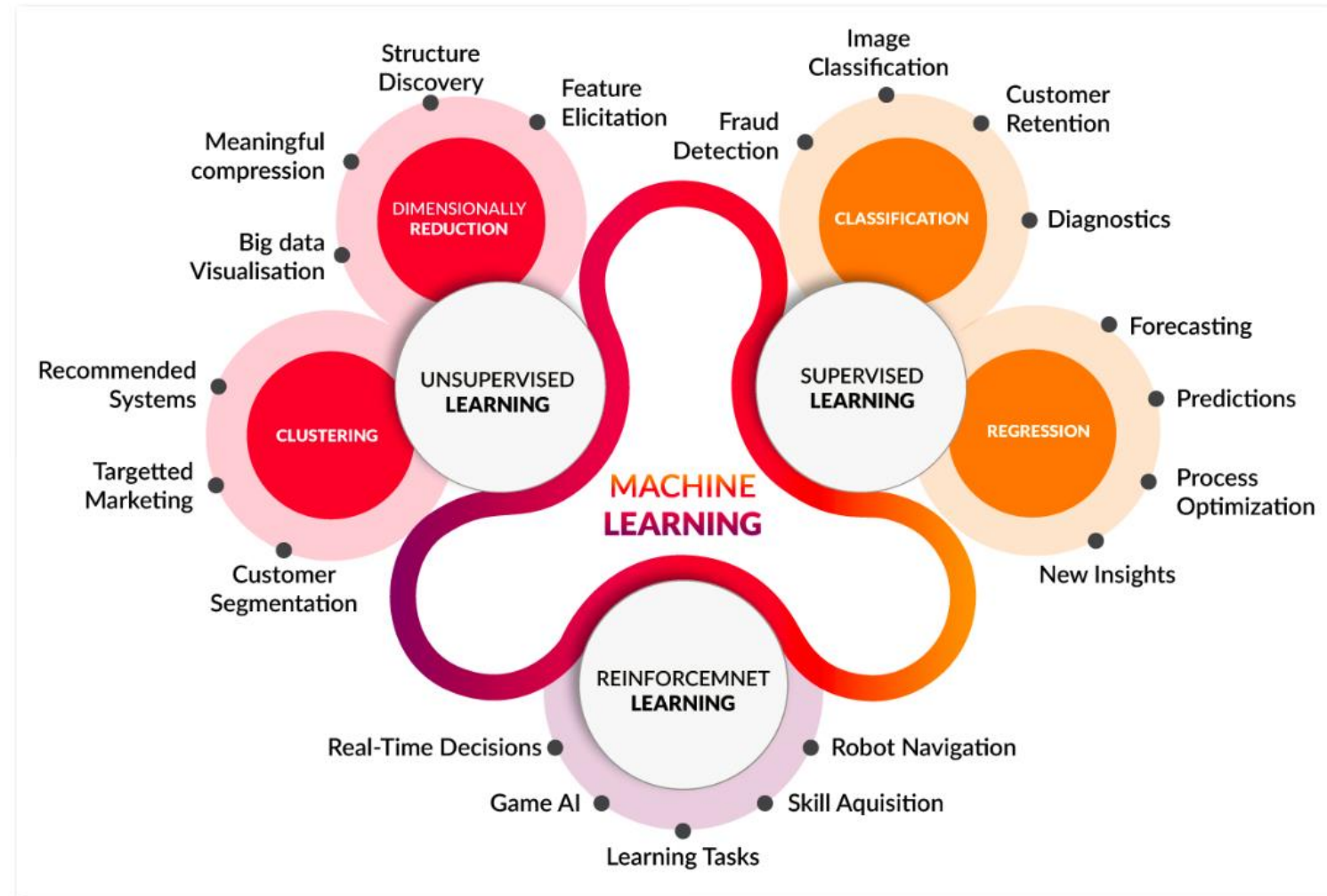
- ▶ **Accuracy** is how close a measured value is to the actual (true) value.
- ▶ **Precision** is how close the measured values are to each other.



Machine learning techniques

Machine learning mainly has three types of learning techniques:

- Supervised learning
- Unsupervised learning
- Reinforcement learning



Machine Learning tasks categories

Supervised Learning

Unsupervised Learning

Reinforcement Learning

1. Classification
2. Regression
3. Clustering
4. Anomaly detection
5. Association
6. Recommendation
7. Dimensionality reduction
8. Computer Vision
9. Text Analytics

Classification [supervised learning]

- Classification is concerned with building models that separate data into distinct classes. These models are built by inputting a set of training data for which the classes are pre-labelled in order for the algorithm to learn from. The model is then used by inputting a different dataset for which the classes are withheld, allowing the model to predict their class membership based on what it has learned from the training set.

Binary classification examples (divide data to two options only)

- spam filtering is a classification task
- Tumor diagnosis can be treated as a classification problem.
- determining credit risk using personal information such as income, outstanding debt

Multi-class classification examples

- handwritten recognition each character is a multi-class classification problem
- image recognition is a multi-class classification task
- Xbox Kinect360, which infers body parts and position

Regression [supervised learning]

- The goal is to predict a numerical label for an unlabeled observation

Regression algorithms: Linear regression, Decision trees

Examples

- home valuation
- Asset trading, and forecasting
- Sales or inventory forecasting

← Features →					Label
Position	Experience	Skill	Country	City	Salary (\$)
Developer	0	1	USA	New York	103100
Developer	1	1	USA	New York	104900
Developer	2	1	USA	New York	106800
Developer	3	1	USA	New York	108700
Developer	4	1	USA	New York	110400
Developer	5	1	USA	New York	112300
Developer	6	1	USA	New York	114200
Developer	7	1	USA	New York	116100
Developer	8	1	USA	New York	117800
Developer	9	1	USA	New York	119700
Developer	10	1	USA	New York	121600

Anomaly Detection [supervised learning]

the goal is to find outliers, noise, deviations in a dataset

Anomaly detection applications

- In manufacturing, it is used for automatically finding defective products.
- In data centers, it is used for detecting bad systems.
- Websites use it for fraud detection.
- Detecting security attacks. Network traffic associated with a security attack is unlike normal network traffic. Similarly, hacker activity on a machine will be different from a normal user activity.

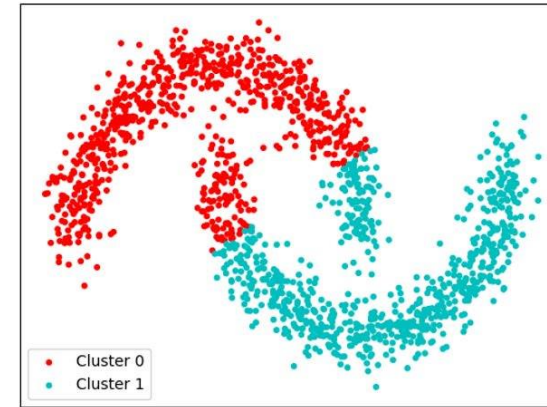
Clustering [unsupervised learning]

- The aim is to split a dataset into a specified number of clusters or segments.
- Elements in the same cluster are more similar to each other than to those in other clusters.

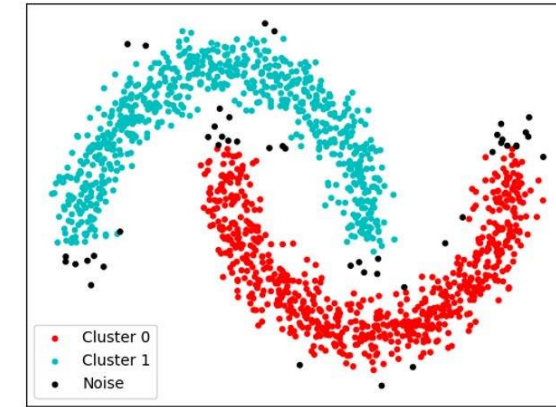
Clustering algorithms

1) k-means

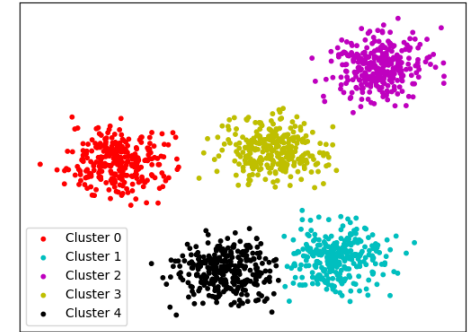
- The number of clusters (k) must be given explicitly.
- Identify the best k cluster centers in an iterative manner
- Clusters are assumed to be spherical.



