Python ??

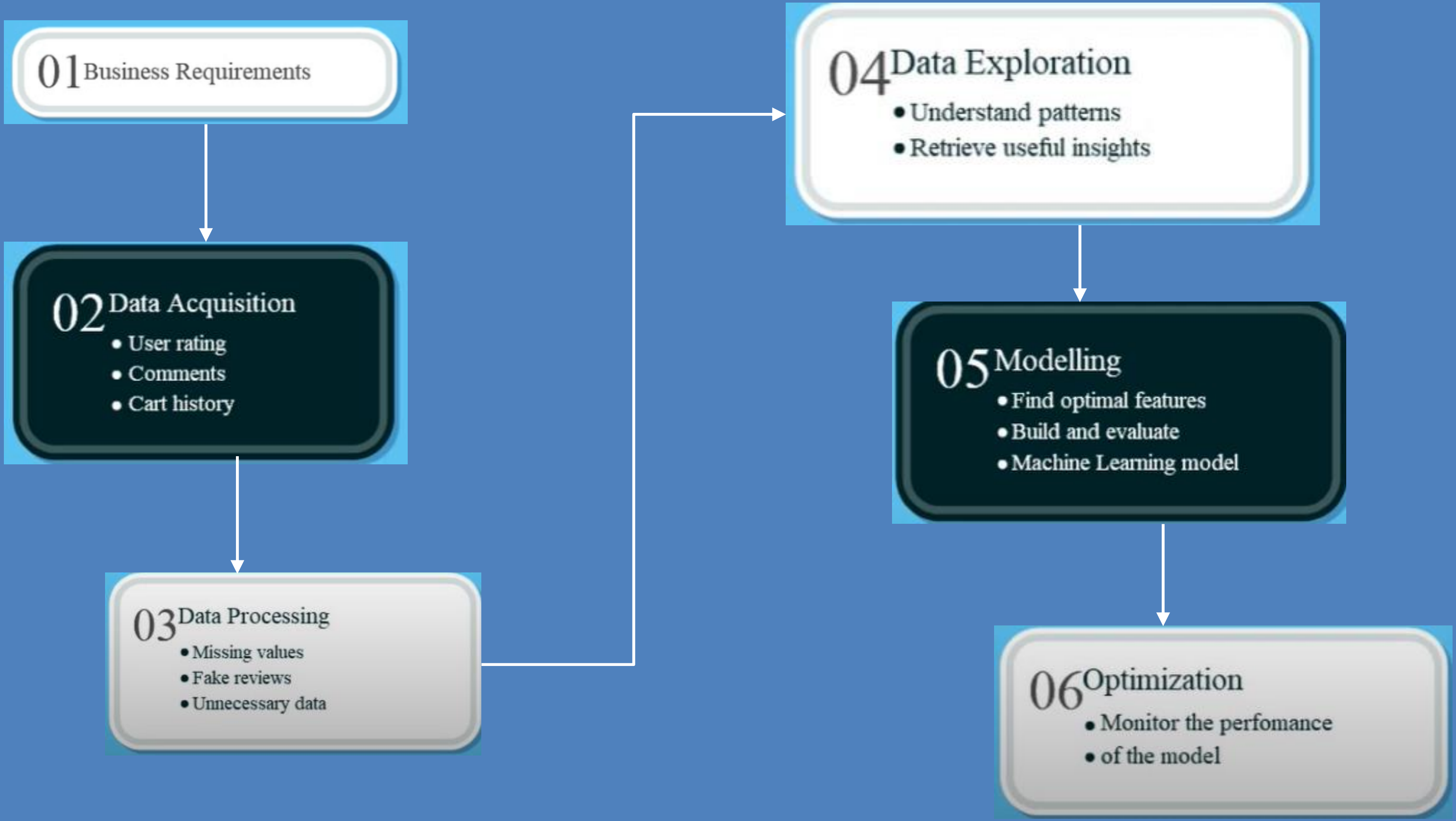
- Inventor- Van Rossum
- General purpose high level language
- Evolved since past 25 years
- Clear readable syntax
- Good practice required – clean readable code
- Dynamic language – everything at runtime
- Interpreted language – quick iteration and test
- Lot less code
- Portable
- Code security
- Slow
- Second most popular language

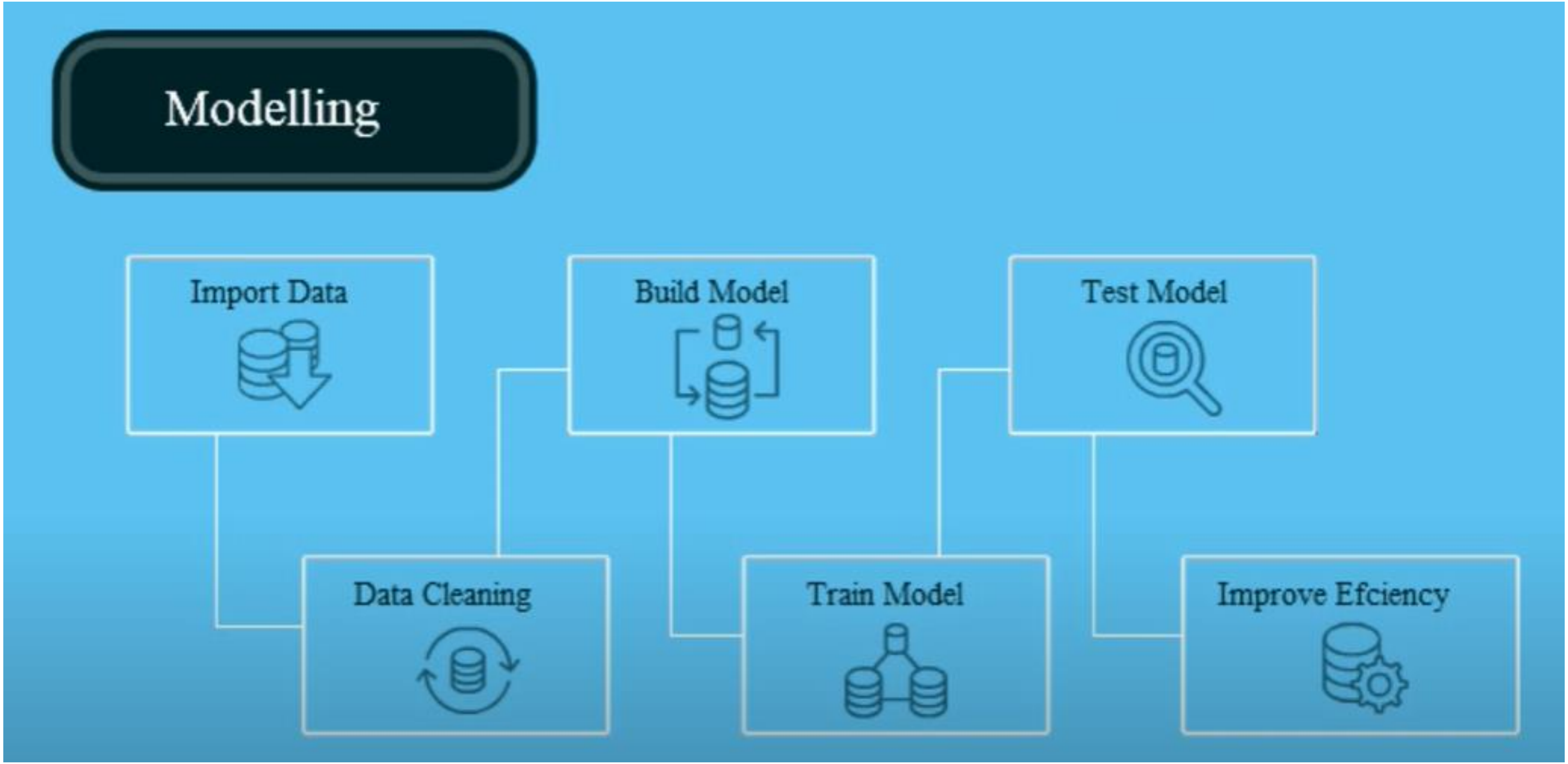
- Instagram,
- Netflix
- Google
- Spotify
- Uber
- OpenShift
- Dropbox

Applications

- Fraud detection.
- Web search results.
- Real-time ads on web pages
- Credit scoring and offers.
- Prediction of equipment failures.
- New pricing models.
- Network intrusion detection.
- Recommendation Engines
- Customer Segmentation
- Text Sentiment Analysis
- Predicting Customer Churn
- Pattern and image recognition.
- Email spam filtering.
- Financial Modeling

DS vs AI vs ML vs DL


01 Business Requirements
02 Data Acquisition
User rating
Comments
Cart history
03 Data Processing
Missing values
Fake reviews
Unnecessary data
04 Data Exploration
Understand patterns
Retrieve useful insights
05 Modelling
Find optimal features
Build and evaluate
Machine Learning model
06 Optimization
Monitor the perfomance
of the model


Modelling
Import Data
Build Model
Test Model
Data Cleaning
Train Model
Improve Efciency

Machine Learning

- Modelling uses machine learning algorithms, in which the machine learns from the data just like humans learn from their experiences.
- Speech Recognition, Image recognition, AI Camera on Phone, Healthcare, Insurance, Customer Churn, Customer Default
- Derived from data - derivatives

