ICDS Spring 2025

Data Science: Part 1

Introduction to Machine Learning

What will be involved in this topic?

In lectures,

- Basic concepts in machine learning (few math involved)
- Unsupervised learning and Supervised learning (e.g., K-means, K-NN, regression)
- Neural networks (e.g., CNN)
- Reinforcement learning (Q-learning)

In recitations,

- The unit project starts: 3 sessions - 15% of final grade
- Programming exercises

The evolution of data science

Data science is a broad, multidisciplinary field, which involves statistics, data analytics, data modeling, machine learning, programming, and etc.

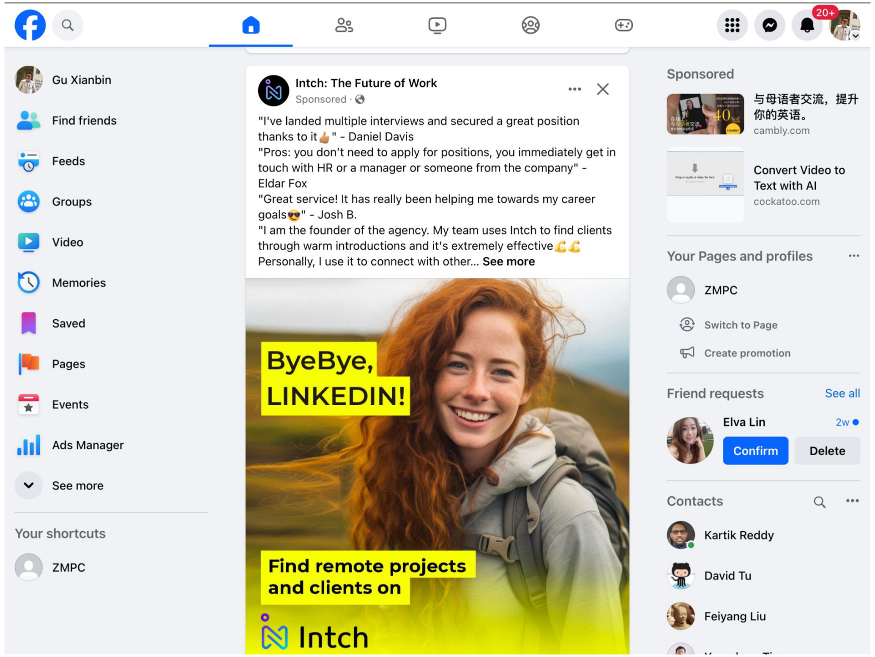
- **Origin:** computer science + statistics (in 1970s)
- **Present:** it becomes a ubiquitous technology in academics and industries. Some hot subjects nowadays:
    - Big data: database tools and techniques developed for big data
    - Machine learning and AI: algorithms and models for data analytics
- **Future:**
    - Generative AI: models learn the patterns and structure from the input data and then generate new data that has similar characteristics
    - Quantum computing

Agenda

- Social network advertising
  - The K Nearest Neighbor algorithm
- What is machine learning?
  - Major tasks in machine learning
- Foundations of ML models
  - Inductive bias
  - Data/Feature vector
  - Distance
- Types of machine learning algorithms

Personalized advertising on social media

- How can we deliver relevant and engaging content to users?



Intch: The Future of Work
Sponsored ·

"I've landed multiple interviews and secured a great position thanks to it👍" - Daniel Davis
"Pros: you don't need to apply for positions, you immediately get in touch with HR or a manager or someone from the company" - Eldar Fox
"Great service! It has really been helping me towards my career goals😎" - Josh B.
"I am the founder of the agency. My team uses Intch to find clients through warm introductions and it's extremely effective💪💪
Personally, I use it to connect with other... **See more**

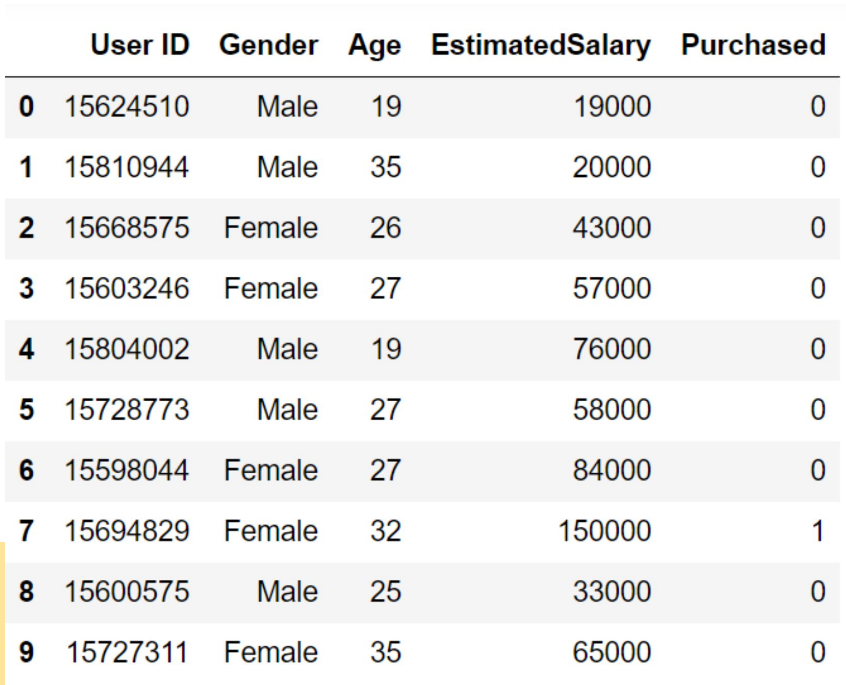
- We can use the data to solve this problem, i.e., a machine learning-based solution.

Example: the social network ad dataset

kaggle

It is a dataset of an advertisement of a product on a social networking site which contains,

- User information: **salary**, **age**, and **gender** of each users in the social networking site;
- Whether the user click the ad and buy the product: **Purchased** (0: not click, 1: click and buy)

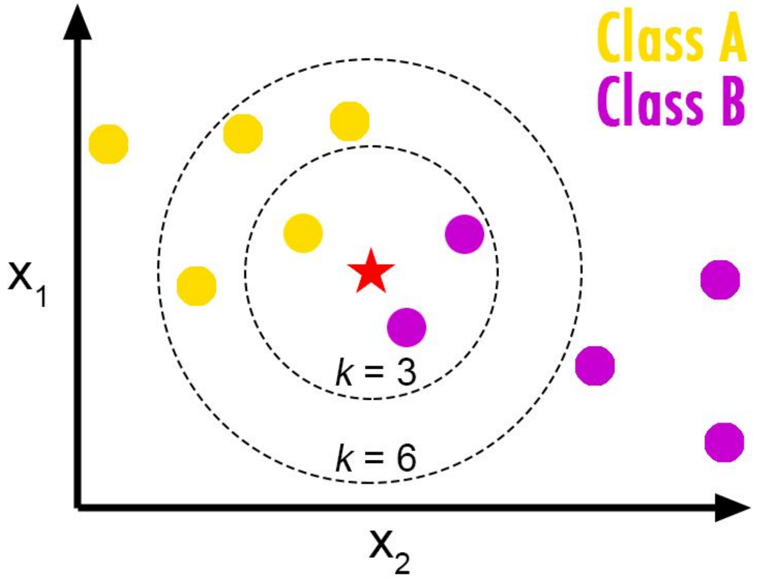


	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0

Now, there is a **new** user registered, and we have the salary, age, and gender. Shall we show this ad to him/her?