K-means



OPTICS



2) OPTICS /DBSCAN

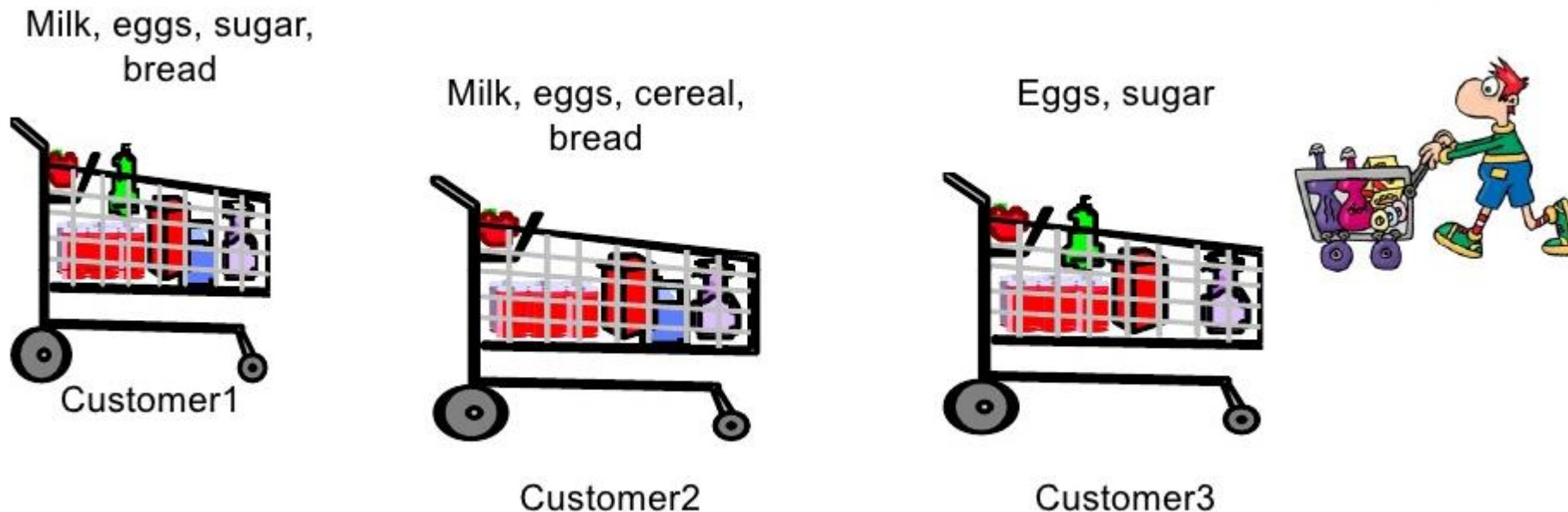
- it is a density-based clustering algorithm. represents clusters by its nature

Example

- creating customer segments, which can be targeted with different marketing programs

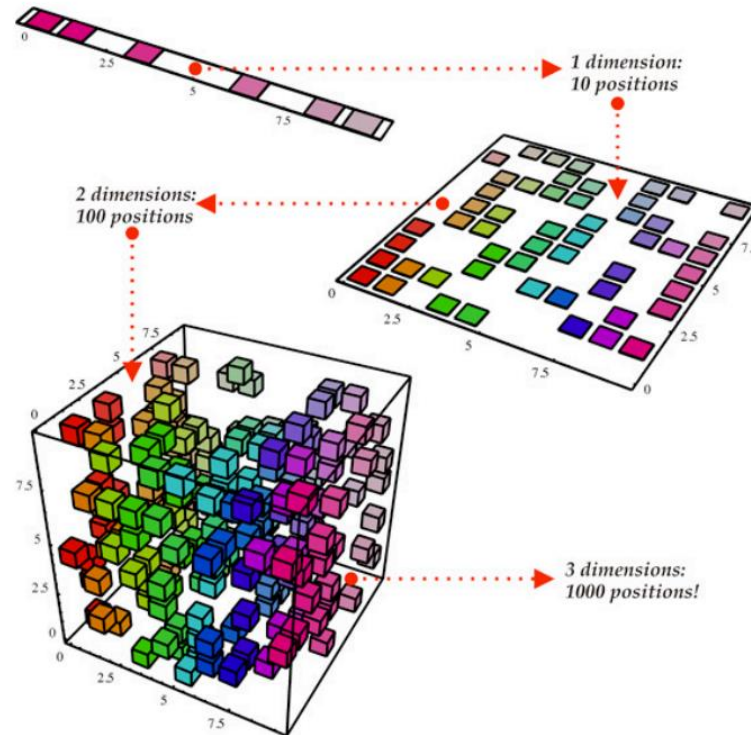
Association [unsupervised learning]

- Association is most easily explained by introducing market basket analysis, a typical task for which it is well-known.
- attempts to identify associations between the various items that have been chosen by a particular shopper and placed in their market basket and assigns support and confidence measures for comparison.
- The value of this lies in cross-marketing and customer behavior analysis.



Dimensionality Reduction [unsupervised learning]

- The goal in dimensionality reduction is to reduce the number of features in a dataset without significantly impacting the predictive performance of a model and this will reduce the computational complexity and cost of machine learning.



Recommendation [Reinforcement]

- The goal of a recommendation system is to recommend a product to a user based on past behavior to determine user preferences.
- Unlike Association, Recommendation focus on user behavior and suggest according to this user behavior.
- It is reinforcement because we not sure of result until user choose one of our recommendation, if not, then our recommendation was not correct.

Supervised machine learning algorithms

Classification

- **Two Class Classification**

- Logistic Regression (Fast)
- Decision Tree (Fast)
- Decision jungle(Accurate)
- SVM (Accurate) (>100 features)
- Boosted Decision Tree (Fast - Large memory)
- Bayes point machine (Fast)

- **Multi Class Classification**

- Decision Tree (Fast)
- Logistic Regression (Fast)
- Random Forest (Accurate)
- Gradient Boosting Tree (Accurate)
- Naive Bayes (Big Data)
- Decision jungle(Accurate)

Regression

- Linear Regression (Fast)
- Decision Tree (Fast)
- Random Forest (Accurate)
- Gradient Boosting Tree (Accurate)
- Ordinal regression
- Bayesian linear regression
- Boosted Decision Tree (Fast - Large memory)
- SGD Regressor (<100K rows)
- Lasso/ ElasticNet (Few Features)
- RidgeRegression
- SVR(kernel=linear/ rbf)
- EnsembleRegressors

Anomaly Detection

- One Class SVM (support vector machine)
- PCA based Anomaly Detection
- Time Series Anomaly Detection

Unsupervised machine learning algorithms

Clustering

- K-means Clustering
- K-modes (Categorical variables)
- DBScan (predict the number of clusters automatically)
- OPTICS (predict the number of clusters automatically)

Association

Apriori

Dimension Reduction

- PCA
- Singular value decomposition

Recommendation

- Matchbox Recommender

Computer Vision

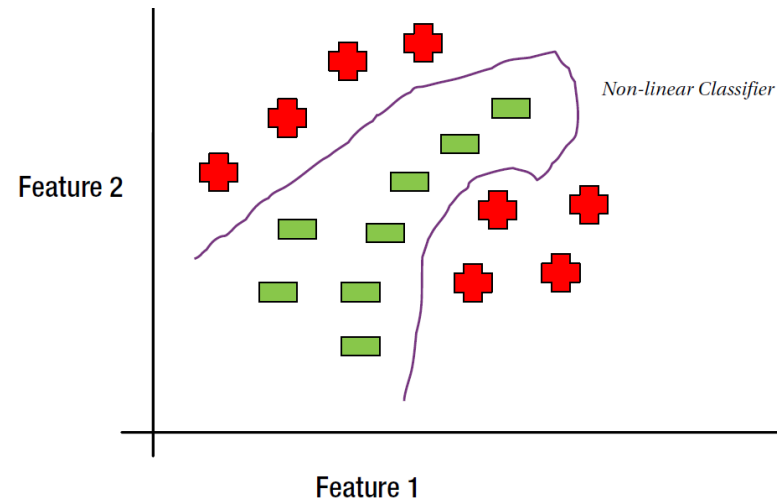
- OpenCV Library

Text Analytics (Supervised Learning)

