Agenda

Day1	1. Fundamentals of Machine Learning: Introduction to Machine Learning and its Applications, Machine Learning Python Tools. Simple machine learning application. 2. Data Preprocessing: Data Sets, importing libraries, Missing Data, Categorical Data, Splitting the Data Set into Training Set and Test Set, Feature Extraction and Preprocessing
Day2	1. Types of Machine Learning Algorithms 2. Bayesian Learning: Bayes Theorem and Concept Learning, ML and LS error Hypothesis, Naïve Bayes Classifier, Bayesian belief networks, EM Algorithm
Day 3	1. Instance Based Learning: Introduction, K-nearest neighbor learning, locally weighted regression, radial basis function, cased based reasoning. 2. Reinforced Learning: Introduction, Learning Task, Q Learning

Day 1

1. Fundamentals of Machine Learning
2. Data Preprocessing

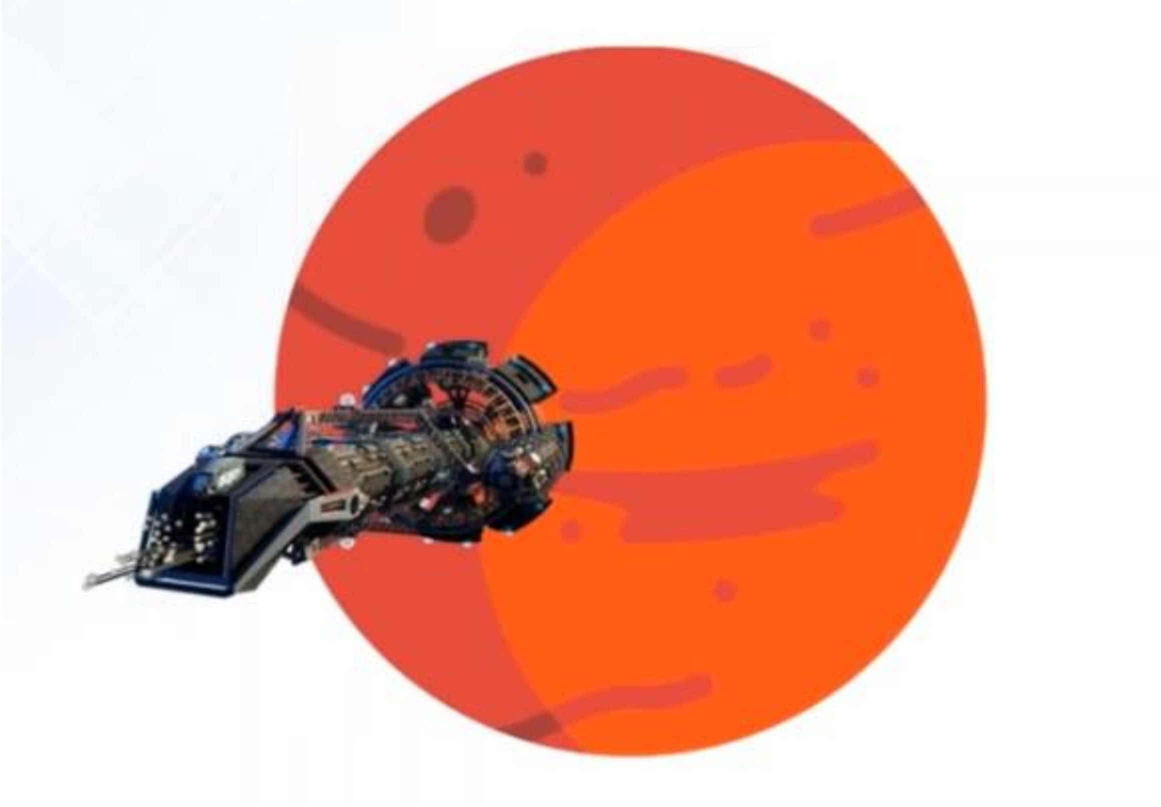
1.Fundamentals of Machine Learning

a) **Why Machine Learning**
b) **What is Machine Learning ?**
c) **How Does ML Works**
d) **ML Applications**
e) **ML Use Cases**
f) **Understanding ML using Analogy**
g) **AI , ML and DL**
h) **Machine Learning Python Tools.**
i) **Simple machine learning application.**

Why Machine Learning ?

a. Why Machine Learning ?

Navigation on Mars or Deep ocean
Where no humans are present

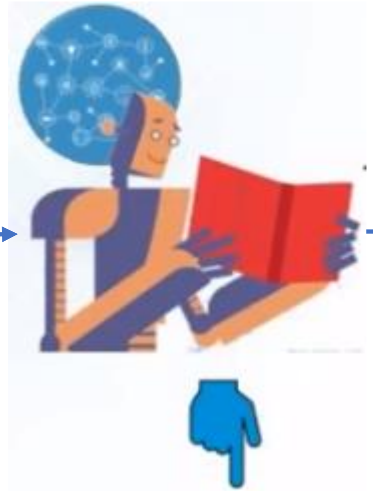


a. Why Machine Learning ?

Analyzing huge sensor data and predicting the outcome eg: in Forecasting Systems is not possible by manual calculations

Recommending a product to buyers e.g. used by amazon , flipkart , Snapdeal etc., is not possible by manual calculations

Purchase History of a Customer



Search list of a Customer

a. Why Machine Learning ?

b. What is Machine Learning ?

b. What is Machine Learning ?

- **Is a type of Artificial Intelligence that provides computers with the ability to learn without being explicitly programmed.**
- Machine Learning is a concept which allows the machine to learn from examples and experience, and that too without being explicitly programmed.
    - So instead of you writing the code, what you do is you feed data to the generic algorithm, and the algorithm/ machine builds the logic based on the given data.
- ML enables the computers or the machines to make data-driven decisions rather than being explicitly programmed for carrying out a certain task. These programs or algorithms are designed in a way that they learn and improve over time when are exposed to new data.

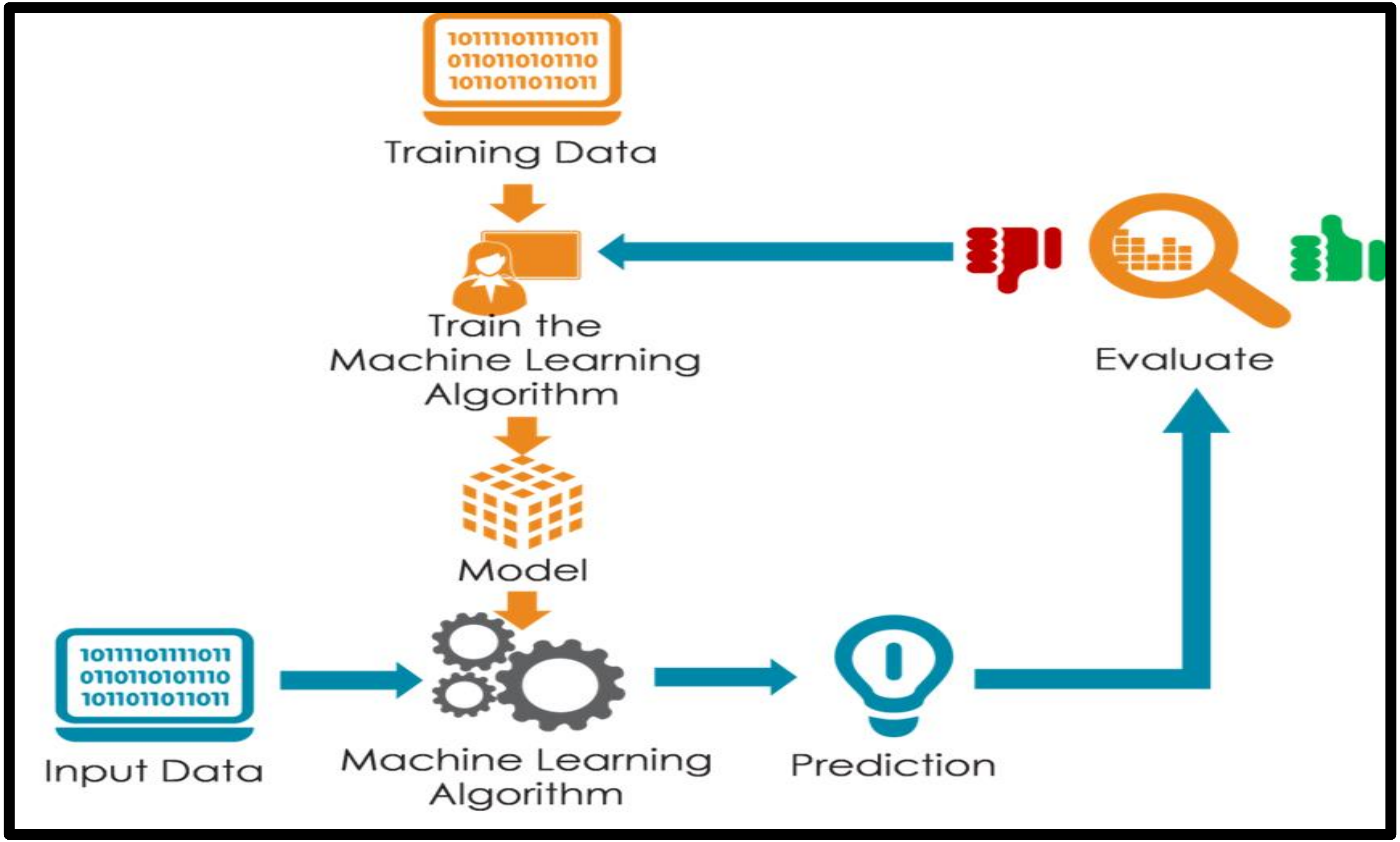
b. What is Machine Learning?

- Have you ever shopped online? So while checking for a product, did you noticed ***when it recommends for a product similar to what you are looking for***? or did you noticed "***the person bought this product also bought this***" combination of products. How are they doing this recommendation? **This is machine learning**.
- Did you ever get a call from any bank or finance company ***asking you to take a loan or an insurance policy***? What do you think, do they call everyone? ***No, they call only a few selected customers who they think will purchase their product. How do they select***? This is target marketing and can be applied using Clustering. **This is machine learning.**

c. How does Machine Learning Works ?

c. How does Machine Learning Work?

- Machine Learning algorithm is trained using a training data set to create a model. ***When new input data is introduced to the ML algorithm, it makes a prediction on the basis of the mode***l.
- The prediction is evaluated for accuracy and if the accuracy is acceptable, the Machine Learning algorithm is deployed. ***If the accuracy is not acceptable, the Machine Learning algorithm is trained again and again with an augmented training data set.***
- This is just a very high-level example as there are many factors and other steps involved.