K-NN Algorithm

The K-Nearest Neighbor Algorithm



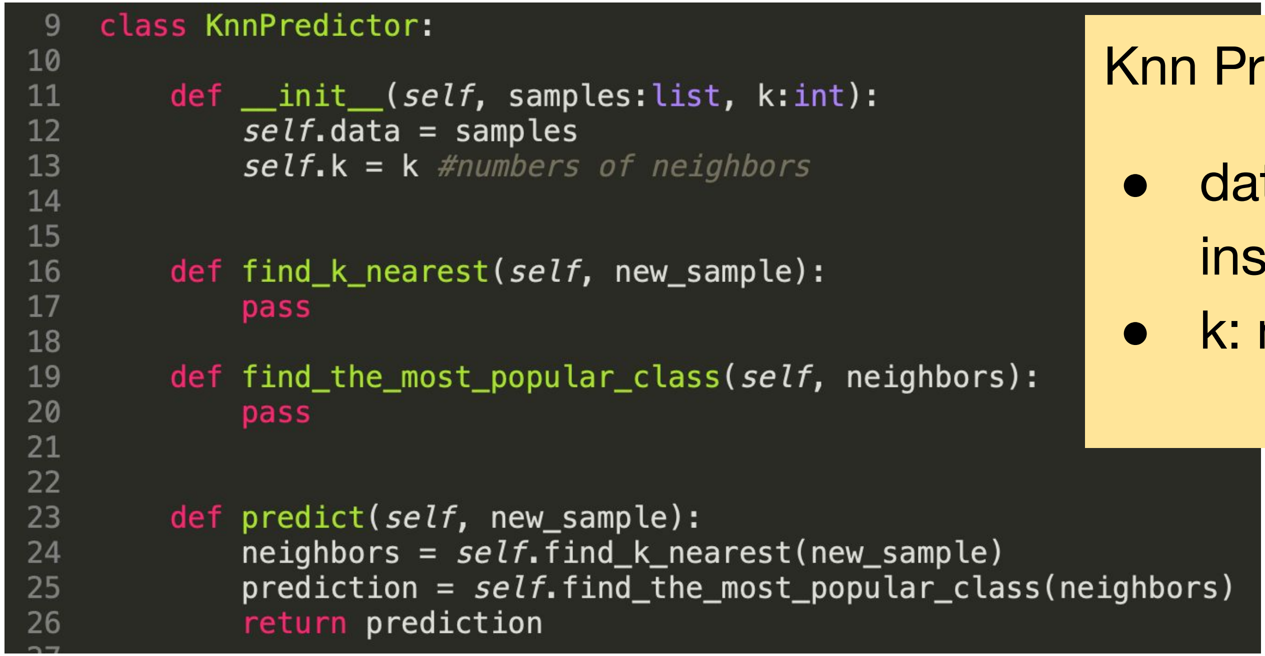
- **The logic:** Similar objects share the same label. ⇒ We predict the label of an object by the label of its nearest neighbor.
- Usually, we increase the confidence of the prediction by referring to a group of closest neighbors because there are noises in observations.
- The majority label among the k nearest neighbors will be the label of the object. (**the majority wins**)

k-NN: the algorithm

For a given new sample,

1. Find the k nearest samples
    - Calculate the distance to every known sample
    - Take the nearest k samples
2. Predict the label
    - find the most popular class of the k nearest samples
    - return it as the prediction

Programming K-NN in OOP



```python
class KnnPredictor:

    def __init__(self, samples:list, k:int):
        self.data = samples
        self.k = k #numbers of neighbors


    def find_k_nearest(self, new_sample):
        pass

    def find_the_most_popular_class(self, neighbors):
        pass


    def predict(self, new_sample):
        neighbors = self.find_k_nearest(new_sample)
        prediction = self.find_the_most_popular_class(neighbors)
        return prediction
```

Knn Predictor class:

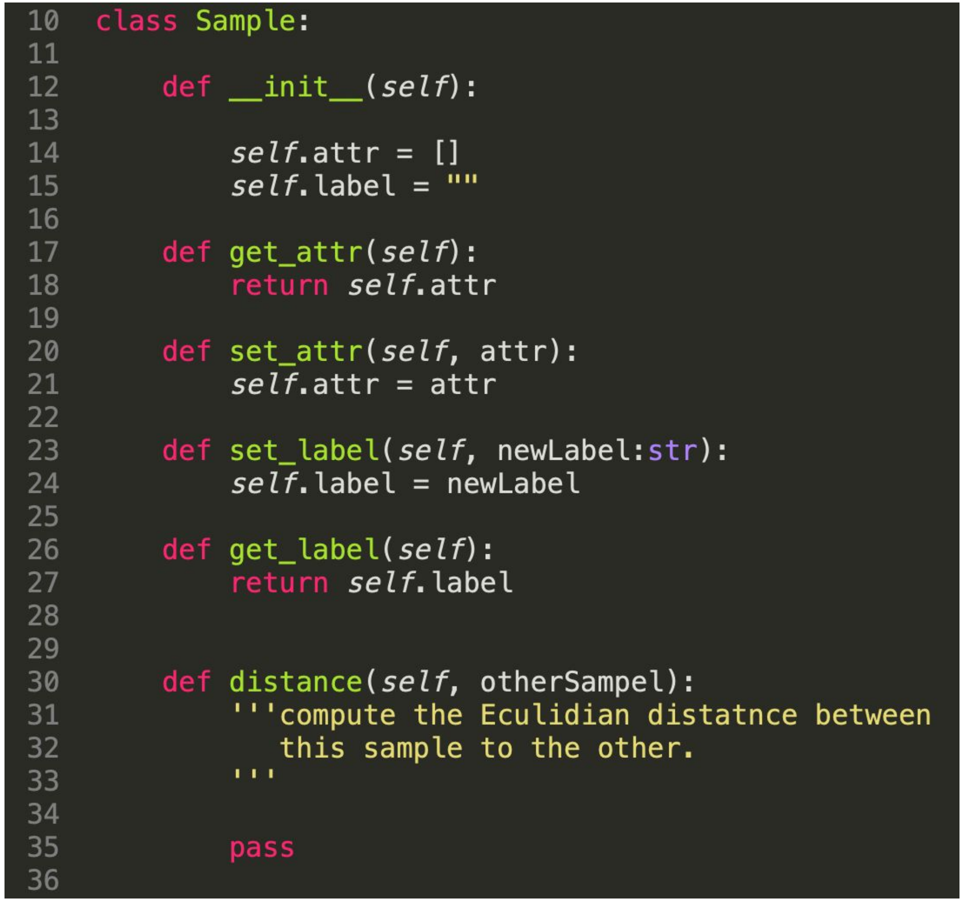
- data: a list of Sample instances
- k: number of neighbors

Sample class

It represents the data that k-NN will process.

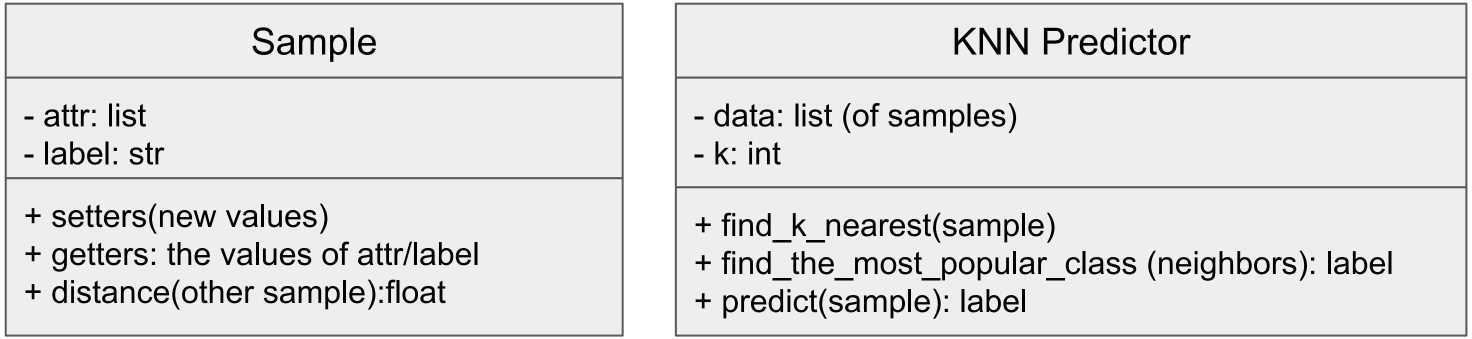
A sample instance should have:

- attributes: gender, age, salary, etc.
- label: the label
- Setters and getters
- distance: the distance between the sample to the other sample.



```
class Sample:

    def __init__(self):

        self.attr = []
        self.label = ""

    def get_attr(self):
        return self.attr

    def set_attr(self, attr):
        self.attr = attr

    def set_label(self, newLabel:str):
        self.label = newLabel

    def get_label(self):
        return self.label


    def distance(self, otherSampel):
        '''compute the Eculidian distatnce between
           this sample to the other.
        '''


        pass