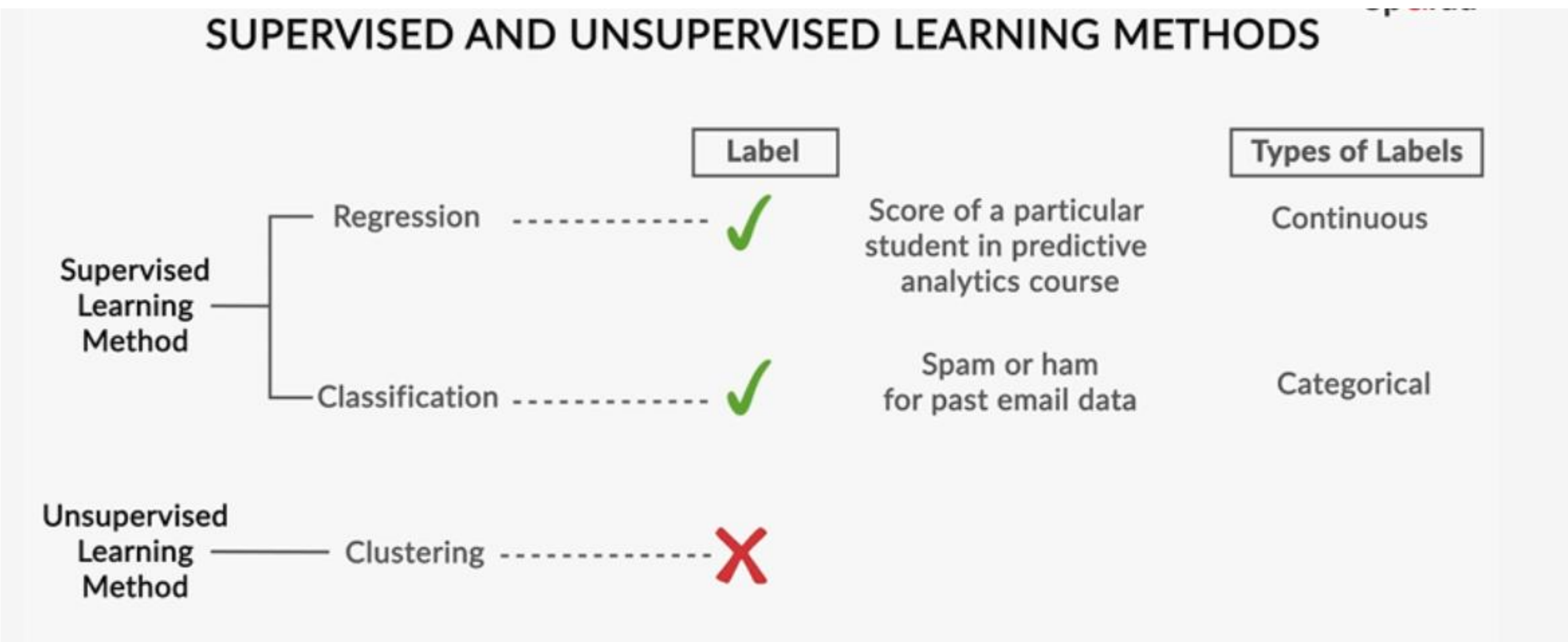
3 broad categories of machine learning

- **Regression**: The output variable to be predicted is a **continuous variable**, e.g. scores of a student
- **Classification**: The output variable to be predicted is a **categorical variable**, e.g. classifying incoming emails as spam or ham
- **Clustering**: **No pre-defined notion of label** allocated to groups/clusters formed, e.g. customer segmentation
-

Supervised / Unsupervised

- **Supervised learning methods**
  - Past data with labels is used for building the model
  - **Regression** and **classification** algorithms fall under this category
- **Unsupervised learning methods**
  - No pre-defined labels are assigned to past data
  - **Clustering** algorithms fall under this category
  - PCA

Illustration



Evaluating Supervised Learning

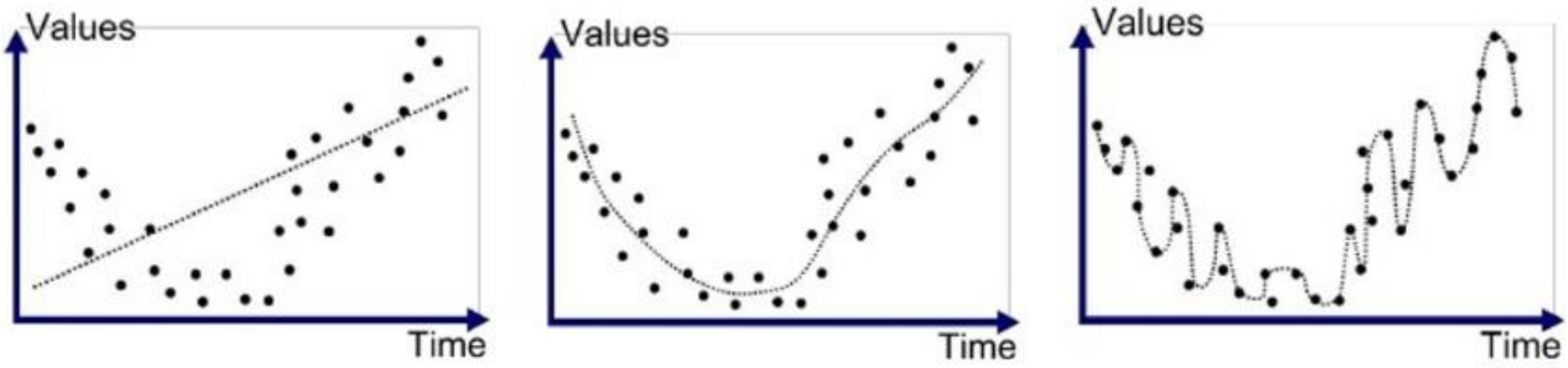
- Split data in two parts: a training set and a test set.
- Train the model using only the training set and then measure (using r-squared, Classification accuracy, Logarithmic loss, confusion matrix, AUC, F1 score, MSE, MAE) the model's accuracy by asking it to predict values for the test set, and compare that to the known, true values.

Train / Test in practice

- Need to ensure both sets are large enough to contain representatives of all the variations and outliers in the data you care about
- The data sets must be selected randomly
- Train/test is a great way to guard against overfitting

Train/Test is not Infallible

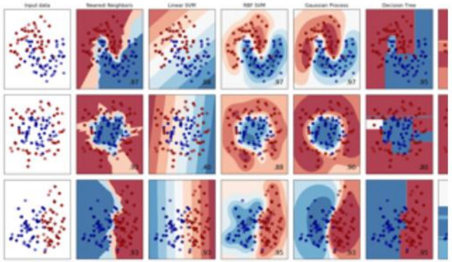
- Sample sizes are too small
- Or due to random chance that train and test sets look remarkably similar
- Overfitting / Underfitting can still happen



Classification

Identifying which category an object belongs to.

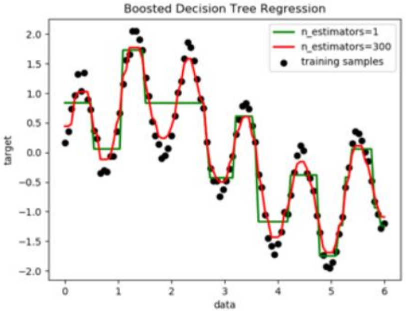
Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

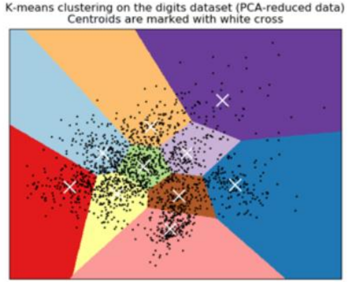
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Clustering

Automatic grouping of similar objects into sets.

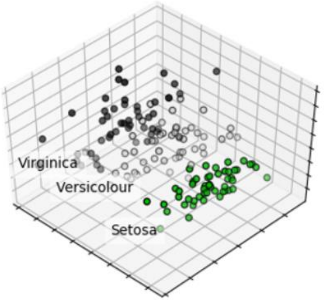
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...



Dimensionality reduction

Reducing the number of random variables to consider.

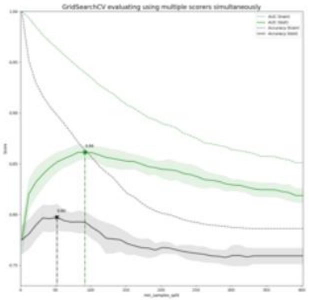
Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...



1.1. Linear Models

- 1.1.1. Ordinary Least Squares
- 1.1.2. Ridge regression and classification
- 1.1.3. Lasso
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic-Net
- 1.1.6. Multi-task Elastic-Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
- 1.1.11. Logistic regression
- 1.1.12. Generalized Linear Regression
- 1.1.13. Stochastic Gradient Descent - SGD
- 1.1.14. Perceptron
- 1.1.15. Passive Aggressive Algorithms
- 1.1.16. Robustness regression: outliers and modeling errors
- 1.1.17. Polynomial regression: extending linear models with basis functions

1.2. Linear and Quadratic Discriminant Analysis

- 1.2.1. Dimensionality reduction using Linear Discriminant Analysis
- 1.2.2. Mathematical formulation of the LDA and QDA classifiers
- 1.2.3. Mathematical formulation of LDA dimensionality reduction
- 1.2.4. Shrinkage
- 1.2.5. Estimation algorithms

1.3. Kernel ridge regression

1.4. Support Vector Machines

- 1.4.1. Classification
- 1.4.2. Regression
- 1.4.3. Density estimation, novelty detection
- 1.4.4. Complexity
- 1.4.5. Tips on Practical Use
- 1.4.6. Kernel functions
- 1.4.7. Mathematical formulation
- 1.4.8. Implementation details

1.5. Stochastic Gradient Descent

- 1.5.1. Classification
- 1.5.2. Regression
- 1.5.3. Stochastic Gradient Descent for sparse dat
- 1.5.4. Complexity
- 1.5.5. Stopping criterion
- 1.5.6. Tips on Practical Use
- 1.5.7. Mathematical formulation
- 1.5.8. Implementation details

1.6. Nearest Neighbors

- 1.6.1. Unsupervised Nearest Neighbors
- 1.6.2. Nearest Neighbors Classification
- 1.6.3. Nearest Neighbors Regression
- 1.6.4. Nearest Neighbor Algorithms
- 1.6.5. Nearest Centroid Classifier
- 1.6.6. Nearest Neighbors Transformer
- 1.6.7. Neighborhood Components Analysis

1.7. Gaussian Processes

- 1.7.1. Gaussian Process Regression (GPR)
- 1.7.2. GPR examples
- 1.7.3. Gaussian Process Classification (GPC)
- 1.7.4. GPC examples
- 1.7.5. Kernels for Gaussian Processes

1.8. Cross decomposition

1.9. Naive Bayes

- 1.9.1. Gaussian Naive Bayes
- 1.9.2. Multinomial Naive Bayes
- 1.9.3. Complement Naive Bayes
- 1.9.4. Bernoulli Naive Bayes
- 1.9.5. Categorical Naive Bayes
- 1.9.6. Out-of-core naive Bayes model fitting

