Data Science

Yogesh Kulkarni

Machine Learning

Introduction to Machine Learning

How do we learn?

- What do we do when we have to prepare for an examination?
- Study. Learn. Imbibe. Take notes. Practice mock papers.
- Thus, prepare for the unseen test.

What is Learning?

"Learning is any process by which a system improves performance from experience."

- Herbert Simon, Turing Award 1975, Nobel in Economics 1978.

What is Machine Learning?

Machine learning is a type of artificial intelligence (AI) which:

- Learns function without being explicitly programmed.
- Can grow and change when exposed to new data.

Quick Definition

Machine learning is a field of study that gives computers the ability to learn, without being explicitly programmed.
- Arthur Samuel, 1959

So, rather than coding each step explicitly, you give computers just some examples, and it figures out the steps.

Another Definition of Machine Learning

Machine learning is manifestation of statistical learning: implemented through software.

WellPosed Definition

A computer program is said to learn from experience E with respect to some class of tasks T and some performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.
- T. Mitchell's book Machine Learning (1997)
In the various problem settings T, P, and E can refer to completely different things.

Intuition

- For any process (Machine ie Artificial or Biological), need to define a task, say Spam Detection. Machine can also do it and humans can also detect it.
- Experience means multiple observations, runs, practices. So in our examples, its more Spam/Non-Spam examples.
- Performance defined is How many we got right ie Accuracy.
- Now if Task of Spam detection improves Accuracy if we look at more samples, then this is Machine Learning.
- Its also called Inductive Learning ("Learning by Experience"). Based on evidence (not Facts), so its suggestive.
- Another type is deductive (inferences based on facts/rules. So, more deterministic. Eg I went to Movie today. Today is Saturday. Inference: I went to movie on Saturday.)

Intuition

- Mitchell's definition, though looks OK for Machine Learning, its not very rigorous.
- Contrary example: Say your task is Bike driving, experiences is more miles you drive, performance is smoothness of drive (vibrations).
- As you put more miles, due to smoothening, performance improves (vibrations go down), is it Learning?
- No!!!
- But for Machine Learning domain, it looks fine.

Tasks

Tasks T in machine learning

- Classification of an instance to one of the categories based on its features;
- Regression – prediction of a numerical target feature based on other features of an instance;
- Clustering – identifying partitions of instances based on the features of these instances so that the members within the groups are more similar to each other than those in the other groups;
- etc

Experience

- Experience E refers to data (we can't go anywhere without it).
- For example, to predict loan defaults based on the data accumulated about our clients.
- Here, the experience E is the available training data: a set of instances (clients), a collection of features (such as age, salary, type of loan, past loan defaults, etc.) for each,
- And a target variable (whether they defaulted on the loan) is (1 or 0),
- This is a (binary) classification problem.

Performance

- Metric of the algorithm's performance evaluation P
- Such metrics differ for various problems and algorithms
- A simple metric for classification algorithms, the proportion of correct answers – accuracy – on the test set.

So, What is Machine Learning?

- Ability of computers to "learn" from "data"
- Learn: Discover patterns, underlying structure
- Data: Comes from sensors, transactions, etc.

Goal of Statistical learning

Dependent variables need to predicted or estimated in terms of independent variables.

- Data in control: independent variables.
- Data not in control: dependent variables.

Example

Goal: to measure sales based on the advertising budget allocated for TV, Radio, and Print.

- Can control: budgets of TV, Radio, and Print.
- Cannot control: how they will impact the sales.
- Express dependent (sales) as function of independent (advertising budget).
- Want to uncover this hidden relationship.
- Statistical learning reveals hidden data relationships.
- Relationships between dependent and independent data.

Mathematical Definition of Machine Learning

Machine Learning comes up with a Model given inputs and targets.

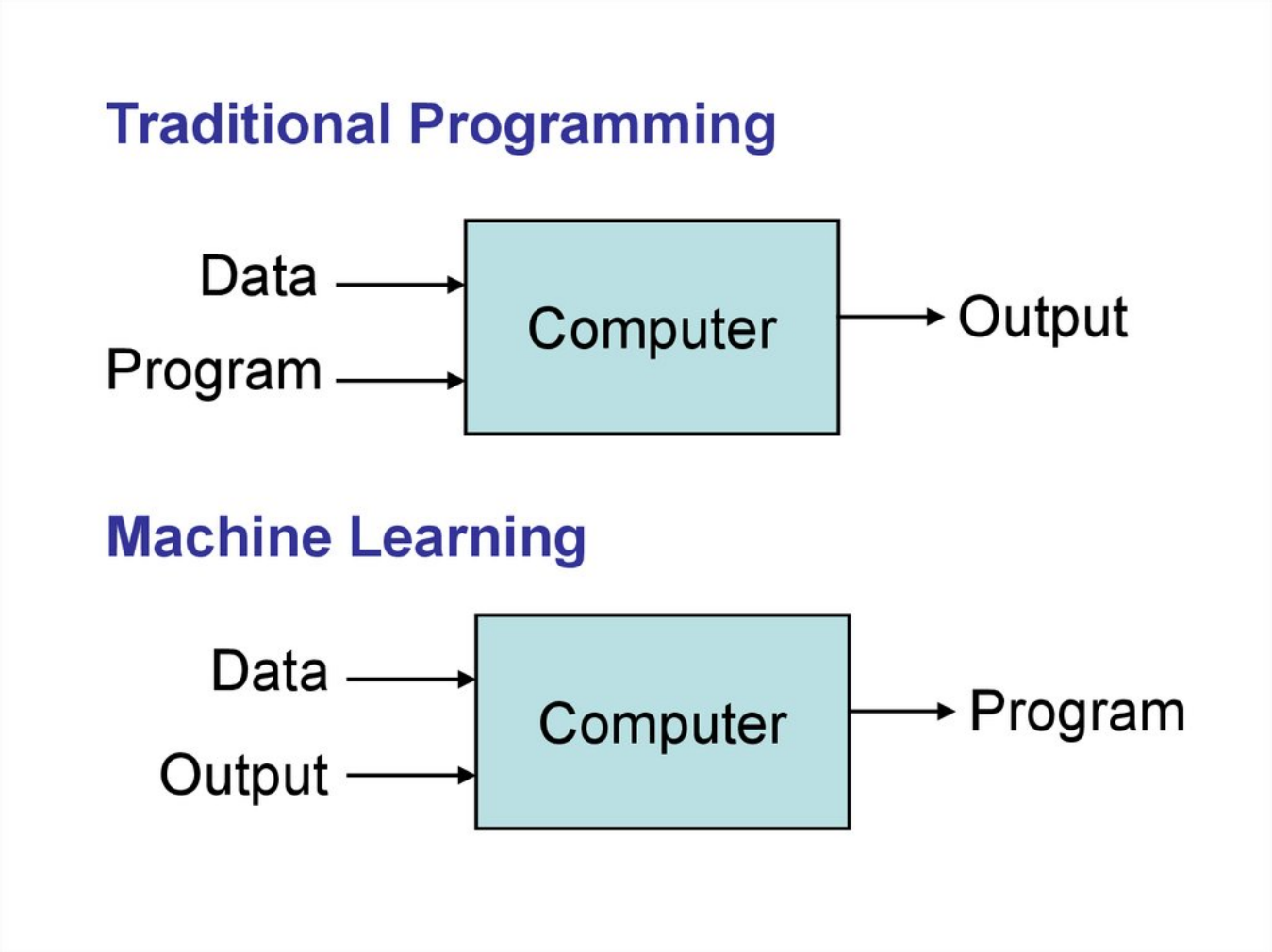
- Input data is available.
- Input data is transformed to get output.
- Output: something that needs to be predicted or estimated.
- Transformation engine is called Model or function.

Model entities

For $income = c + \beta_0 \times education + \beta_1 \times experience$

- Inputs: Education and experience, also called as features or attributes or dimensions or variables.
- Mathematical entities added to input data, are Parameters.. β_0 and β_1 are parameters
- Income is target, also called as outcome or class.

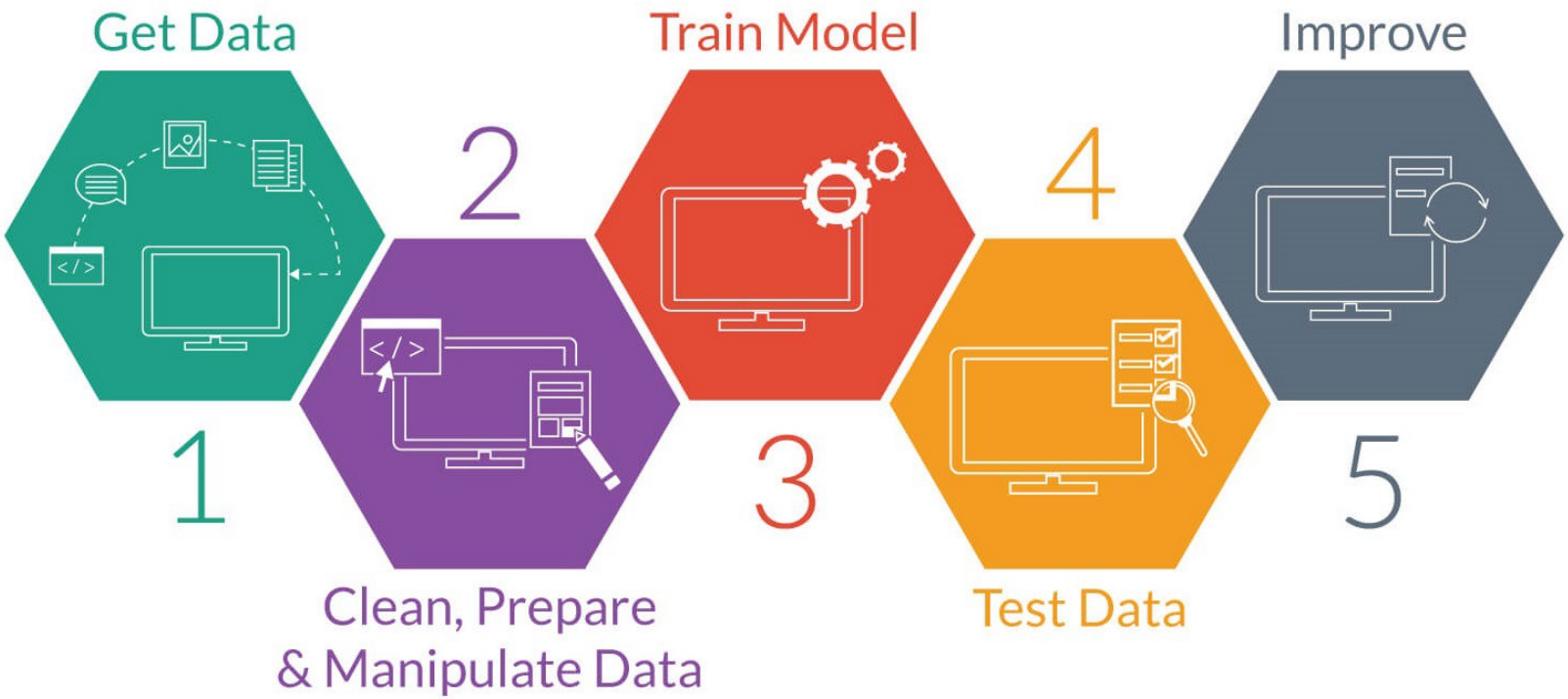
Traditional vs. Machine Learning?



Why Machine Learning?

- Problems with High Dimensionality
- Hard/Expensive to program manually
- Techniques to model 'ANY' function given 'ENOUGH' data.
- Job $$$

Machine Learning Process



(Reference: The Role of Big Data in Strengthening Machine Learning Projects - Techno FAQ)

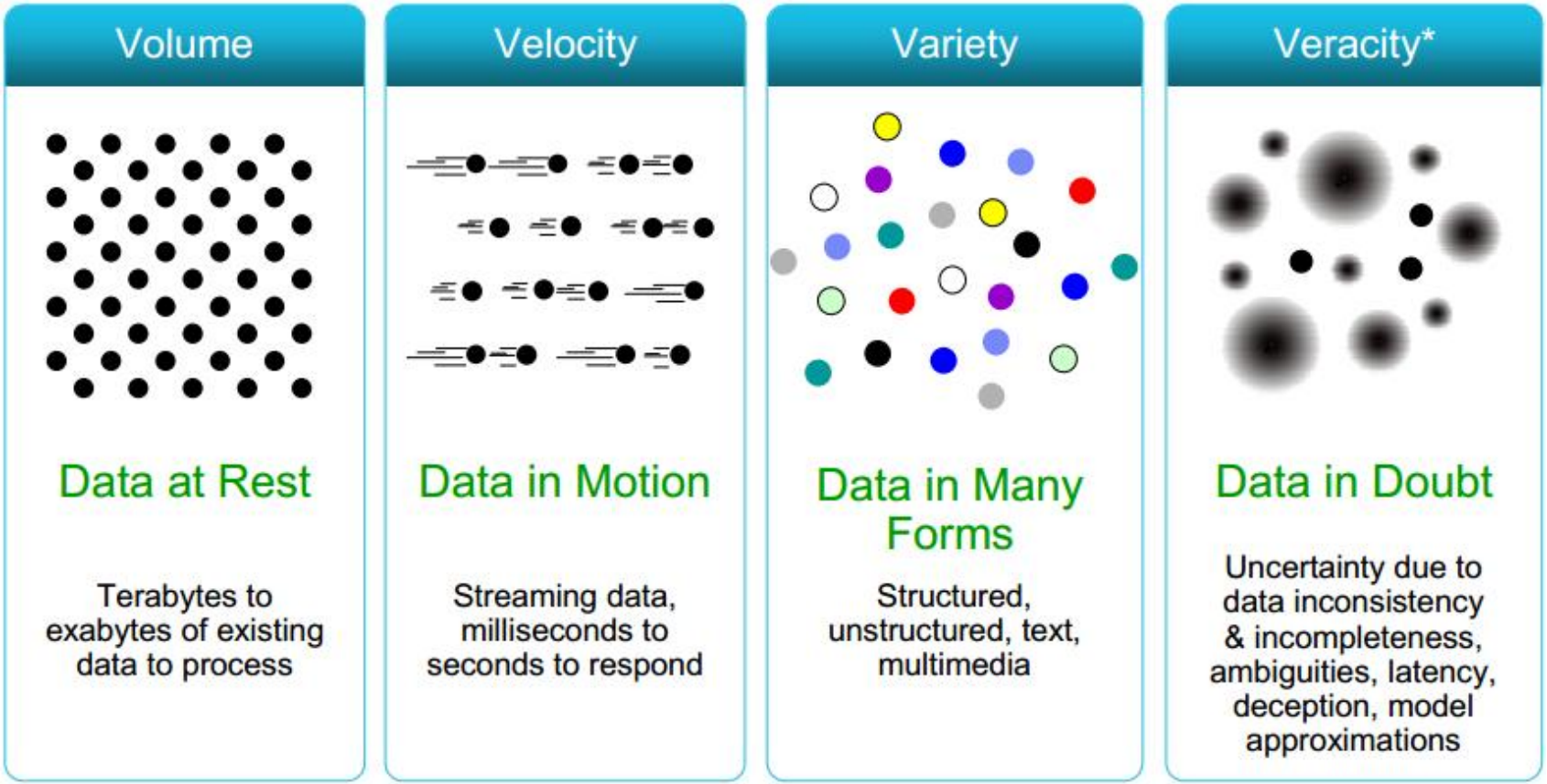
Why now?