```

Objects in k-NN



Sample

- attr: list
- label: str

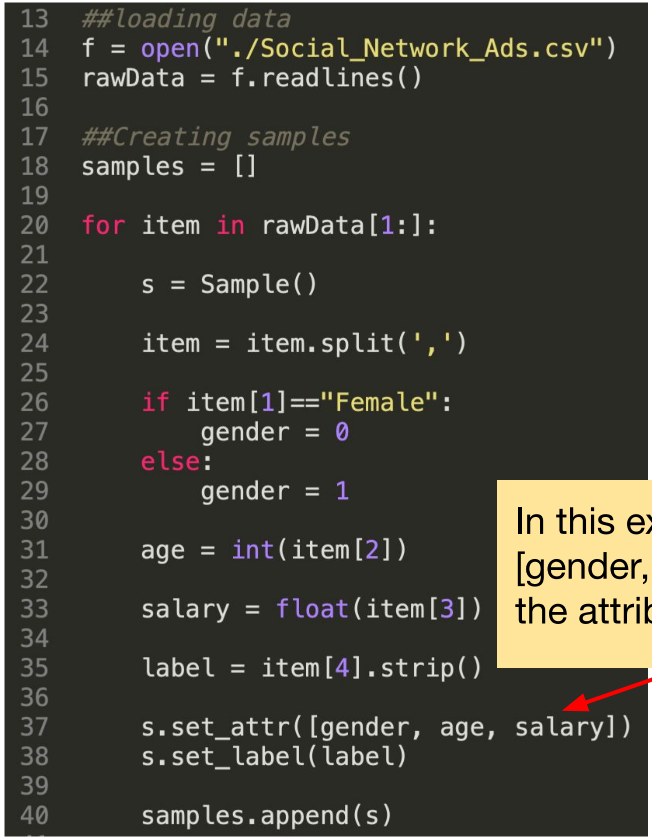
+ setters(new values)
+ getters: the values of attr/label
+ distance(other sample):float

KNN Predictor

- data: list (of samples)
- k: int

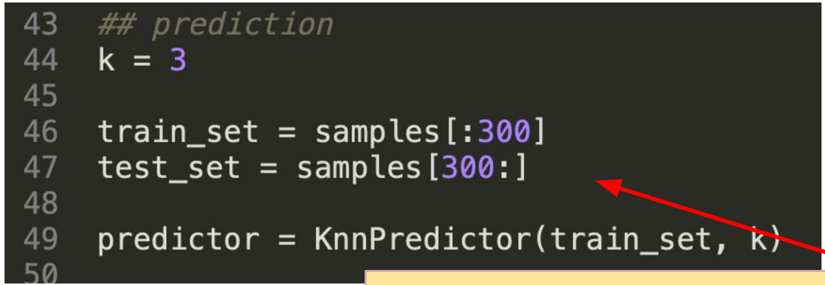
+ find_k_nearest(sample)
+ find_the_most_popular_class (neighbors): label
+ predict(sample): label

Run kNN on the dataset



```python
##loading data
f = open("./Social_Network_Ads.csv")
rawData = f.readlines()

##Creating samples
samples = []

for item in rawData[1:]:

    s = Sample()

    item = item.split(',')

    if item[1]=="Female":
        gender = 0
    else:
        gender = 1

    age = int(item[2])

    salary = float(item[3])

    label = item[4].strip()

    s.set_attr([gender, age, salary])
    s.set_label(label)

    samples.append(s)
```

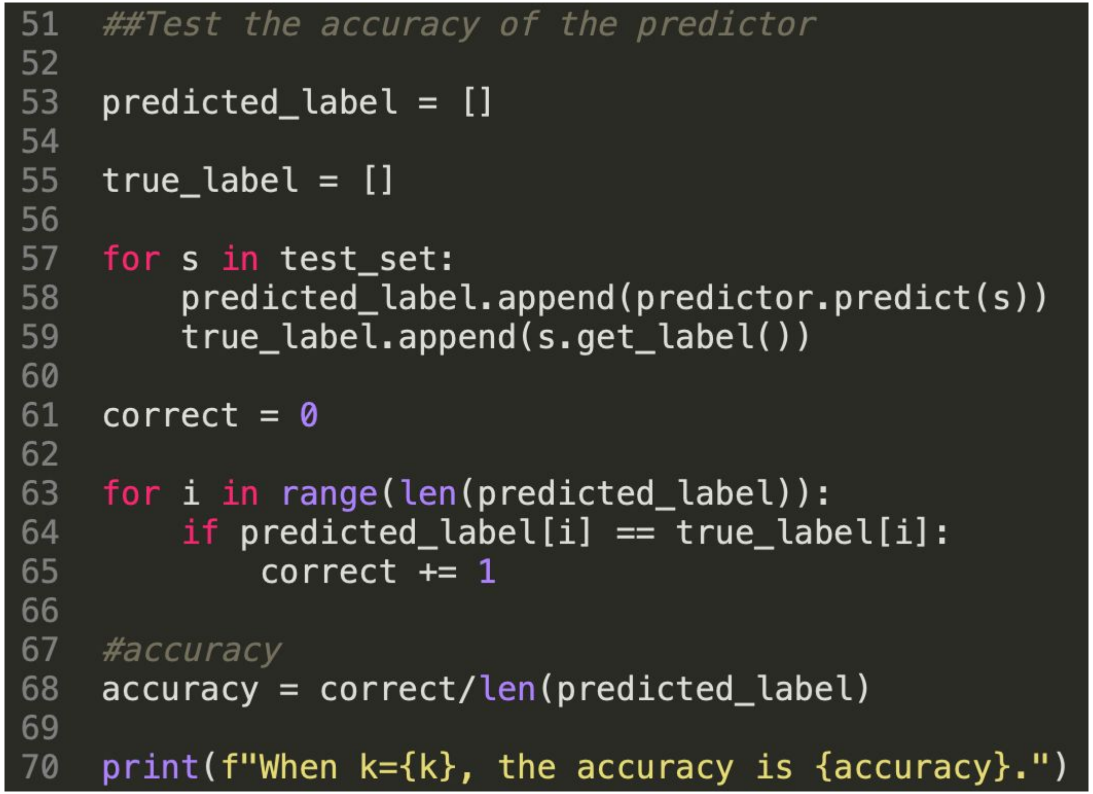
In this example, we use [gender, age, and salary] as the attributes of a sample.



```python
## prediction
k = 3

train_set = samples[:300]
test_set = samples[300:]

predictor = KnnPredictor(train_set, k)
```

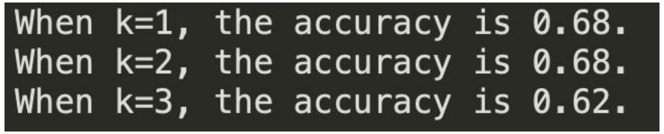
- We build a predictor using knn and set k = 3;
- We split the dataset into two parts: One is used for building the model (often called as training set); the other is for testing the model's performance (often called as test set).

How is the performance of our knn?



```python
##Test the accuracy of the predictor

predicted_label = []

true_label = []

for s in test_set:
    predicted_label.append(predictor.predict(s))
    true_label.append(s.get_label())

correct = 0

for i in range(len(predicted_label)):
    if predicted_label[i] == true_label[i]:
        correct += 1

#accuracy
accuracy = correct/len(predicted_label)

print(f"When k={k}, the accuracy is {accuracy}.")
```

- A simple measure of performance is called accuracy.
- Accuracy = number of correct predictions / number of total predictions
- The performance of the model is sensitive to the k.



```
When k=1, the accuracy is 0.68.
When k=2, the accuracy is 0.68.
When k=3, the accuracy is 0.62.
```

What is machine learning?

Definition: Machine Learning is PET

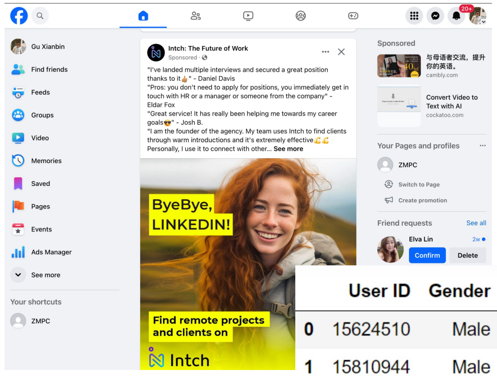
A computer program is said to **learn** from experience E with respect to some class of task T and performance measure P; if its performance at tasks in T, as measured by P, improves with experience E.