1.10. Decision Trees

- 1.10.1. Classification
- 1.10.2. Regression
- 1.10.3. Multi-output problems
- 1.10.4. Complexity

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
OPTICS	minimum cluster membership	Very large `n_samples`, large `n_clusters`	Non-flat geometry, uneven cluster sizes, variable cluster density	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

Linear Regression

- R2 / RSS / TSS

```
In [144]: import pandas as pd
```

```
In [145]: advertising = pd.read_csv("tvmarketing.csv")
```

```
In [150]: # Display the first 5 rows
          advertising.head()
```

Out[150]:

	TV	Sales
0	230.1	22.1
1	44.5	10.4
2	17.2	9.3
3	151.5	18.5
4	180.8	12.9

```
In [147]: # Display the last 5 rows
          advertising.tail()
```

...

```
In [110]: # Let's check the columns
          advertising.info()
```

Linear Regression

In [155]:
```
# Let's look at some statistical information about our dataframe.
advertising.describe()
```

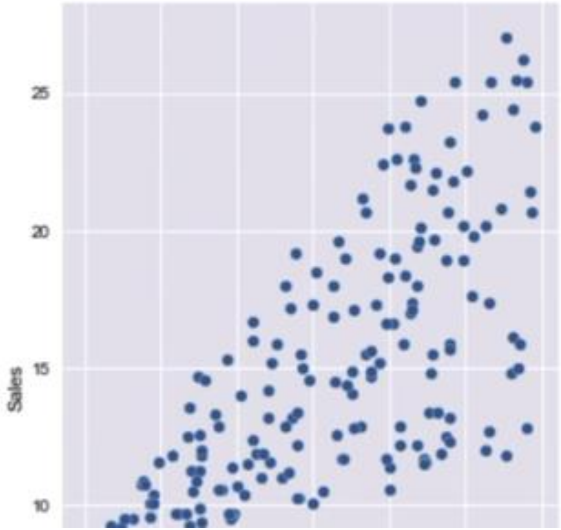
Out[155]:

	TV	Sales
count	200.000000	200.000000
mean	147.042500	14.022500
std	85.854236	5.217457
min	0.700000	1.600000
25%	74.375000	10.375000
50%	149.750000	12.900000
75%	218.825000	17.400000
max	296.400000	27.000000

Linear Regression

```
In [114]: # Visualise the relationship between the features and the response using scatterplots
          sns.pairplot(advertising, x_vars=['TV'], y_vars='Sales',size=7, aspect=0.7, kind='scatter')

Out[114]: <seaborn.axisgrid.PairGrid at 0x113c2beb8>
```



Equation

Performing Simple Linear Regression

Equation of linear regression
$y = c + m_1x_1 + m_2x_2 + \ldots + m_nx_n$

- y is the response
- c is the intercept
- m_1 is the coefficient for the first feature
- m_n is the coefficient for the nth feature

In our case:

$y = c + m_1 \times TV$

The m values are called the model coefficients.

Preparing X and y

- The scikit-learn library expects X (feature variable) and y (response variable) to be NumPy arrays.
- However, X can be a dataframe as Pandas is built over NumPy.

Linear Regression

- **Create X_train, y_train, X_test and Y_test**

```
# 1. Create the datasets X_train, y_train, X_test and y_test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7 , random_state=100)
```

- **Create or instantiate an object, e.g. lr = LinearRegression**

```
# 2. Create (or instantiate) an object of the model you want to build, e.g.
lr = LinearRegression()
```

Model Steps

- **Fit the model using the object, e.g. lr.fit(X_train, y_train)**

```
# 3. Fit the model using the training data
lr.fit(X_train, y_train)
```

- **Predict the labels of X_test, e.g. lr.predict(X_test)**

```
# 4. Predict the labels using the test data X_test
y_pred = lr.predict(X_test)
```

- **Evaluate the model by comparing the predictions with the actual labels**

```
# 5. Evaluate the model using an appropriate metric by comparing y_test and y_predicted
r_squared = r2_score(y_test, y_pred)
```

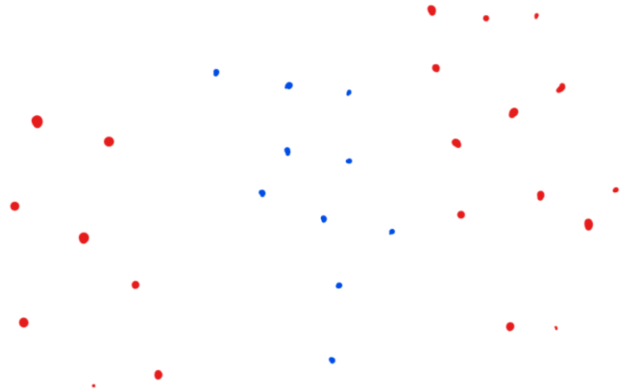
Models

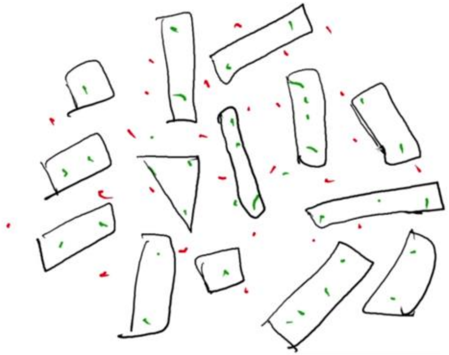
- Logistic Regression
- Naïve Bayes
- K Means Clustering
- Hierarchical Clustering / K Mode Clustering / DB Scan Clustering
- SVM
- Tree Model
- Model Selection
- Boosting

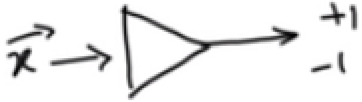
NLP / Deep Learning

- Lexical Processing
- Syntatic Processing
- Semantic Processing
- Chatbots
- Neural Network
- CNN
- RNN