- Named Entity Recognition
- Sentimental Analysis

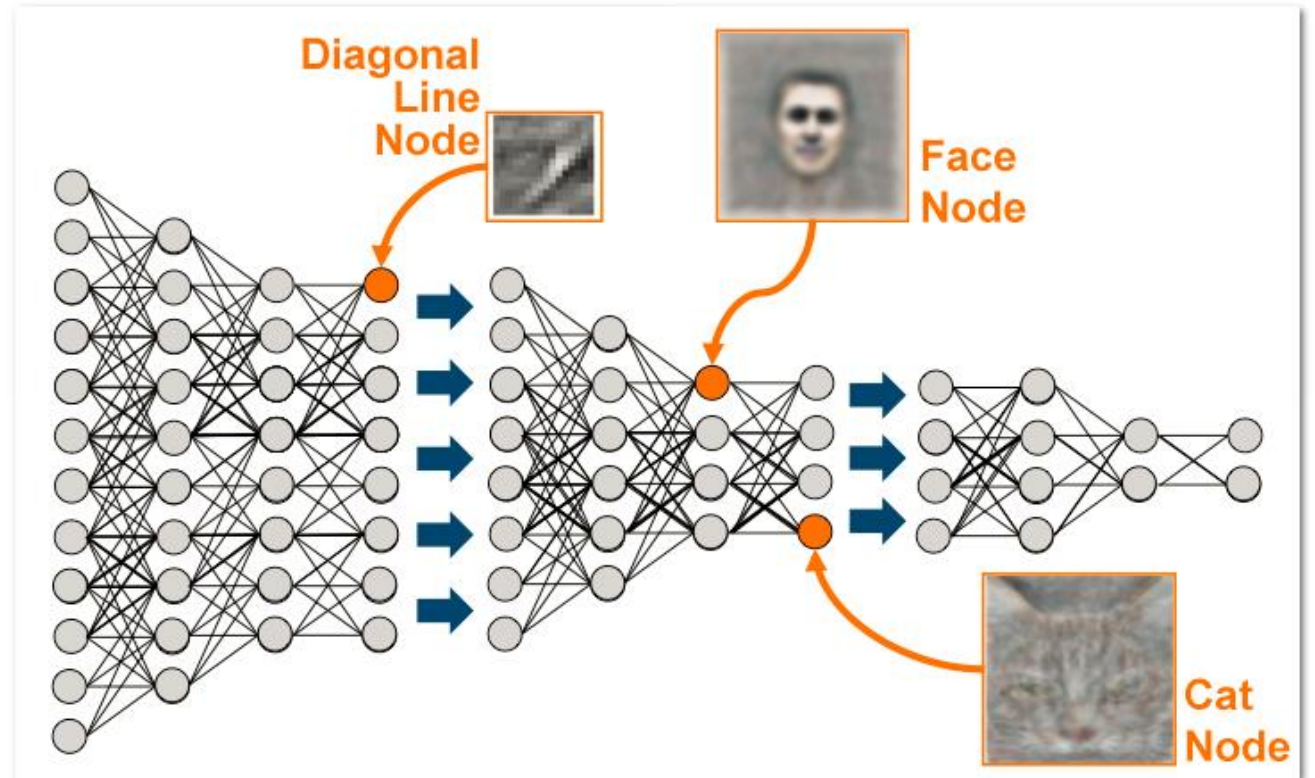
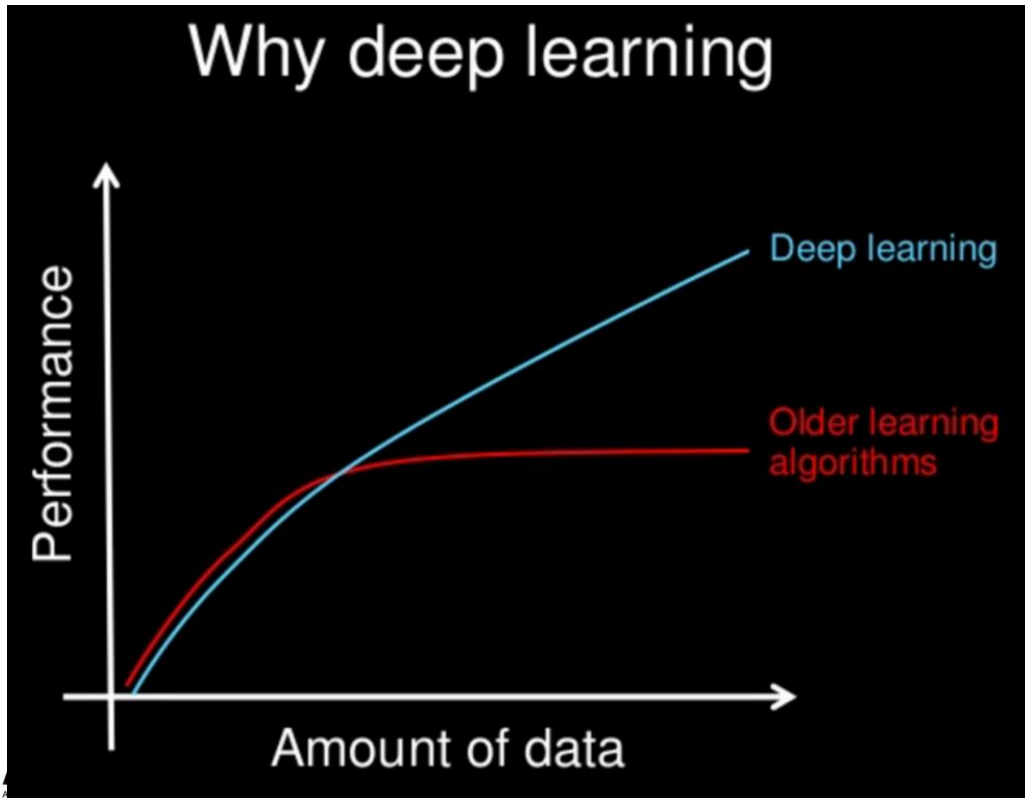
Neural Network

- The feedforward neural network algorithm uses a technique known as backpropagation to train a model. During the training phase, prediction errors are fed back to the network. The algorithm uses this information for adjusting the weights of the edges connecting the nodes to minimize prediction errors. This process is repeated until the prediction errors converge to value less than a predefined threshold. Generally, a neural network with one layer is sufficient in most cases. If more than one hidden layers are used, it is recommended to have the same number of nodes in each hidden layer. Neural networks are better suited for classifying data that is not linearly separable.



Deep learning

- It consists of many layers/nodes of neural networks.
- Deep learning automatically finds out the features which are important for classification, but in Machine Learning we had to manually give the features
- deep learning algorithms need a large amount of data to work perfect.

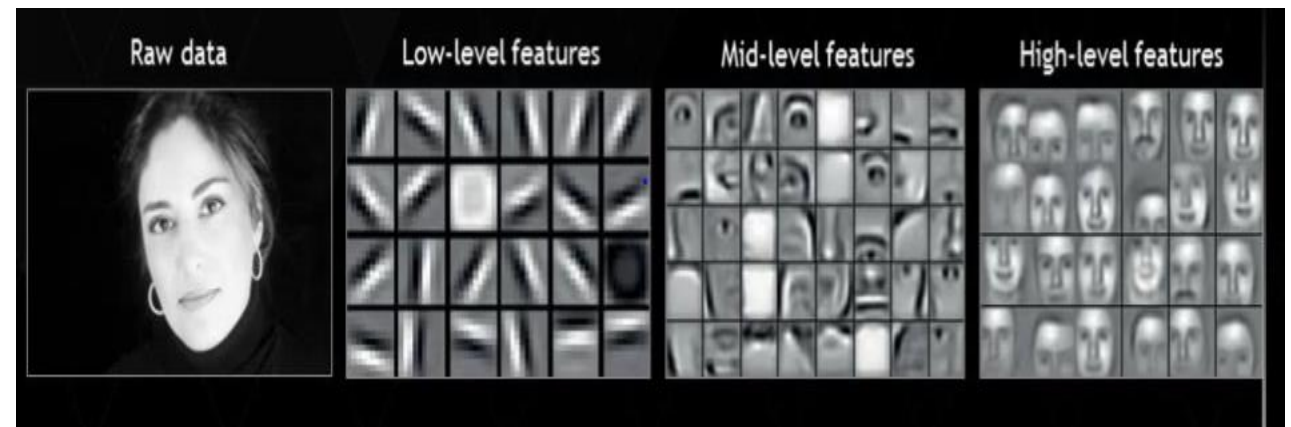


Traditional ML Algorithms

- can work on low-end machines
- break the problem down into different parts, solve them individually and combine them to get the result.
- takes much less time to train, ranging from a few seconds to a few hours.
- gives us the selection rules, so it is easy to interpret, safe to use in industry for interpretability.

Deep learning algorithms

- need a large amount of data to work perfect
- heavily depend on GPUs
- solve the problem end-to-end.
- takes a long time to train. (like two weeks)
- It is excellent and is near human performance, but no guarantee to be always like this!, because most of operations are hidden and we can't review selection logic!



No Free Lunch

- In machine learning, there's something called the “No Free Lunch” theorem. In a nutshell, it states that no one algorithm works best for every problem.
- As a result, one should **try many different algorithms for the problem**, while using a hold-out “test set” of data to evaluate performance and select the winner.

Python Machine learning

Why Python?

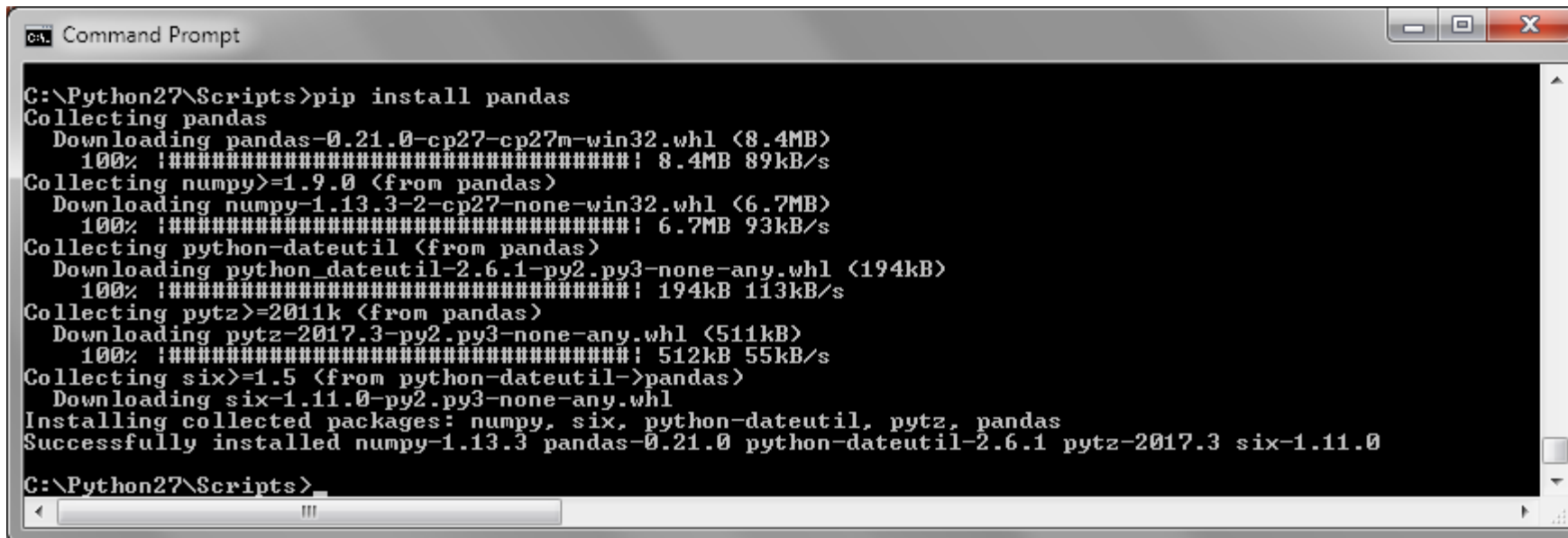
- A large community
- tons of machine learning specific libraries
- easy to learn
- **TensorFlow** make Python the **leading language** in the data science community.