--- Tom Mitchell

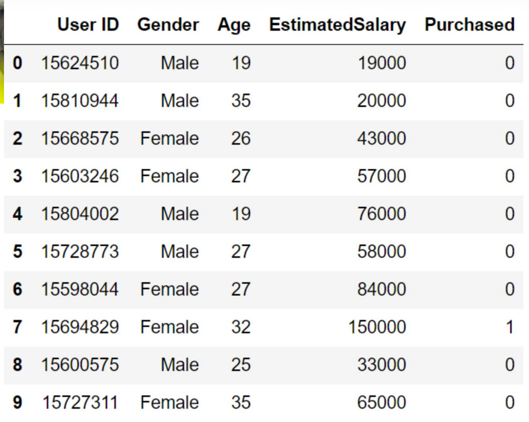
Tom Mitchell (1951-),
E. Fredkin University Professor, Machine learning department, School of Computer Science, CMU

Task: the job that the program is expected to do
Experience: the observations/dataset
Performance measure: some statistics that evaluate the program’s achievements

Solving social media advertising



"I've landed multiple interviews and secured a great position thanks to it👍" - Daniel Davis
"Pros: you don't need to apply for positions, you immediately get in touch with HR or a manager or someone from the company" - Eldar Fox
"Great service! It has really been helping me towards my career goals😎" - Josh B.
"I am the founder of the agency. My team uses Intch to find clients through warm introductions and it's extremely effective💪💪
Personally, I use it to connect with other... See more



	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0

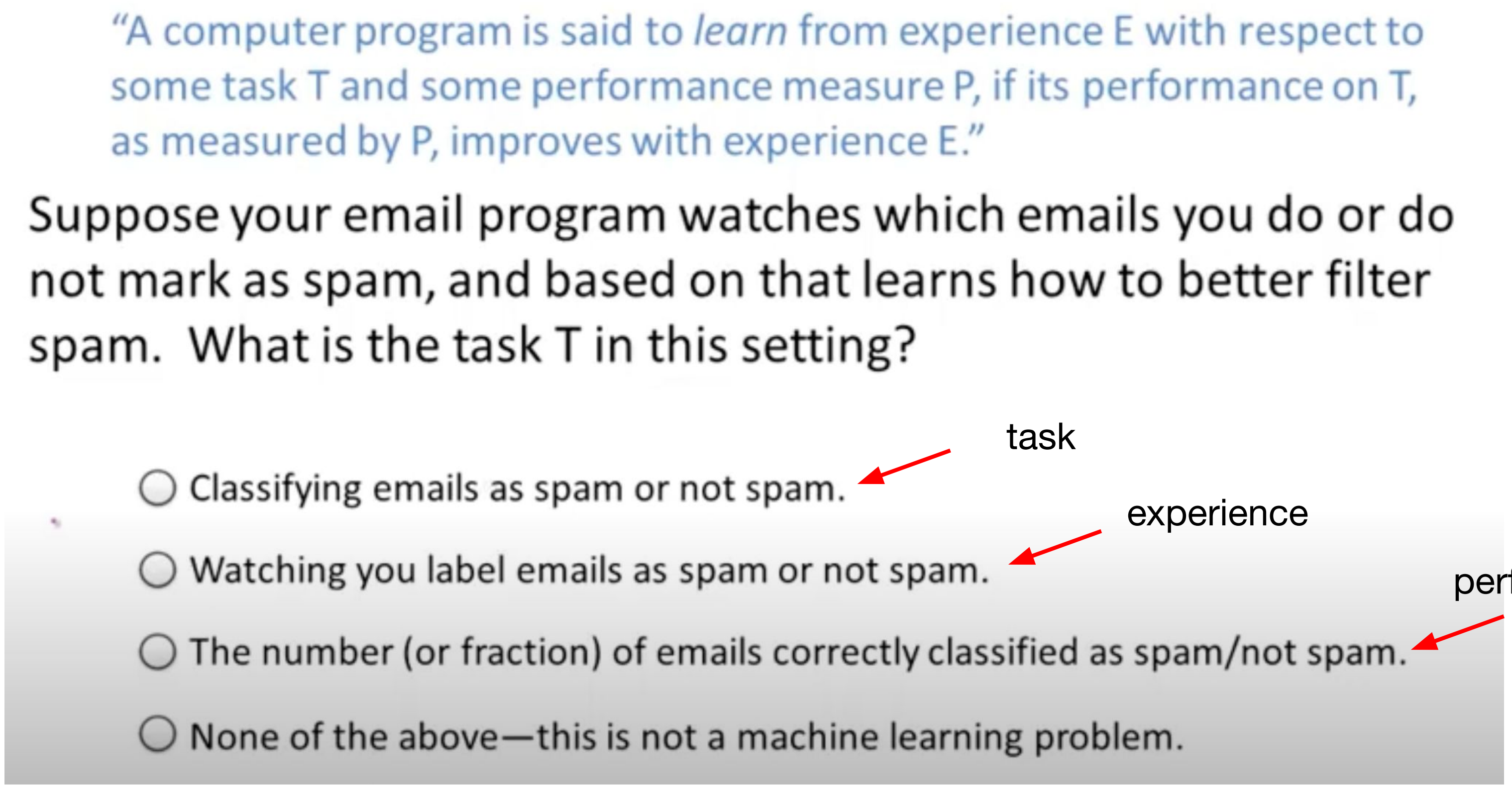
Task: pushing ads effectively to users

Experience: the dataset of users responses to the ads

Performance: accuracy of the algorithm

```
When k=1, the accuracy is 0.68.
When k=2, the accuracy is 0.68.
When k=3, the accuracy is 0.62.
```

Exercise: Spam Classifier



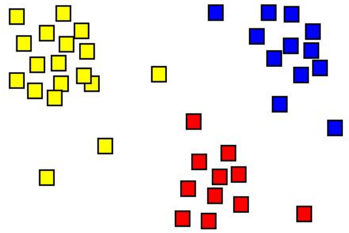
"A computer program is said to *learn* from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. What is the task T in this setting?

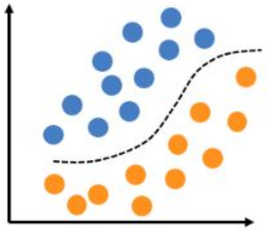
- ○ Classifying emails as spam or not spam. ← task
- ○ Watching you label emails as spam or not spam. ← experience
- ○ The number (or fraction) of emails correctly classified as spam/not spam. ← performance
- ○ None of the above—this is not a machine learning problem.

Major tasks for machine learning

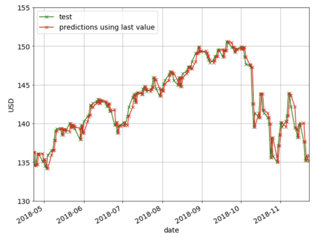
- Clustering
  - discover partitions of a dataset
- Classification/recognition
  - determine which class a data item should belong to;
- Regression/Prediction
  - predict the value of a variable base on some relationship;



Clustering: finding a partition so the data points can be grouped into subsets



Classification: finding a boundary between two classes so it can tell which class a data point belongs to



Regression: predict a value based on the observations.

Foundations for building a machine learning model

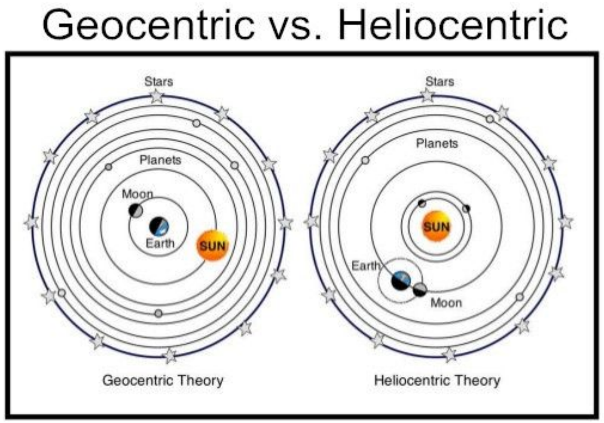
We can break down a machine learning model into the following components,

- Inductive bias (model design)
- Features (data representation)
- Distance metrics (measurement)