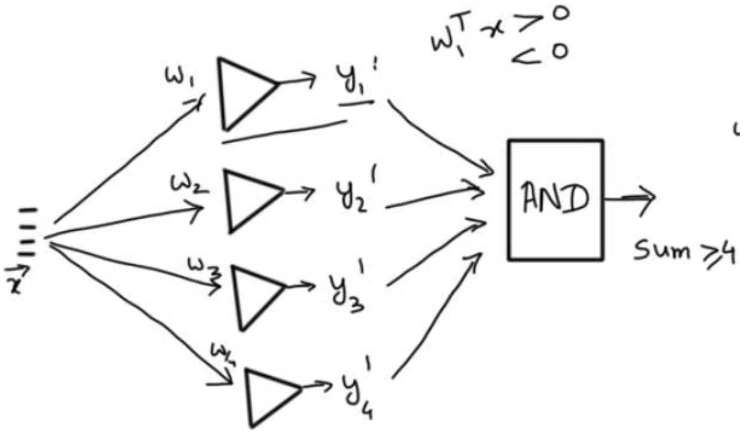
Neural Network

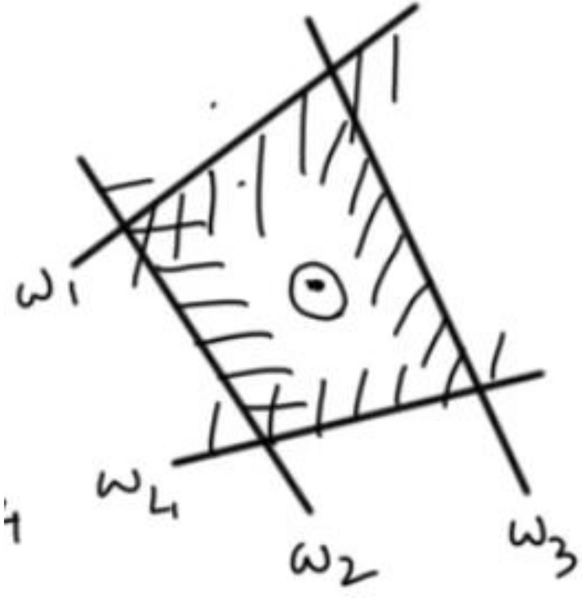




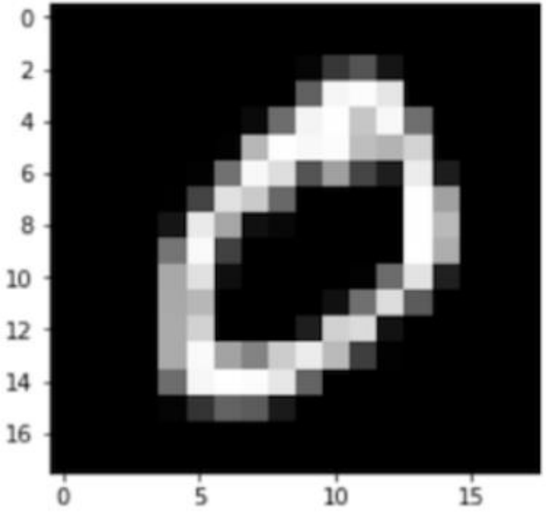


Neural Network





Character Recognition

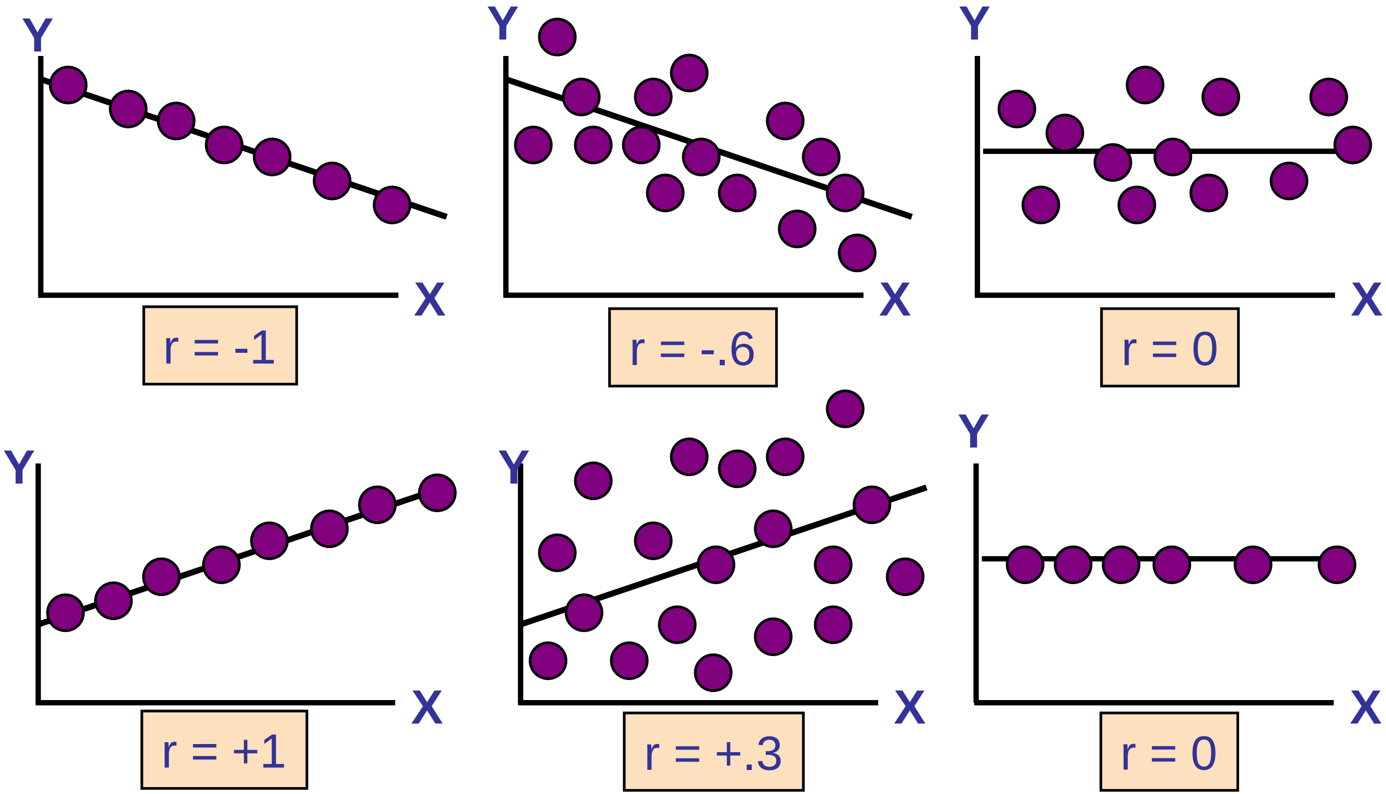



```
[[   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   6  55  84  22   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0  96 244 250 228   0   0   0   0   0]
 [   0   0   0   0   0   0   0   9 108 243 252 196 247 110   0   0   0   0]
 [   0   0   0   0   0   0   2 181 252 247 251 189 178 210   0   0   0   0]
 [   0   0   0   0   0   2 112 247 220  84 159  69  30 234  29   0   0   0]
 [   0   0   0   0   1  68 223 201 103   0   0   0   0 252 160   0   0   0]
 [   0   0   0   0  21 232 166  17   7   0   0   0   0 252 184   0   0   0]
 [   0   0   0   0 116 248  65   0   0   0   0   0   0 253 172   0   0   0]
 [   0   0   0   0 167 223  15   0   0   0   0   2 107 225  33   0   0   0]
 [   0   0   0   0 168 182   0   0   0   0  16 111 219  90   0   0   0   0]
 [   0   0   0   0 169 208   0   0   0  30 207 217  18   0   0   0   0   0]
 [   0   0   0   0 169 248 162 130 202 234 184  62   2   0   0   0   0   0]
 [   0   0   0   0 108 245 253 251 229  99   0   0   0   0   0   0   0   0]
 [   0   0   0   0   5  52  98  91  26   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

Correlation

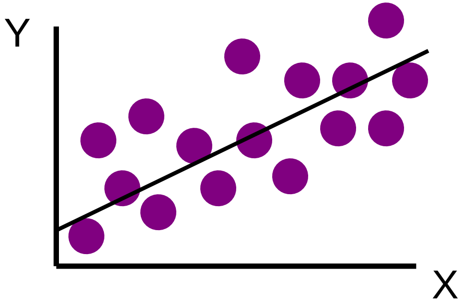
- Measures the relative strength of the *linear* relationship between two variables