- Flood of data (Internet, IoT)
- Increasing computational power
- Easy/free availability of algorithms
- Increasing support from industries

The storm: The Big Data is coming

- In 2012, HBR put Data Scientists on the radar
- "The Sexiest Job of the 21st Century".
- Industry, trying to be data-driven, than manual.

(Big) Data Characteristics



(Image Credit: http://www.rosebt.com/blog/data-veracity)

What's the answer?

AI-ML-DL

- Machines showing intelligence of Humans
- Machine Learning: part of AI
- Logic is not programmed by hand,
- Gets emerged in training with data.

A Puzzle

How different is Machine Learning?

Maths Puzzle

Math Quiz #1 - Teacher's Answer Key

1) 2 4 5 = 3	5) 6 2 2 = 10
2) 5 2 8 = 2	6) 3 1 1 = 2
3) 2 2 1 = 3	7) 5 3 4 = 11
4) 4 2 2 = 6	8) 1 8 1 = 7

Maths Puzzle

- Letting the computer work out that relationship for you.
- 'Learn' to solve such problems,
- 'Test' with any other problem of the same type!

Types of Machine Learning

Two kinds of learning

- Supervised
- Unsupervised

Supervised

- Training data with correct answers
- Both used to train the model
- Then apply unseen data on model

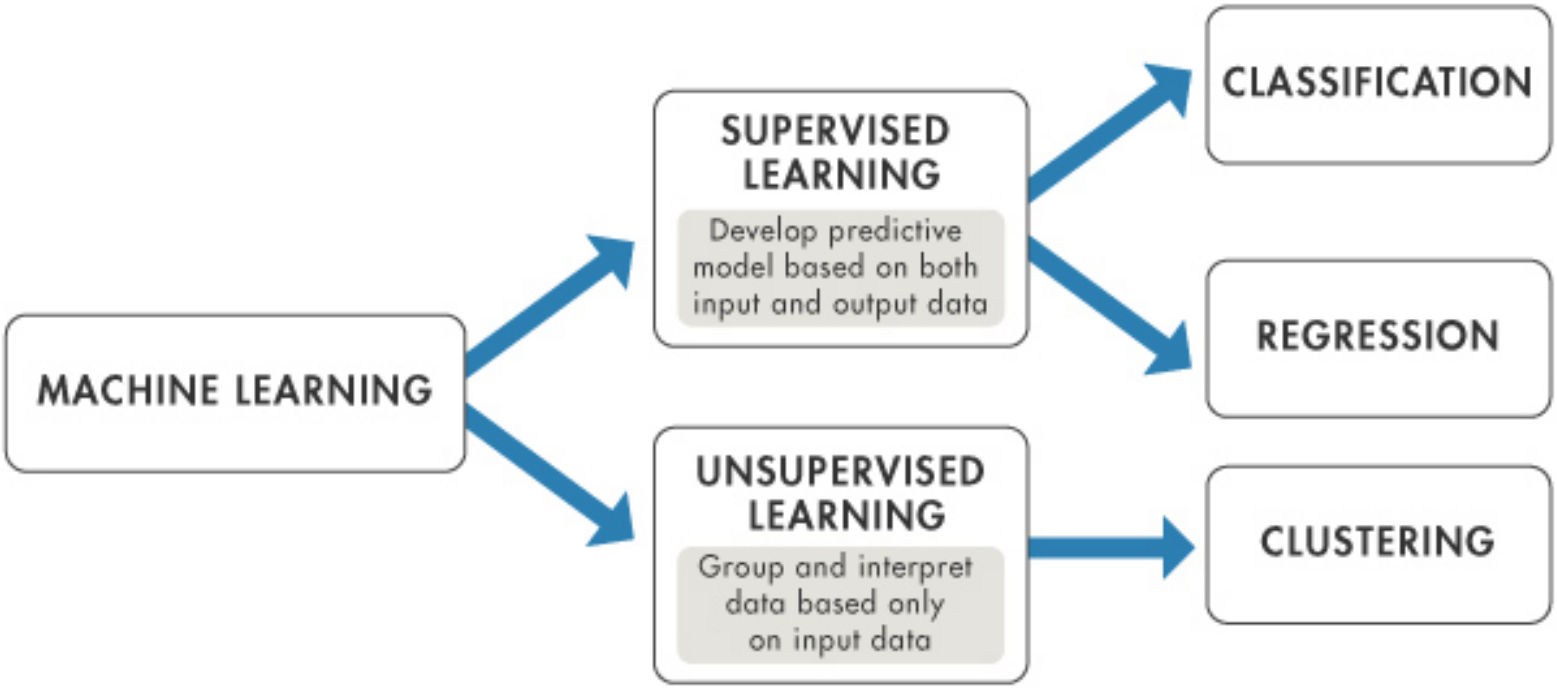
Unsupervised

- Training data with no answers
- Extract patterns, groups

Some types of algorithms

- Prediction: predicting a continuous variable from data
- Classification: assigning records to predefined groups
- Clustering: splitting records into groups based on similarity
- Association learning: seeing what often appears together

Machine Learning Learning Algorithms



(Reference: Machine Learning in MATLAB - MATLAB & Simulink - MathWorks)

Machine Learning Learning Algorithms

- Is this A or B? : Classification algorithms
- Is this weird? : Anomaly detection algorithms
- How much—or—How many? : Regression algorithms
- How is this organized? : Clustering algorithms, Dimensionality reduction
- What should I do next? : Reinforcement learning algorithms

(Ref: Brandon Rohrer's breakdown of the "5 questions data science answers")

Classification

- **Description:** Identifying the category an object belongs to.
- **Applications:** Spam detection, Image recognition.
- **Algorithms:** SVM, nearest neighbors, random forest, Logistic Regression

Regression

- **Description:** Predicting a continuous-valued attribute associated with an object.
- **Applications:** Drug response, Stock prices.
- **Algorithms:** Linear Regression

Clustering

- **Description:**Automatic grouping of similar objects into sets.
- **Applications:** Customer segmentation, Grouping experiment outcomes
- **Algorithms:** k-Means

Dimensionality Reduction

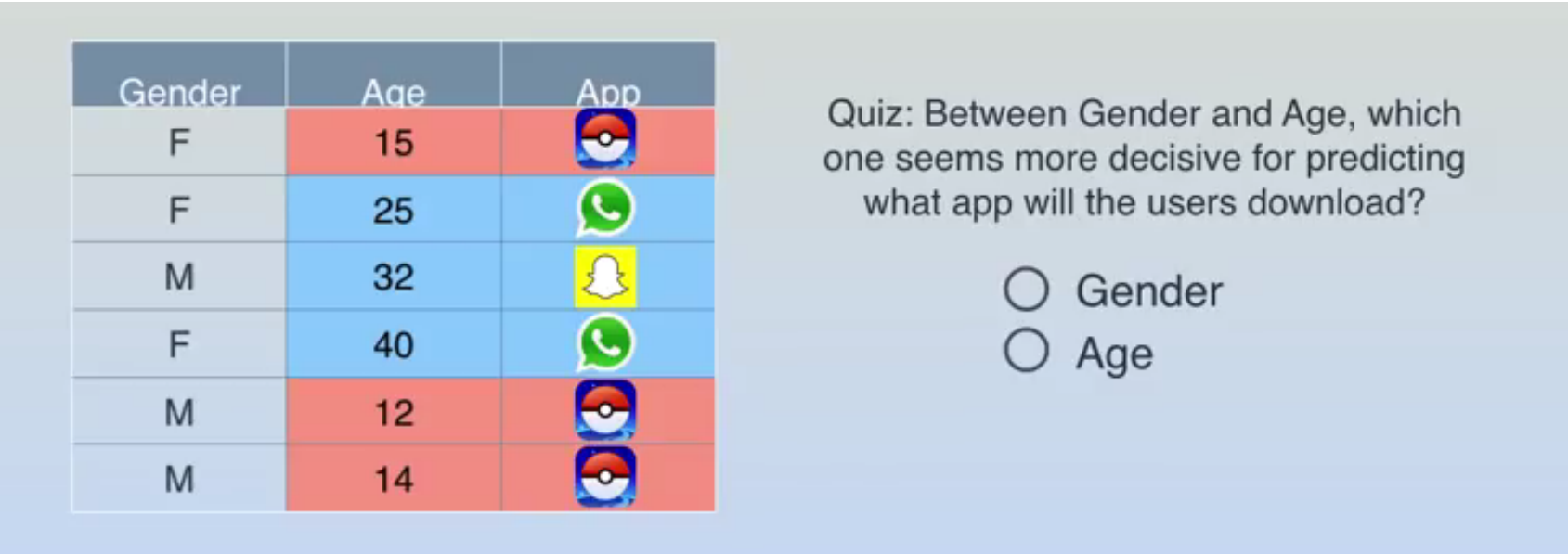
- **Description:** Reducing the number of random variables to consider.
- **Applications:** Visualization, Increased efficiency
- **Algorithms:** PCA, Singular Value Decomposition

Popular Algorithms in Machine Learning

- Linear, Logistic Regression
- Decision Trees
- SVM - Support Vector Machines, Naive Bayes
- K-Means

Popular Algorithms in Machine Learning

Recommendation: Is Gender the decider or Age?



Gender is not much, but all below 20 years downloaded Pokemon Go.Age splits data best.

(Image Credit: A Gentle Introduction To Machine Learning; SciPy 2013 Presentation - Kastner, Kyle)

Popular Algorithms in Machine Learning

Decision Tree.

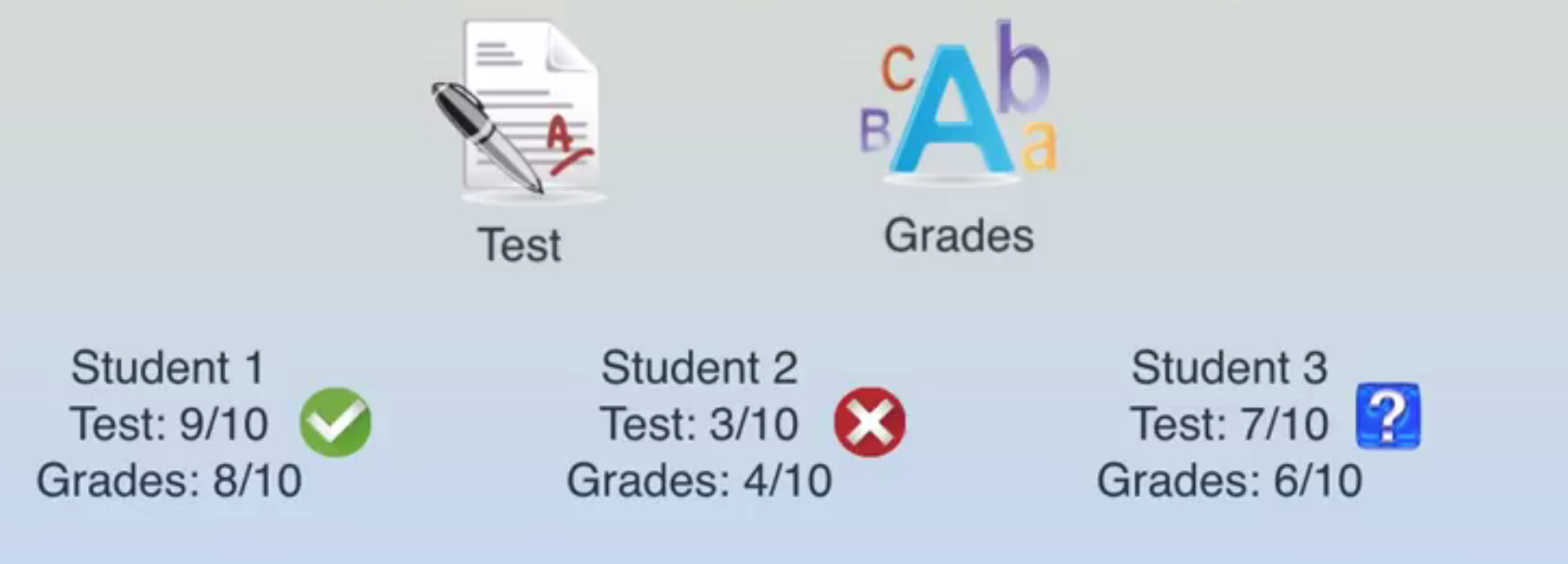
Gender	Age	App
F	15	
F	25	
M	32	
F	40	
M	12	
M	14	

Any new person can walk through the Tree and predict.

(Image Credit: A Gentle Introduction To Machine Learning; SciPy 2013 Presentation - Kastner, Kyle)

Popular Algorithms in Machine Learning

Deciding acceptance to Univ based on Test scores and grade.