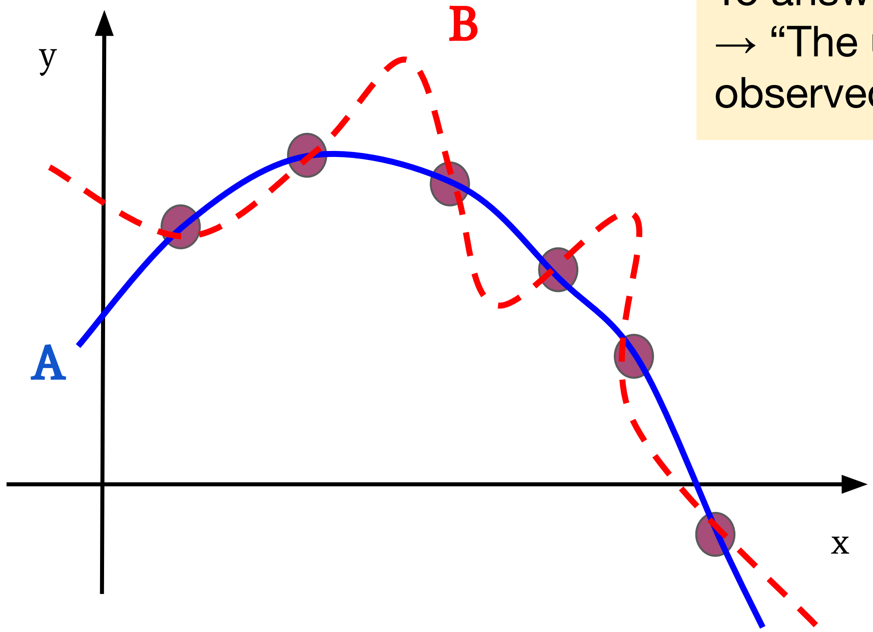
Inductive bias (learning bias)

Inductive bias (learning bias): a set of assumptions that the learner uses to predict outputs given inputs that it has not encountered. (Wikipedia)

- e.g., geocentric theory (地心说) and heliocentric theory (日心说), both describe how stars move in the sky but with different assumptions.



Which model is better, A or B?

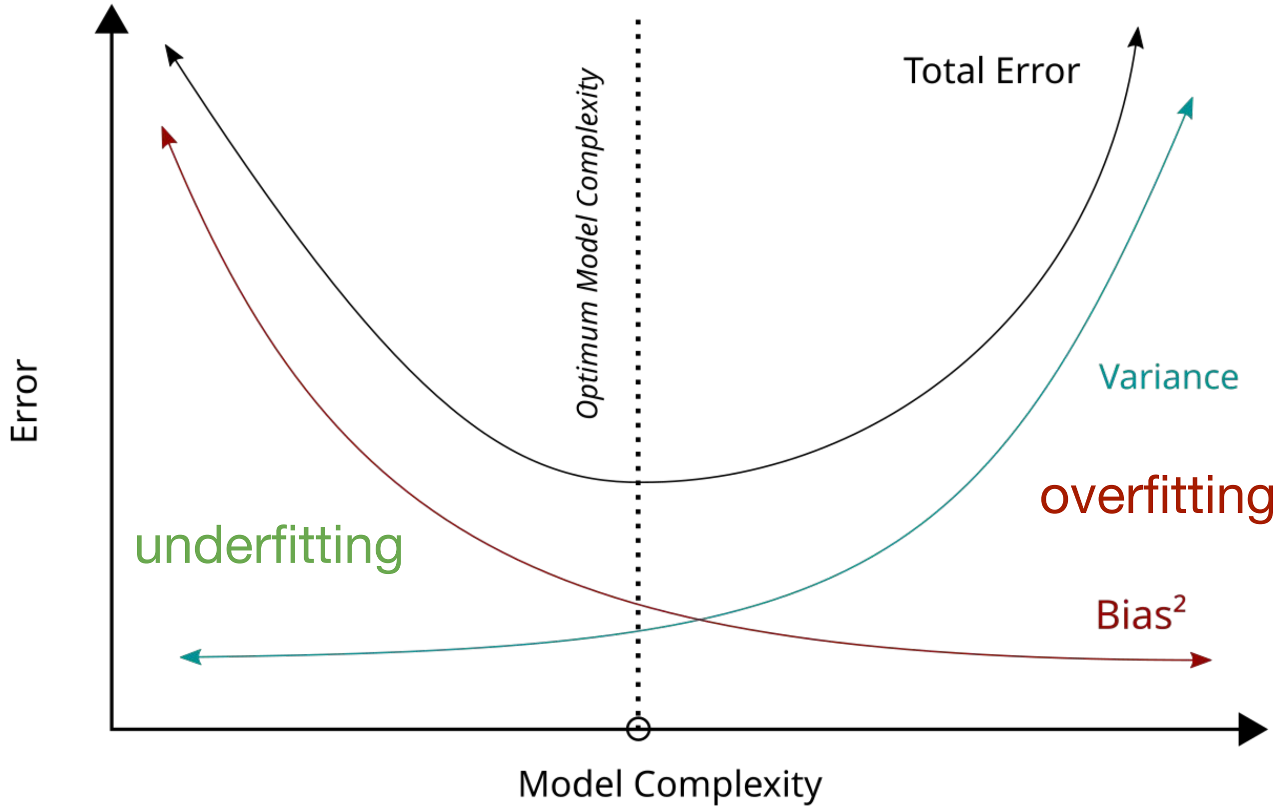


To answer it, we have to make some assumptions. → "The unseen data points are very similar to the observed data points."(inductive bias) → A is better

Inductive bias is a logic foundation of machine learning → it explains why and how we design, train, and test our models with the existing data.

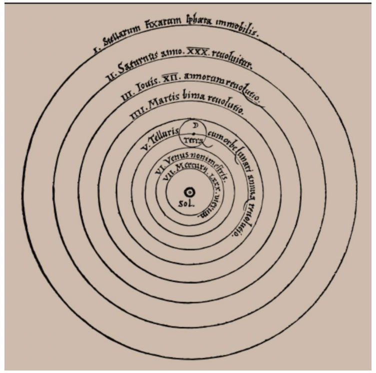
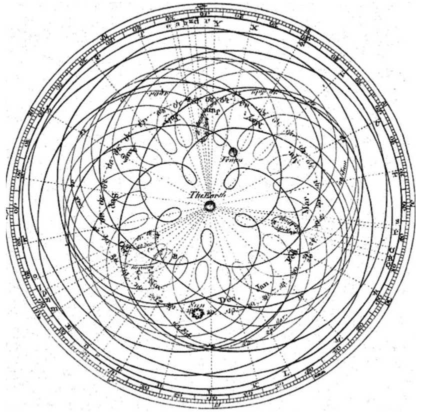
(One assumption we always have for machine learning: we believe that the the future data has some relation with the existing data).

Bias-Variance Tradeoff



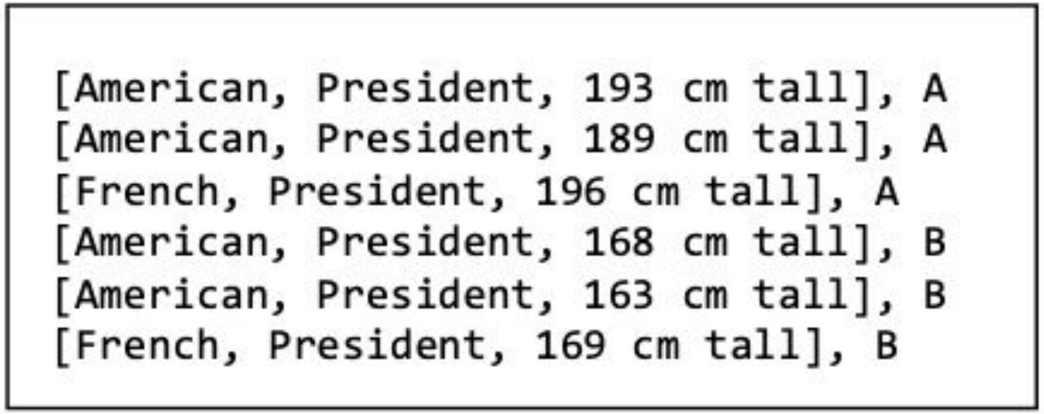
How to choose learning bias: Occam’s razor

- **Occam’s razor** (*lex parsimoniae*): the **simplest** consistent hypothesis about the target function is actually the best. → <u>Other things being equal, simpler explanations are generally better than more complex ones.</u>
- So, the heliocentric model is preferred, not the geocentric model (i.e., We prefer to use the assumptions that simply explanations)


I. Stellarum Fixarum sphæra immobilis.
II. Saturnus anno. XXX. reuoluitur.
III. Iouis. XII. annorum reuolutio.
IIII. Martis bima reuolutio.
V. Telluris cum orbe lunari annua reuolutio.
Terra
VI. Venus nono mense reducitur.
VII. Mercurii. LXXX. dierum.
Sol.