- Unit-less
- Ranges between –1 and 1
- The closer to –1, the stronger the negative linear relationship
- The closer to 1, the stronger the positive linear relationship
- The closer to 0, the weaker any positive linear relationship

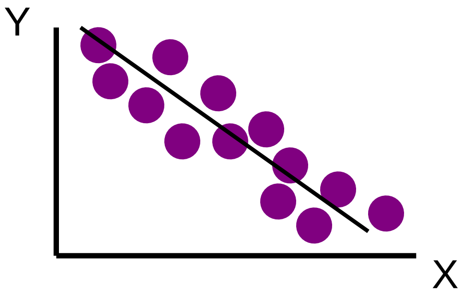
Scatter Plots of Data with Various Correlation Coefficients



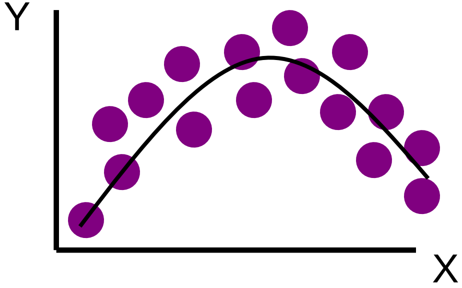
Linear Correlation

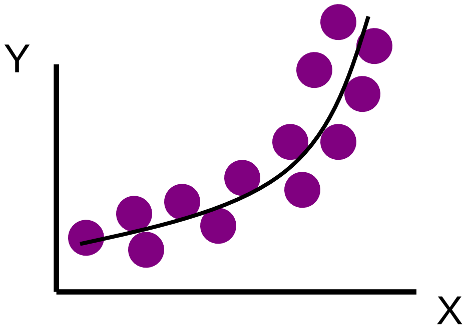
Linear relationships





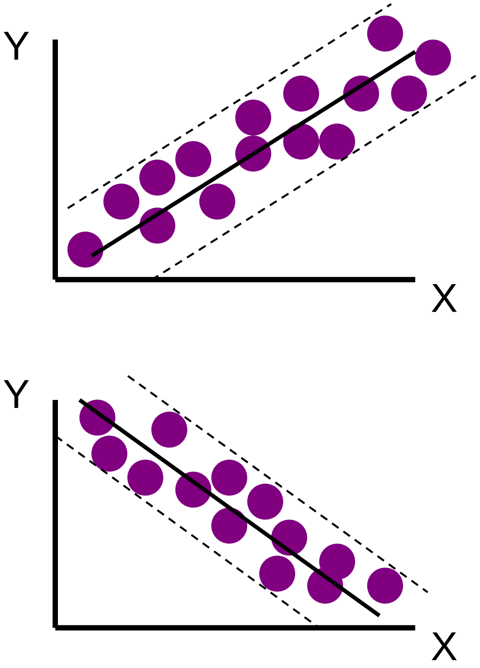
Curvilinear relationships



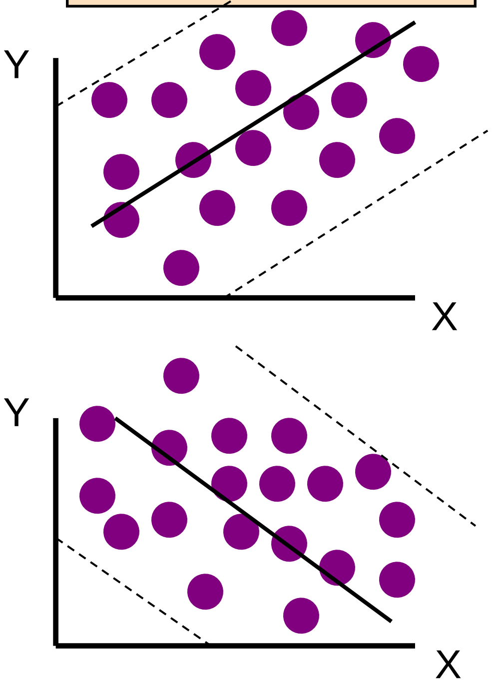


Linear Correlation

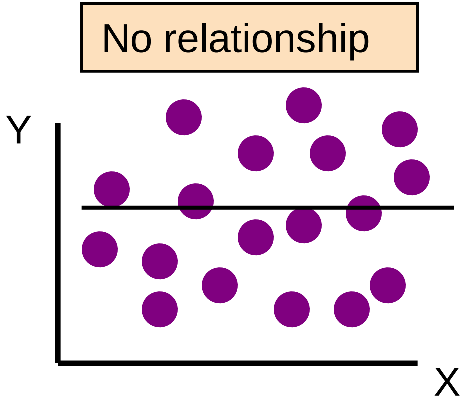
Strong relationships

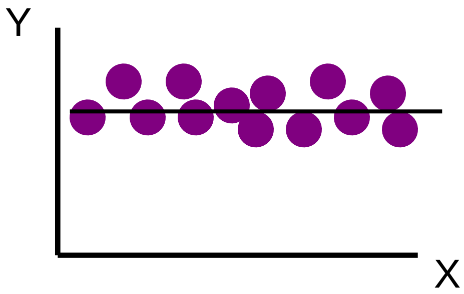


Weak relationships



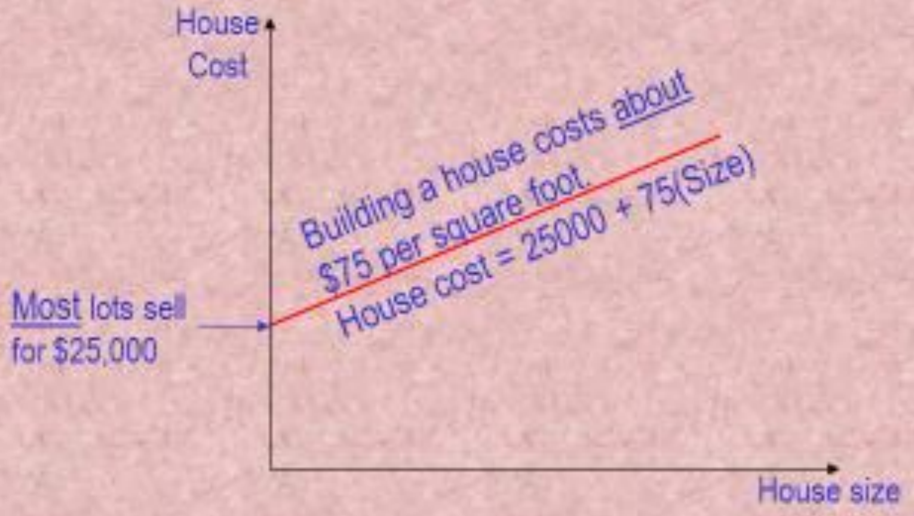
Linear Correlation





The Model

The model has a deterministic and a probabilistic components



3

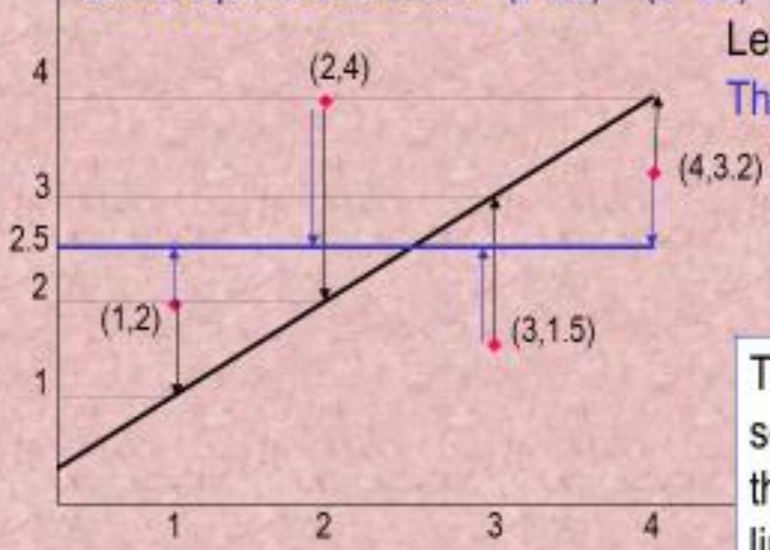
The Least Squares (Regression) Line

Sum of squared differences = $(2 - 1)^2 + (4 - 2)^2 + (1.5 - 3)^2 + (3.2 - 4)^2 = 6.89$

Sum of squared differences = $(2 - 2.5)^2 + (4 - 2.5)^2 + (1.5 - 2.5)^2 + (3.2 - 2.5)^2 = 3.99$

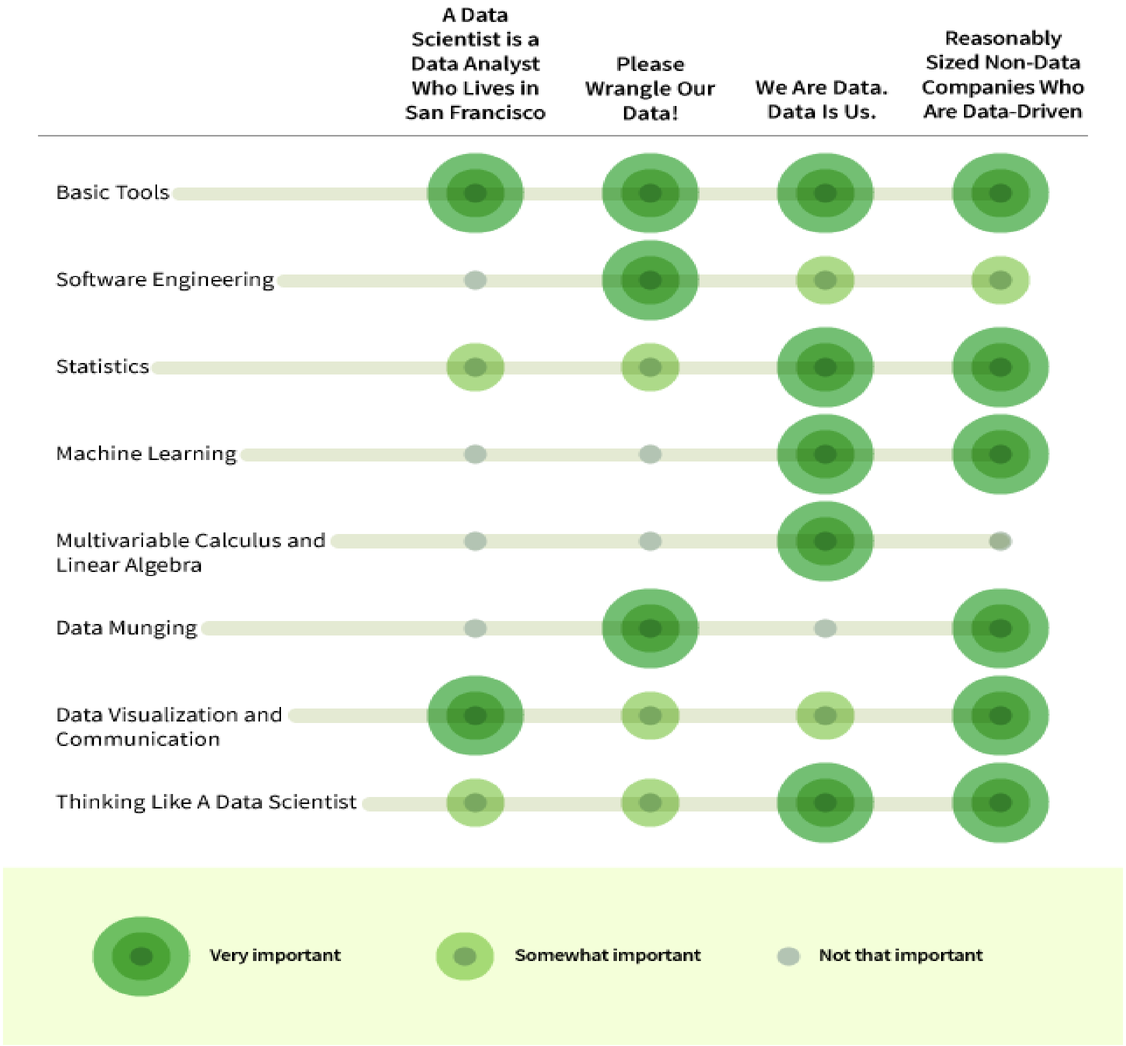
Let us compare two lines

The second line is horizontal

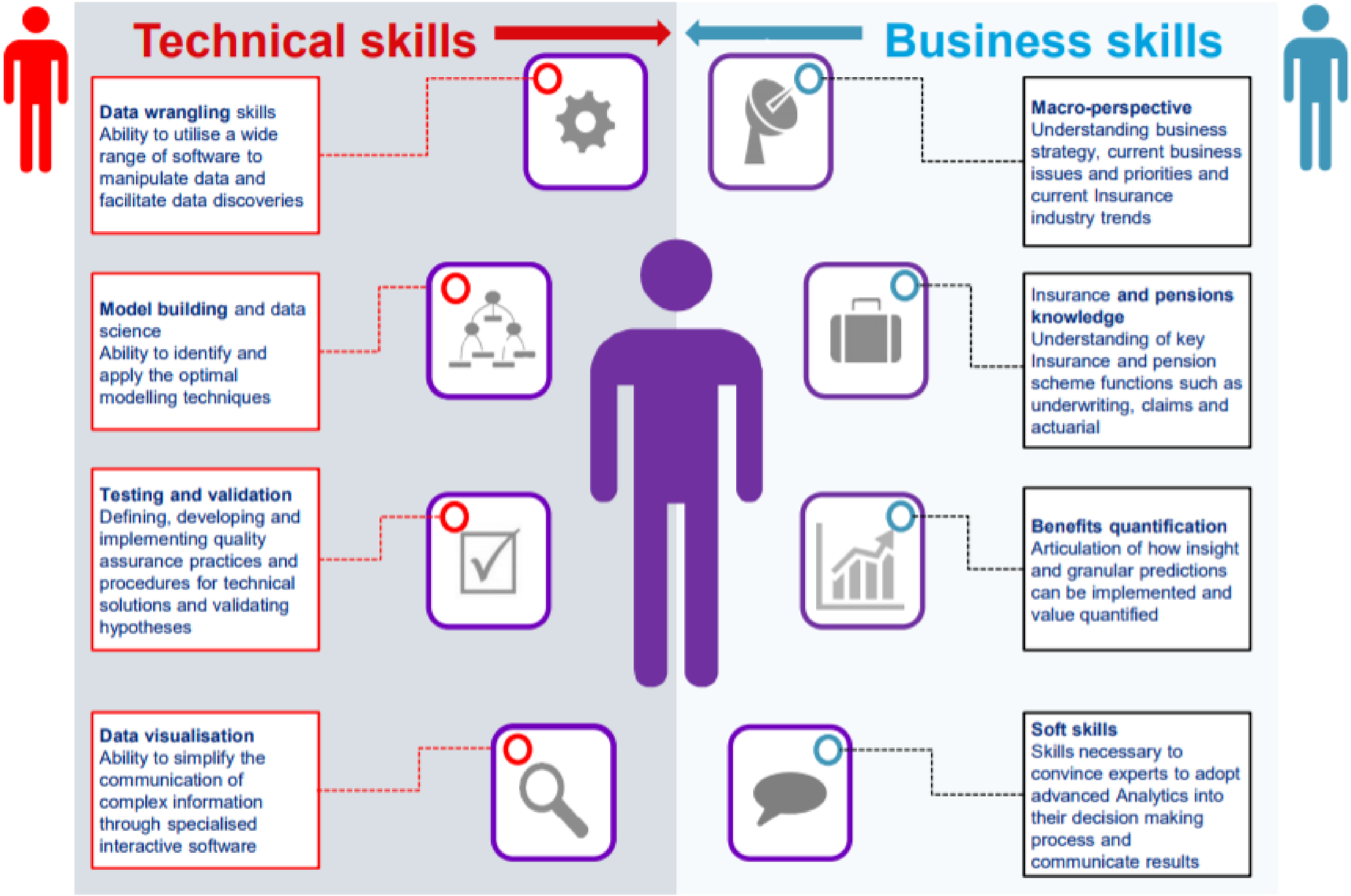


The smaller the sum of squared differences the better the fit of the line to the data.

Skill Mappings



Data Science Insurance Perspective



Unsupervised Learning