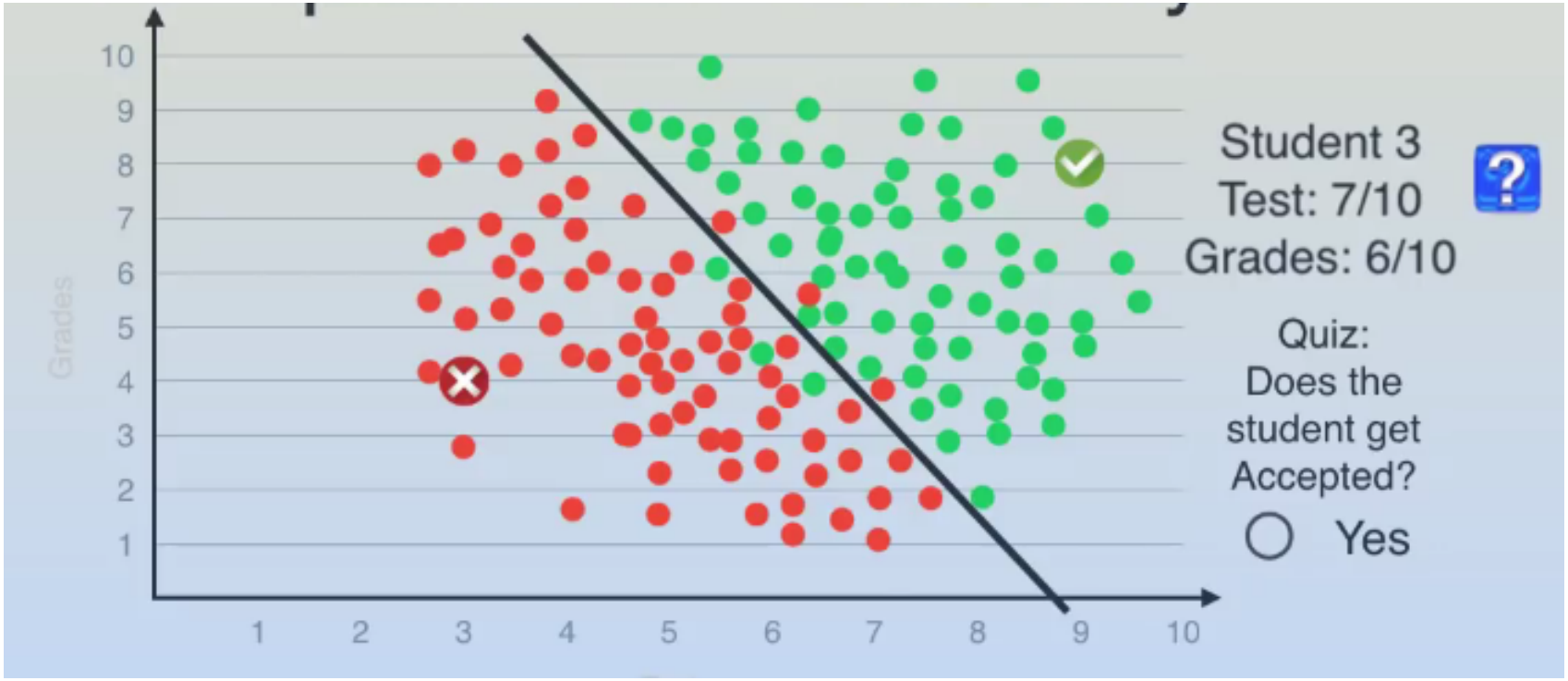


Will Student 3 get accepted?

(Image Credit: A Gentle Introduction To Machine Learning; SciPy 2013 Presentation - Kastner, Kyle)

Popular Algorithms in Machine Learning

Putting it in a grid. Test score as X and Grades as Y. Prev data shown with acceptance colored.

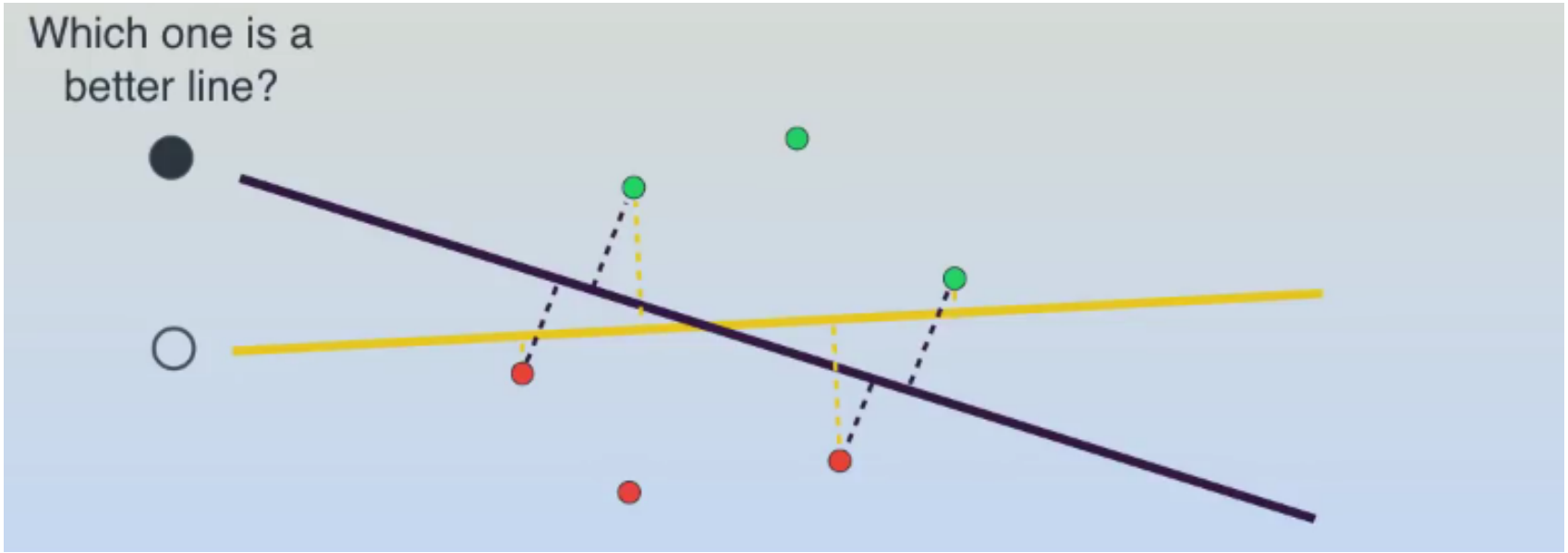


After separating line, Student 3's fate can be predicted. That's Logistic Regression.

(Image Credit: A Gentle Introduction To Machine Learning; SciPy 2013 Presentation - Kastner, Kyle)

Popular Algorithms in Machine Learning

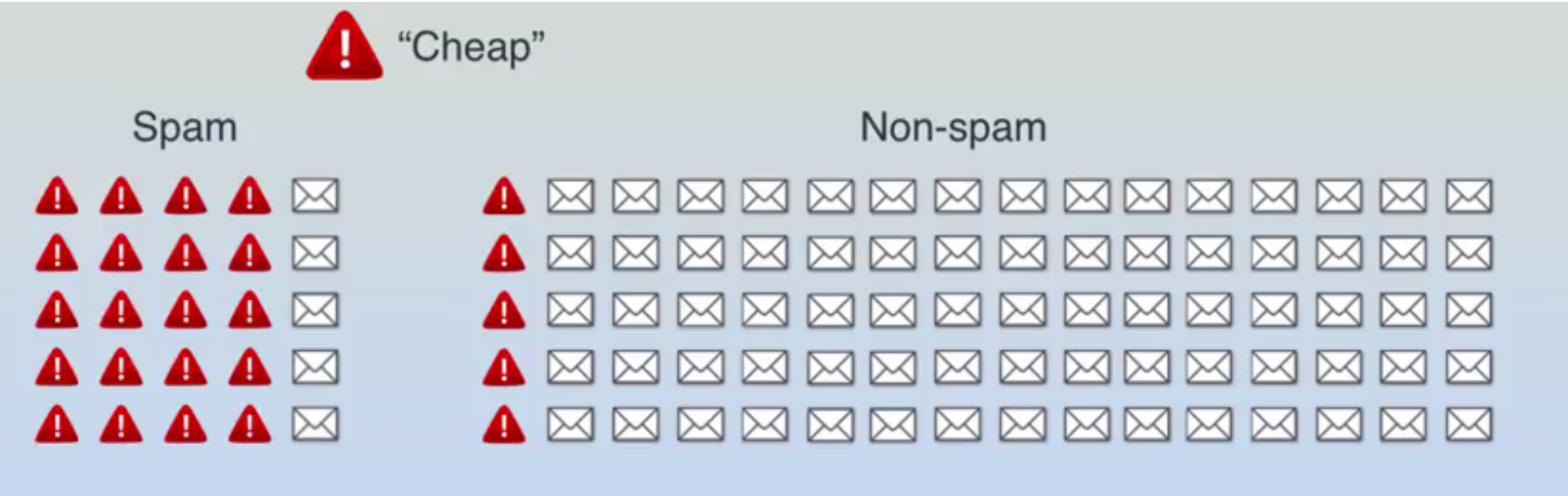
Which line separates best?



The one with max separation in the middle. That's Support Vector Machine.
(Image Credit: A Gentle Introduction To Machine Learning; SciPy 2013 Presentation - Kastner, Kyle)

Popular Algorithms in Machine Learning

Classification: Detecting if mail is spam or not.

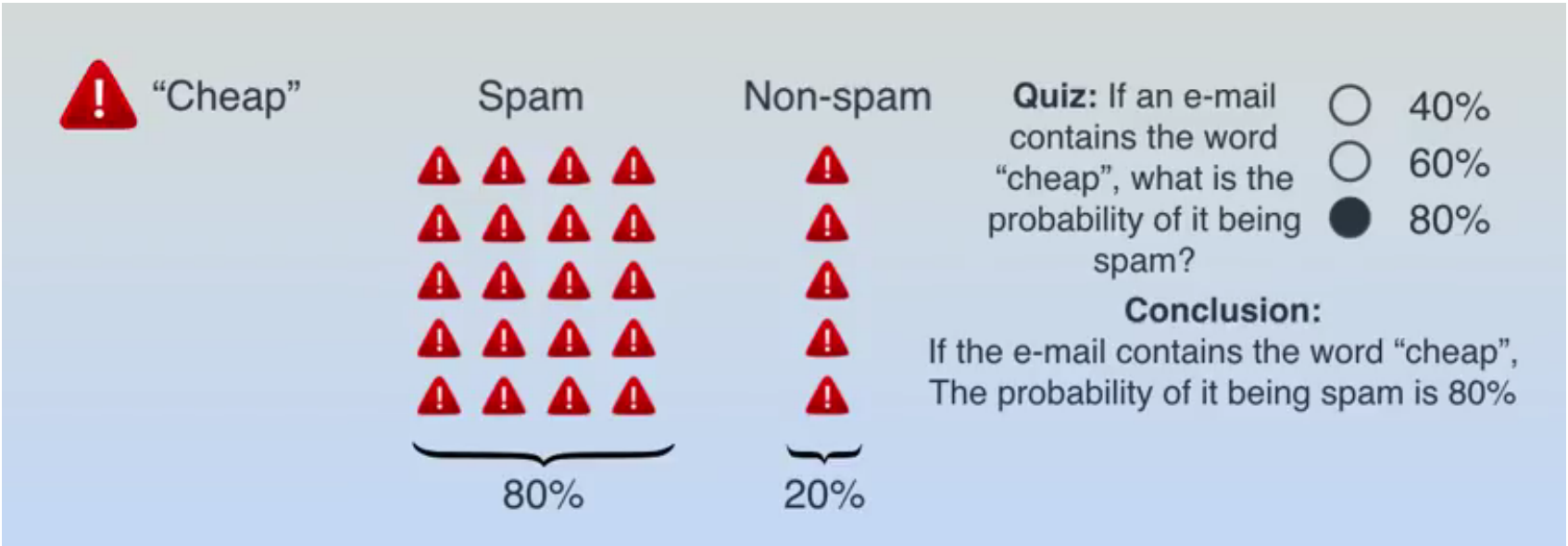


Features: mail containing "cheap".
Most past spam mails contain it.

(Image Credit: A Gentle Introduction To Machine Learning; SciPy 2013 Presentation - Kastner, Kyle)

Popular Algorithms in Machine Learning

Easy to compute probability of being a Spam, if it contains “cheap”.

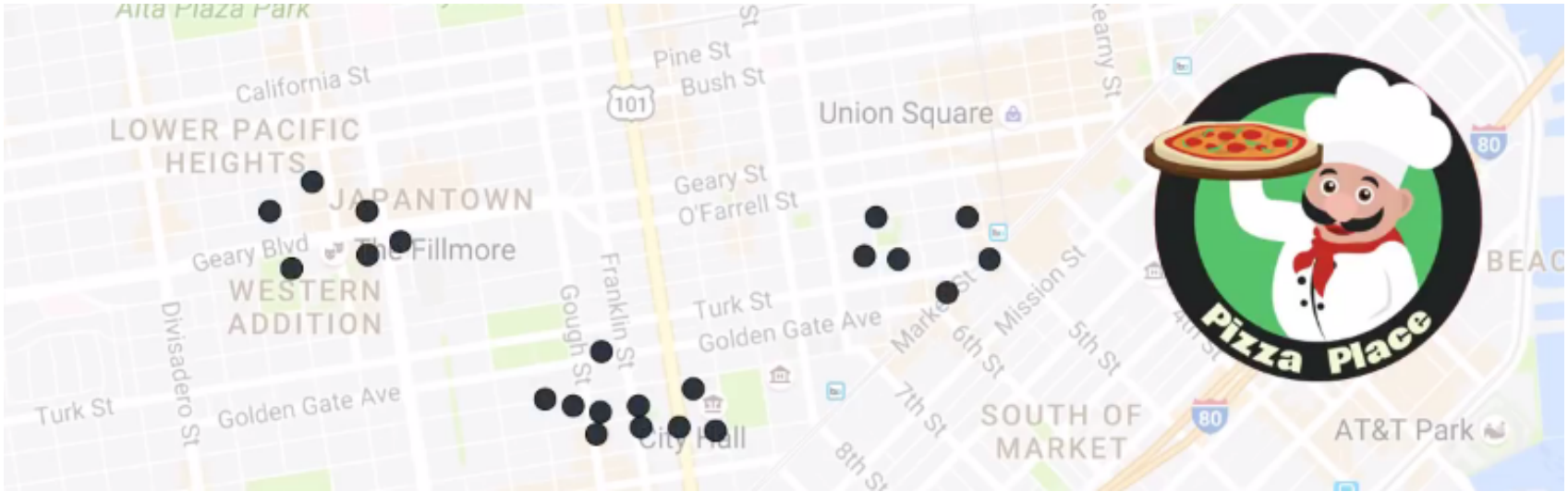


That’s Naive Bayes.