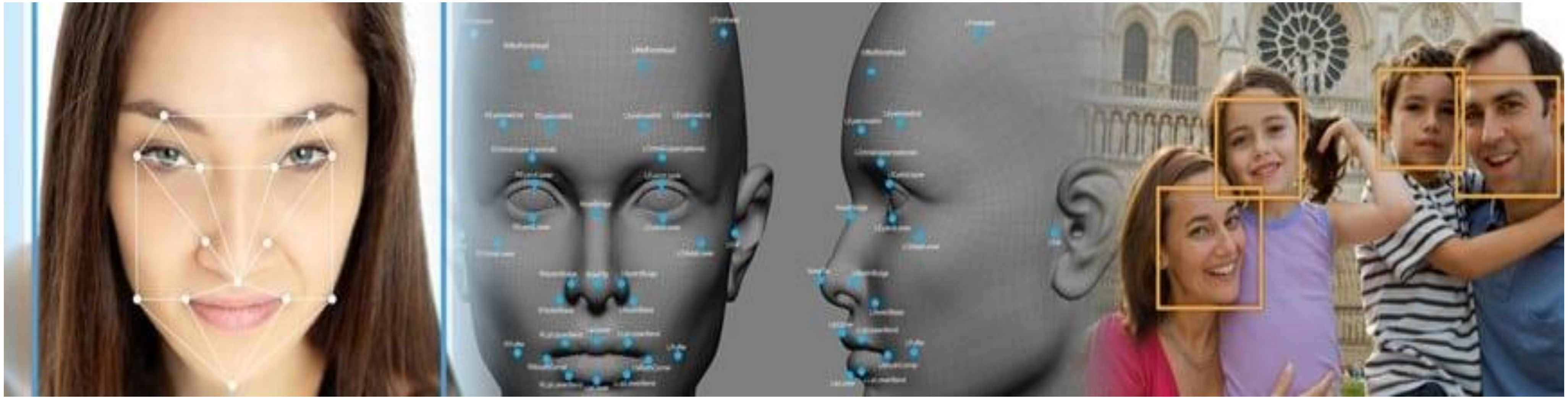
1011110111011
0110110101110
1011011011011
Training Data
Train the
Machine Learning
Algorithm
Evaluate
Model
1011110111011
0110110101110
1011011011011
Input Data
Machine Learning
Algorithm
Prediction

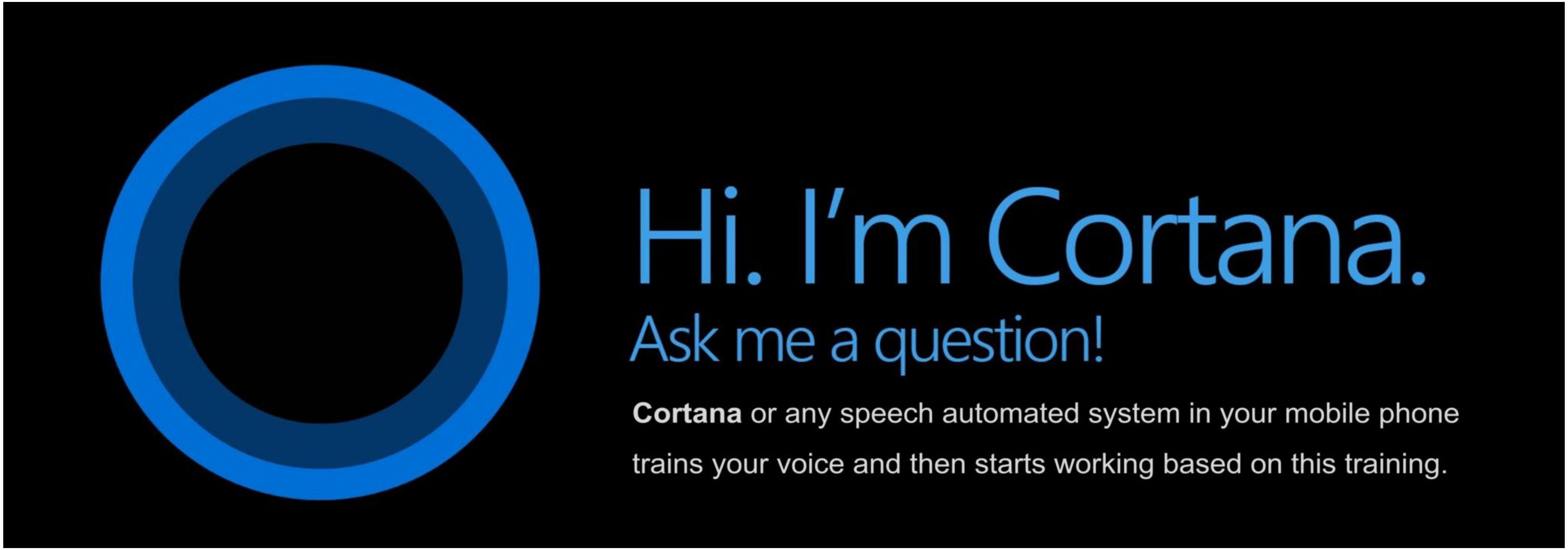
d. Machine Learning Applications

- **Virtual Personal Assistants : Siri, Alexa and Google Now**
- **Predictions while Commuting:** *Traffic Predictions, Online Transportation Networks*
- **Videos Surveillance :**
- **Social Media Services:** *People You May Know, Face Recognition, Similar Pins*
- **Email Spam and Malware Filtering**
- **Online Customer Support**
- **Search Engine Result Refining**
- **Product Recommendations**
- **Online Fraud Detection**

d. Machine Learning Use Cases

Face Detection.

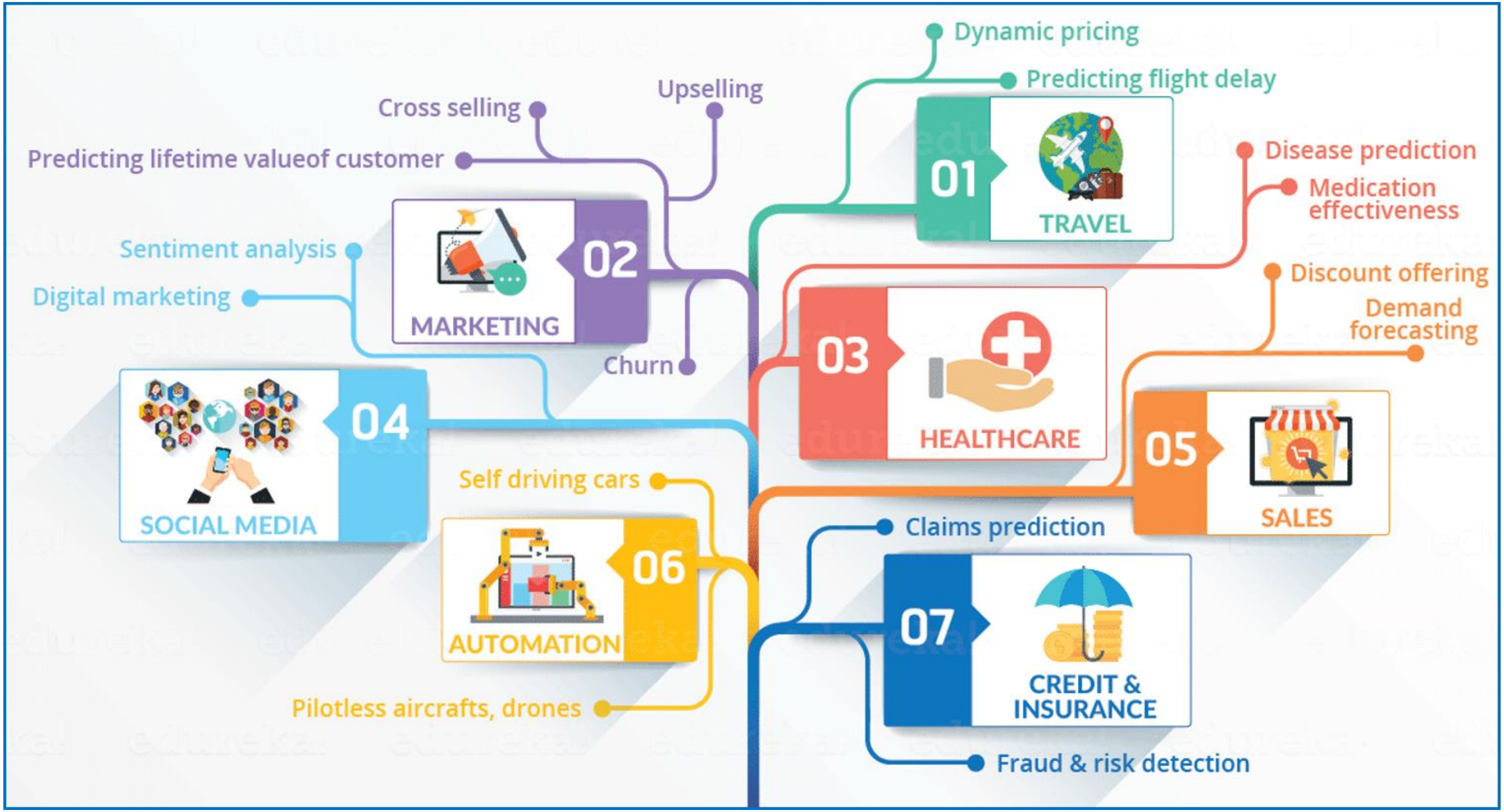




Cortana is a virtual assistant created by Microsoft for Windows 10, Windows 10 Mobile, Windows Phone 8.1,Invoke smart speaker, Microsoft Band, Xbox One, iOS, Android, Windows Mixed Reality, and soon Amazon Alexa.

Cortana can set reminders, recognize natural voice without the requirement for keyboard input, and answer questions using information from the Bing search engine.

Cortana is currently available in English, Portuguese, French, German, Italian, Spanish, Chinese, and Japanese language editions, depending on the software platform and region in which it is used.



01
TRAVEL
Dynamic pricing
Predicting flight delay
02
MARKETING
Cross selling
Upselling
Predicting lifetime valueof customer
Churn
03
HEALTHCARE
Disease prediction
Medication effectiveness
04
SOCIAL MEDIA
Sentiment analysis
Digital marketing
05
SALES
Discount offering
Demand forecasting
06
AUTOMATION
Self driving cars
Pilotless aircrafts, drones
07
CREDIT & INSURANCE
Claims prediction
Fraud & risk detection

e. Understanding Machine Learning with an Analogy

- **As a Human:** Let's suppose one day you went for shopping mangoes. The vendor had a cart full of mangoes from where you could handpick the mangoes, get them weighed and pay according to the rate fixed per Kg.
- **Task: How will you choose the best mangoes?**

Set of learning,

- Given below is set of learning, human gains from his experience of shopping mangoes, you can drill it down to have a further look at it in detail. Go through it once, you will relate it to machine learning very easily.

  - Learning 1: Bright yellow mangoes are sweeter than pale yellow ones
  - Learning 2: The smaller and bright yellow mangoes are sweet only half the time
  - Learning 3: Small, pale yellow ones are the sweetest of all
  - Learning 4: Soft mangoes are jucier
  - Learning 5: Green mangoes are tastier than yellow ones
  - Learning 6: You don't need mangoes anymore

Learning 1

- **Experience 1:**
- You were informed that bright and yellow mangoes are sweeter than pale and yellow ones.
- So you make a simple **rule: pick only from the bright yellow mangoes.** You check the colour of the mangoes, pick the bright yellow ones, pay up, and return home. Right?

Learning2

- **Experience 2:**
- Now when you went home and tasted the mangoes, some of them were not sweet as you thought. You are worried as your wisdom was insufficient. You concluded that when it comes shopping mangoes, you have to look for more than just the colours.
- After a lot of pondering and tasting different types of mangoes,**you concluded that** the bigger and bright yellow mangoes are guaranteed to be sweet, while the smaller, bright yellow mangoes are sweet only half the time (i.e. if you bought 100 bright yellow mangoes (50 will be big in size and rest 50 will be small), then the 50 big mangoes will all be sweet, while out of the 50 small ones, only 25 mangoes will turn out to be sweet). You will then update your rule about the mango shopping and from next time you will keep this in mind.