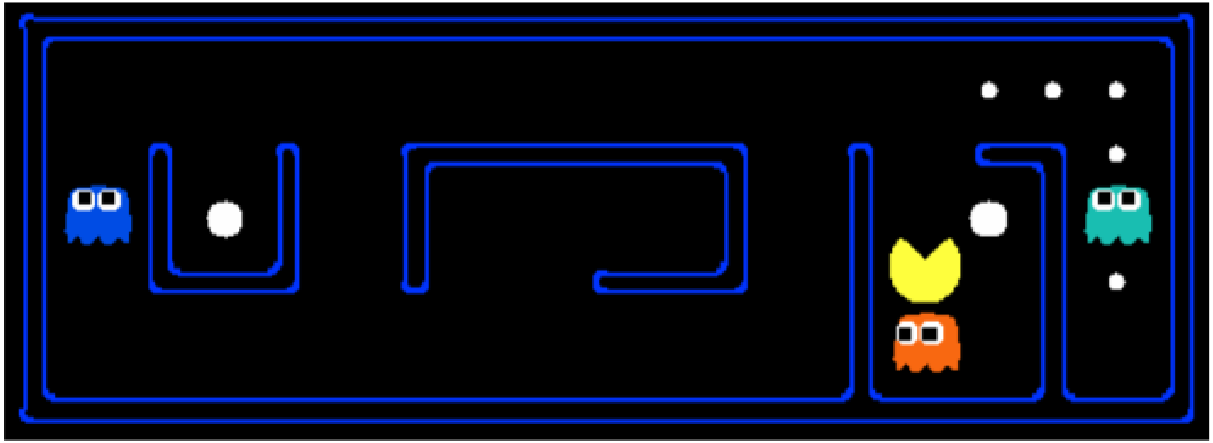
- **Unsupervised learning is used against data that has no historical labels.**
- **The goal is to explore the data and find some structure within.**

Reinforcement Learning

- **Reinforcement learning is often used for robotics, gaming and navigation.**
- **With reinforcement learning, the algorithm discovers through trial and error which actions yield the greatest rewards.**

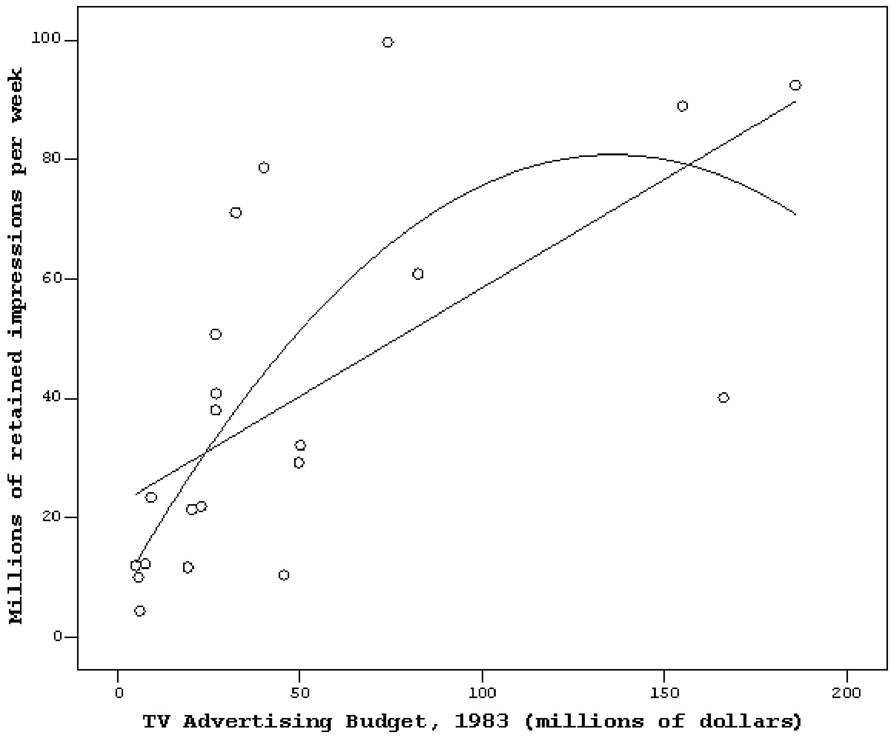
Reinforcement Learning

- You have some sort of agent that "explores" some space
- Learns the value of different state changes in different conditions
- Those values inform subsequent behavior of the agent
- Yields fast on-line performance once the space has been explored

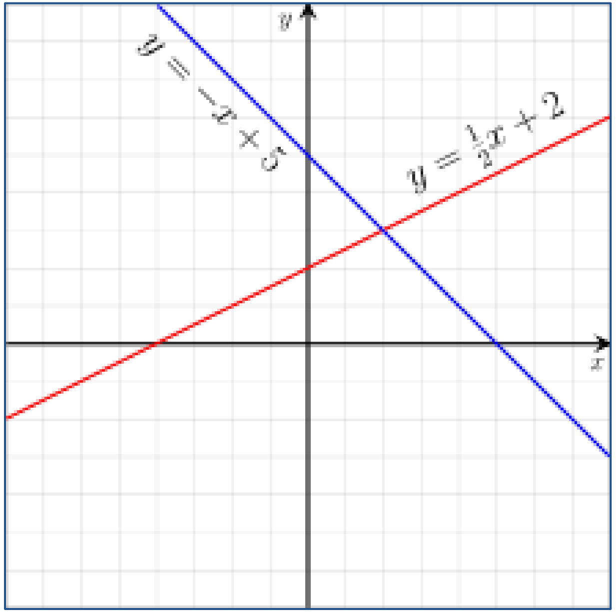


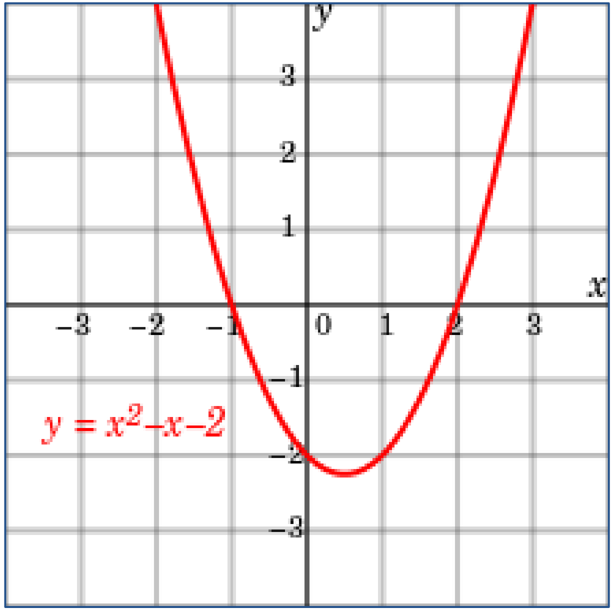
Regression

- Relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables)
- Commonly used for predictive analysis



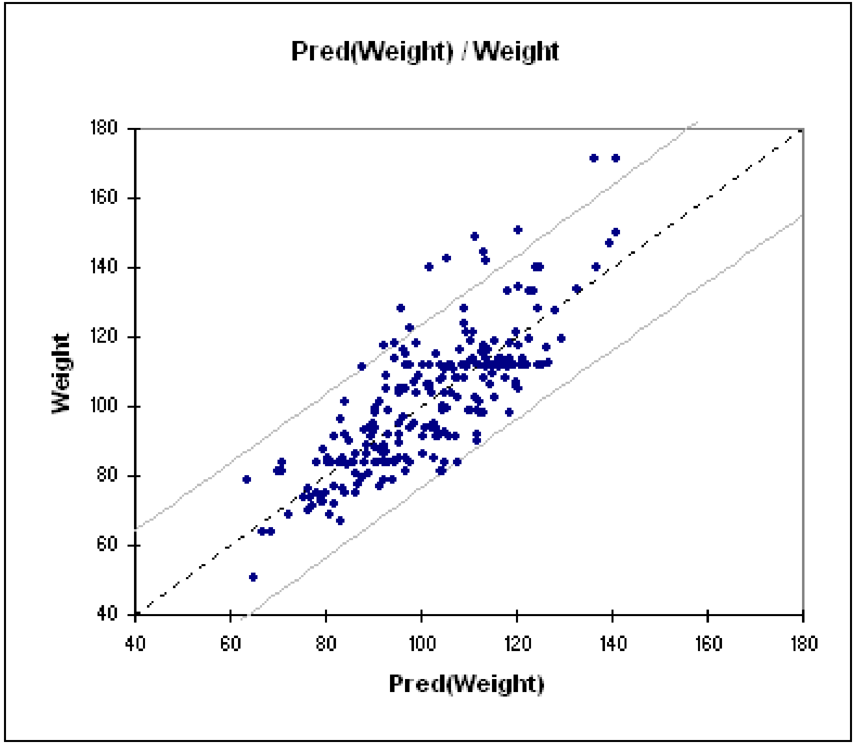
Regression





How this works

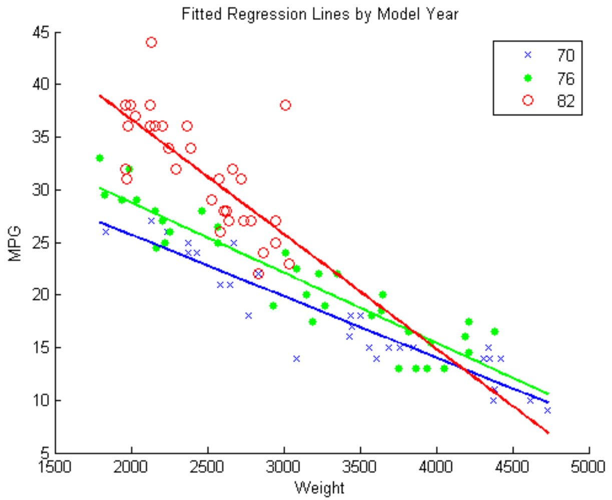
- Usually : “least squares”
- Minimizes the squared-error between each point and the line
- The slope is the correlation between the two variables
- This is the same as maximizing the likelihood of the observed data



http://localhost:8890/notebooks/Desktop/COE_29th%20March%202017/Data%20science/Python/ML_Demos/LinearRegression.ipynb

Multivariate regression (Multiple Regression)

- What if more than one variable influences