(Image Credit: A Gentle Introduction To Machine Learning; SciPy 2013 Presentation - Kastner, Kyle)

Popular Algorithms in Machine Learning

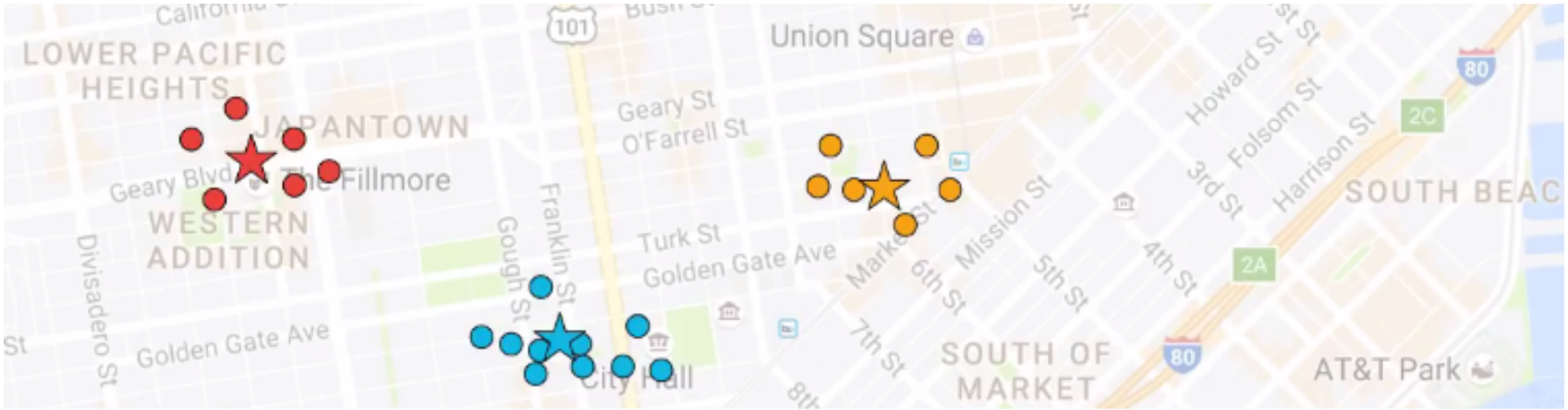
Wish to put 3 pizza shops in a city. Customers are plotted. What are the best locations?



Start with random locations and set ownership. Update locations. Repeat.

Popular Algorithms in Machine Learning

Locations settle at the centroids of the clusters.



That's K-means.

(Image Credit: A Gentle Introduction To Machine Learning; SciPy 2013 Presentation - Kastner, Kyle)

Example: Predicting House Price

(Ref: Machine Learning is Fun! - Adam Geitgey)

Numerical way

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Normaltown	$250,000
2	800	Hipsterton	$300,000
2	850	Normaltown	$150,000
1	550	Normaltown	$78,000
4	2000	Skid Row	$150,000

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Hipsterton	???

Machine Learning Type

- This is supervised learning.
- Knew how much each house sold for,
- So, knew the answer to the problem
- Need work backwards to figure out the logic.

Traditional way

BUT, how to decide which numbers to put? PRAY!!!

```
def estimate_house_sales_price(num_of_bedrooms, sqft, neighborhood):
  price = 0
  price_per_sqft = 200

  if neighborhood == "hipsterton":
    price_per_sqft = 400
  elif neighborhood == "skid row":
    price_per_sqft = 100

  price = price_per_sqft * sqft
  if num_of_bedrooms == 0:
    price = price - 20000
  else:
    price = price + (num_of_bedrooms * 1000)

  return price
```

Prayer

- Wouldn't it be better if computer figures out?
- Treat it as black box
- Feed Inputs and outputs
- That's it!!

```
def estimate_house_sales_price(num_of_bedrooms, sqft, neighborhood):
    price = <computer, plz do some math for me>
    return price
```

Prayer granted!!

- Notice the magic numbers
- .841, 1231.123, 2.324, and 201.234.
- These are weights.
- Better the weights - better the prediction!
- Done!!

```
def estimate_house_sales_price(num_of_bedrooms, sqft, neighborhood):
  price = 0
  price += num_of_bedrooms * .841
  price += sqft * 1231.123
  price += neighborhood * 2.324
  price += 201.234
  return price
```

How to figure out? A dumb way

Step 1: Start with each weight set to 1.0:

```
def estimate_house_sales_price(num_of_bedrooms, sqft, neighborhood):
  price = 0
  price += num_of_bedrooms * 1.0
  price += sqft * 1.0
  price += neighborhood * 1.0
  price += 1.0
  return price
```

How to figure out? A dumb way

Step 2: Guess/predict for all houses

Bedrooms	Sq. feet	Neighborhood	Sale price	My Guess
3	2000	Normaltown	$250,000	$178,000
2	800	Hipsterton	$300,000	$371,000
2	850	Normaltown	$150,000	$148,000
1	550	Normaltown	$78,000	$101,000
4	2000	Skid Row	$150,000	$121,000

Predictions NOT good, right?

What to do?

- Actual $250,000, but guessed $178,000
- Off by $72,000 for that single house.
- Diff can be positive or negative, so square it
- Add squared diffs of all houses.
- Total: $86,123,373.
- Thats whole error in the system.
- That's how "wrong" your function currently is.

What Next?

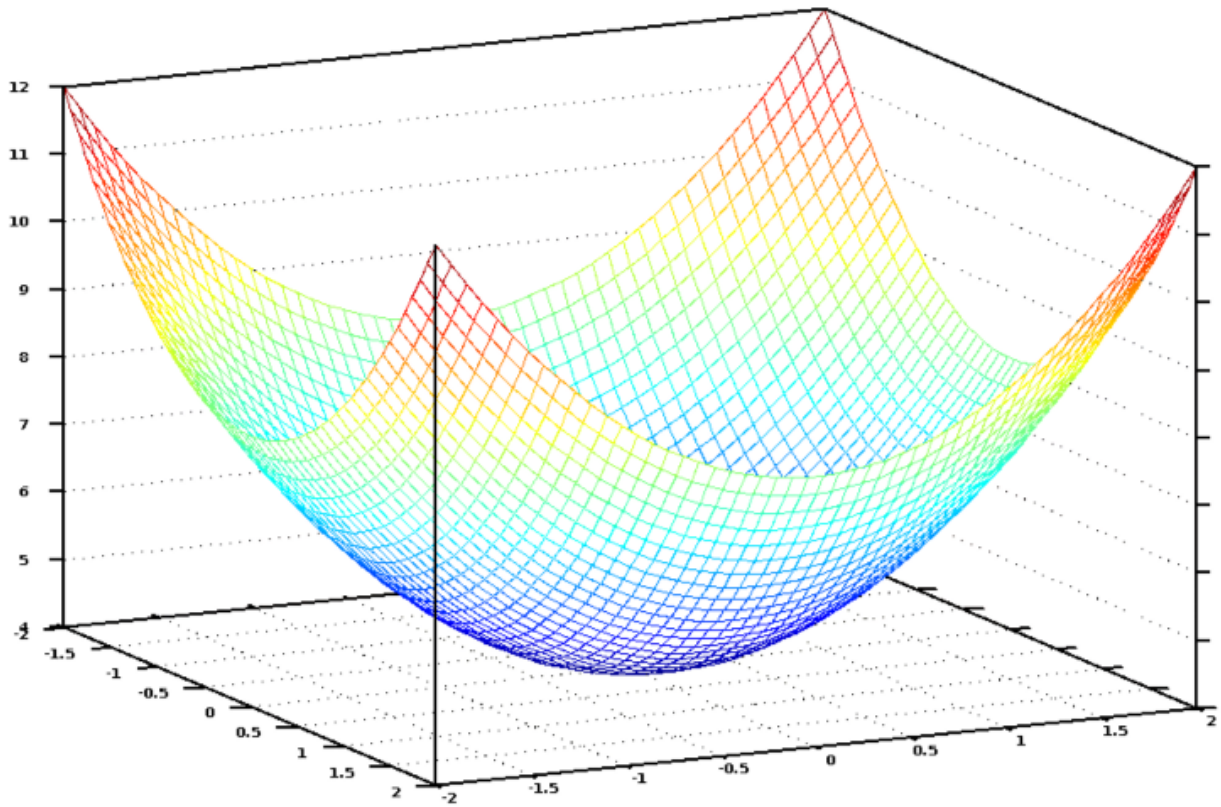
- Average per house error is “cost”.
- Get cost to be zero by playing with the weights.
- Thats the Goal!!!

Mathematically

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

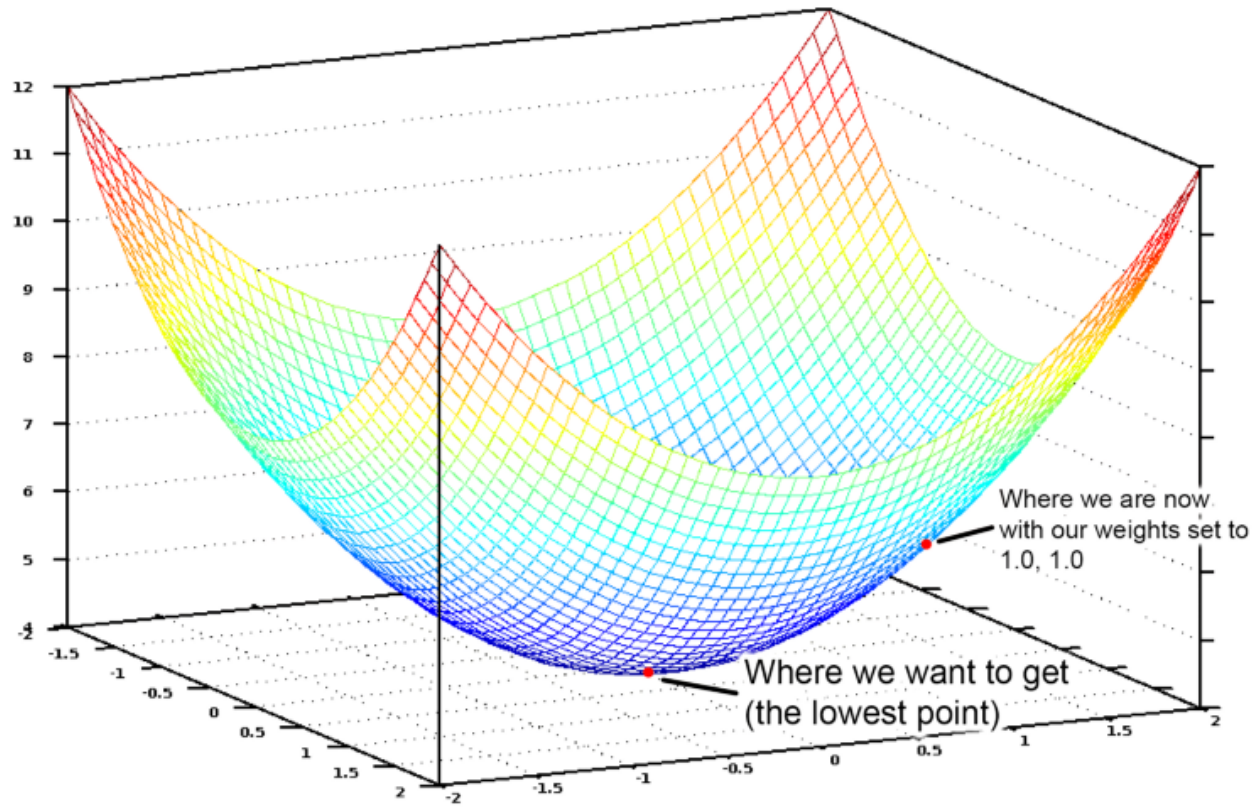
Graphically

Plotting cost values, for all possible ranges of weights for number_of_bedrooms and sqft:



Graphically

Cost is lowest at lowest point of the surface. Ideal.
Weights at that point are the answers!



How to find the lowest cost point?

- Start somewhere.
- Find direction (slope? Derivative? Partial?)
- Derivative: tells us which way is downhill for any given point on our graph.
- Move in slope direction.
- Adjust our weights to get to next point
- "walking down hill" towards the lowest point.
- That's gradient descent.
- Scikit Learn does this for you, hushsh!!

Calculus, anybody?

Repeat until convergence {

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

Applications of Machine Learning

Everyday Applications of Machine Learning

- Face Recognition (Facebook)
- Spam recognition in Emails
- Recommender Systems
- Feelings Analysis, Sentiments
- Natural language: Translate a sentence from Hindi to English, question answering, etc.
- Speech: Recognise spoken words, speaking sentences naturally
- Game playing: Play games like chess
- Robotics: Walking, jumping, displaying emotions, etc.
- Driving a car, flying a plane, navigating a maze, etc.

Cool-down: Summary

SO . . .

- What is Machine learning, after-all?
- Its usage in your domain?

Machine Learning Concepts

Machine Learning way of Solution, Mathematically

General Form

$Y = F(X_1, X_2, \ldots, X_p)$

- We have:
  - Observation of quantitative (numerical) response Y
  - Observation of p different predictors $X_1, X_2, \ldots, X_p$
- Find function F between Y and $X = X_1, X_2, \ldots, X_p$

General Form

As one may not get exact function F, approximate it to f, thus introducing some error.

- So, general form: $Y = f(X) + \epsilon$
- f is unknown function of $X_1, X_2, \ldots, X_p$
- ϵ is a random error term, Independent of X, Has mean equal to zero
- Statistical learning refers to approaches for estimating f

Estimate f

- Not concerned if f is linear, quadratic, etc.
- We only care that our predictions are "near accurate".
- So, f often treated as a black box.
- The aim is of minimizing reducible error

How do we estimate f?

Most statistical learning methods classified as:

- Parametric
- Non-parametric

How do we estimate f?

- Parametric:
  - Assume that the functional form, or shape, of f in linear in X, $f(X) = \sum_{i=1}^{p} \beta_i X_i$
  - This is a linear model, for p predictors $X = X_1, X_2, \ldots, X_p$
  - Model fitting involves estimating the parameters $\beta_0, \beta_1, \ldots, \beta_p$
  - Reduces the problem of estimating f to estimating a set of parameters
- Non-Parametric: No explicit assumptions about the functional form of f

Comparison

Parametric

- Only need to estimate set of parameters
- Bias such as linear model
- May not fit data well

Non-Paremetric

- Potential of many shapes for f
- Lots of observations needed
- Complex models can overfit

Simple Example

- Linear regression is a simple and useful tool for predicting a quantitative response. The relationship between input variables $X = (X_1, X_2, \ldots X_p)$ and output variable Y takes the form: $Y \approx \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon$
- $\beta_0, \beta_1, \ldots \beta_p$ are the unknown coefficients (parameters) which we are trying to determine.
- The best coefficients will lead us to the best "fit", which can be found by minimizing the residual sum squares (RSS), or the sum of the difference between the actual ith value and the predicted ith value.
- $RSS = \sum_{i=1}^{n} = e_i$, where $e_i = y_i - y'$

How to find best fit?

Matrix Form:

- We can solve the closed-form equation for coefficient vector w: $w = (X^T X^{-1}) X^T Y$.
- X represents the input data and Y represents the output data.
- This method is used for smaller matrices, since inverting a matrix is computationally expensive.

How to find best fit?

Gradient Descent:

- First-order optimization algorithm.
- We can find the minimum of a convex function by starting at an arbitrary point and repeatedly take steps in the downward direction, which can be found by taking the negative direction of the gradient.
- After several iterations, we will eventually converge to the minimum. In our case, the minimum corresponds to the coefficients with the minimum error, or the best line of fit.
- The learning rate α determines the size of the steps we take in the downward direction.

How to find best fit?

Gradient descent algorithm in two dimensions (meaning with 2 features, as example):

- Repeat until convergence.

$$w_0^{t+1} = w_0^t - \alpha \frac{\partial J(w_0, w_1)}{\partial w_0}$$

$$w_1^{t+1} = w_1^t - \alpha \frac{\partial J(w_0, w_1)}{\partial w_1}$$

- For non convex functions, gradient descent no longer guarantees an optimal solutions since there may be local minimas.
- Instead, we should run the algorithm from different starting points and use the best local minima we find for the solution.

How to find best fit?

Stochastic Gradient Descent:

- Instead of taking a step after sampling the entire training set, we take a small batch of training data at random to determine our next step.
- Computationally more efficient and may lead to faster convergence.

Cross Validation

Background: Imp Terms

- **Learning algorithm:** identifies a model that best fits the relationships between inputs and outputs.
- **Training set:** consists of records with features and with/without class labels
- **Test set:** consists of records with only features, class labels to be computed.

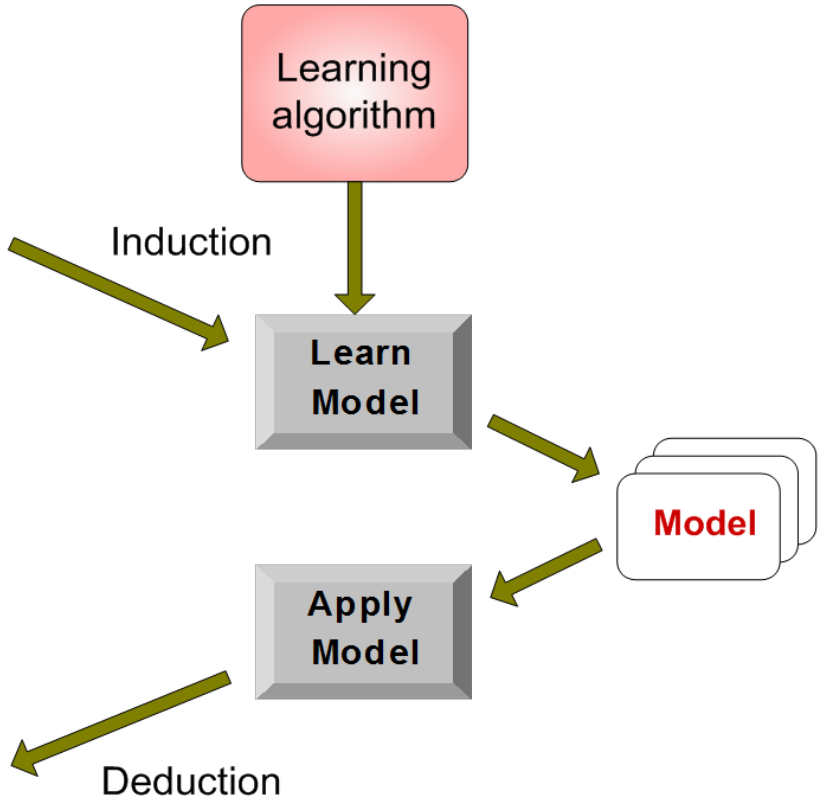
Training - Testing

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

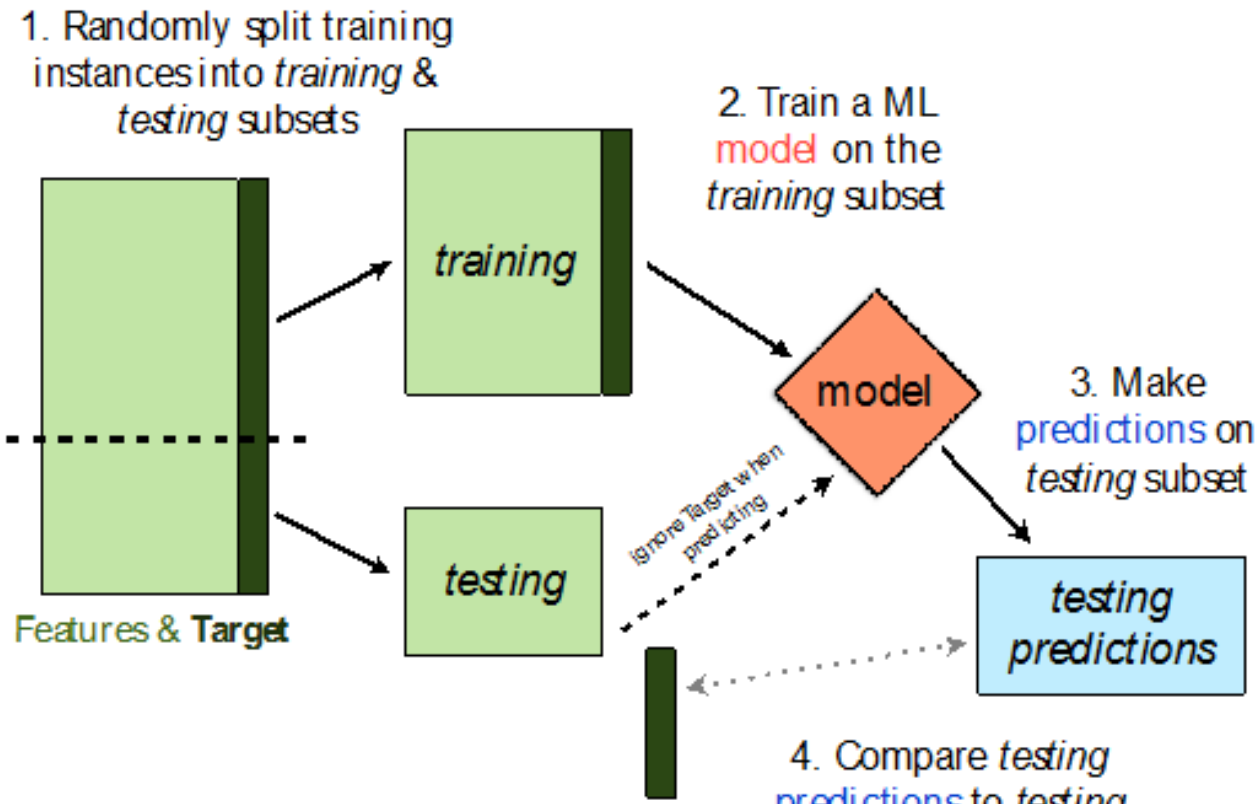


(Reference: Data Mining Classification - Gun Ho Lee)

Cross Validation: Measuring Performance

- Labeled training data is finite.
- The model built needs to be validated.
- So, training data itself is split
  - Use one part for model building: Training Data.
  - Use the other part: Testing Data. (actually called as Validation Data, but within Cross Validation process, it can be called as Testing data. Otherwise Testing data is the one which does not have output values. Validation data has output values for comparison.)

Machine Learning Technique

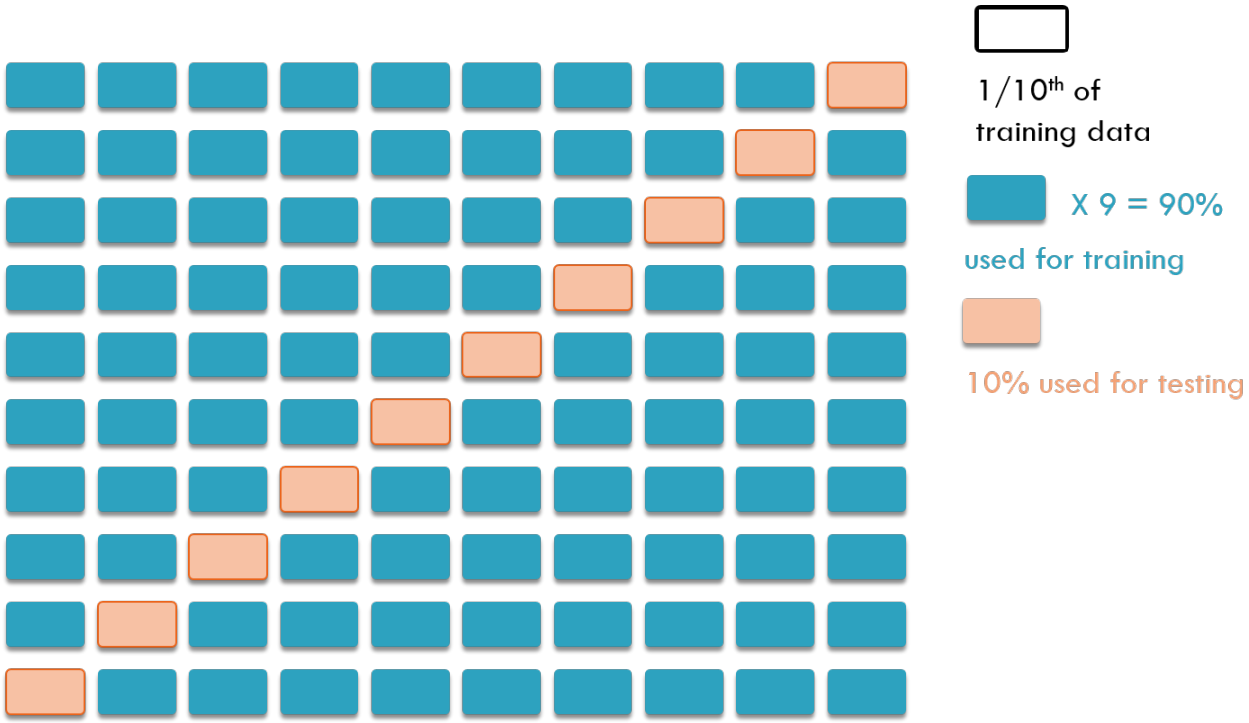


Cross Validation

How to divide dataset into Training, Validation, Testing sets?

- Problems?
  - Validation set should "represent" the Training set.
  - "Lucky split" is possible: Difficult instances were chosen for the training set, Easy instances put into the testing set
- Multiple evaluations using different portions of data for training and validating/testing
- k-fold Cross Validation

K-Fold



Cross Validation Error Estimate: $= \frac{1}{k} \sum Error_i$

Note: Here weights or all 10 models are neither averaged nor the last weights are taken as final model. [PTO]

Practical Tip

- Cross Validation is NOT used for Model Building, but choosing best model amongst choices such as Linear models, decision tree, SVM, etc.
- Once a specific model with known hyper parameters is finalized using least average error, a new model is built using FULL training data.
- Not splits are done there.

Machine Learning Evaluation

Evaluation

Given Actuals and Predictions, how to find out how good the performance was?

- Accuracy: Misclassification Rate
- Confusion Matrix
- Precision, Recall, F1

Evaluation Metrics: Accuracy

Accuracy

Model Evaluation on Test Set (Regression) - Mean Squared Error

- Mean Squared Error: measuring the "quality of fit"
- Will be small if the predicted responses are very close to the true responses $MSE = \frac{1}{n}\sum(y_i - f(x_i))^2$

Accuracy

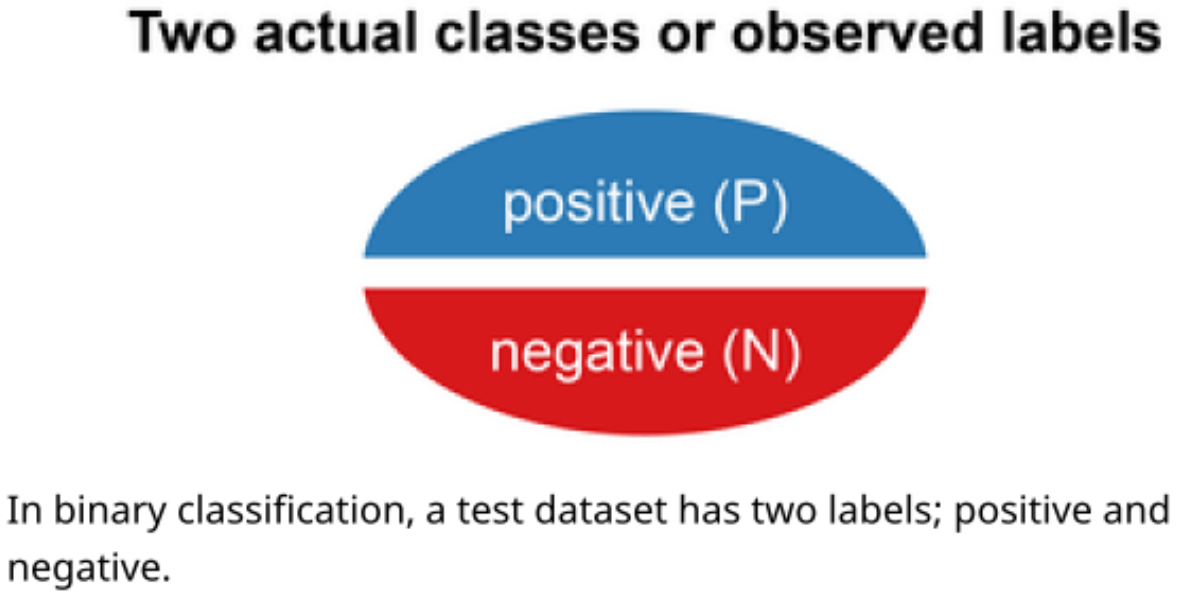
How should classifier be quantitatively evaluated?

- Misclassification Rate $= \frac{NumIncorrectPrediction}{TotalPrediction}$
- Issues with misclassification rate?
  - If Accuracy is used: Example: in CC fraud domain, there are many more legitimate transactions than fraudulent transactions
  - Then: Classifier that predicts every transaction as GOOD would have 99% accuracy! Seems great, but it's not

Evaluation Metrics: Confusion Matrix

Evaluation Metrics

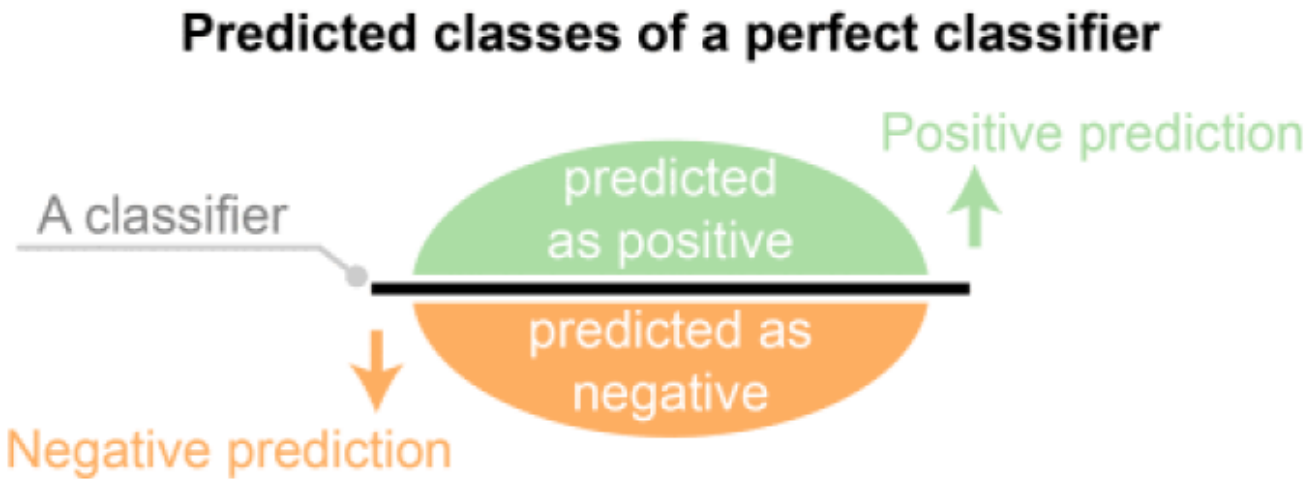
There are predicted labels (Positive / negative) as well as actual labels (Positive / negative) .



In binary classification, a test dataset has two labels; positive and negative.

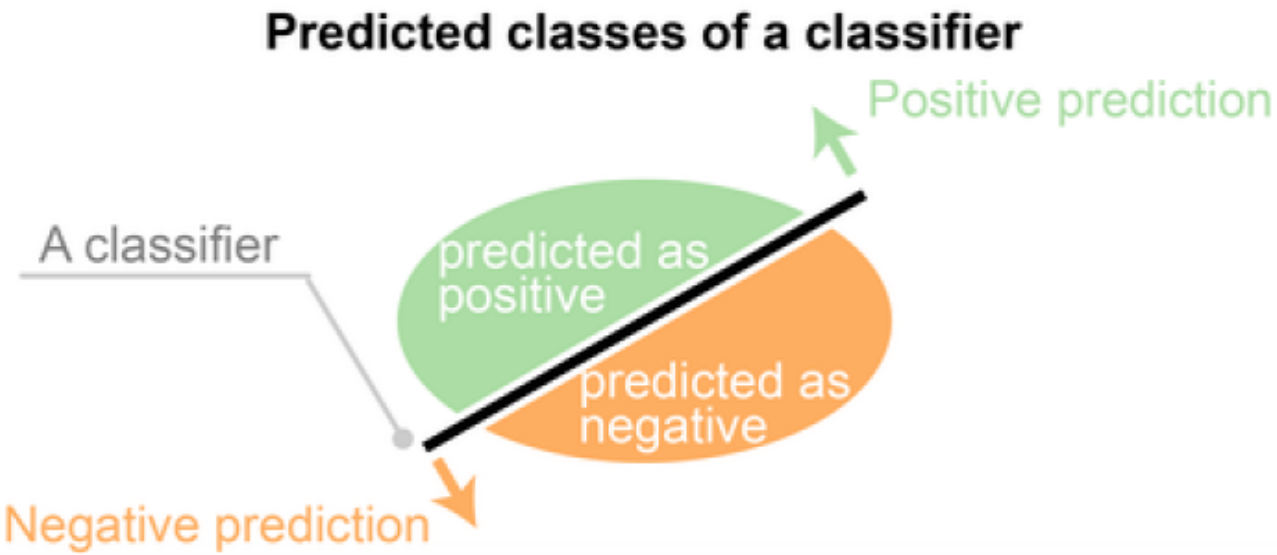
Evaluation Metrics

The predicted labels will be exactly the same if the performance of a binary classifier is perfect, but it is uncommon to be able to develop a perfect binary classifier that is practical for various conditions.



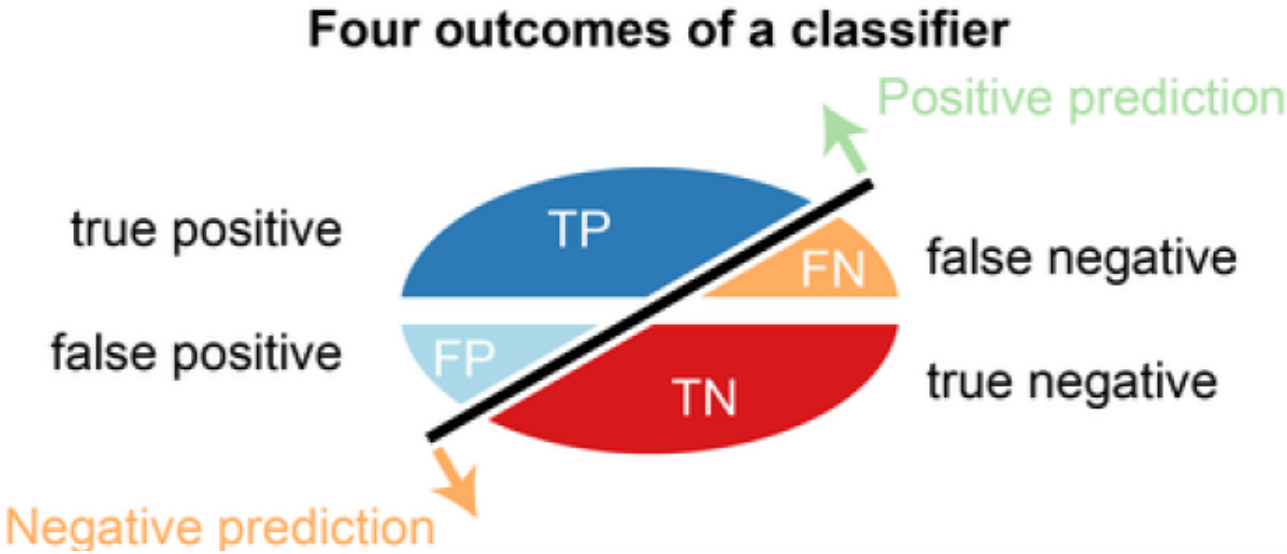
Evaluation Metrics

Hence, the predicted labels usually match with part of the observed labels



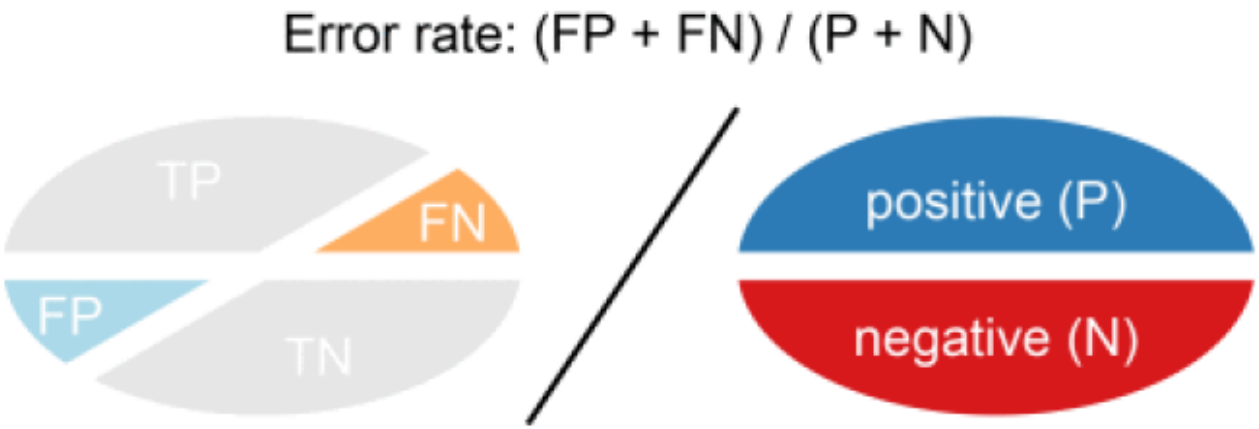
Evaluation Metrics

Classification of a test dataset produces four outcomes - true positive, false positive, true negative, and false negative.



Error rate (ERR) and accuracy (ACC) are the most common and intuitive measures derived from the confusion matrix.

Error rate (ERR)

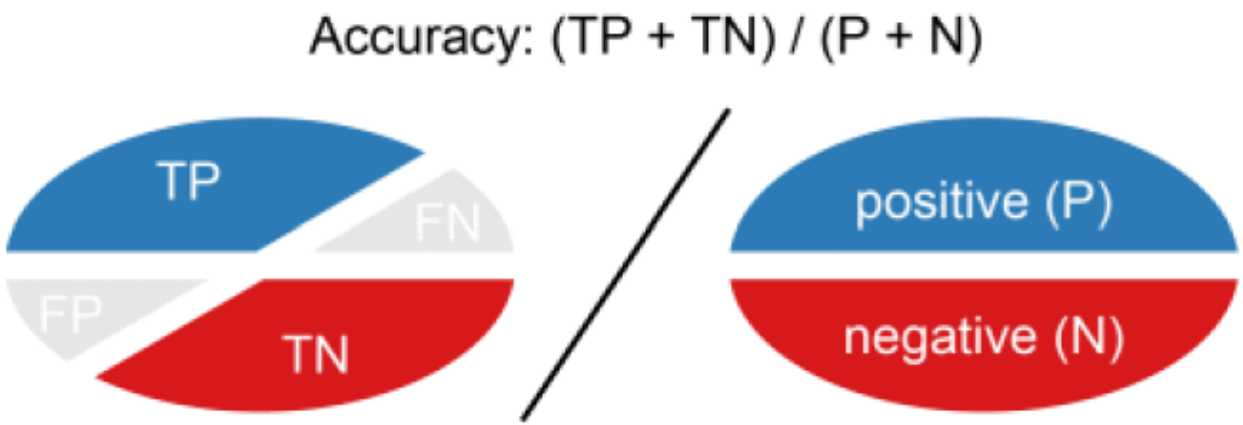


It is calculated as the number of all incorrect predictions divided by the total number of the dataset. The best error rate is 0.0, whereas the worst is 1.

$$ERR = \frac{FP+FN}{TP+TN+FN+FP} = \frac{FP+FN}{P+N}$$

Accuracy

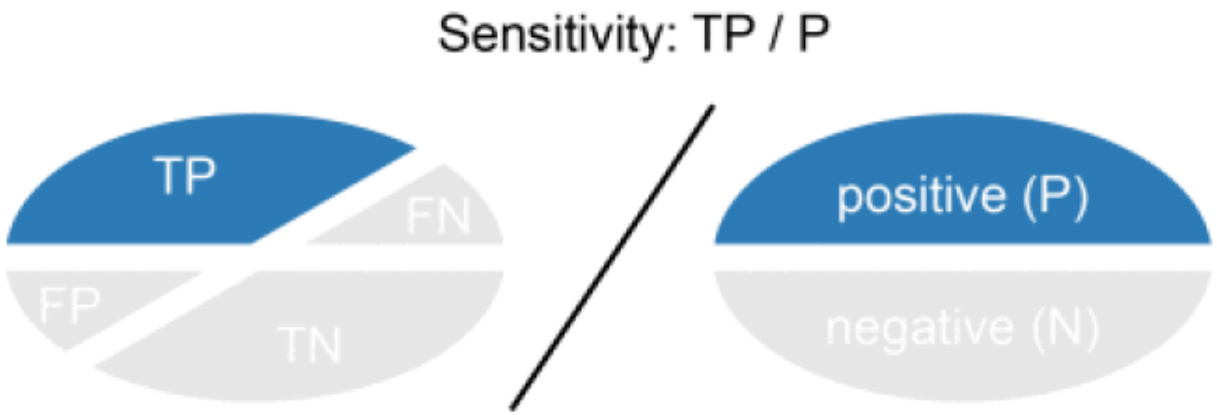
It is calculated as the number of all correct predictions divided by the total number of the dataset. The best accuracy is 1.0, whereas the worst is 0.0. It can also be calculated by $1 - ERR$.



$$Accuracy = \frac{TP+TN}{P+N}$$

Sensitivity (Recall or True positive rate)

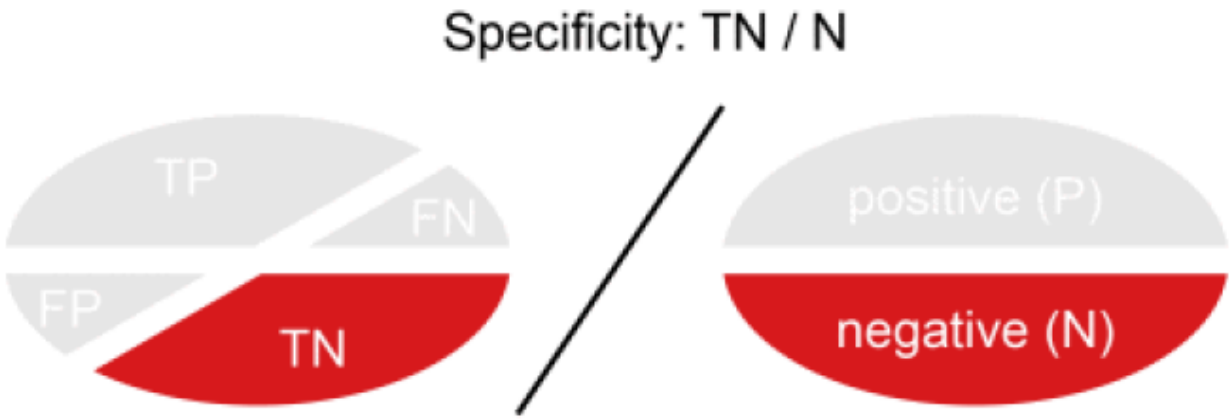
Sensitivity (SN) is calculated as the number of correct positive predictions divided by the total number of positives. It is also called recall (REC) or true positive rate (TPR). The best sensitivity is 1.0, whereas the worst is 0.0.



$$SN = \frac{TP}{TP+FN} = \frac{TP}{P}$$

Specificity (True negative rate)

Specificity (SP) is calculated as the number of correct negative predictions divided by the total number of negatives. It is also called true negative rate (TNR). The best specificity is 1.0, whereas the worst is 0.0.



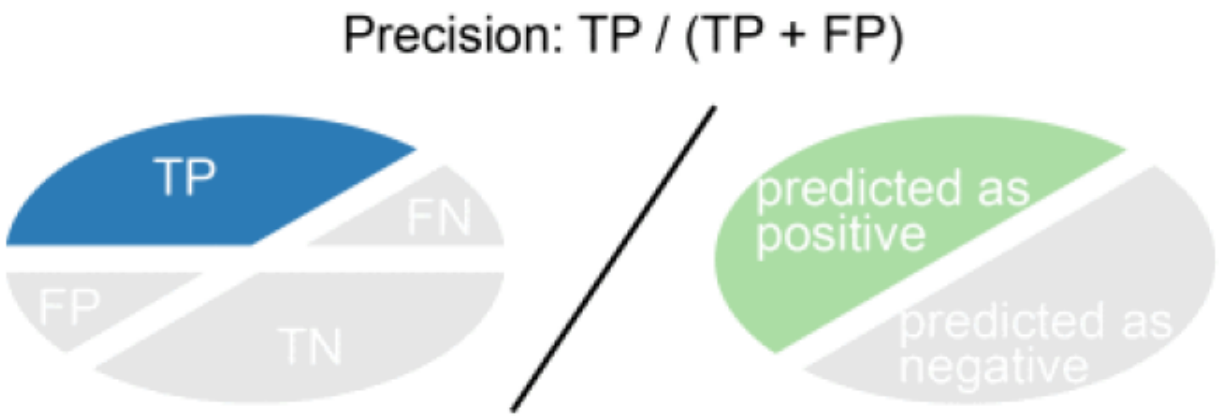
$$SP = \frac{TN}{TN+FP} = \frac{TN}{N}$$

Sensitivity vs Specificity

- For sensitivity we will look oat only *P, all that were predicted positive, rest are all in the negatives group.
- For highly sensitive test, 100% sensitive, within the Negatives group, there is no mistake.
- Meaning anyone who has tested negative, they surely don't have disease.
- All are truly negative. Meaning FN are 0. Recall thus can be made 1 by marking all positive. So there is no one in the Negatives group. So no FN. But thats not good.
- But still its not a perfect test, because within Positives, you may have some FPs.
- Similarly 100% Specific test: We will send all negative tested guys to a different group. Within them we had FN guys as well. The Positives were perfect. All of them had disease. $FP = 0$. So those who tested positive surely had the disease.
- In reality there is no perfect test!!

Precision (Positive predictive value)

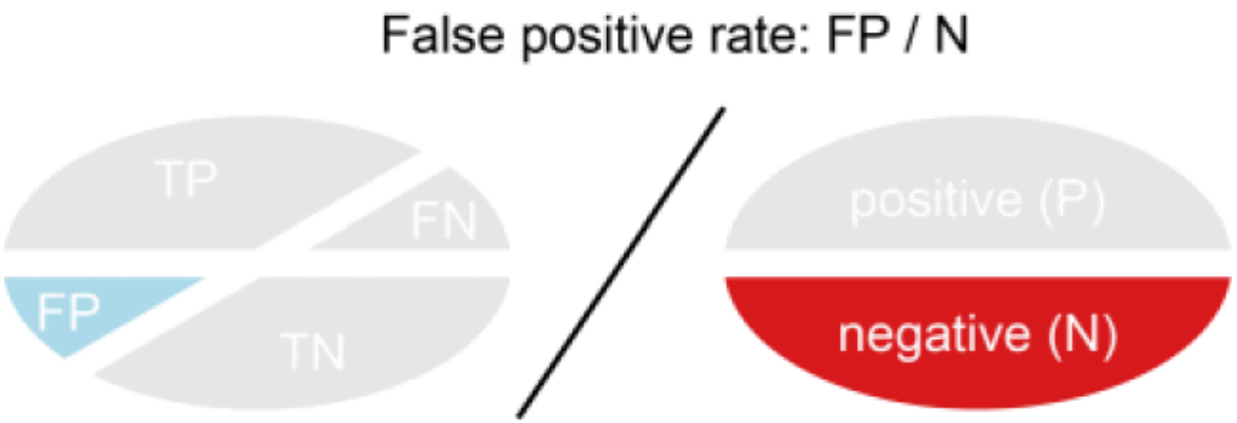
Precision (PREC) is calculated as the number of correct positive predictions divided by the total number of positive predictions. It is also called positive predictive value (PPV). The best precision is 1.0, whereas the worst is 0.0.



$$PREC = \frac{TP}{TP+FP}$$

False positive rate

False positive rate (FPR) is calculated as the number of incorrect positive predictions divided by the total number of negatives. The best false positive rate is 0.0 whereas the worst is 1.0. It can also be calculated as 1 - specificity.



$$FPR = \frac{FP}{TN+FP} = 1 - SP$$

Example

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**

Type I Error

Type I Error is equivalent to False Positive (FP).

- Test is Positive but actually diesese is not there.
- A Fire alarm going off when in fact there is no fire.

This kind of error is synonymous to "believing a lie" or "a false alarm".

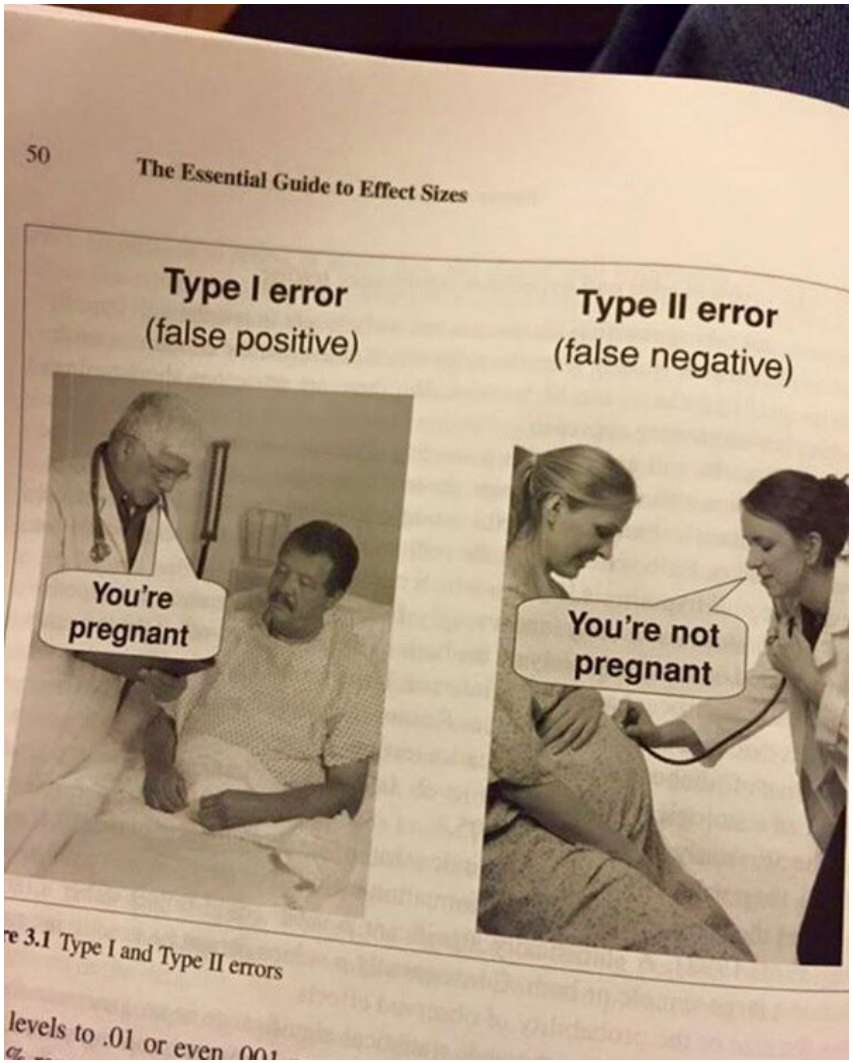
Type II Error

Type II Error is equivalent to False Negative (FN).

- Test is Negative but actually diesese is there..
- A Fire breaking out and the fire alarm does not ring.

This kind of error is synonymous to "failing to believe a truth" or "a miss".

Errors


50
The Essential Guide to Effect Sizes
Type I error
(false positive)
Type II error
(false negative)
You're pregnant
You're not pregnant
re 3.1 Type I and Type II errors
levels to .01 or even .001

Evaluation Metrics: Precision and Recall

Precision and Recall

- Precision: fraction of records that are positive in the set of records that classifier predicted as positive
- Interpretation: If the prediction is very stringent then getting all of them correct is likely, so more precise. Less (no) mis-predictions (FP) happens.
- Recall: fraction of positive records that are correctly predicted by classifier, out of all actual.
- Interpretation: Criteria is relaxed so that maximum catch is made. Less (no) mis-classification (FN) happens.

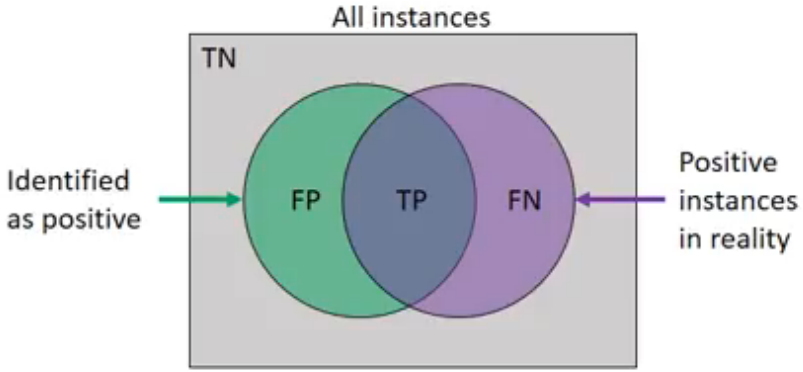
Precision and Recall

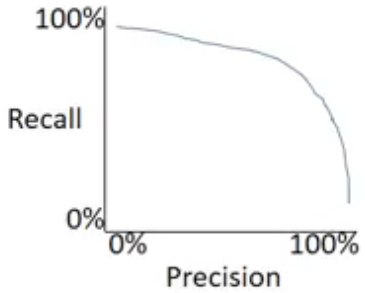
$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{F-measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

High precision: relevant results >> irrelevant results
High recall: we got most of the relevant results
F-measure: harmonic mean of precision and recall



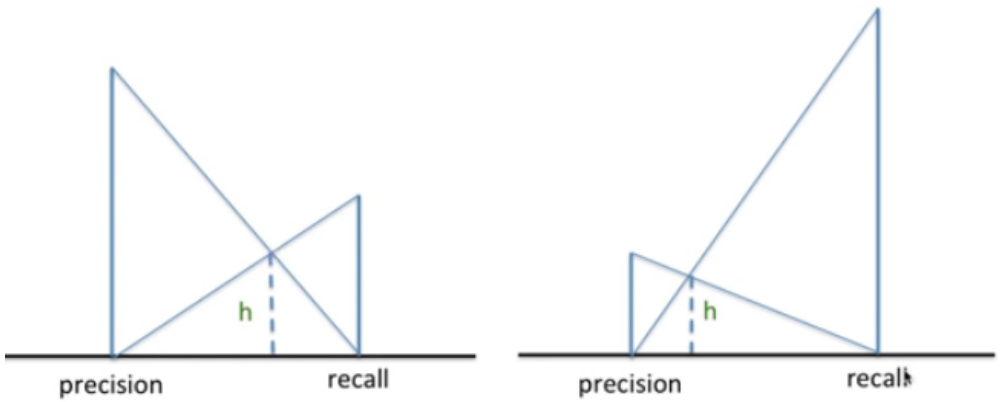


1: (Powers, 2007)

(Reference: Text Mining - Jeff Shaul)

F1 Measure

- F-measure: combines precision and recall into a single metric, using the harmonic mean
- Harmonic Mean of two numbers tends to be closer to the smaller of two numbers, so the only way F1 is high is for both precision and recall to be high. $F_1 = \frac{2rp}{(r+p)} = \frac{2 \times TP}{(2 \times TP + FP + FN)}$

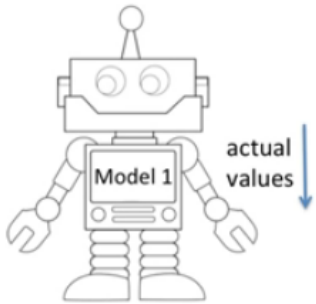


h is half the harmonic mean

Even if anyone is very high, we get balanced F1 score
(Ref: Performance measure on multiclass classification - Minsuk Heo)

F1 Measure

- Model 1's F1 is better but Accuracy is lower than Model 2. Which is better?
- F1 is better as it takes care of any imbalance in the targets.

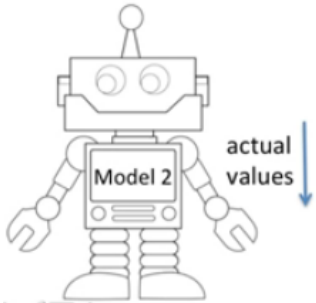


predictions →

actual values ↓

	A	B	C	D
A	100	80	10	10
B	0	9	0	1
C	0	1	8	1
D	0	1	0	9

F1 Score = **0.601** accuracy = 0.547



predictions →

actual values ↓

	A	B	C	D
A	198	2	0	0
B	7	1	0	2
C	0	8	1	1
D	2	3	4	1

F1 Score = **0.342** accuracy = 0.87

(Ref: Performance measure on multiclass classification - Minsuk Heo)

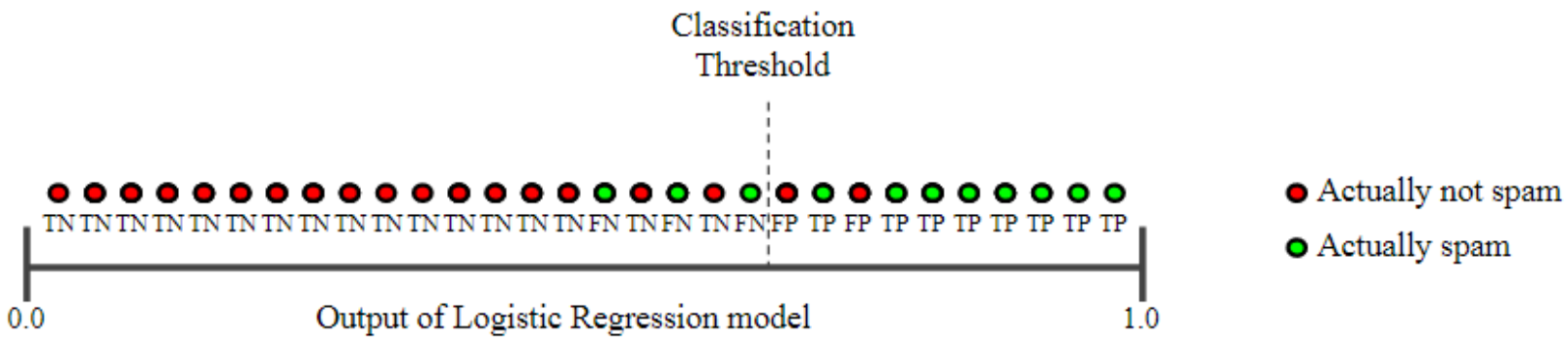
Precision and Recall: Tug of War

Precision and Recall: A Tug of War

- Recall and Precision seem to be opposing each other.
- Just knowing one value is not enough, but both
- And we need to do better at both.