Learning 3

- **Experience 3: *Tragedy:*** *Next time at the market, you see that your favorite vendor has gone out of town. You decide to buy from a different vendor, who supplies mangoes grown from a different part of the country. Now, you realize that the rule which you had learnt (that big, bright yellow mangoes are the sweetest) is no longer applicable. You have to learn from scratch. You taste a mango of each kind from this vendor and realize that the small, pale yellow ones are in fact the sweetest of all.*

Learning 4

- **Experience 4:** *One day your cousin visits you from another city. You decide to treat her with mangoes. But she is like "I don't care about the sweetness of a mango, I only want the juiciest ones". Now once again, you run your experiments, tasting all kinds of mangoes, and realizing that the softer ones are juicier.*

Learning 5

- **Experience 5:** *Later on, you move to a different part of the world and you found that the mangoes here taste surprisingly different from your home country. You realized that for this country the green mangoes are tastier than the yellow ones.*

Learning 6

- **Experience 6:** *You marry someone who hates mangoes but loves apples instead. Now you go for shopping oranges instead of mangoes. Now, all your accumulated knowledge about mangoes is worthless. Now you have to learn everything about the correlation between the physical characteristics and the taste of apples, by the same method of experimentation.*

What if you have to write a code for it?

- **As a Human Written Code:** Now, imagine you were asked to write a computer program to choose your mangoes (or oranges). You might write the following rules/algorithm:

- if is bright yellow **and** size is big **and** sold by: mango is sweet.
  if (soft): mango is juicy

- You would use these rules to choose the mangoes.

Training and Testing Phase

- Machine Learning algorithms are an evolution of normal algorithms. They make your programs “smarter”, by allowing them to automatically learn from the data you provide. The algorithm is mainly divided into:
  - Training Phase
  - Testing phase

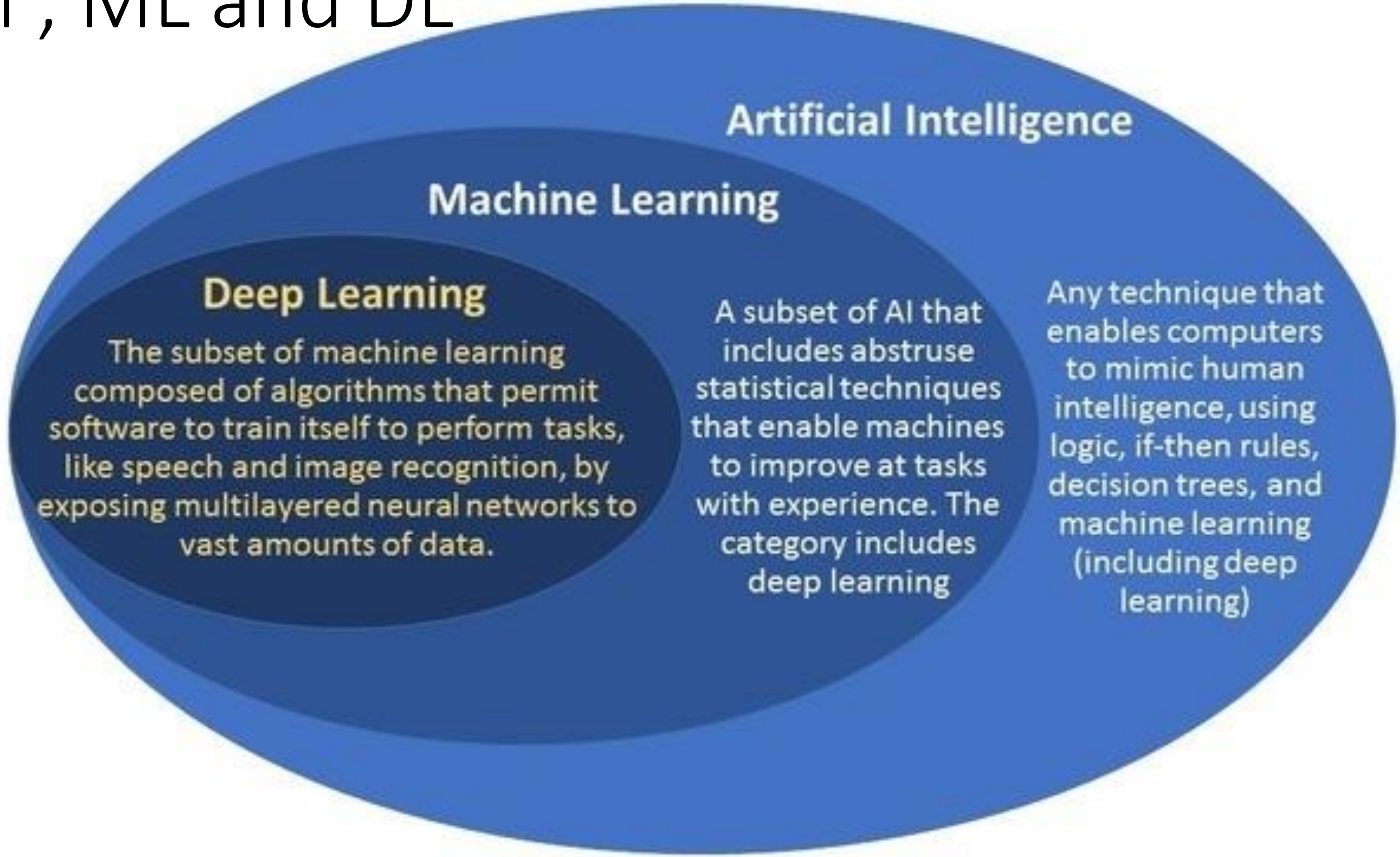
Training Phase

- You take a randomly selected specimen of mangoes from the market (**training data**), make a table of all the physical characteristics of each *mango, like color, size, shape, grown in which part of the country, sold by which vendor, etc (**features**), along with the sweetness, juiciness, ripeness of that mango (**output variables**)*.
- You feed this data to the machine learning algorithm (**classification/regression**), and it learns a model of the correlation between an average mango's physical characteristics, and its quality.

Testing Phase

- Next time when you go shopping, you will measure the characteristics of the mangoes which you are purchasing(**test data**)and feed it to the Machine Learning algorithm.
- It will use the model which was computed earlier *to predict if the mangoes are sweet, ripe and/or juicy*.
- The algorithm may internally use the rules, similar to the one you manually wrote earlier (for eg, a **decision tree**).
- Finally, you can now shop for mangoes with great confidence, without worrying about the details of how to choose the best mangoes.

f. AI , ML and DL


Artificial Intelligence
Machine Learning
Deep Learning
The subset of machine learning composed of algorithms that permit software to train itself to perform tasks, like speech and image recognition, by exposing multilayered neural networks to vast amounts of data.
A subset of AI that includes abstruse statistical techniques that enable machines to improve at tasks with experience. The category includes deep learning
Any technique that enables computers to mimic human intelligence, using logic, if-then rules, decision trees, and machine learning (including deep learning)

g. Open Source and Commercial Machine Learning Tools

Open Source Tools

1. Scikit Learn
2. Shogun
3. Accord.NET Framework
4. Spark MLlib
5. H20
6. Coudera Oryx
7. GoLearn
8. Weka
9. Deep Learn.js
10. ConvNet.Js
11. OpenAI
12. TensorFlow
13. Keras
14. Charnn
15. PaddlePAddle
16. CNTK
17. R
18. Monte Carlo ML Library
19. Octave Forge

Commercial Tools

1. Microsoft Azure Machine Learning
2. SAS Enterprise Miner
3. IBM SPSS Modeler
4. RapidMiner
5. Apache Mahout
6. MATLAB
7. Oracle Data Mining

Steps involved in machine learning

1. Collecting Data
2. Cleaning Data
3. Analyze Data
4. Build Model
5. Train the Algorithm
6. Test the Algorithm
7. Use it

i. Simple Machine Learning Application

Training data

Features

Weight	Texture	Label
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
...	...	...

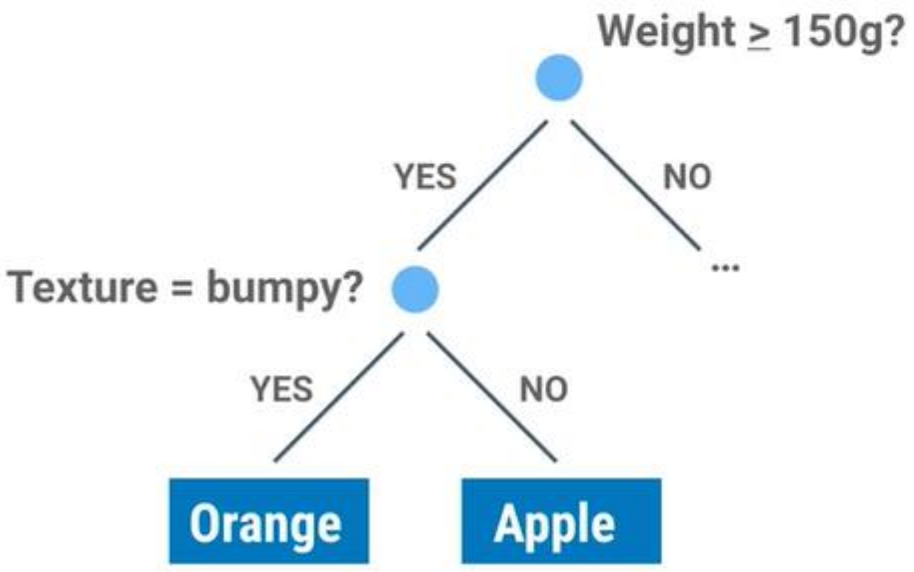
Supervised Learning Recipe

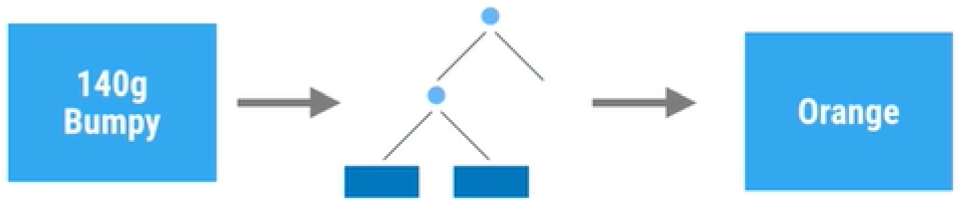
Training data

Features

Weight	Texture	Label
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple
...	...	...

Decision Tree





Goals

1. Import dataset.
2. Train a classifier.
3. Predict label for new flower.
4. Visualize the tree.

Testing Data

Just like in programming, testing is a very important part of ML.

Testing Data

- Examples used to “test” the classifier’s accuracy.
- Not part of the training data.