- Example: predicting a price for a car based on its many attributes (body style, brand, mileage, etc.)




DEMO

K-Means Clustering

- Attempts to split data into K groups that are closest to K centroids
- Unsupervised learning – uses only the positions of each data point
- Can uncover interesting groupings of people/ things / behavior

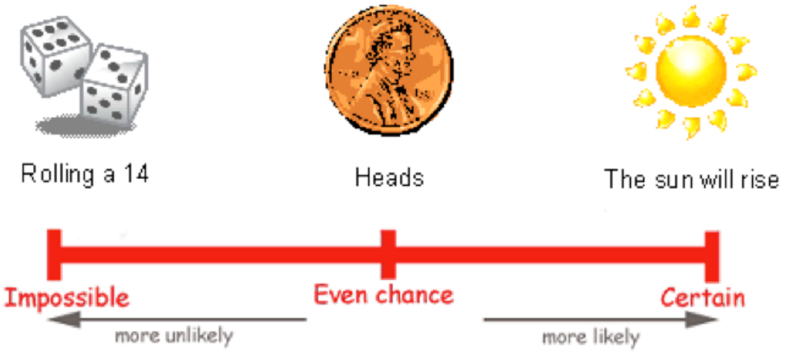
Example: Where do millionaires live?

- What genres of music / movies / Cars ?
- Create your own stereotypes from demographic data

http://localhost:8890/notebooks/Desktop/COE_29th%20March%202017/Data%20science/Python/ML_Demos/KMeans.ipynb

Bayesian Methods

- $P(A|B) = \frac{P(A)P(B|A)}{P(B)}$
- Let's use it for machine learning! I want a spam classifier.
- Example: how would we express the probability of an email being spam if it contains the word "free"?
- $P(Spam \mid Free) = \frac{P(Spam)P(Free \mid Spam)}{P(Free)}$



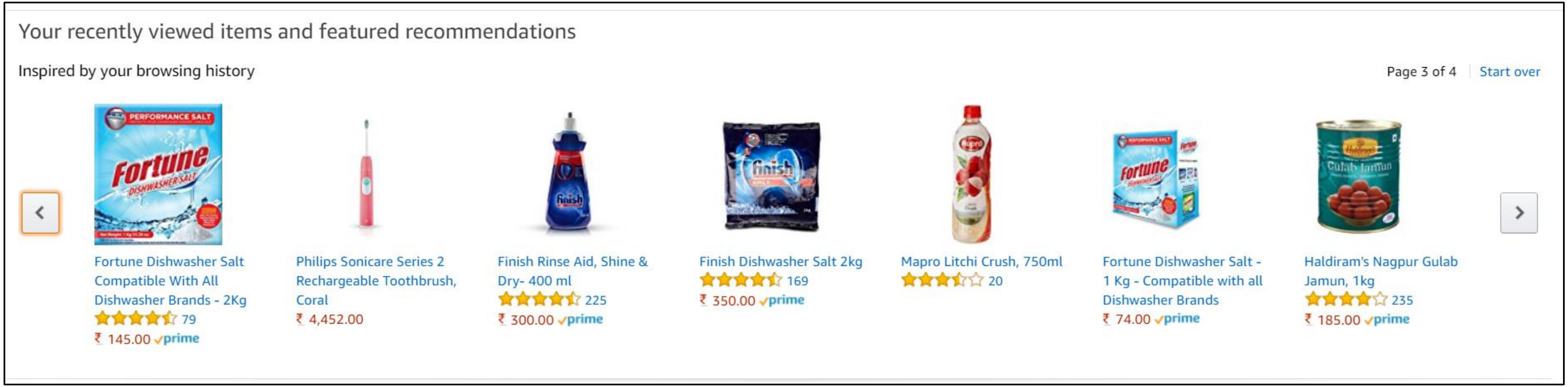
What about all the other words?

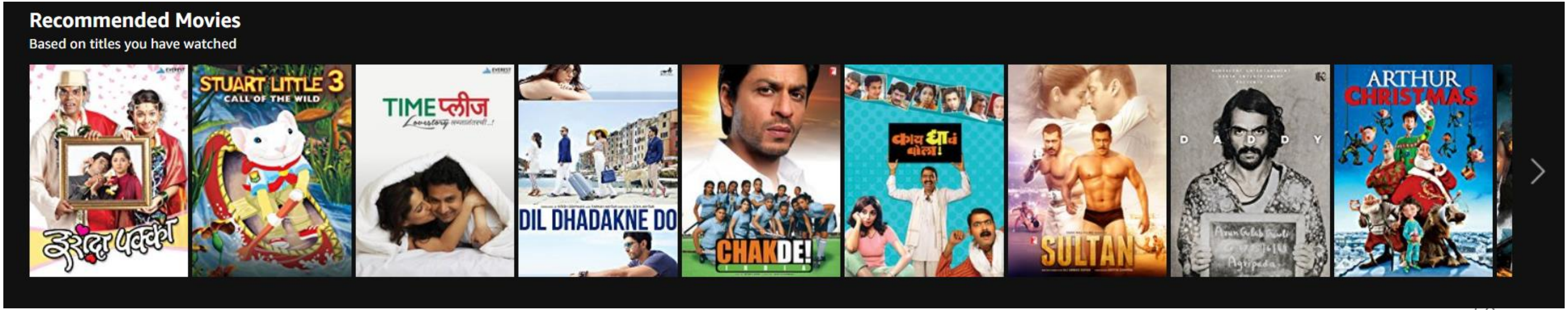
- We can construct P(Spam | Word) for every (meaningful) word we encounter during training
- Then multiply these together when analyzing a new email to get the probability of it being spam.

Classification problem

http://localhost:8890/notebooks/Desktop/COE_29th%20March%202017/Data%20science/Python/ML_Demos/NaiveBayes.ipynb

Recommender Systems





User-Based Collaborative Filtering

- Build a matrix of things each user bought/viewed/rated