About Python

- It is case sensitive
- Text indentation is important in logic flow!
- Use **#** symbol to add a code comment
- Use **""" """** to comment a block of code

Prepare your machine

- Install **Python 3.7.xx**
- Pip install pandas
- Pip install matplotlib Sklearn Statistics scipy seaborn lpython
pip install --no-cache-dir pystan #to force redownload exist package in case of error
- Get Data set for training through
<http://archive.ics.uci.edu/ml/datasets.html>

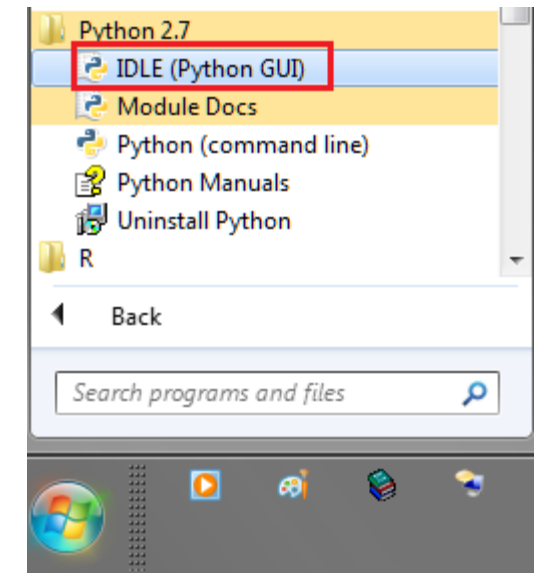


```

C:\Python27\Scripts>pip install pandas
Collecting pandas
  Downloading pandas-0.21.0-cp27-cp27m-win32.whl (8.4MB)
    100% |#####| 8.4MB 89kB/s
Collecting numpy>=1.9.0 (from pandas)
  Downloading numpy-1.13.3-2-cp27-none-win32.whl (6.7MB)
    100% |#####| 6.7MB 93kB/s
Collecting python-dateutil (from pandas)
  Downloading python_dateutil-2.6.1-py2.py3-none-any.whl (194kB)
    100% |#####| 194kB 113kB/s
Collecting pytz>=2011k (from pandas)
  Downloading pytz-2017.3-py2.py3-none-any.whl (511kB)
    100% |#####| 512kB 55kB/s
Collecting six>=1.5 (from python-dateutil->pandas)
  Downloading six-1.11.0-py2.py3-none-any.whl
Installing collected packages: numpy, six, python-dateutil, pytz, pandas
Successfully installed numpy-1.13.3 pandas-0.21.0 python-dateutil-2.6.1 pytz-2017.3 six-1.11.0

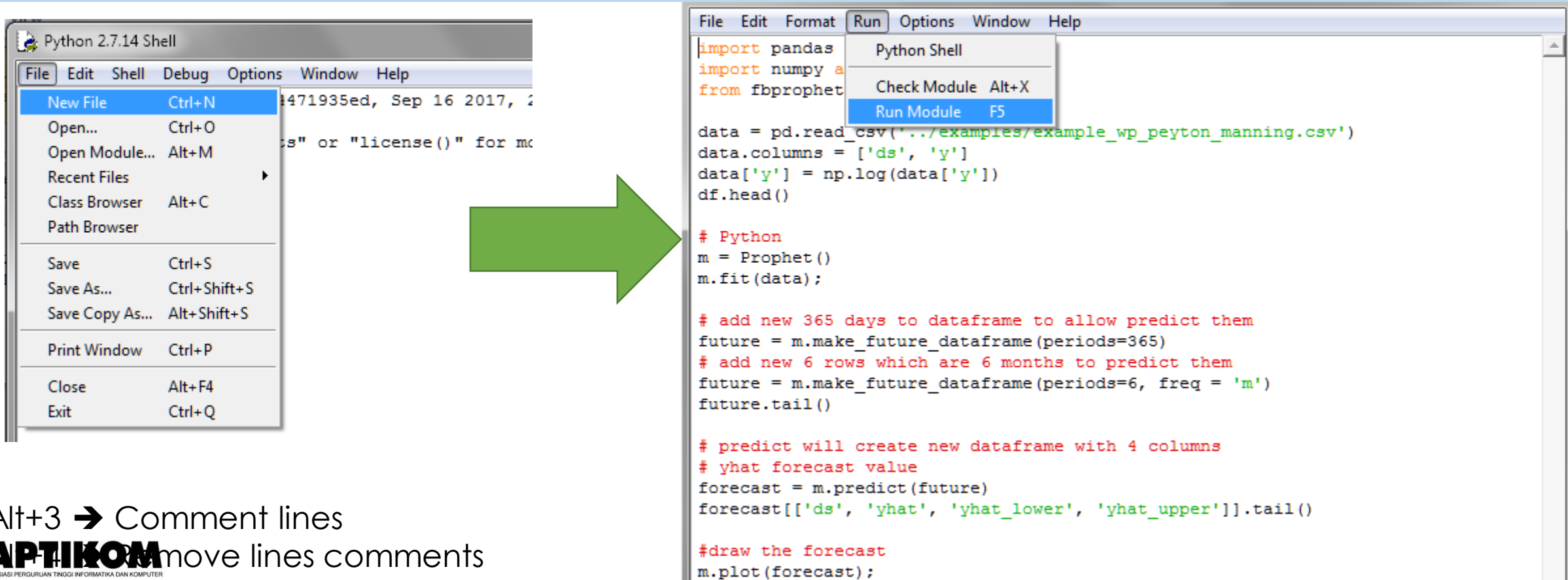
C:\Python27\Scripts>_

```



How to Run Python ML application

- Open “IDLE (Python GUI)” → File → New File
- Write your code on the new window then → Run → Run Module



The image shows two screenshots of the Python IDLE environment. The left screenshot shows the 'File' menu with 'New File' highlighted. The right screenshot shows the 'Run' menu with 'Run Module' highlighted. A large green arrow points from the 'File' menu to the 'Run' menu. Below the screenshots, there is a text instruction: 'Alt+3 → Comment lines' and 'move lines comments'. The bottom left corner features the APTIKOM logo and the text 'ASOSIASI PERGURUAN TINGGI INFORMATIKA DAN KOMPUTER'.

```
File Edit Shell Debug Options Window Help
New File Ctrl+N
Open... Ctrl+O
Open Module... Alt+M
Recent Files
Class Browser Alt+C
Path Browser

Save Ctrl+S
Save As... Ctrl+Shift+S
Save Copy As... Alt+Shift+S

Print Window Ctrl+P

Close Alt+F4
Exit Ctrl+Q
```

```
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5

import pandas
import numpy as np
from fbprophet

data = pd.read_csv('../examples/example_wp_pegton_manning.csv')
data.columns = ['ds', 'y']
data['y'] = np.log(data['y'])
df.head()

# Python
m = Prophet()
m.fit(data);

# add new 365 days to dataframe to allow predict them
future = m.make_future_dataframe(periods=365)
# add new 6 rows which are 6 months to predict them
future = m.make_future_dataframe(periods=6, freq = 'm')
future.tail()

# predict will create new dataframe with 4 columns
# yhat forecast value
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()

#draw the forecast
m.plot(forecast);
```

Alt+3 → Comment lines
move lines comments

APTIKOM
ASOSIASI PERGURUAN TINGGI INFORMATIKA DAN KOMPUTER

Anaconda

- Anaconda is Python distribution for data scientists .
- It has around 270 packages including the most important ones for most scientific applications, data analysis, and machine learning such as NumPy, SciPy, Pandas, IPython, matplotlib, and scikit-learn.
- Download it for free from <https://www.continuum.io/downloads>

Import Statistical libraries

```
import time
```

```
import random
```

```
import datetime
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import statistics
```

```
from scipy import stats
```

```
import sklearn
```

```
import seaborn
```

```
from IPython.display import Image
```

```
import numpy as np
```

COMPUTER VISION

Open CV for computer vision

- To install OpenCV `pip install opencv-python`
- Normal images are 4 dimensional array for each pixel (Blue ,Green, Red, and Alpha)
- It is normal to process images in Gray scale for fast performance
- Video is just a loop of images. So, any code processing images can process video too

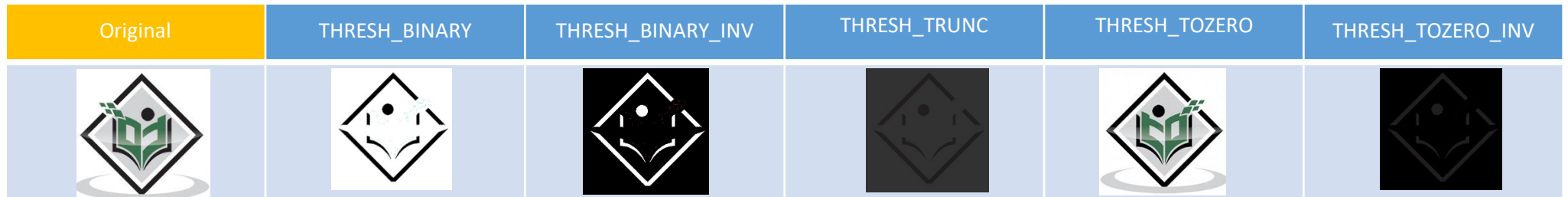
For more information about open CV please visit

- <https://www.tutorialspoint.com/opencv/index.htm>
- <https://www.youtube.com/playlist?list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqIQq>

Common OpenCV functions

- `image = cv2.imread(ImgPATH)` #Read image
- `cv2.cvtColor(ImgPATH, cv2.COLOR_BGR2GRAY)` #Convert to gray scale
- `cv2.imshow('Window Title', image)` #Show Image in a window
- `cv2.imwrite(ImgPATH, image)` # to write image to HD
- `_ , image = cv2.threshold(image, threshold, maxval, thresholdType)` #change pixel to maxval if value greater than threshold
- `image = cv2.adaptiveThreshold(image, maxValue, adaptiveMethod, thresholdType, blockSize, C)` #Threshold is automatic calculated

thresholdType



adaptiveMethod

- `ADAPTIVE_THRESH_MEAN_C` : threshold value is the mean of neighborhood area.
- `ADAPTIVE_THRESH_GAUSSIAN_C` : threshold value is the weighted sum of neighborhood values where weights are a Gaussian window.

blockSize : A variable of the integer type representing size of the pixelneighborhood used to calculate the threshold value.

C : A variable of double type representing the constant used in the both methods (subtracted from the mean or weighted mean).

OpenCV Video Stream Example

#VideoStream1 = cv2.VideoCapture('c:/boxed-correct.avi') → read stream from avi video file

VideoStream1 = cv2.VideoCapture(0) → read stream from the first connected video cam

while True:

 IsReturnFrame1,ImgFrame1=VideoStream1.read()

 if IsReturnFrame1 == 0: break

 GrayImgFrame1=cv2.cvtColor(ImgFrame1,cv2.COLOR_BGR2GRAY)

 cv2.imshow('ImageTitle1',ImgFrame1)

 cv2.imshow('ImageTitle2',GrayImgFrame1)

 if cv2.waitKey(0): break

VideoStream1.release()

cv2.destroyAllWindows()

Image processing using OpenCV

```
import cv2
import numpy as np

img = cv2.imread('c:/bookpage.jpg')
cv2.imshow('original',img)

retval, threshold_Color = cv2.threshold(img, 12, 255, cv2.THRESH_BINARY)
cv2.imshow('threshold_Color',threshold_Color)

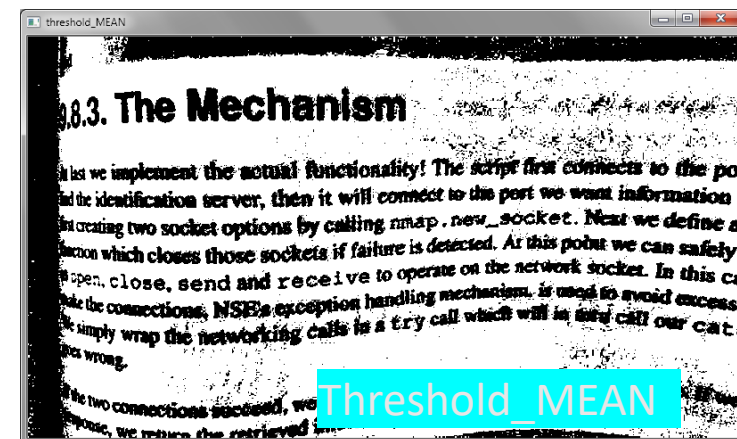
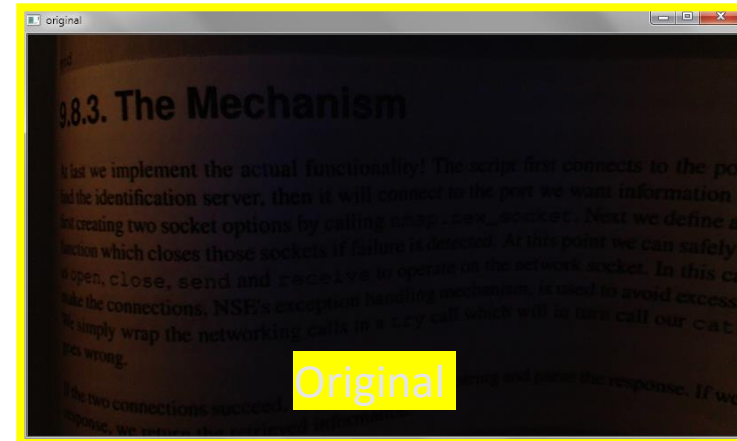
grayscaled=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

retval, thresholdGray = cv2.threshold(grayscaled, 8, 255, cv2.THRESH_BINARY)
cv2.imshow('thresholdGray',thresholdGray)

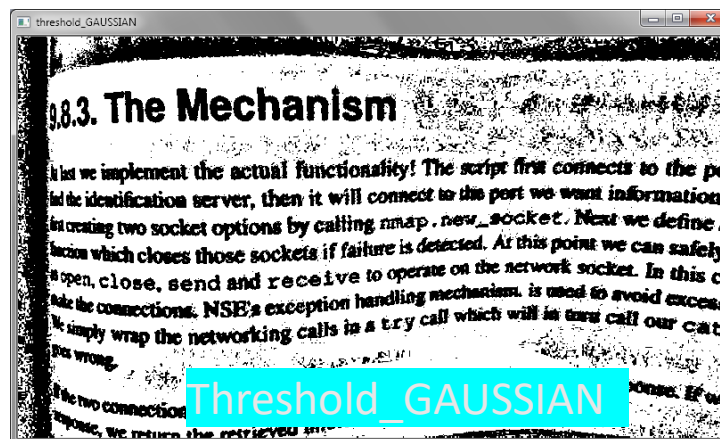
thr_GU = cv2.adaptiveThreshold(grayscaled,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 115, 1)
cv2.imshow('threshold_GAUSSIAN',thr_GU)

thr_MEAN = cv2.adaptiveThreshold(grayscaled, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 115, 1)
cv2.imshow('threshold_MEAN',threshold_MEAN)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

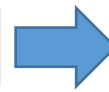


ASOSIASI PERGURUAN TINGGI INFORMATIKA DAN KOMPUTER



Object Recognition

MainImage



template

Search for image with exact lighting/scale/angle

```
img_rgb = cv2.imread('c:/MainImage.jpg')
img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)

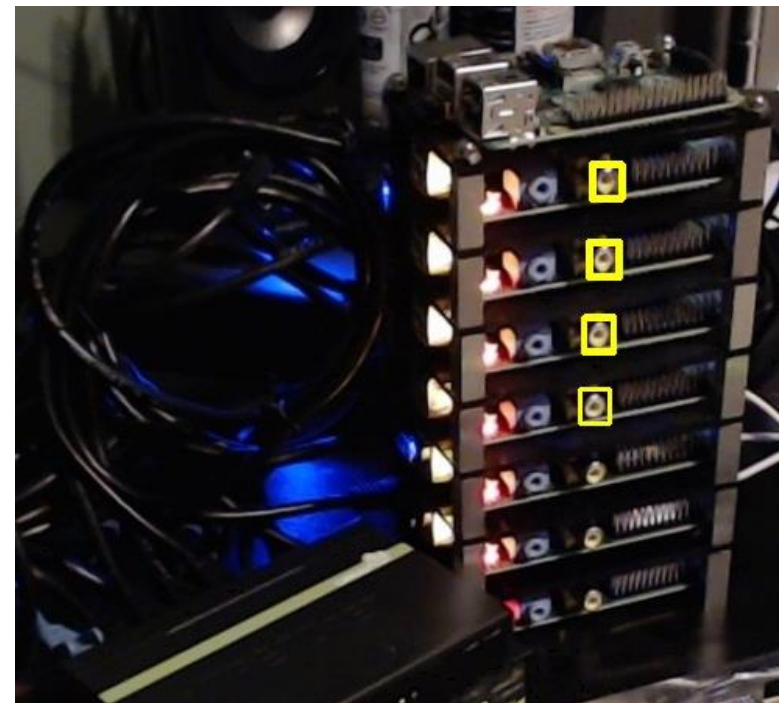
template = cv2.imread('c:/template-for-matching.jpg', 0)
w, h = template.shape[::-1]

res = cv2.matchTemplate(img_gray, template, cv2.TM_CCOEFF_NORMED)
threshold = 0.75
loc = np.where(res >= threshold)

for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0, 255, 255), 2)

cv2.imshow('Detected', img_rgb)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Result



NLP

Natural Language Processing

NLP (Natural Language Processing)

Why we need NLP packages?

- NLP Package handle a wide range of tasks such as Named-entity recognition , part-of-speech (POS) tagging, sentiment analysis, document classification, topic modeling, and much more.

Named-entity recognition: means extract names of persons, organizations, locations, time, quantities, percentages, etc.

example: **Jim bought 300 shares of Acme Corp. in 2006.** → [Jim]**Person** bought 300 shares of [Acme Corp.]**Organization** in [2006]**Time**.

Part-of-speech tagging: used to identification of words as nouns, verbs, adjectives, adverbs, etc.

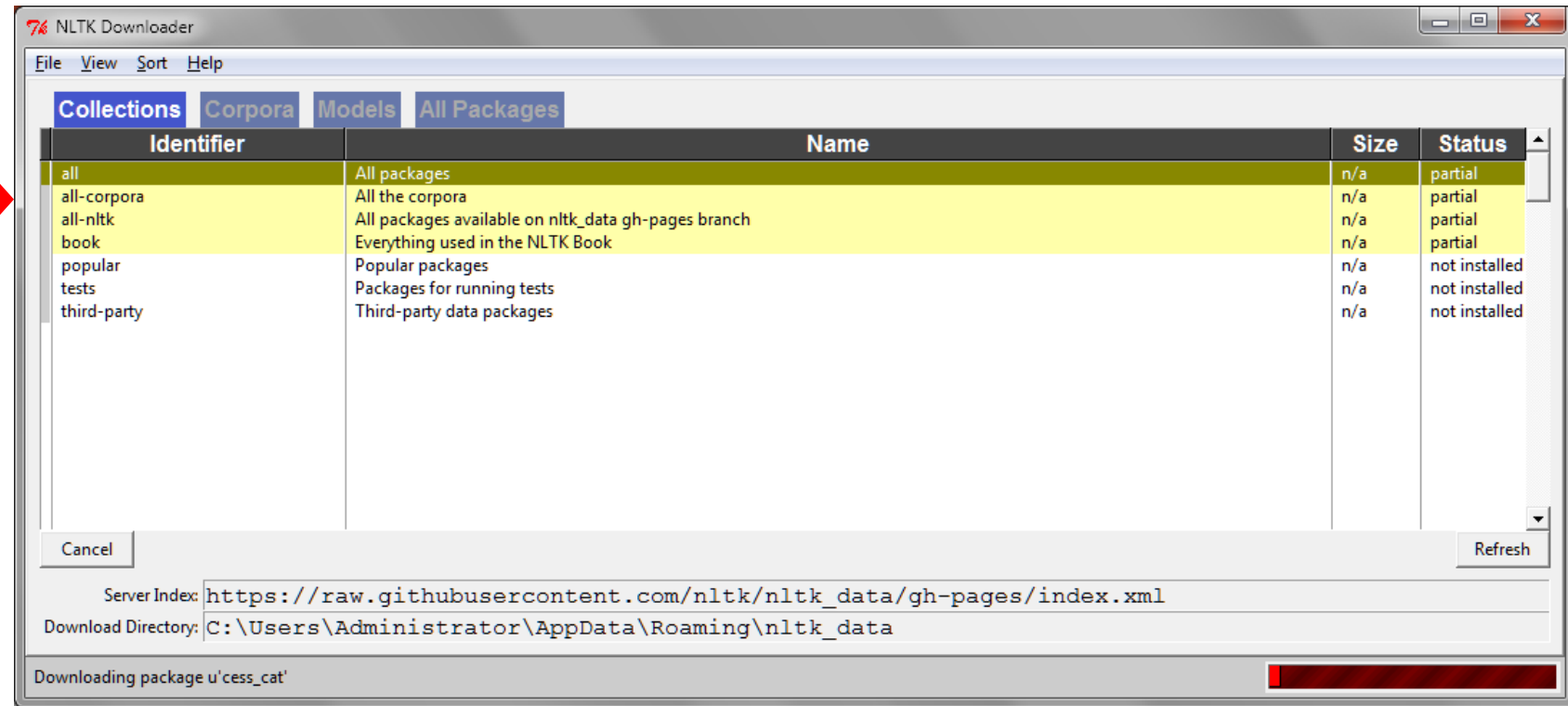
Top 5 Python NLP libraries

- **NLTK** good as education and research tool. Its modularized structure makes it excellent for learning and exploring NLP concepts, but it's not meant for production.
- **TextBlob** is built on top of NLTK, and it's more easily-accessible. good library for fast-prototyping or building applications that don't require highly optimized performance. Beginners should start here.
- **Stanford's CoreNLP** is a Java library with Python wrappers. It's in many existing production systems due to its speed.
- **SpaCy** is a new NLP library that's designed to be fast, streamlined, and production-ready. It's not as widely adopted.
- **Gensim** is most commonly used for topic modeling and similarity detection. It's not a general-purpose NLP library, but for the tasks it does handle, it does them well.

NLTK Package for Natural Language Processing

- Pip install nltk
- To download all packages using GUI, write → `nltk.download()`

```
C:\Python27\python.exe
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:19:30) [MSC v.1500 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> nltk.download()
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```



NLTK concepts

Tokenizing

- Word tokenizers : split text by words
- Sentence tokenizers : split text by paragraphs

Lexicon

- Get words and their actual means in the context

Corpora

- Text classification. Ex: medical journals, presidential speech

Lemmatizing (stemming)

- return the word to its root, ie (gone, went, going) → go

WordNet

- List of different words that have the same meaning for the given word

NLTK Simple example

```
from nltk.tokenize import sent_tokenize, word_tokenize  
from nltk.corpus import stopwords
```

```
EXAMPLE_TEXT = "Hello Mr. Smith, how are you doing today? The weather is great, and Python is awesome. The sky is pinkish-blue. You shouldn't eat cardboard."
```

```
WordsArray = sent_tokenize(EXAMPLE_TEXT)  
print(WordsArray)
```

```
ListOfAvailableStopWords= set(stopwords.words("english"))  
print(ListOfAvailableStopWords)
```

textblob

- pip install textblob
- python -m textblob.download_corpora

```
from textblob import TextBlob

EXAMPLE_TEXT = ("Hello Mr. Mohamed, how are you doing today? The weather is great, "
                "and Python is awesome. The sky is pinkish-blue. You shouldn't eat cardboard.")

blob = TextBlob(EXAMPLE_TEXT)

print (blob.words) #split to words array
print (blob.sentences) #split to paragraphs array

for sentence in blob.sentences:
    print (sentence)

animals = TextBlob("cat dog octopus")
print animals.words           #WordList(['cat', 'dog', 'octopus'])
print animals.words.pluralize() #WordList(['cats', 'dogs', 'octopodes'])
print animals.words.singularize() #WordList(['cat', 'dog', 'octopus'])

SpellingCorrection= TextBlob("I havv goood speling!")
print(SpellingCorrection.correct()) # I have good spelling!

from textblob import Word
w = Word("went")
print w.lemmatize("v") #go

w = Word("doing")
print w.lemmatize("v") #go

#Use internet to translate!
print (blob.translate(to="es") ) #Spanish
```



Sentimental Analysis Example

- We have a sample of 3000 random users comments on imdb.com, amazon.com, and yelp.com website <http://archive.ics.uci.edu/ml/machine-learning-databases/00331/>
- Each comment has a score, Score is either 1 (for positive) or 0 (for negative)

```
# -*- coding: utf-8 -*-
from textblob.classifiers import NaiveBayesClassifier
from textblob import TextBlob
import numpy

#Dataset contains 3000 ==> 1:positive, 0:negative
#http://archive.ics.uci.edu/ml/machine-learning-databases/00331/
train_dataset = numpy.loadtxt("c:/amazon_cells_labelled.txt", delimiter="\t", skiprows =0, dtype='str')
test_dataset1 = numpy.loadtxt("c:/imdb_labelled.txt", delimiter="\t", skiprows =0, dtype='str')
test_dataset2 = numpy.loadtxt("c:/yelp_labelled.txt", delimiter="\t", skiprows =0, dtype='str')

cl = NaiveBayesClassifier(train_dataset)
Current_accuracy = cl.accuracy(test_dataset1)
print("Dataset 1 Accuracy: {0}".format(Current_accuracy))

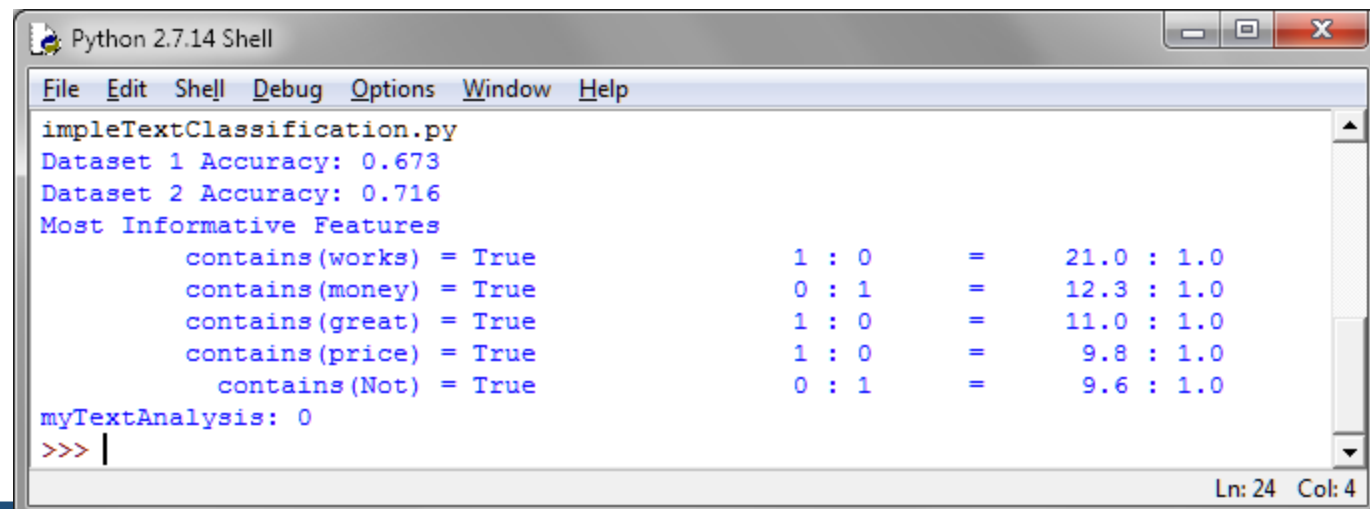
Current_accuracy = cl.accuracy(test_dataset2)
print("Dataset 2 Accuracy: {0}".format(Current_accuracy))

# Most Informative Features, top 5
cl.show_informative_features(5)

blob = TextBlob("The beer was amazing. But the hangover was horrible. My boss was not happy.",
                classifier=cl)
```

```
myTextAnalysis = blob.classify()
print("myTextAnalysis: {0}".format(myTextAnalysis))
```

```
##for sentence in blob.sentences:
##    print(sentence)
##    print(sentence.classify())
```



```
Python 2.7.14 Shell
File Edit Shell Debug Options Window Help
impleTextClassification.py
Dataset 1 Accuracy: 0.673
Dataset 2 Accuracy: 0.716
Most Informative Features
contains(works) = True 1 : 0 = 21.0 : 1.0
contains(money) = True 0 : 1 = 12.3 : 1.0
contains(great) = True 1 : 0 = 11.0 : 1.0
contains(price) = True 1 : 0 = 9.8 : 1.0
contains(Not) = True 0 : 1 = 9.6 : 1.0
myTextAnalysis: 0
>>> |
```

Python Deep Learning

Common Deep learning fields

Main Fields

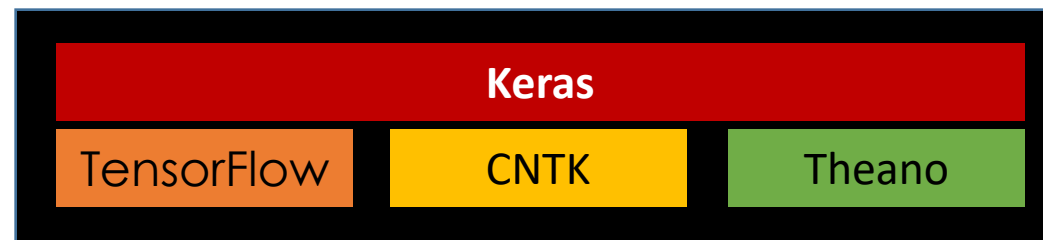
- Computer vision
- Speech recognition
- Natural language processing
- Audio recognition
- Social network filtering
- Machine translation

Visual examples:

- ▶ Colorization of Black and White Images.
[https://www.youtube.com/watch?v= MJU8VK2PI4](https://www.youtube.com/watch?v=MJU8VK2PI4)
- ▶ Adding Sounds To Silent Movies.
<https://www.youtube.com/watch?v=0FW99AQmMc8>
- ▶ Automatic Machine Translation.
- ▶ Object Classification in Photographs.
- ▶ Automatic Handwriting Generation.
- ▶ Character Text Generation.
- ▶ Image Caption Generation.
- ▶ Automatic Game Playing.
<https://www.youtube.com/watch?v=TmPfTpjtdgg>

Common Deep learning development tool (Keras)

- Keras is a free Artificial Neural Networks (ANN) library (deep learning library).
- it is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
- It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.



Deep learning Computer vision (image processing)

3 common ways to detect objects

- median based features
- edge based features
- threshold based features

How to use Neural Network Models in Keras

Five steps

1. Define Network.
2. Compile Network.
3. Fit Network.
4. Evaluate Network.
5. Make Predictions.

Step 1. Define Network

- Neural networks are defined in Keras as a sequence of layers.
- The first layer in the network must define the number of inputs to expect. for a Multilayer Perceptron model this is specified by [the input_dim](#) attribute.
- Example of small Multilayer Perceptron model (2 inputs, 5 hidden layers, 1 output)

```
model = Sequential()  
model.add(Dense(5, input_dim=2))  
model.add(Dense(1))
```

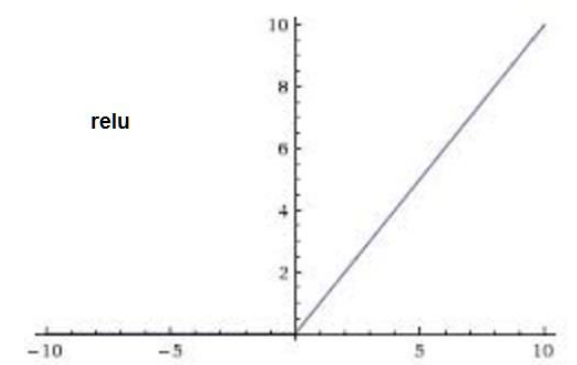
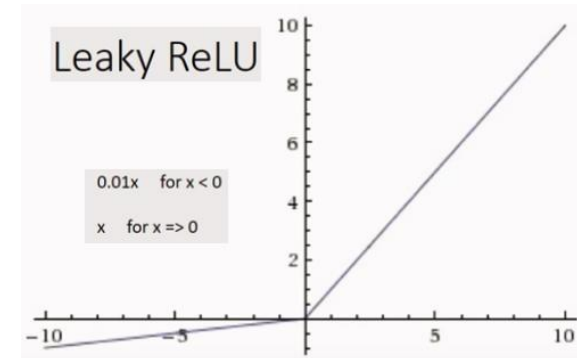
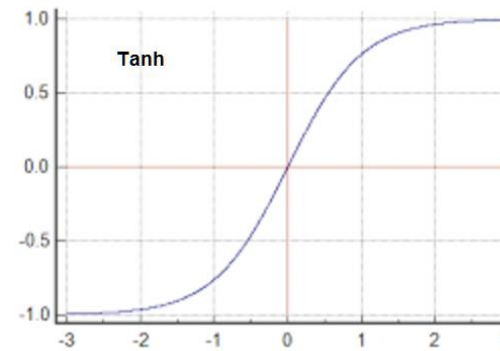
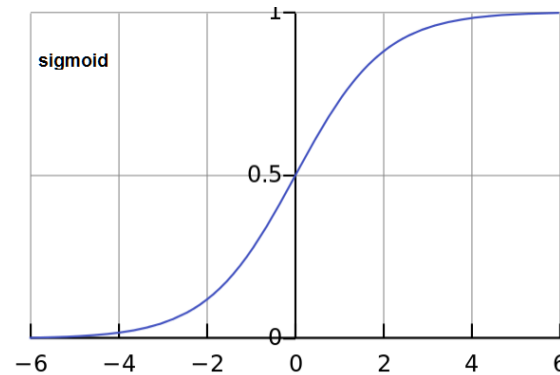
- Re-write after add activation function

```
model = Sequential()  
model.add(Dense(5, input_dim=2, activation='relu'))  
model.add(Dense(1, activation='sigmoid'))
```

Available Activation Functions

Optional, it is like a filter, used to solve some common predictive modeling problem, to get significant boost in performance.

- Sigmoid: used for Binary Classification (2 class) one neuron the output layer. What ever the input it will map to zero or one.
- Softmax: used for Multiclass Classification (>2 class), one output neuron per class value.
- Linear: used for Regression, the number of neurons matching the number of outputs.
- Tanh: what ever the input it will convert to number between -1 and 1
- Relu: either 0 for $x < 0$ or x for $x \geq 0$. so, it just remove the negative values and pass the positive as it is.
- LeakyReLU: minimize the value of negative values and pass as it is if positive
- elu
- selu
- softplus
- softsign
- hard_sigmoid
- PReLU
- ELU
- ThresholdedReLU



Step 2. Compile Network

- Specifically the optimization algorithm to use to train the network and the loss function used to evaluate the network that is minimized by the optimization algorithm.

```
model.compile(optimizer='sgd', loss='mse')
```

Optimizers tool to minimize loss between prediction and real value. Commonly used optimization algorithms:

- 'sgd' (Stochastic Gradient Descent) requires the tuning of a learning rate and momentum.
- ADAM requires the tuning of learning rate.
- RMSprop requires the tuning of learning rate.

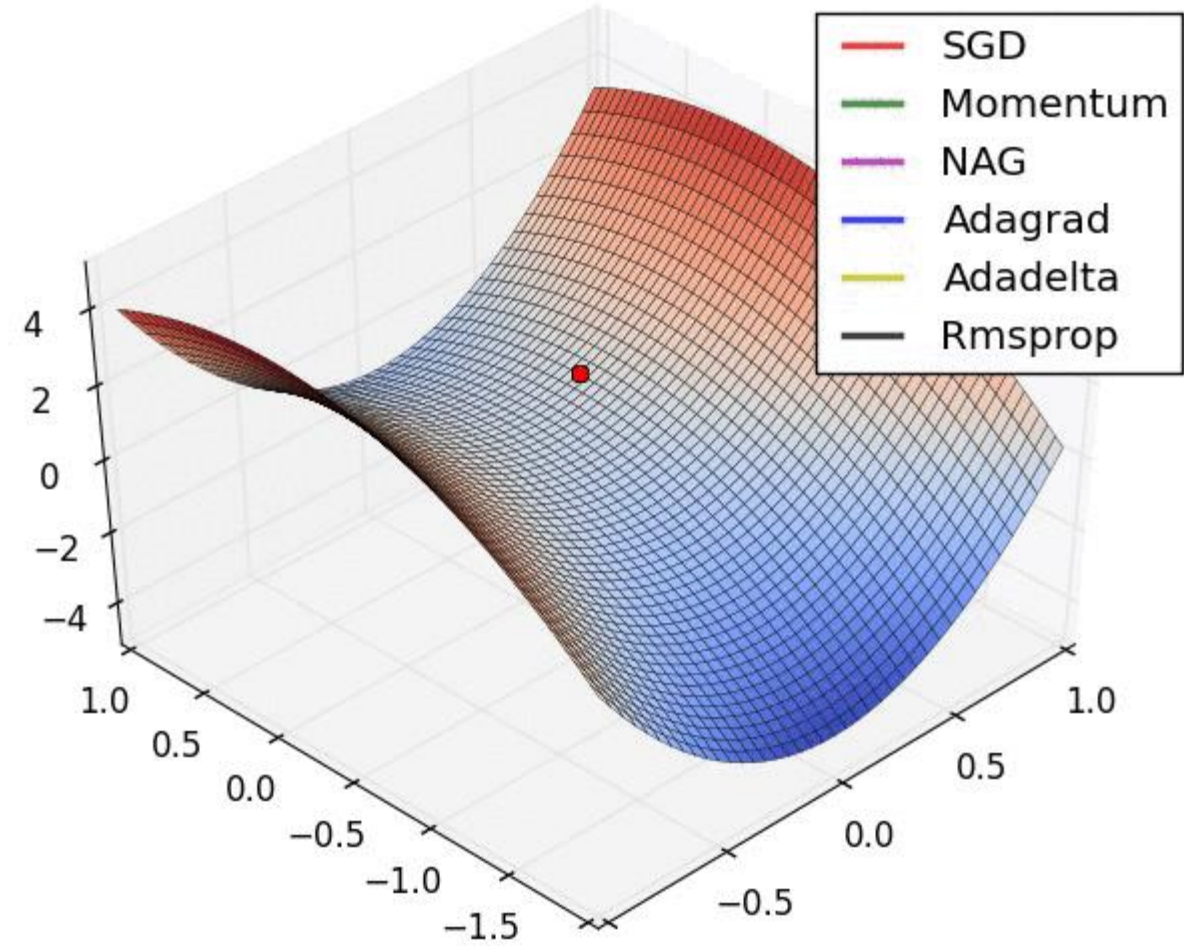
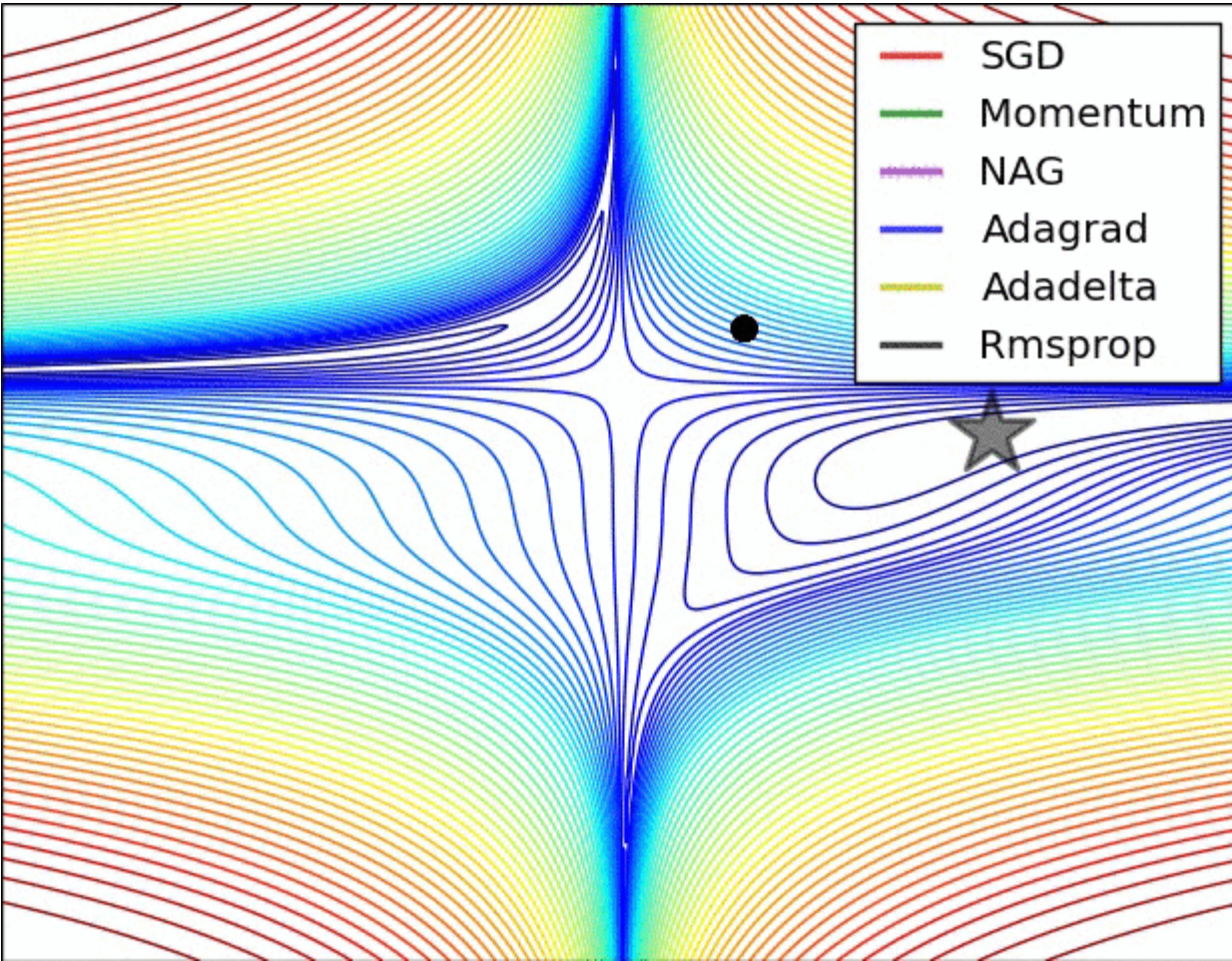
Loss functions:

- Regression: Mean Squared Error or 'mse'.
- Binary Classification (2 class): 'binary_crossentropy'.
- Multiclass Classification (>2 class): 'categorical_crossentropy'.

Finally, you can also specify metrics to collect while fitting the model in addition to the loss function. Generally, the most useful additional metric to collect is accuracy.

```
model.compile(optimizer='sgd', loss='mse', metrics=['accuracy'])
```


Optimizers



Step 3. Fit Network

```
history = model.fit(X, y, batch_size=10,  
                    epochs=100)
```

The network is trained using the backpropagation algorithm

- Batch size is the number of samples that going to be propagated through the network.
- epochs is the number of training times (for **ALL** the training examples)

Example: if you have 1000 training examples, and your batch size is 500, then it will take 2 iterations to complete 1 epoch.

Step 4. Evaluate Network

- The model evaluates the loss across all of the test patterns, as well as any other metrics specified when the model was compiled.
- For example, for a model compiled with the accuracy metric, we could evaluate it on a new dataset as follows:

```
loss, accuracy = model.evaluate(X, Y)  
print("Loss: %.2f, Accuracy: %.2f%%" % (loss, accuracy*100))
```

Step 5. Make Predictions

```
probabilities = model.predict(X)
predictions = [float(round(x)) for x in probabilities]
accuracy = numpy.mean(predictions == Y) #count the number of True and divide by the
total size
print("Prediction Accuracy: %.2f%%" % (accuracy*100))
```

Binary classification using Neural Network in Keras

Diabetes Data Set

Detect Diabetes Disease based on analysis

Dataset Attributes:

1. Number of times pregnant
2. Plasma
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

```
1 # Sample Multilayer Perceptron Neural Network in Keras
2 from keras.models import Sequential
3 from keras.layers import Dense
4 import numpy
5 # load and prepare the dataset
6 dataset = numpy.loadtxt("pima-indians-diabetes.csv", delimiter=",")
7 X = dataset[:,0:8]
8 Y = dataset[:,8]
9 # 1. define the network
10 model = Sequential()
11 model.add(Dense(12, input_dim=8, activation='relu'))
12 model.add(Dense(1, activation='sigmoid'))
13 # 2. compile the network
14 model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
15 # 3. fit the network
16 history = model.fit(X, Y, epochs=100, batch_size=10)
17 # 4. evaluate the network
18 loss, accuracy = model.evaluate(X, Y)
19 print("\nLoss: %.2f, Accuracy: %.2f%%" % (loss, accuracy*100))
20 # 5. make predictions
21 probabilities = model.predict(X)
22 predictions = [float(round(x)) for x in probabilities]
23 accuracy = numpy.mean(predictions == Y)
24 print("Prediction Accuracy: %.2f%%" % (accuracy*100))
```

Save prediction model

- After train our model, ie, `model.fit(X_train, Y_train)`, we can save this training to use later.
- This task can be done by **Pickle** package (Python Object Serialization Library), using dump and load methods. Pickle can save any object not just the prediction model.

```
import pickle
```

```
.....
```

```
model.fit(X_train, Y_train)
```

```
# save the model to disk
```

```
pickle.dump(model, open("c:/data.dump", 'wb'))    #wb= write bytes
```

```
# some time later... load the model from disk
```

```
model = pickle.load(open("c:/data.dump", 'rb'))    #rb= read bytes
```



Terima Kasih