Feature vector: a format of data

- Feature: a description/representation of a data item
- Each dimension represents an attribute of the data item



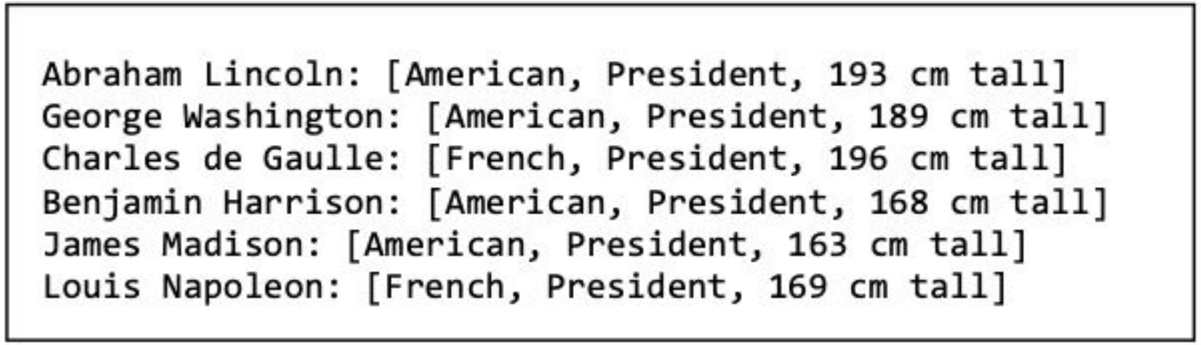
```
[American, President, 193 cm tall], A
[American, President, 189 cm tall], A
[French, President, 196 cm tall], A
[American, President, 168 cm tall], B
[American, President, 163 cm tall], B
[French, President, 169 cm tall], B
```

Figure 22.3 Feature vector/label pairs for presidents

Feature engineering:

- Designing specific features for specific tasks (for different tasks, different attributes are needed)
- Goals: Picking out the most relevant attributes

Feature engineering: a simple example



```
Abraham Lincoln: [American, President, 193 cm tall]
George Washington: [American, President, 189 cm tall]
Charles de Gaulle: [French, President, 196 cm tall]
Benjamin Harrison: [American, President, 168 cm tall]
James Madison: [American, President, 163 cm tall]
Louis Napoleon: [French, President, 169 cm tall]
```

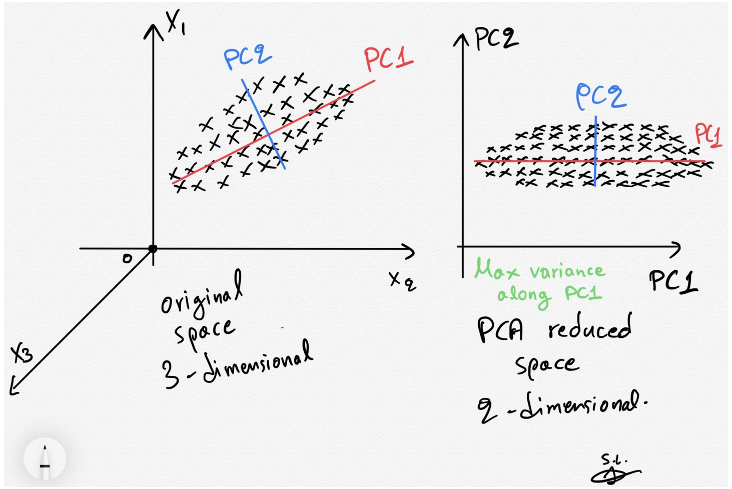
If the goal is to identify the nationalities of these people’s parents, what attribute shall we take?

- title “president” provides nothing, so we may ignore it.
- “height” is not that critical in this task, so we may reduce it weight
- “nationality” should be given the highest weight

Dimension reduction (a feature engineering technique)

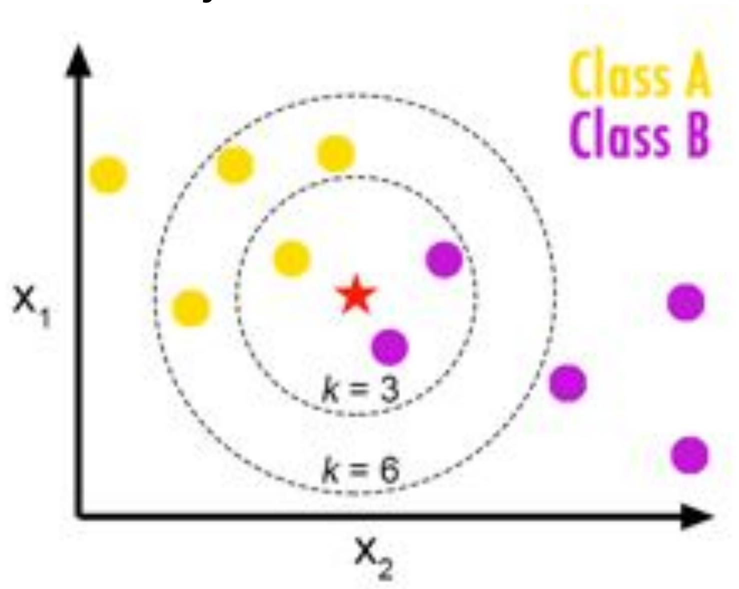
Many real-world datasets are high dimensions. It often needs to transform the data from a high-dimension space into a low-dimension space.

- Ideally, the transformation should reduce the dimensions but keep the intrinsic structure of the dataset.
- Techniques:
  - Principal Component Analysis (PCA)
  - Manifold learning



Distance metrics

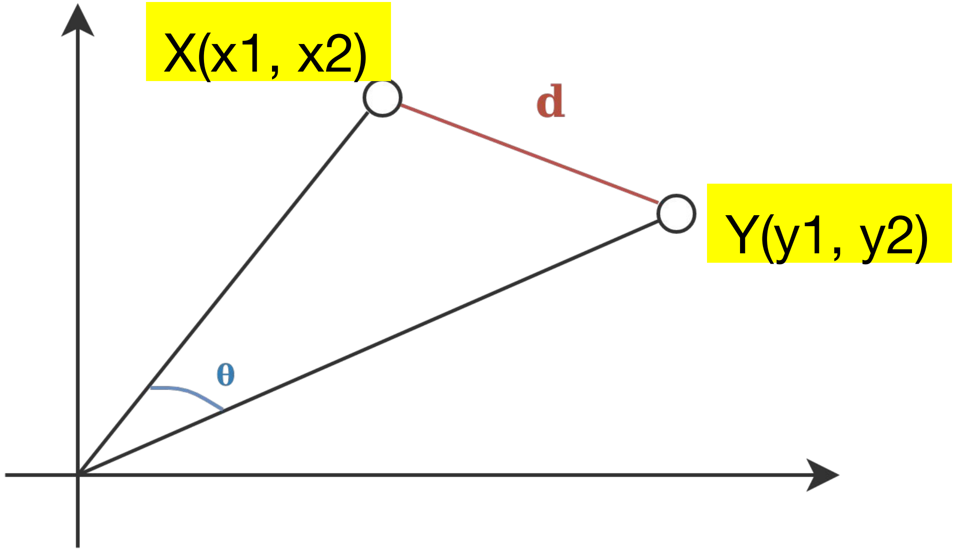
- Since we represent data by “feature vectors” ⇒ data items are points in a coordinate system.
- Relations between data items are represented by distance.
  - close ⇔ similar



Euclidean distance

- Very intuitive

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

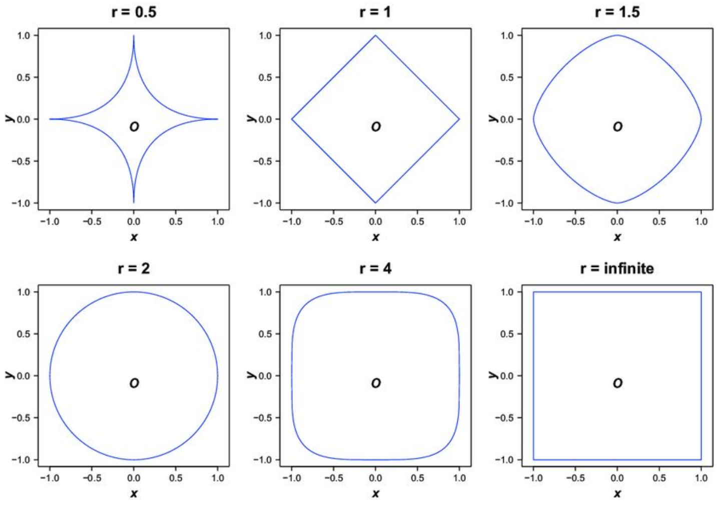


Minkowski distance: a generalized distance metric

Let $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$ and $\mathbf{y} = \{y_1, y_2, \cdots, y_n\}$ be two vectors. Minkowski distance is defined as

$$d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}$$

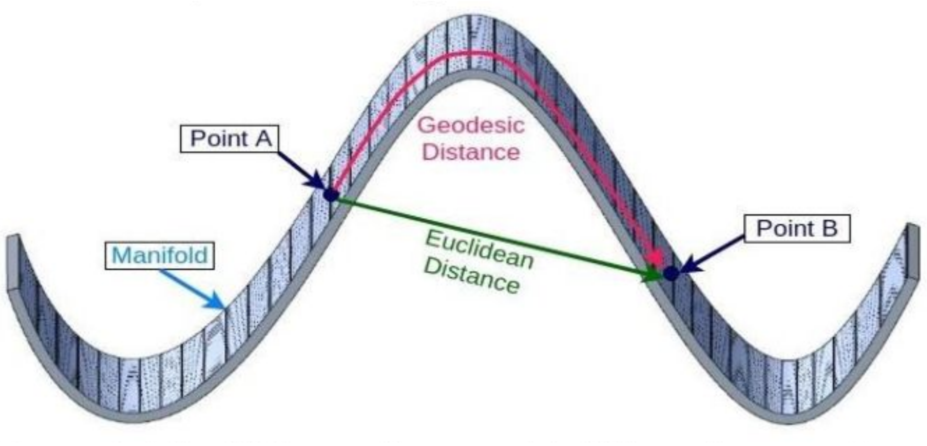
- p = 2: Euclidean distance
- p = 1: Manhattan distance

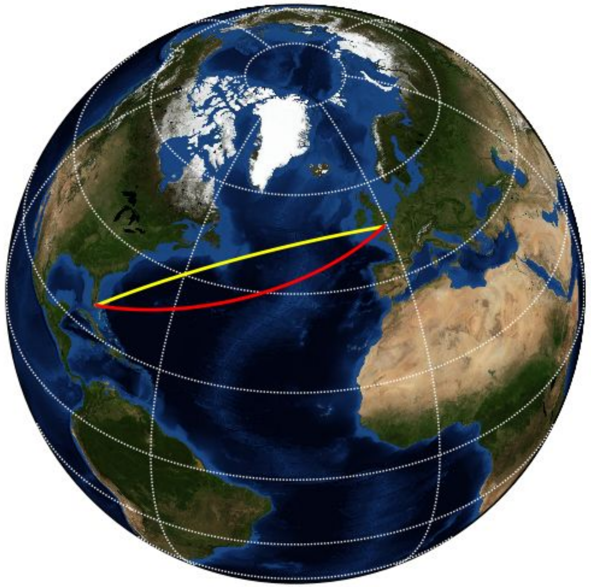


Empirical analysis:
https://link.springer.com/chapter/10.1007/978-3-319-22572-2_24

Other non-Euclidean distance

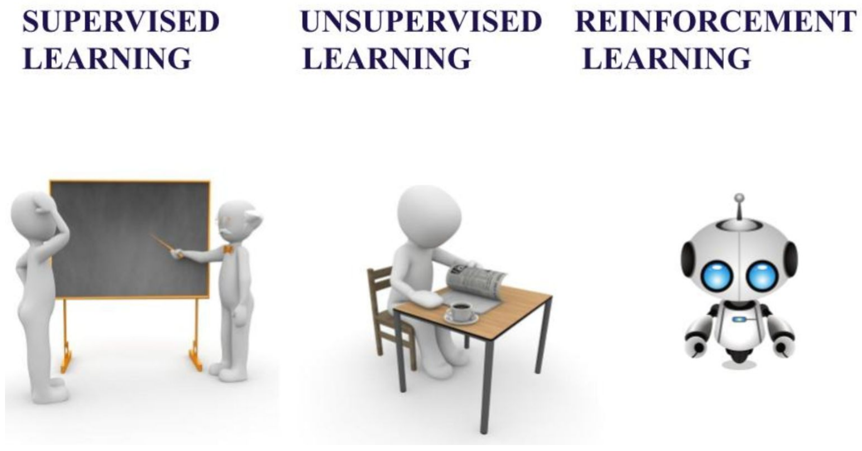
- Geodesic distance → distance measured along the surface





Types of machine learning methods

Depending on how much guidance is needed when they “learn”:



- Supervised learning (need clear guidance): works with labelled data sets; it needs the labels to tell it whether its prediction is correct or not.
- Unsupervised learning (no guidance needed): no labels are needed; it is self-organized, using a predefined way to process the data.
- Reinforcement learning (vague guidance is ok; like the intelligent creatures) : it “labels” the data items during learning; “learn by doing”.

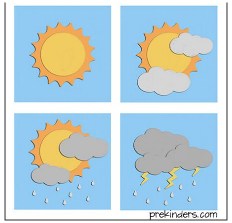
Supervised and unsupervised learning

Most of the learning algorithms are supervised learning or unsupervised learning models.

- Supervised learning:
    - models use the given {input, true output} pairs to train its parameters so that, when the model is given an input, its output will be close to the true output.
    - Widely used in tasks of prediction (e.g., regression, and classification)
- Unsupervised learning:
    - models are self-adapted; they don’t need {input, true output} pairs to train their parameters. (It uses hand-crafted rules to complete the task.)
    - Used in clustering (i.e., partition a dataset into several subsets by some predefined rules)

Using supervised or unsupervised learning?

- Given a set of records of traffics, weather conditions, and the time spent traveling from your apartment to the campus under corresponding traffic and weather, build a model which can tell the commute time by inputting the traffic and weather.



Using supervised or unsupervised learning?

- Given a set of images of different breeds of dogs, build a model that can identify the type of a dog.

Using supervised or unsupervised learning?

- You want to group students on campus into several small groups based on their features including weight, height, and color of hair.

K-NN is a supervised learning approach

- K-NN is a **supervised learning** algorithm but has **no training** step
  - We just take the given dataset as the model (a **“lazy”** learner, not “learn” but “memorize”; no need to train, it is “trained” while making predictions (i.e., computing the distance to the other samples)
  - **k is a hyper-parameter**, i.e., it has been decided before we “train” the model. (By trying different ks, we can set the k to the one that has the highest performance.)

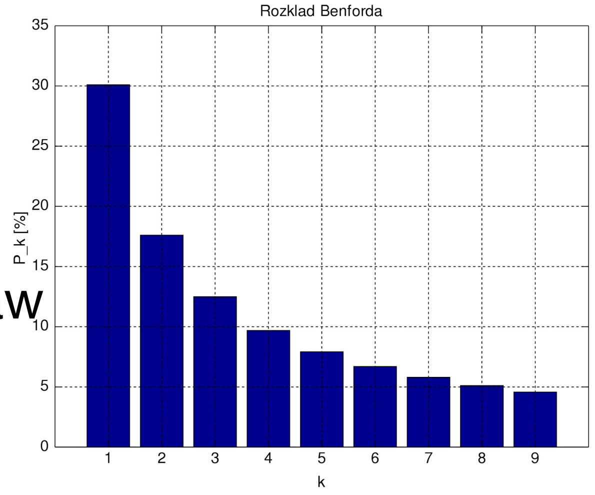
Appendix:

- Exercise: Benford’s law
- Andrew Ng’s online machine learning course

Exercise: Benford's law

Benford's law, also called, the first-digit law.