- Compute similarity scores between users
- Find users similar to you
- Recommend stuff they bought/viewed/rated that you haven’t yet.

User-Based Collaborative Filtering

User-Based Collaborative Filtering

Problems with User-Based CF

- People are fickle; tastes change
- There are usually many more people than things
- People do bad things odd times

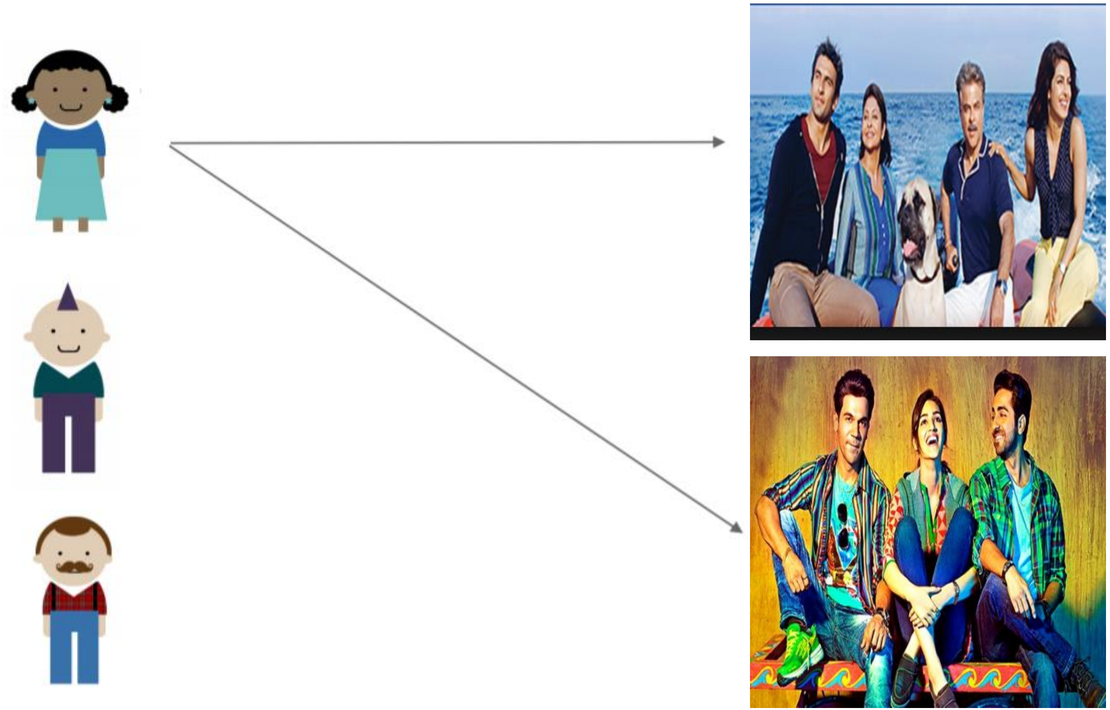
Item-Based Collaborative Filtering

- A movie will always be the same movie – Does it change?
- There are usually fewer things than people
- Harder to game the system

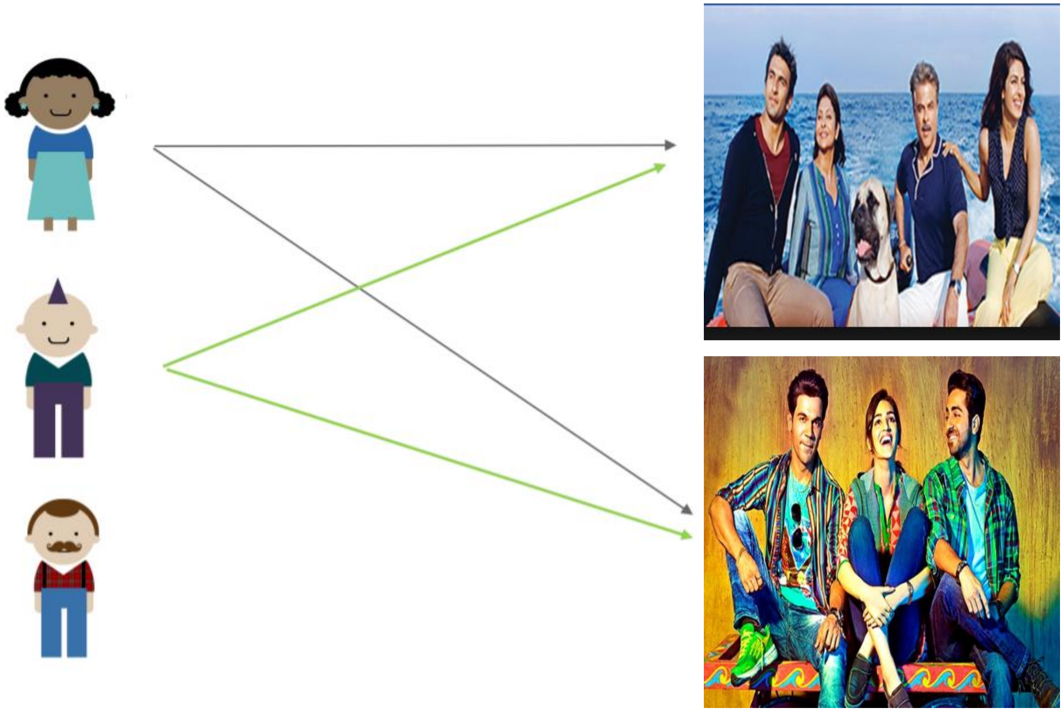
Item-Based Collaborative Filtering

- Find every pair of movies that were watched by the same person
- Measure the similarity of their ratings across all users who watched both
- Sort by movie, then by similarity strength

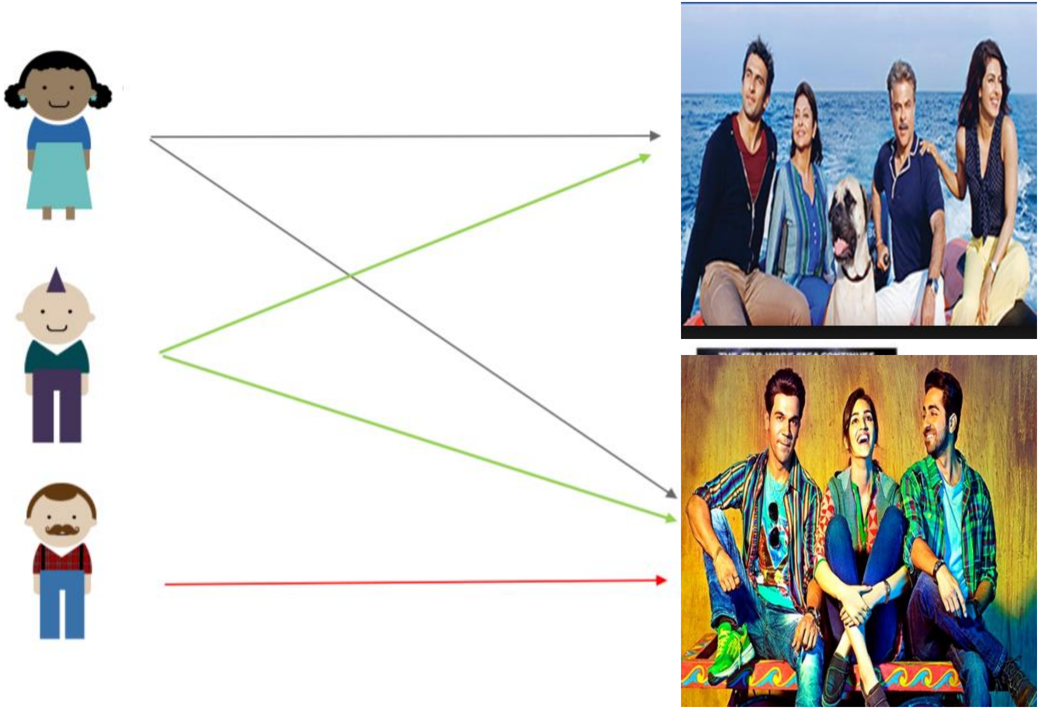
Item-Based Collaborative Filtering



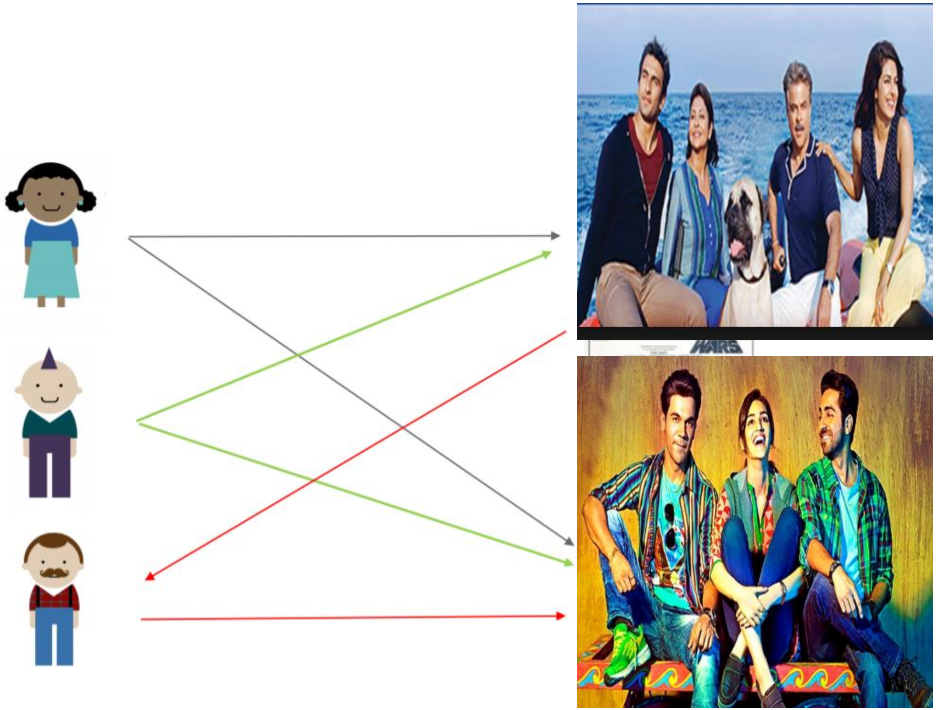
Item-Based Collaborative Filtering



Item-Based Collaborative Filtering



Item-Based Collaborative Filtering

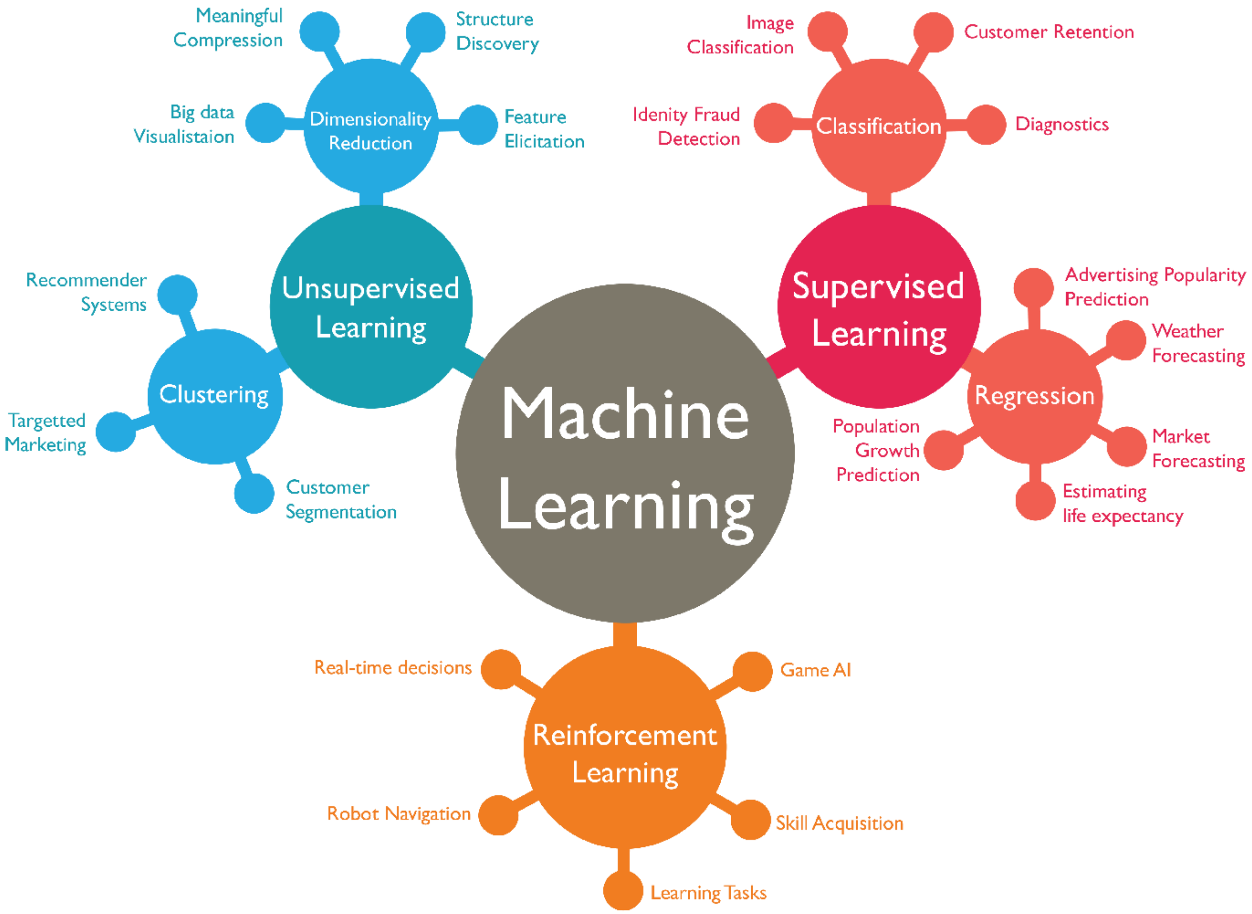


K-Nearest Neighbor (KNN)

- Used to classify new data points based on “distance” to known data
- Find the K nearest neighbors, based on your distance metric

http://localhost:8890/notebooks/Desktop/COE_29th%20March%202017/Data%20science/Python/ML_Demos/SimilarMovies.ipynb

Back up slides


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks