**BigData One:**

● 1 terabyte (TB) = 1012 bytes = 1000 GB.

● 1 petabyte (PB) = 1015 bytes = 1 million GB.

● 1 exabyte (EB) = 1018 bytes = 1 billion GB.

● 1 zettabyte (ZB) = 1021 bytes = 1000 billion GB.

● 1 yottabyte (YB) = 1024 bytes = million billion GB.

Characterizing Big Data by various V’s:

**Volume: scale of data**

**Variety: different forms of data**

**Velocity: analysis of streaming data**

**Veracity: uncertainty of data**

Other V’s:

**Variability:**

**Visualization:**

**Value:**

**BigData Two:**

**Data science process:**

1. **Collecting and (pre)processing:** 
   1. **Structured data**
   2. **Unstructured data: natural language text. Video, audio**
   3. **Semi-structured data: JSON, XML, CSV**
2. **Cleaning and analyzing:**
   1. **Cleaning the data**
   2. **Exploratory data analysis**
   3. **Models & algorithms**
3. **Data product, communicating**

**Machine learning vs data mining:**

Data mining: Computational data analysis to find interesting properties from data. Emphasis is on analyzing current data.

Machine learning: A class of computational analysis methods that build (and update) a

general data model based on known data. Emphasis is on analyzing future data. Two aspects, training or fitting and prediction.

**Hadoop Distributed File System (HDFS)：**

Amdahl's law:

maximum parallel speedup using n computers: 1 / (f + (1 - f) / n), f is the fraction of code that can not be parallelized.

lines\_one = [line.split(**','**) **for** line **in** open(**'tracks.csv'**)]

**with** open(**'./busdata.json'**, **'r'**) **as** busdata\_json:  
 databuf = io.StringIO() *# Initialize an empty StringIO file.* **for** line **in** busdata\_json: *# Read one line from the original file.  
 # print(line)* databuf.write(line) *# Write the read line into the StringIO file.* **if** line == **'}\n'**: *# Did we reach the end of a top-level dictionary?* databuf.seek(0) *# Set the StringIO to enable reading from its beginning.* data = json.load(databuf) *# Read the single top-level JSON dictionary.* databuf.close() *# Discard the current StringIO object.* databuf = io.StringIO() *# Initialize a new empty StringIO object.  
 # Now data can be used as a normal Python dictionary.  
 # print(data.keys()) # One example: lists the keys of the data dictionary.  
 # print(data['body'][0].keys())* **for** i **in** data[**'body'**]:  
 **if** i[**'monitoredVehicleJourney'**][**'lineRef'**] **not in** bus\_vehicles:  
 bus\_vehicles[i[**'monitoredVehicleJourney'**][**'lineRef'**]] = [i[**'monitoredVehicleJourney'**][**'vehicleRef'**]]  
 **if** i[**'monitoredVehicleJourney'**][**'vehicleRef'**] **not in** bus\_vehicles[i[**'monitoredVehicleJourney'**][**'lineRef'**]]:  
 bus\_vehicles[i[**'monitoredVehicleJourney'**][**'lineRef'**]] += [i[**'monitoredVehicleJourney'**][**'vehicleRef'**]]  
 **if** i[**'monitoredVehicleJourney'**][**'lineRef'**] **not in** bus\_data\_points:  
 bus\_data\_points[i[**'monitoredVehicleJourney'**][**'lineRef'**]] = [i[**'recordedAtTime'**] + i[**'monitoredVehicleJourney'**][**'vehicleRef'**]]  
 **if** (i[**'recordedAtTime'**] + i[**'monitoredVehicleJourney'**][**'vehicleRef'**]) **not in** bus\_data\_points[i[**'monitoredVehicleJourney'**][**'lineRef'**]]:  
 bus\_data\_points[i[**'monitoredVehicleJourney'**][**'lineRef'**]] += [i[**'recordedAtTime'**] + i[**'monitoredVehicleJourney'**][**'vehicleRef'**]]  
 databuf.close()