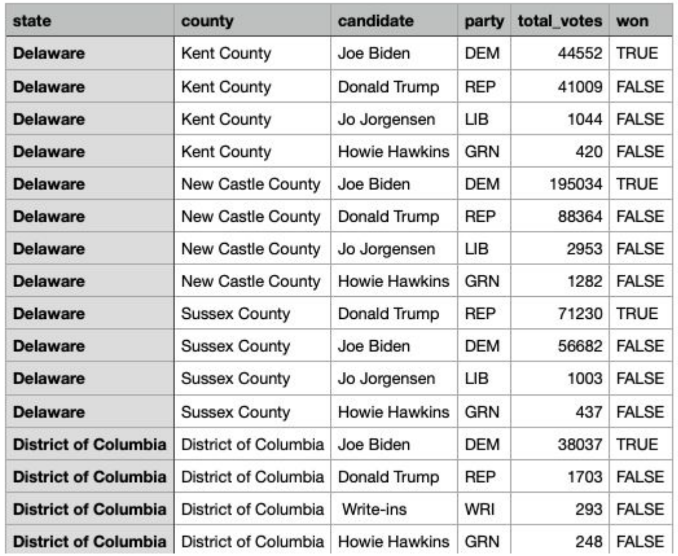
- is about the distribution of leading digits in many real-life sets of numerical data
- the first digit is likely to be small
  - about 30% observations starts by 1
  - about 5% observations starts by 9
  - P(d) = lg(1 + 1/d) (note: 10-based)
- Many actual datasets seem to observe this law
  - Fibonacci number
  - The population of countries



US Election 2020 dataset

- Data source: https://www.kaggle.com/unanimad/us-election-2020 (not official)
- Contains the number of votes for different candidates of each county over all states
- Might have missing values

president_county_candidate



state	county	candidate	party	total_votes	won
Delaware	Kent County	Joe Biden	DEM	44552	TRUE
Delaware	Kent County	Donald Trump	REP	41009	FALSE
Delaware	Kent County	Jo Jorgensen	LIB	1044	FALSE
Delaware	Kent County	Howie Hawkins	GRN	420	FALSE
Delaware	New Castle County	Joe Biden	DEM	195034	TRUE
Delaware	New Castle County	Donald Trump	REP	88364	FALSE
Delaware	New Castle County	Jo Jorgensen	LIB	2953	FALSE
Delaware	New Castle County	Howie Hawkins	GRN	1282	FALSE
Delaware	Sussex County	Donald Trump	REP	71230	TRUE
Delaware	Sussex County	Joe Biden	DEM	56682	FALSE
Delaware	Sussex County	Jo Jorgensen	LIB	1003	FALSE
Delaware	Sussex County	Howie Hawkins	GRN	437	FALSE
District of Columbia	District of Columbia	Joe Biden	DEM	38037	TRUE
District of Columbia	District of Columbia	Donald Trump	REP	1703	FALSE
District of Columbia	District of Columbia	Write-ins	WRI	293	FALSE
District of Columbia	District of Columbia	Howie Hawkins	GRN	248	FALSE

Benford's law and US Election 2020

- Task: show the distribution of leading digits of a candidate's votes over all counties.

Candidate

Attributes:
(please design them by yourself)

Methods:
(please design them by yourself)

A candidate must have a name, and the information about the votes of each (state, county).

Main function: benford_analysis of a candidate

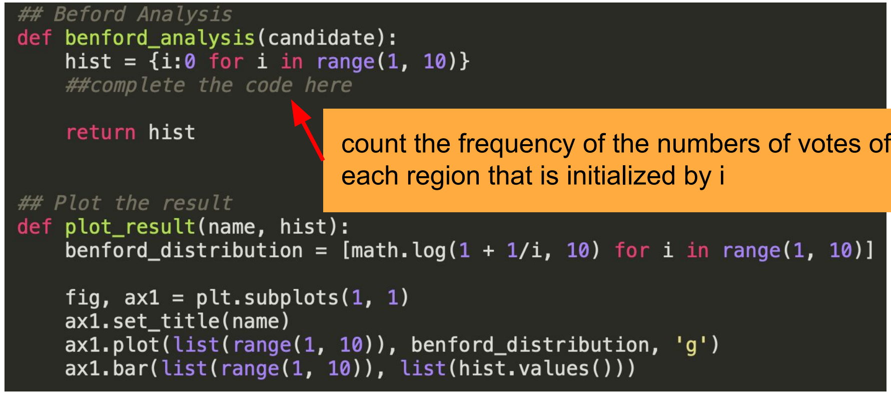
- for i in range(1, 10), count the **frequency** of the numbers of votes of each region that is initialized by i
- return the histogram

Hist

In this example, we simply use the built-in data type: dict

- hist is a Python dictionary with the keys are the numbers from 1 to 9, the values are the frequency the numbers that are initialized by the corresponding keys.

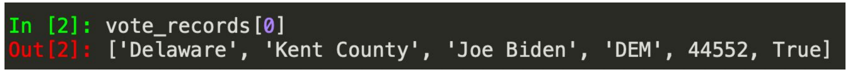
Benford’s law and US Election 2020



```python
## Beford Analysis
def benford_analysis(candidate):
    hist = {i:0 for i in range(1, 10)}
    ##complete the code here

    return hist


## Plot the result
def plot_result(name, hist):
    benford_distribution = [math.log(1 + 1/i, 10) for i in range(1, 10)]

    fig, ax1 = plt.subplots(1, 1)
    ax1.set_title(name)
    ax1.plot(list(range(1, 10)), benford_distribution, 'g')
    ax1.bar(list(range(1, 10)), list(hist.values()))
```

count the frequency of the numbers of votes of each region that is initialized by i

- `vote_records`: it is a list of vote records which is given in the starting code; each element is a list of 6 elements. (Print it to see details.)
- from the left to the right, they are [state, country, candidate, party, total_votes, won]

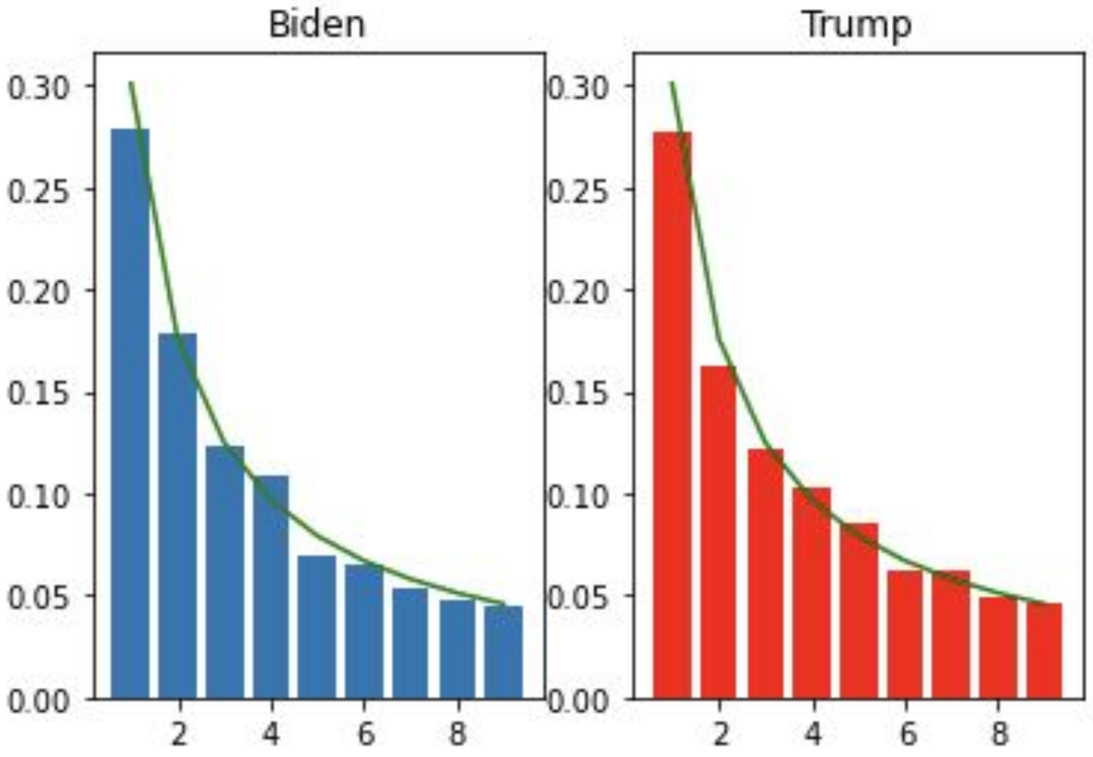


```
In [2]: vote_records[0]
Out[2]: ['Delaware', 'Kent County', 'Joe Biden', 'DEM', 44552, True]
```

In this example, we need to do the Benford analysis on the total_vote of each (state, country) of a candidate (e.g., Trump).

Please download the starting code and dataset in Brightspace/Lectures/Week 8/.

Benford's law and US Election 2020



Which one do you think fits better?

Homeworks

[OOP] Finish the classes we covered in today's lecture:

- KNN predictor & sample class: page 10-11
- Candidate class for Benford's law: page 41

[ML] Andrew Ng's online machine learning course for beginners:

- https://www.youtube.com/playlist?list=PL6xNS6KCt75ccBh6dB1mkSr3V69OP7p6B
- On Coursera: https://www.coursera.org/specializations/machine-learning-introduction