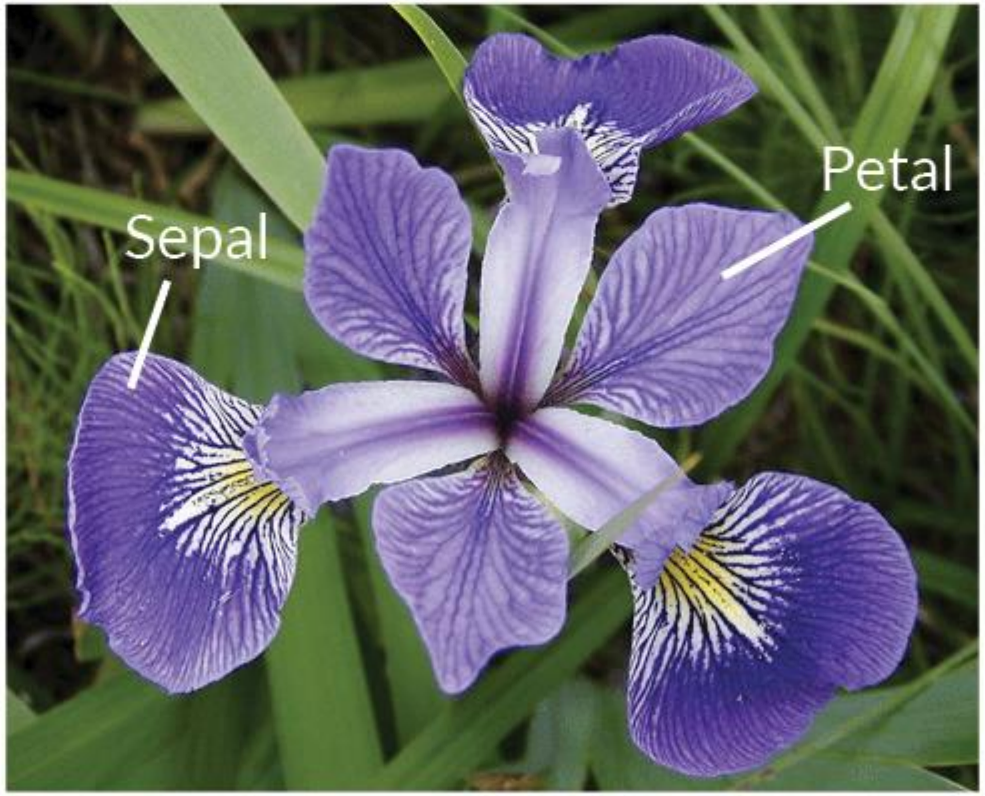
Hands On

End of Day1 Part1

2 .Data Preprocessing

Data Sets

- A **data set** (or **dataset**) is a collection of data.
- Most commonly a data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the data set in question.
- The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a **datum**.



Iris Versicolor

Iris Setosa

Iris Virginica

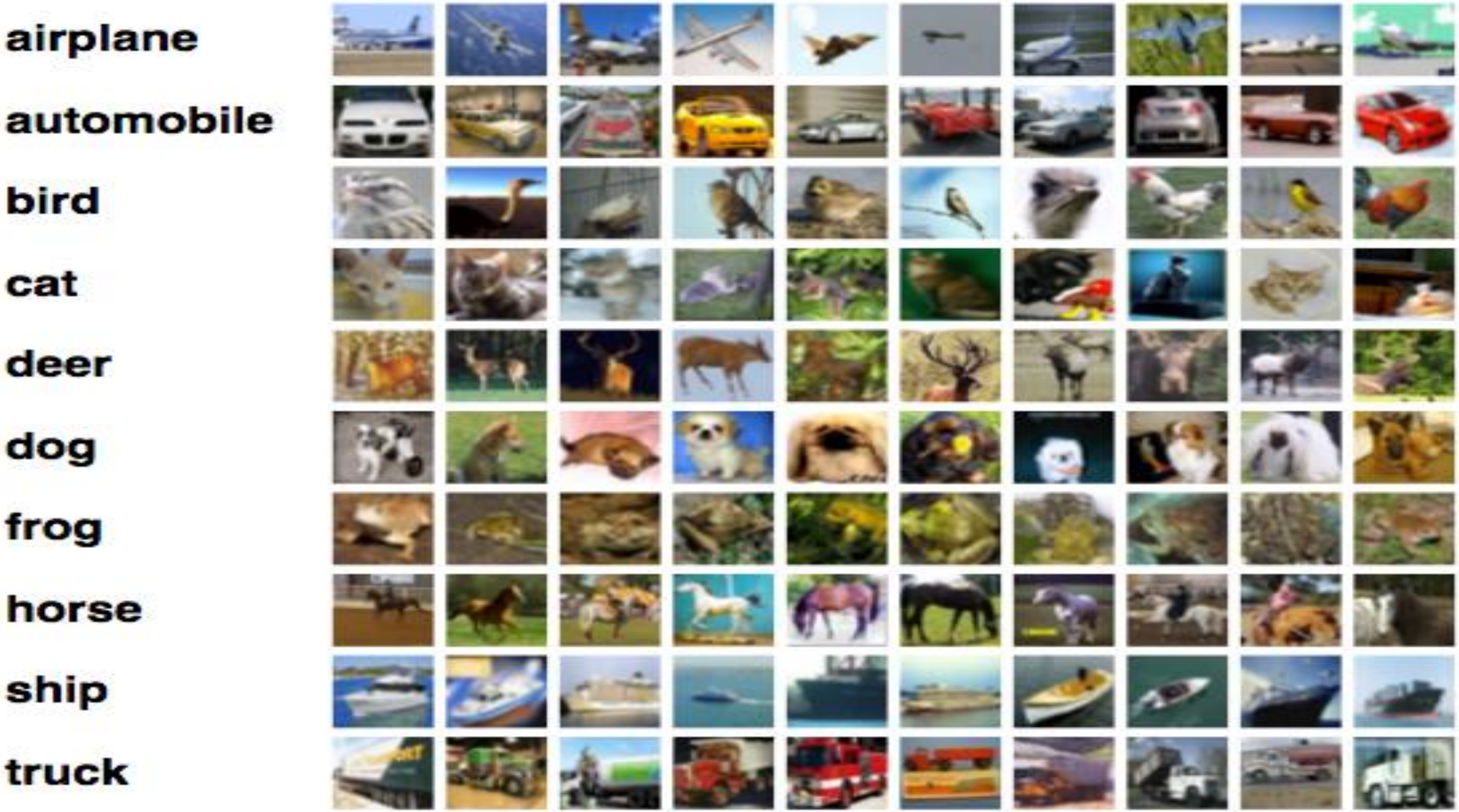
Attributes

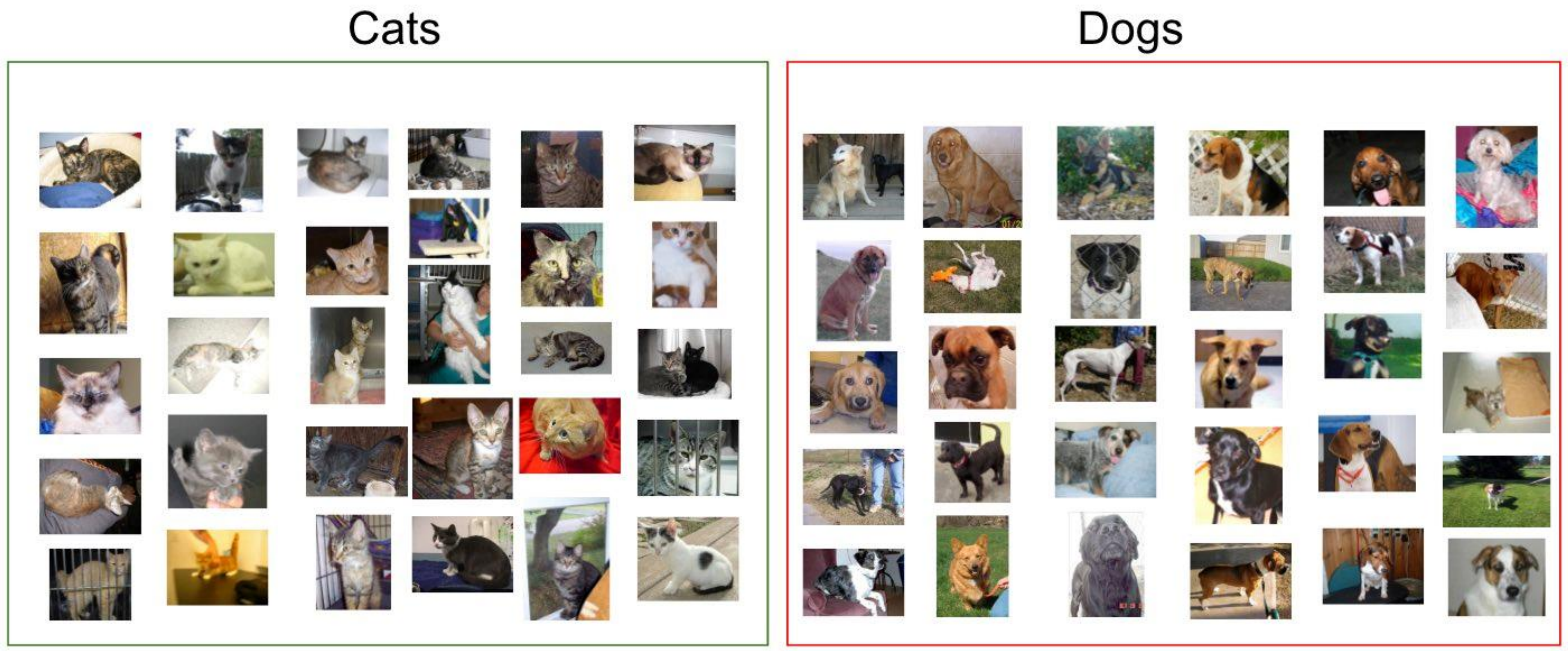
sepal_length	sepal_width	petal_length	petal_width	Iris_class
5	2	3.5	1	versicolor
6	2.2	4	1	versicolor
6.2	2.2	4.5	1.5	versicolor
6	2.2	5	1.5	virginica
4.5	2.3	1.3	0.3	setosa
5.5	2.3	4	1.3	versicolor
6.3	2.3	4.4	1.3	versicolor
5	2.3	3.3	1	versicolor
4.9	2.4	3.3	1	versicolor
5.5	2.4	3.8	1.1	versicolor
5.5	2.4	3.7	1	versicolor
5.6	2.5	3.9	1.1	versicolor
6.3	2.5	4.9	1.5	versicolor
5.5	2.5	4	1.3	versicolor
5.1	2.5	3	1.1	versicolor
4.9	2.5	4.5	1.7	virginica
6.7	2.5	5.8	1.8	virginica
5.7	2.5	5	2	virginica
6.3	2.5	5	1.9	virginica
5.7	2.6	3.5	1	versicolor
5.5	2.6	4.4	1.2	versicolor
5.8	2.6	4	1.2	versicolor

Data point /example

Numerical value

Categorical value


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



Sample of cats & dogs images from Kaggle Dataset

Predictors/Attributes

Target

Outlook	Temperature	Humidity	Windy	Play Tennis
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	High	TRUE	No

Weekend (Example)	Weather	Parents	Money	Decision (Category)
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay in
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

Features **Label**

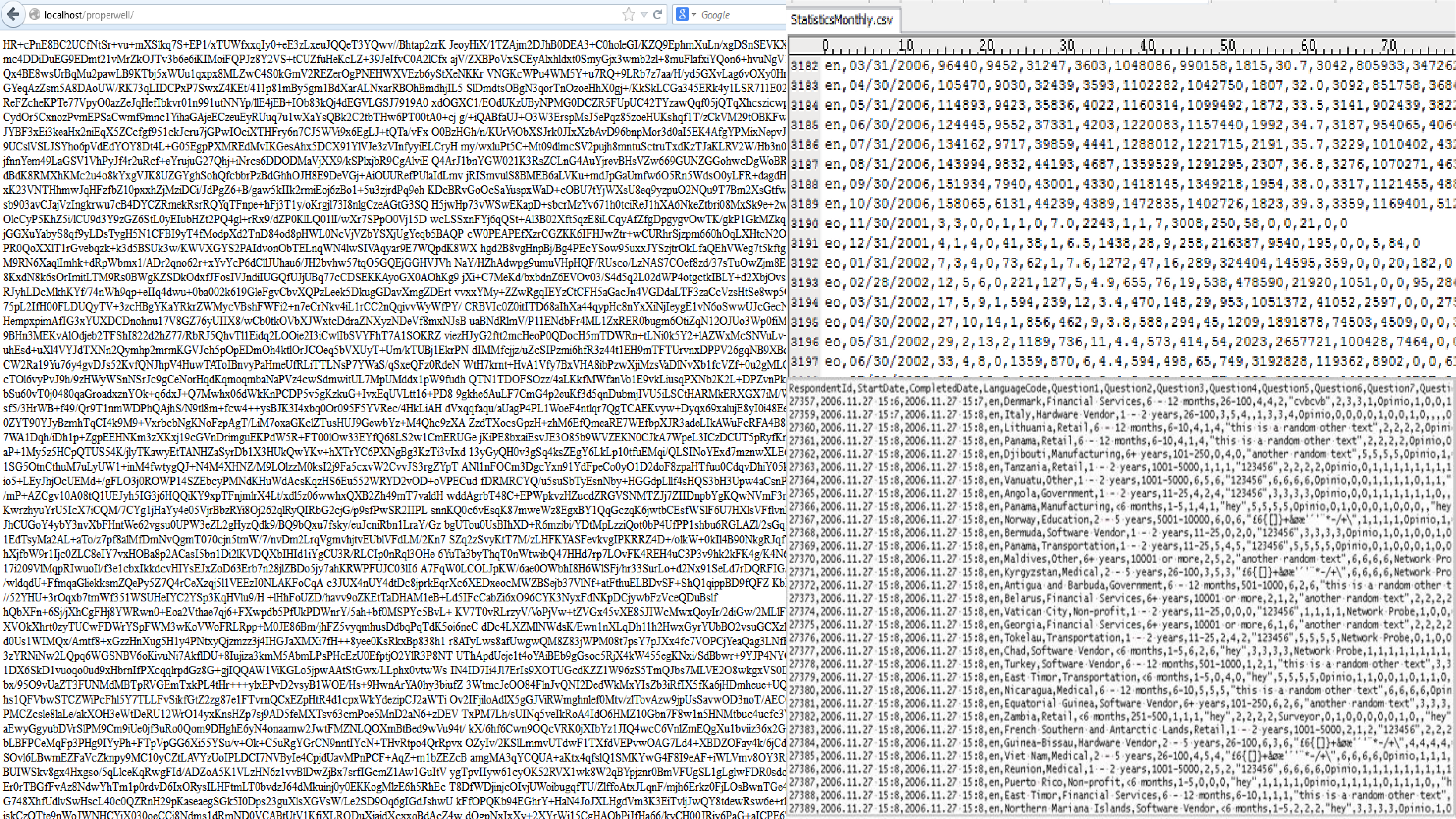
Position	Experience	Skill	Country	City	Salary ($)
Developer	0	1	USA	New York	103100
Developer	1	1	USA	New York	104900
Developer	2	1	USA	New York	106800
Developer	3	1	USA	New York	108700
Developer	4	1	USA	New York	110400
Developer	5	1	USA	New York	112300
Developer	6	1	USA	New York	114200
Developer	7	1	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

Classic Data Sets

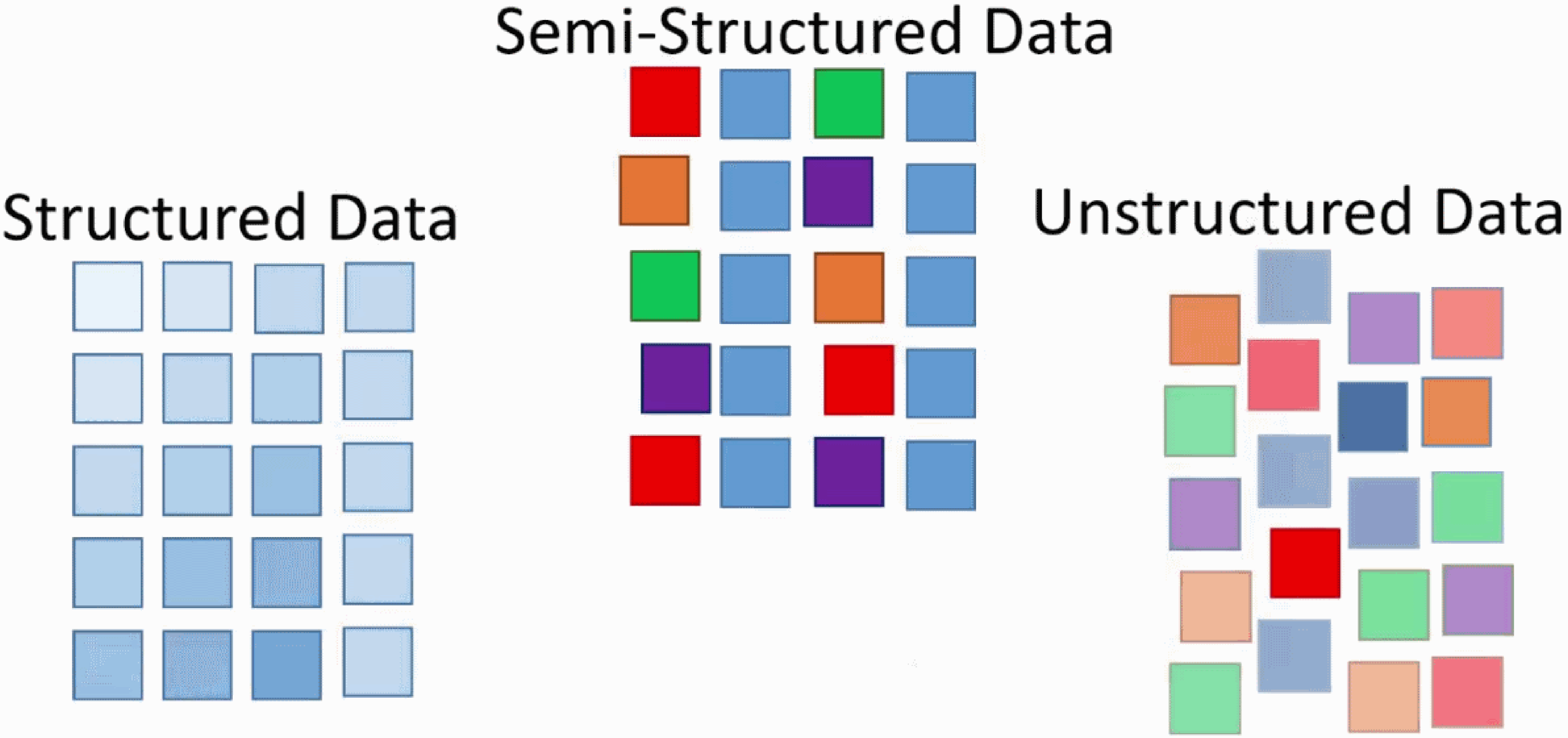
- Iris flower data set – Multivariate data set introduced by Ronald Fisher (1936).[7]
- MNIST database – Images of handwritten digits commonly used to test classification, clustering, and image processing algorithms
- World University Rankings – Ranking universities can be difficult and controversial. There are hundreds of ranking systems, and they rarely reach a consensus. This dataset contains three global university rankings.
- IMDB 5000 Movie Dataset – This dataset explores the question of whether we can anticipate a movie's popularity before it's even released.
- Wine Quality (Regression) – Properties of red and white vinho verde wine samples from the north of Portugal. The goal is to model wine quality based on physicochemical tests. (We also have a tutorial.)
- Credit Card Default (Classification) – Predicting credit card default is a valuable and common use for machine learning. This rich dataset includes demographics, payment history, credit, and default data.
- US Census Data (Clustering) – Clustering based on demographics is a tried and true way to perform market research and segmentation.
- ImageNet – ImageNet hosts a computer vision competition every year, and many consider it to be the benchmark for modern performance. The current image dataset has 1000 different classes.
- YouTube 8M – Ready to tackle videos, but can't spare terabytes of storage? This dataset contains millions of YouTube video ID's and *billions* of audio and visual features that were pre-extracted using the latest deep learning models.
- EOD Stock Prices - End of day stock prices, dividends, and splits for 3,000 US companies, curated by the Quandl community.
- Zillow Real Estate Research - Home prices and rents by size, type, and tier, sliced by zip code, neighborhood, city, metro area, county and state.
- Global Education Statistics - Over 4,000 internationally comparable indicators for education access, progression, completion, literacy, teachers, population, and expenditures.
- Million Song Dataset - Large, rich dataset for music recommendations. You can start with a pure collaborative filter and then expand it with other methods such as content-based models or web scraping.

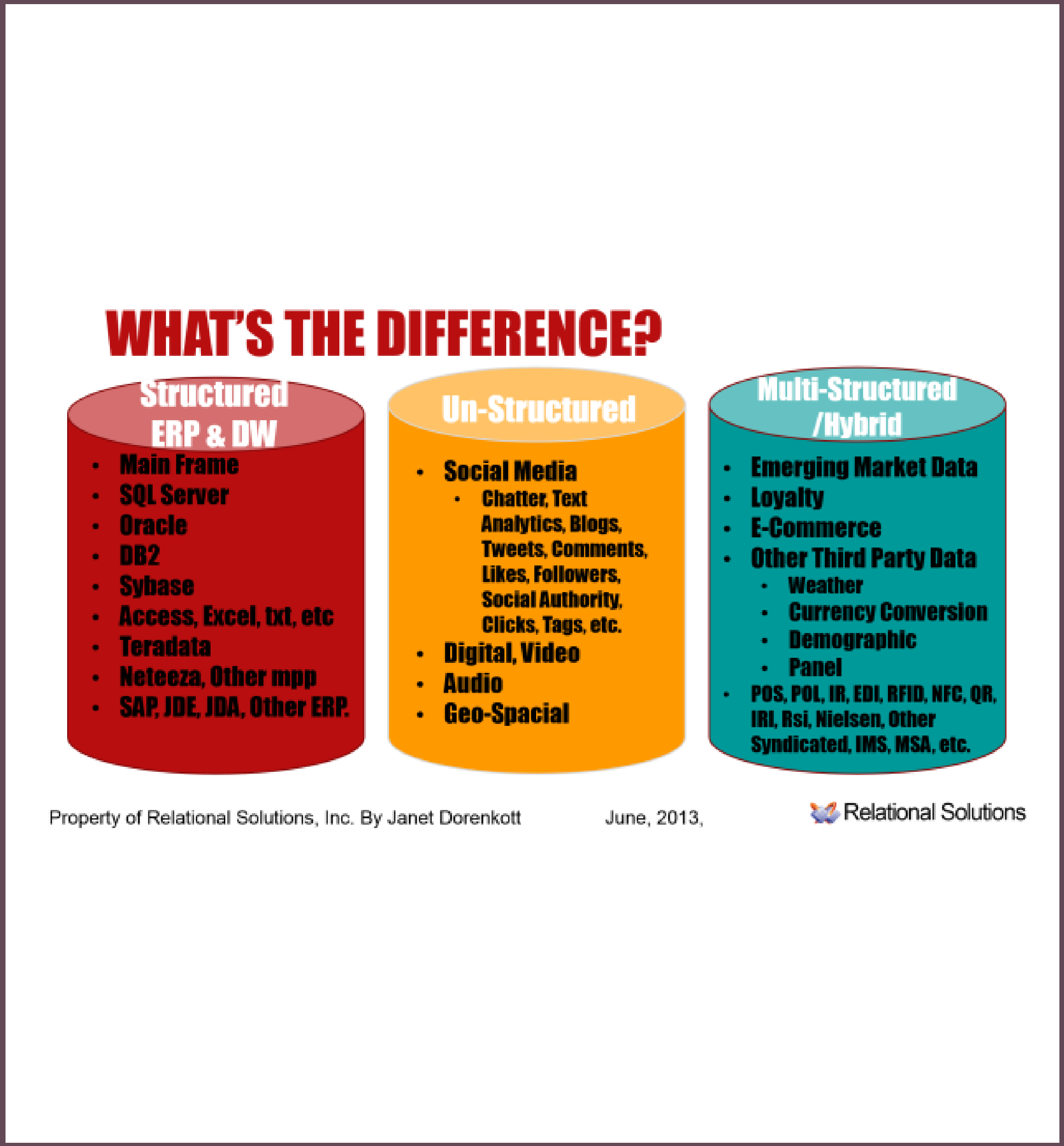
Types of Data

- Raw Data
- Structured Data
- Unstructured Data


localhost/properwell/
Google
StatisticsMonthly.csv

Structured, Unstructured and Semi-Structured

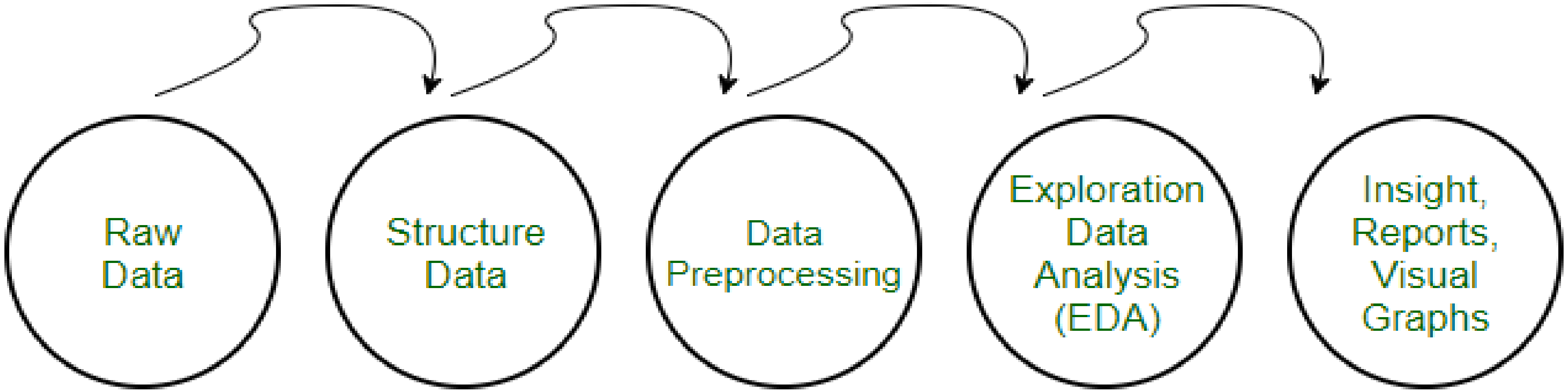



WHAT'S THE DIFFERENCE?
Structured
ERP & DW
Main Frame
SQL Server
Oracle
DB2
Sybase
Access, Excel, txt, etc
Teradata
Neteeza, Other mpp
SAP, JDE, JDA, Other ERP.
Un-Structured
Social Media
Chatter, Text Analytics, Blogs, Tweets, Comments, Likes, Followers, Social Authority, Clicks, Tags, etc.
Digital, Video
Audio
Geo-Spacial
Multi-Structured /Hybrid
Emerging Market Data
Loyalty
E-Commerce
Other Third Party Data
Weather
Currency Conversion
Demographic
Panel
POS, POL, IR, EDI, RFID, NFC, QR, IRI, Rsi, Nielsen, Other Syndicated, IMS, MSA, etc.
Property of Relational Solutions, Inc. By Janet Dorenkott
June, 2013,
Relational Solutions


Structured Data
Unstructured Data
0.103 0.176 0.387 0.300 0.379
0.333 0.384 0.564 0.587 0.857
0.421 0.309 0.654 0.729 0.228
0.266 0.750 1.056 0.936 0.911
0.225 0.326 0.643 0.337 0.721
0.187 0.586 0.529 0.340 0.829
0.153 0.485 0.560 0.428 0.628
PDF

Data Preprocessing for Machine learning in Python

- Pre-processing refers to the transformations applied to our data before feeding it to the algorithm.
- Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.



Need of Data Preprocessing

- For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner.
- Some specified Machine Learning model needs information in a specified format, for example, *Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set*.
- Another aspect is that data set should be formatted in such a way *that more than one Machine Learning and Deep Learning algorithms* are executed in one data set, and best out of them is chosen.

Data Quality: Why Preprocess the Data?

- Measures for data quality: A multidimensional view
  - **Accuracy:** correct or wrong, accurate or not
  - **Completeness:** not recorded, unavailable, …
  - **Consistency:** some modified but some not, dangling, …
  - **Timeliness:** timely update?
  - **Believability:** how trustable the data are correct?
  - **Interpretability:** how easily the data can be understood?

Major Tasks in Data Preprocessing

- **Data cleaning**
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- **Data integration**
  - Integration of multiple databases, data cubes, or files
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
  - Data compression
- **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation

Data Cleaning

- **Data in the Real World Is Dirty:** Lots of potentially incorrect data, e.g., instrument faulty, human or computer error, transmission error
  - **Incomplete(missing):** lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., *Occupation*=“ ” (missing data)
  - **noisy:** containing noise, errors, or outliers
    - e.g., *Salary*=“−10” (an error)
  - **inconsistent:** containing discrepancies in codes or names, e.g.,
    - *Age*=“42”, *Birthday*=“03/07/2010”
    - Was rating “1, 2, 3”, now rating “A, B, C”
    - discrepancy between duplicate records
  - **Intentional (e.g., *disguised missing* data)**
    - Jan. 1 as everyone’s birthday?

Incomplete (Missing) Data

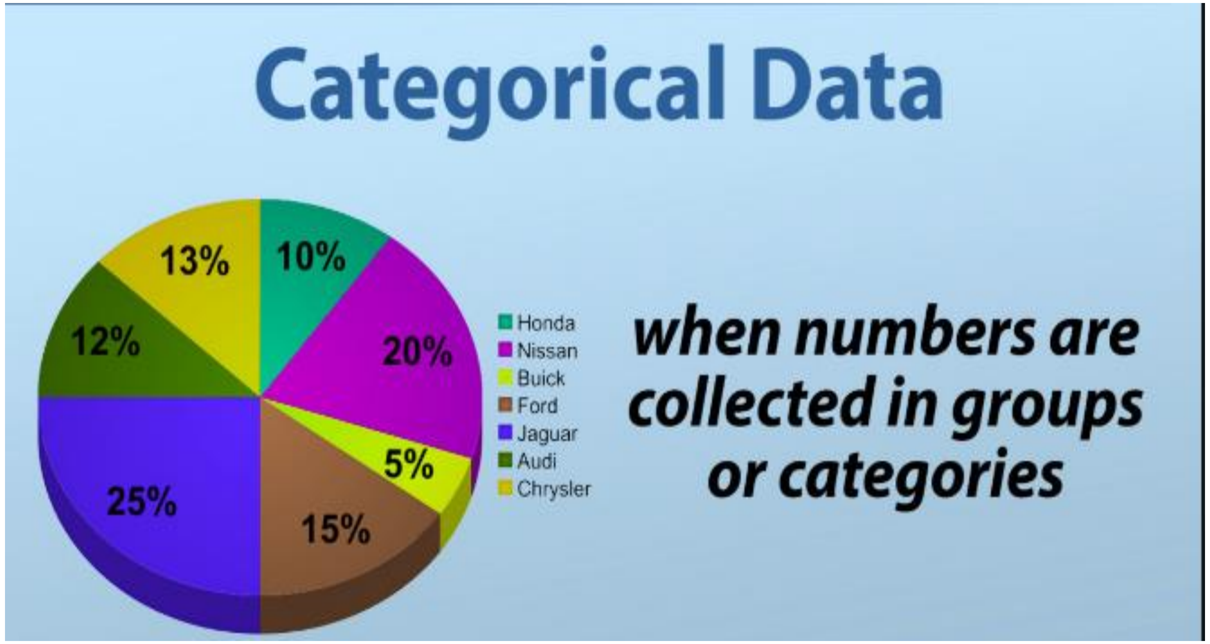
- **Data is not always available**
  - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- **Missing data may be due to**
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data
- **Missing data may need to be inferred**

How to Handle Missing Data?

- **Ignore the tuple:** usually done when class label is missing (when doing classification)—not effective when the % of missing values per attribute varies considerably
- **Fill in the missing value manually**: tedious + infeasible?
- **Fill in it automatically with**
    - a global constant : e.g., “unknown”, a new class?!
    - the attribute mean
    - the attribute mean for all samples belonging to the same class: smarter
    - the most probable value: inference-based such as Bayesian formula or decision tree

	name	gender	height	weight	age
0	Michael	None	123.0	10.0	14.0
1	Jessica	F	145.0	NaN	NaN
2	Sue	NaN	100.0	30.0	29.0
3	Jake	F	NaN	NaN	NaN
4	Amy	NaN	NaN	NaN	52.0
5	Tye	M	150.0	20.0	45.0

	A	B	C	D	E	F	G
3	No missing data				Missing data		
4							
5	Id	math	science		Id	math	science
6	1	14	27		1	14	27
7	2	13	29		2	13	29
8	3	23	49		3	23	49
9	4	19	37		4	19	37
10	5	21	31		5	21	31
11	6	25	40		6	25	40
12	7	18	35		7	18	35
13	8	22	44		8	22	x
14	9	18	32		9	18	x
15	10	28	48		10	28	x
16	11	25	43		11	25	x
17	12	17	35		12	17	x
18	mean	20.25	37.5		mean	20.25	35.42857
19	stdev	4.57513	7.317476		stdev	4.57513	7.524563
20	correl	0.845864			correl	0.769171	

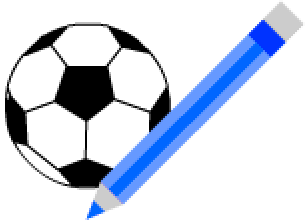


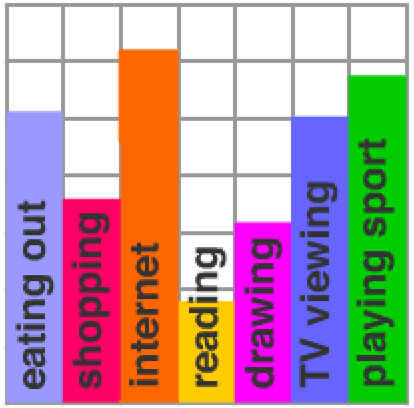
Degree	Frequency
High School	2
Bachelor's	7
MBA	20
Master's	3
Law	4
PhD	4
	40

categorical data

also known as qualitative data

Leisure Activities

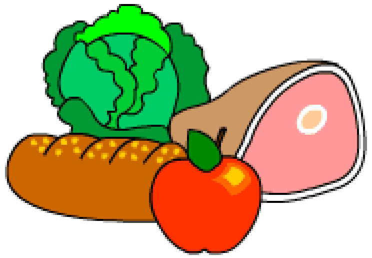




categorical data

data categories which may include things like skills, preferences, homes, schools, food and hobbies.

Favourite Food Groups



categorical data

Noisy Data

- **Noise**: random error or variance in a measured variable
- **Incorrect attribute values** may be due to
  - faulty data collection instruments
  - data entry problems
  - data transmission problems
  - technology limitation
  - inconsistency in naming convention
- **Other data problems** which require data cleaning
  - duplicate records
  - incomplete data
  - inconsistent data

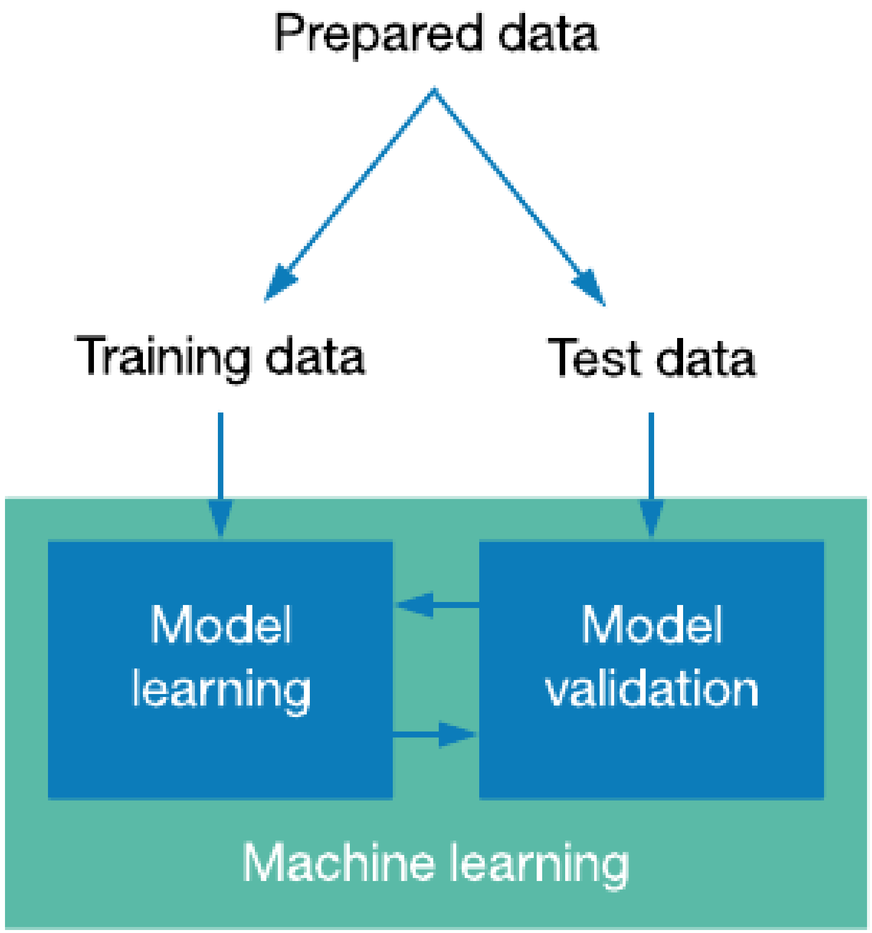
Preprocessing the Data in Python

- In our daily life, we deal with lots of data but this data is in raw form. To provide the data as the input of machine learning algorithms, we need to convert it into a meaningful data.
- That is where data preprocessing comes into picture. In other simple words, we can say that before providing the data to the machine learning algorithms we need to preprocess the data.

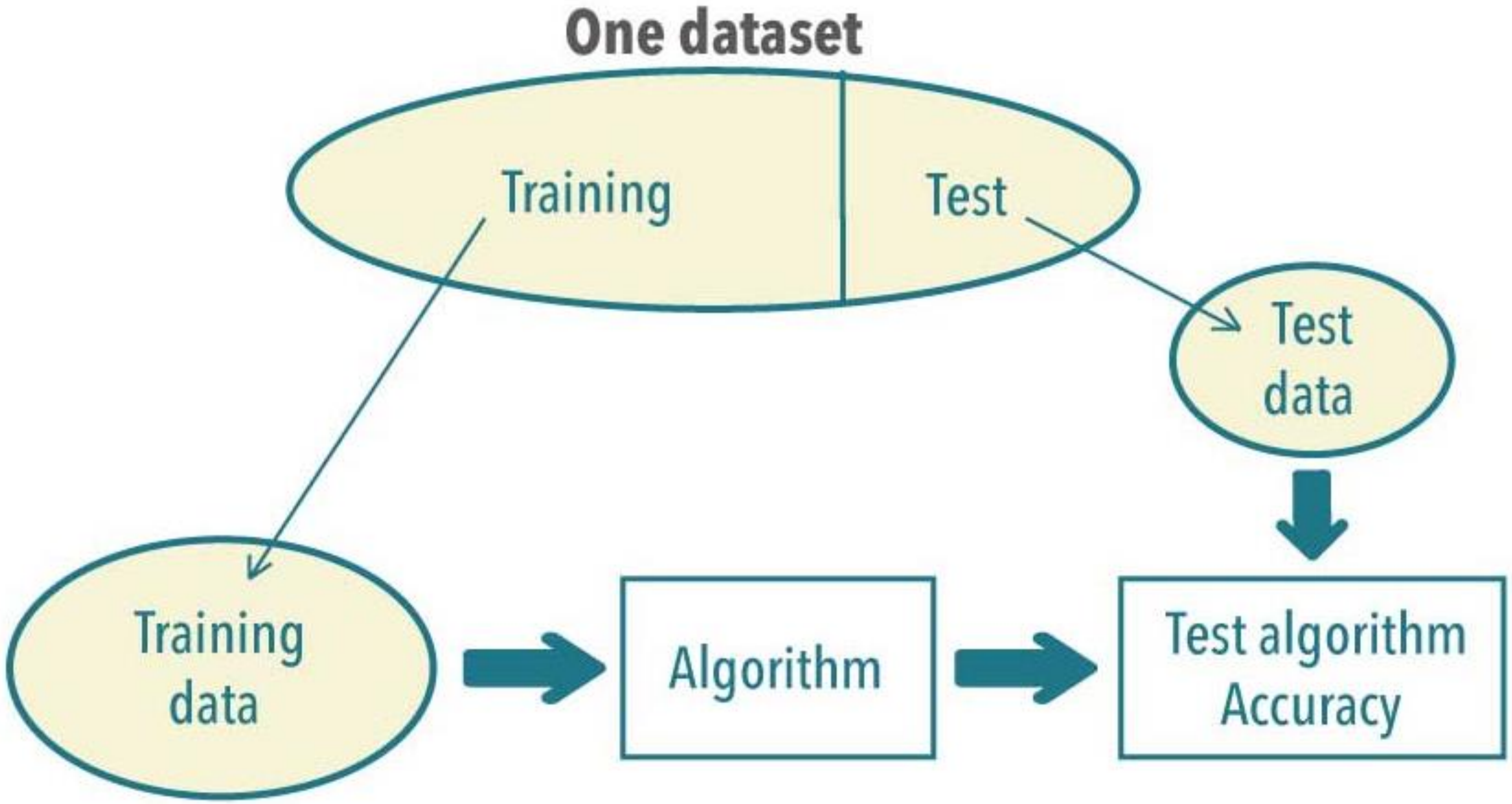
Importing Libraries

Hands On

Training and test data



Training data vs. test data

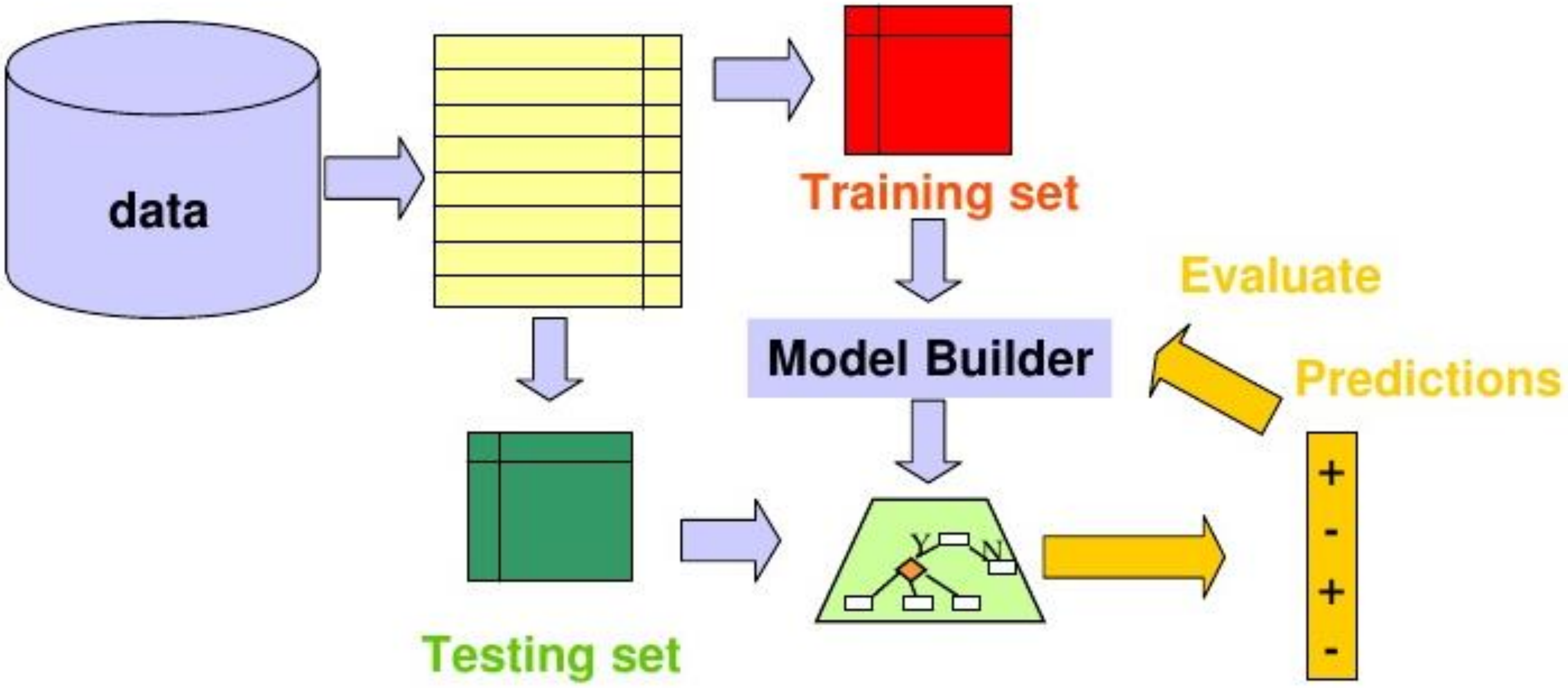


A sample dataset to be used is divided like this:

Training Dataset

Testing Dataset

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2


data
Training set
Model Builder
Evaluate
Predictions
Y
N
+
-
+
-
Testing set

Training , Testing and Validation Data set

- **Training Dataset**: *A set of examples used for learning, that is to fit the parameters [i.e., weights] of the classifier or model.*
- **Validation Dataset**: *The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. A set of examples used to tune the parameters [i.e., architecture, not weights] of a classifier, for example to choose the number of hidden units in a neural network.*
- **Test Dataset**: *The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset*. *A set of examples used only to assess the performance [generalization] of a fully specified classifier.*
  - **Training set:** Is used for finding Nearest neighbors.
  - **Validation set:** Is for finding different k which is applying to train set.
  - **Test set**: Is used for finding the maximum accuracy and unseen data in future.

Hands on

Data preprocessing steps

- **Step 1 – Importing the useful packages**
- **Step 2 – Defining sample data**
- **Step3 – Applying preprocessing technique**

Hands on

Step 1 – Importing the useful packages

- If we are using Python then this would be the first step for converting the data into a certain format, i.e., preprocessing. It can be done as follows −
  - **import numpy as np**
  - **from sklearn import preprocessing**
- Here we have used the following two packages −
  - **NumPy** − Basically NumPy is a general purpose array-processing package designed to efficiently manipulate large multi-dimensional arrays of arbitrary records without sacrificing too much speed for small multi-dimensional arrays.
  - **Sklearn.preprocessing** − This package provides many common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for machine learning algorithms.

Step 2 – Defining sample data

- After importing the packages, we need to define some sample data so that we can apply preprocessing techniques on that data.
- We will now define the following sample data –

```
input_data = np.array([2.1, -1.9, 5.5],
                      [-1.5, 2.4, 3.5],
                      [0.5, -7.9, 5.6],
                      [5.9, 2.3, -5.8 ]])
```

Step3 – Applying preprocessing technique

- In this step, we need to apply any of the preprocessing techniques.

Techniques for Data Preprocessing

1. Scaling /Rescaling : **# Min max scaling**
2. **Standardization**
3. Normalization
    1. **L1 Normalization**
    2. **L2 Normalization**
4. Binarization
5. Mean Removal
6. Labeling the Data

1. Scaling or Rescaling Data

- When your data is comprised of attributes with varying scales, many machine learning algorithms can benefit from rescaling the attributes to all have the same scale.
- The attributes are often rescaled into the ***range between 0 and 1***. This is useful for optimization algorithms in used in the core of machine learning algorithms like gradient descent. It is also useful for algorithms that weight inputs like regression and neural networks and algorithms that use distance measures like K-Nearest Neighbors.
- You can rescale your data using ***scikit-learn*** using the ***MinMaxScaler*** class.

Example : Min max scaled data

Min max scaled data

```
data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0,1))
data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
print ("\nMin max scaled data:\n", data_scaled_minmax)
```

```
Min max scaled data:
 [[1.         0.         1.         0.        ]
 [0.         1.         0.27118644 1.        ]
 [0.33333333 0.84444444 0.         0.2       ]]
```

2 . Standardize Data

- Standardization is a useful technique to transform attributes with a Gaussian distribution and differing means and standard deviations to a standard Gaussian distribution with a *mean of 0 and a standard deviation of 1.*
- It is most suitable for techniques that assume a Gaussian distribution in the input variables and work better with rescaled data, such as linear regression, logistic regression and linear discriminate analysis.
- You can standardize data using scikit-learn with the StandardScaler class.

```python
# Python code to Standardize data (0 mean, 1 stdev)
from sklearn.preprocessing import StandardScaler
import pandas
import numpy
#url = "https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data"
url = "C:\\Users\\Administrator\\Desktop\\Data\\pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values

# separate array into input and output components
X = array[:,0:8]
Y = array[:,8]
scaler = StandardScaler().fit(X)
rescaledX = scaler.transform(X)

# summarize transformed data
numpy.set_printoptions(precision=3)
print(rescaledX[0:5,:])
```

```
[[ 0.64   0.848  0.15   0.907 -0.693  0.204  0.468  1.426]
 [-0.845 -1.123 -0.161  0.531 -0.693 -0.684 -0.365 -0.191]
 [ 1.234  1.944 -0.264 -1.288 -0.693 -1.103  0.604 -0.106]
 [-0.845 -0.998 -0.161  0.155  0.123 -0.494 -0.921 -1.042]
 [-1.142  0.504 -1.505  0.907  0.766  1.41   5.485 -0.02 ]]
```

3. Normalization

- Normalizing in scikit-learn refers to rescaling each observation (row) to have a length of 1 (called a unit norm in linear algebra).
- This preprocessing can be useful for sparse datasets (lots of zeros) with attributes of varying scales when using algorithms that weight input values such as neural networks and algorithms that use distance measures such as K-Nearest Neighbors.
- You can normalize data in Python with scikit-learn using the Normalizer class.
- It is another data preprocessing technique that is used to modify the feature vectors. Such kind of modification is necessary to measure the feature vectors on a common scale. Followings are two types of normalization which can be used in machine learning –

L1 Normalization

- It is also referred to as Least Absolute Deviations. This kind of normalization modifies the values so that the sum of the absolute values is always up to 1 in each row. It can be implemented on the input data with the help of the following Python code –

```
# Normalize data
data_normalized_l1 = preprocessing.normalize(input_data, norm = 'l1')
print("\nL1 normalized data:\n", data_normalized_l1)
```

```
L1 normalized data:
 [[ 0.21582734 -0.10791367  0.21582734 -0.46043165]
 [ 0.          0.35714286 -0.1547619   0.48809524]
 [ 0.0952381   0.21904762 -0.27619048 -0.40952381]]
```

L2 Normalization

- It is also referred to as least squares. This kind of normalization modifies the values so that the sum of the squares is always up to 1 in each row. It can be implemented on the input data with the help of the following Python code –

```
# Normalize data
data_normalized_l2 = preprocessing.normalize(input_data, norm = 'l2')
print("\nL2 normalized data:\n", data_normalized_l2)
```

```
L2 normalized data:
 [[ 0.38345117 -0.19172558  0.38345117 -0.81802916]
 [ 0.          0.57207755 -0.24790027  0.78183932]
 [ 0.17357868  0.39923096 -0.50337816 -0.74638831]]
```

```python
# Normalize data (length of 1)
from sklearn.preprocessing import Normalizer
import pandas
import numpy
#url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
url = "C:\\Users\\Administrator\\Desktop\\Data\\pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
# separate array into input and output components
X = array[:,0:8]
Y = array[:,8]
scaler = Normalizer().fit(X)
normalizedX = scaler.transform(X)
# summarize transformed data
numpy.set_printoptions(precision=3)
print(normalizedX[0:5,:])
```

```
[[0.034 0.828 0.403 0.196 0.    0.188 0.004 0.28 ]
 [0.008 0.716 0.556 0.244 0.    0.224 0.003 0.261]
 [0.04  0.924 0.323 0.    0.    0.118 0.003 0.162]
 [0.007 0.588 0.436 0.152 0.622 0.186 0.001 0.139]
 [0.    0.596 0.174 0.152 0.731 0.188 0.01  0.144]]
```

4.Binarization

- You can transform your data using a binary threshold. All values above the threshold are marked 1 and all equal to or below are marked as 0.
- This is called binarizing your data or threshold your data. It can be useful when you have probabilities that you want to make crisp values. It is also useful when feature engineering and you want to add new features that indicate something meaningful.
- You can create new binary attributes in Python using scikit-learn with the Binarizer class.

```
data_binarized = preprocessing.Binarizer(threshold = 0.5).transform(Input_data)
print("\nBinarized data:\n", data_binarized)
```

```
Binarized data:
 [[1. 0. 1.]
 [0. 1. 1.]
 [0. 0. 1.]
 [1. 1. 0.]]
```

5. Mean Removal

- It is another very common preprocessing technique that is used in machine learning. Basically it is used to eliminate the mean from feature vector so that every feature is centered on zero.
- We can also remove the bias from the features in the feature vector. For applying mean removal preprocessing technique on the sample data, we can write the Python code shown below.
- The code will display the Mean and Standard deviation of the input data – Now, the code below will remove the Mean and Standard deviation of the input data –

```
print("Mean = ", Input_data.mean(axis = 0))
print("Std deviation = ", Input_data.std(axis = 0))
```

```
Mean =  [ 1.75  -1.275  2.2  ]
Std deviation =  [2.714 4.2   4.694]
```

6. Labelling the Data

- We already know that data in a certain format is necessary for machine learning algorithms. Another important requirement is that the data must be labelled properly before sending it as the input of machine learning algorithms. For example, if we talk about classification, there are lot of labels on the data. Those labels are in the form of words, numbers, etc. Functions related to machine learning in sklearn expect that the data must have number labels. Hence, if the data is in other form then it must be converted to numbers. This process of transforming the word labels into numerical form is called label encoding.
- **Label encoding steps**
- Follow these steps for encoding the data labels in Python −
- **Step 1 − Importing the useful packages**¶
- If we are using Python then this would be first step for converting the data into certain format, i.e., preprocessing. It can be done as follows −

Label encoding steps

- **Step 1 – Importing the useful packages**
- **Step 2 – Defining sample labels**
- **Step 3 – Creating & training of label encoder object**
- **Step 4 – Checking the performance by encoding random ordered list**
- **Step 5 – Checking the performance by decoding a random set of numbers –**

Step 1 – Importing the useful packages¶

If we are using Python then this would be first step for converting the data into certain format, i.e., preprocessing. It can

```
import numpy as np
from sklearn import preprocessing
```

Step 2 – Defining sample labels

After importing the packages, we need to define some sample labels so that we can create and train the label encoder. labels –

```
# Sample input labels
input_labels = ['red','black','red','green','black','yellow','white']
```

Step 3 – Creating & training of label encoder object

In this step, we need to create the label encoder and train it. The following Python code will help in doing this –

```
# Creating the label encoder
encoder = preprocessing.LabelEncoder()
encoder.fit(input_labels)
```

```
LabelEncoder()
```

Step 4 – Checking the performance by encoding random ordered list

This step can be used to check the performance by encoding the random ordered list. Following Python code can be written to do the same
get printed as follows –

```
# encoding a set of labels
test_labels = ['green','red','black']
encoded_values = encoder.transform(test_labels)
print("\nLabels =", test_labels)
```

Step 4 – Checking the performance by encoding random ordered list

This step can be used to check the performance by encoding the random ordered list. Following Python code can be written to do the same – The labels would get printed as follows –

```
# encoding a set of labels
test_labels = ['green','red','black']
encoded_values = encoder.transform(test_labels)
print("\nLabels =", test_labels)
```

```
Labels = ['green', 'red', 'black']
```

```
print("Encoded values =", list(encoded_values))
```

```
Encoded values = [1, 2, 0]
```

Step 5 – Checking the performance by decoding a random set of numbers –

This step can be used to check the performance by decoding the random set of numbers. Following Python code can be written to do the same –

```
# decoding a set of values
encoded_values = [3,0,4,1]
decoded_list = encoder.inverse_transform(encoded_values)
print("\nEncoded values =", encoded_values)
```

```
Encoded values = [3, 0, 4, 1]
```

```
C:\Users\Administrator\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWarning: The truth value of an empty array is ambiguous. Returning False, but in future this will result in an error. Use `array.size > 0` to check that an array is not empty.
  if diff:
```

```
print("\nDecoded labels =", list(decoded_list))
```

```
Decoded labels = ['white', 'black', 'yellow', 'green']
```

Hands on