Precision and Recall: A Tug of War

- Explore this notion by looking at the following figure, which shows 30 predictions made by an email classification model.
- Those to the right of the classification threshold are classified as "Spam", while those to the left are classified as "not Spam."



Precision and Recall trade-off

- Typically to increase precision for a given model implies lowering recall, though this depends on the precision-recall curve of your model.
- Generally, if you want higher precision you need to restrict the positive predictions to those with highest certainty in your model, which means predicting fewer positives overall (which, in turn, usually results in lower recall).
- If you want to maintain the same level of recall while improving precision, you will need a better classifier.

Precision and Recall trade-off : Summary

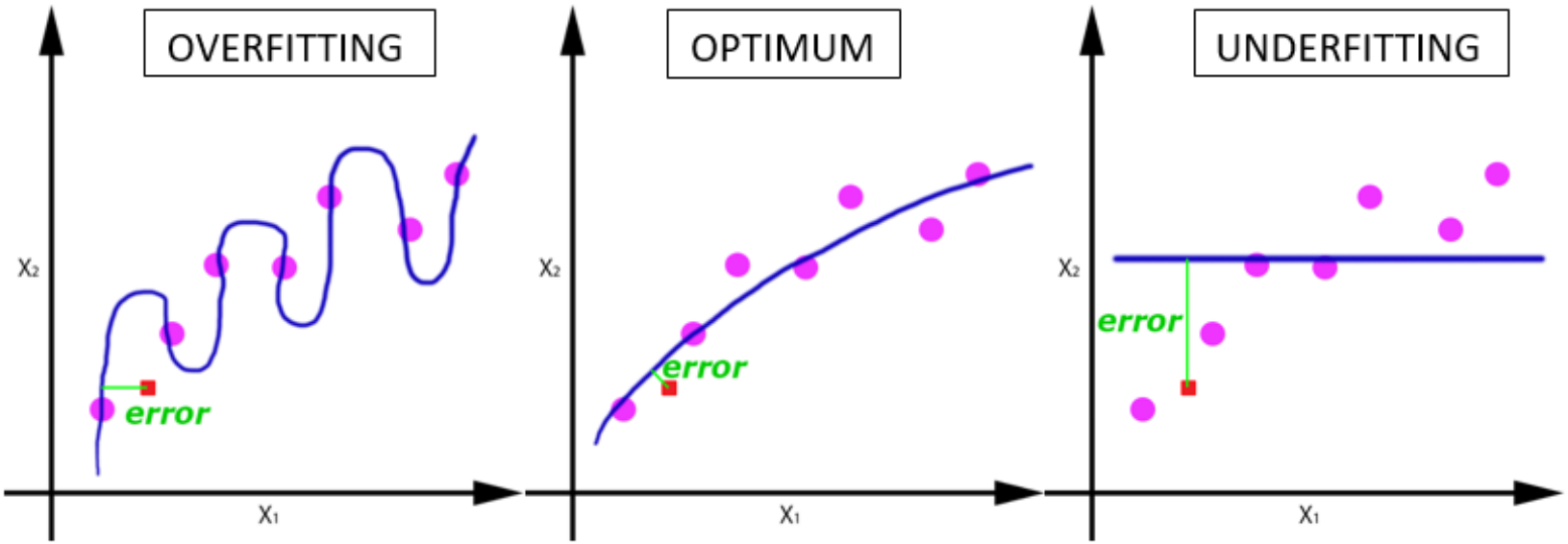
- Intuitively, if you cast a wider net (meaning your classifier threshold is relaxed), you will detect more relevant documents/positive cases (higher recall)
- But you will also get more false alarms (lower precision).
- If you classify everything in the positive category, you have 100% recall (no FN here), a bad precision because some of them could be wrong, so there will be many FPs, making precision lower.
- If you adjust threshold strict so as to have good precision, you will have lower making of positives and mark many actually positive values as negative, falsely. Making more FNs, thus lower recall.

Machine Learning Model Generalization

Generalization in Machine Learning

- Generalization refers to how well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning.
- The goal of a good machine learning model is to generalize well from the training data to any data from the problem domain.
- This allows us to make predictions in the future on data the model has never seen
- In statistics, a fit refers to how well you approximate a target function.

Under-fitting vs Over-fitting



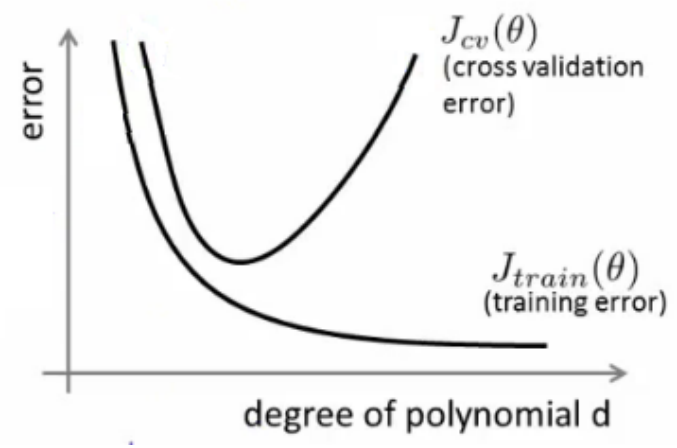
(Reference: What is Machine Learning? - An Introduction- itdadao)

Over-fitting

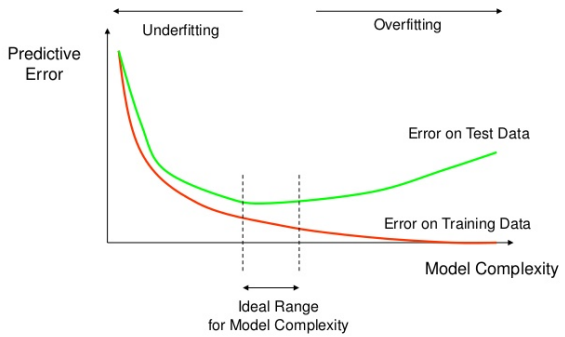
- Over-fitting: occurs when model "memorizes" training data
- Model does not generalize to the overall problem
- This is bad! We wish to avoid over-fitting
- Techniques: Regularization (Penalty) and Drop Out (Neural Network)

Detection

Roughly speaking, over-fitting typically occurs when the ratio $\frac{ComplexityOfTheModel}{TrainingSize}$ is too high.



How Overfitting affects Prediction



- If your data is in two dimensions, you have 10000 points in the training set

Solution

- In many texts, "iterations" is used on X axis, which is easy to imagine in case of Neural Networks, where each epoch refines model by fitting better question/weights.
- In case of Machine Learning, there no iterations. How to detect Over-fitting in case of Polynomial Regression?
- Initially when, polynomial equation to start with is simple, it has not fit the data yet, so its under fitting phase. Losses are HIGH for both, training as well as testing (validation)
- As more levels get added in tree or more degrees in polynomial regression, under-fitting reduces and losses come down.
- At one point, training loss still reduces but validation los starts jumping up. Thats over-fitting. Model is fitting training data TOO tightly and is OFF the validation data.

(Ref:Why Is Over-fitting Bad in Machine Learning? - StackOverflow)

Regularization

- In regularization, apart from cost function, weighted sum of parameters and features is added
- For the features which need to be suppressed, their corresponding coefficient is made large. So their cost component increases
- As the Cost needs to be minimized, to reduce these weighted sum, corresponding feature values are reduced,
- So their importances gets lowered dramatically.

Regularization

- Instead of simply aiming to minimize loss (empirical risk minimization): $minimize(Loss(Data|Model))$
- We'll now minimize loss+complexity, which is called structural risk minimization: $minimize(Loss(Data|Model) + complexity(Model))$

Regularization

- Complexity of model can be represented by many ways, here, Model complexity is taken as a function of the weights of all the features in the model.
- Then, a feature weight with a high absolute value is more complex than a feature weight with a low absolute value.

Under-fitting

- Under-fitting: occurs when model cannot fit training data well
- Model does generalize to the overall problem but not that well
- This is also bad! We wish to avoid under-fitting
- Techniques: more features

Bias and Variance

Evaluation Metrics: Prediction Bias

Types of Errors

The prediction error can be broken down into:

- Bias Error
- Variance Error

$$\text{error}(X) = \text{noise}(X) + \text{bias}(X) + \text{variance}(X)$$

Bias

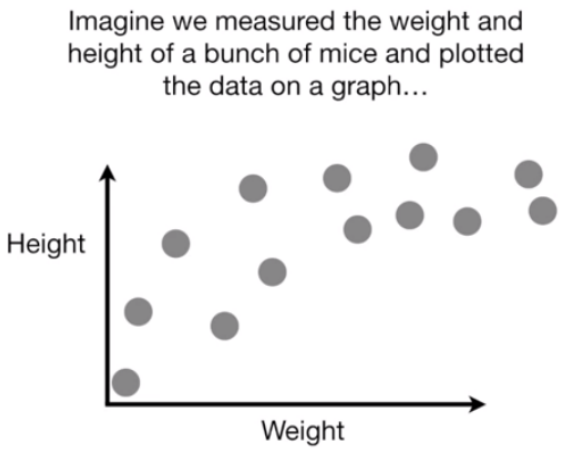
- Bias is the simplifying assumption made by a model to make the target function easier to learn.
- A high bias makes it fast to learn and easier to understand but is generally less flexible. In turn, it has lower predictive performance on complex problems.
- Low Bias: Suggests less assumptions about the form of the target function.
- High-Bias: Suggests more assumptions about the form of the target function.

Variance

- Variance is the amount that the estimate of the target function will change if different training data was used.
- Ideally, it should not change too much from one training dataset to the next, meaning that the algorithm is good at picking out the hidden underlying mapping between the inputs and the output variables.
- Low Variance: Suggests small changes to the estimate of the target function with changes to the training dataset.
- High Variance: Suggests large changes to the estimate of the target function with changes to the training dataset.

Intuition: Example

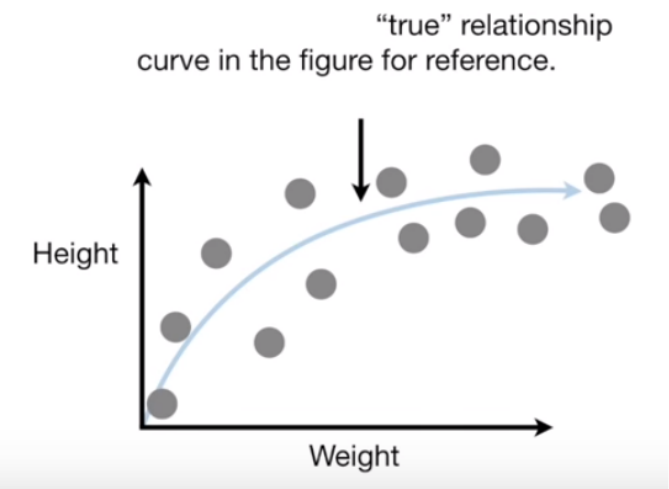
- We have collected past observations of Mice's Heights and Weights.
- Clearly,the relationship is sort of curved.
- Wish to predict Height given Weight.
- Different algorithms will have different underlying structures to fit this regression data.



(Ref: Machine Learning Fundamentals: Bias and Variance -StatQuest with Josh Starmer)

Intuition: True Relationship

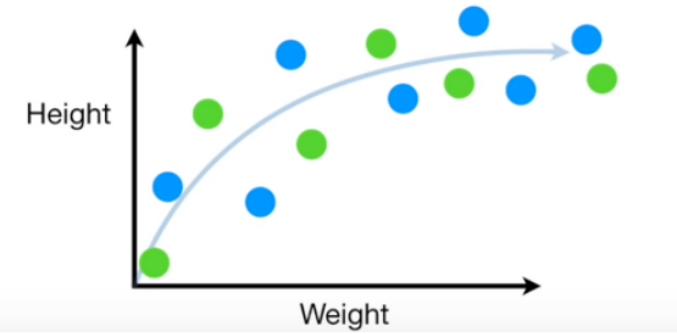
- We have collected past observations of Mice's Heights and Weights.
- Clearly,the relationship is sort of curved.
- Different algorithms will have different underlying structures to fit this regression data.



(Ref: Machine Learning Fundamentals: Bias and Variance -StatQuest with Josh Starmer)

Intuition: Machine Learning

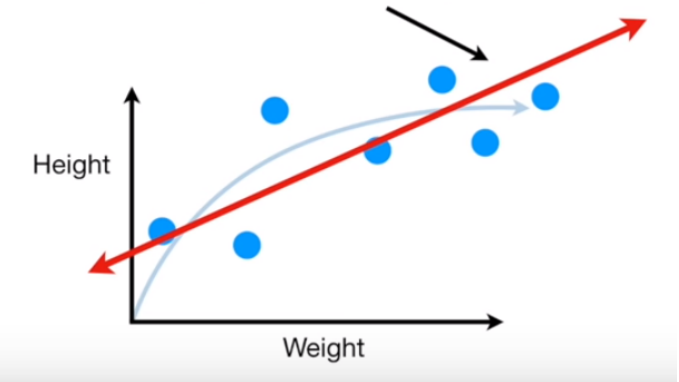
- Split data into training and testing (validation)
- Blue for training
- Green for testing



(Ref: Machine Learning Fundamentals: Bias and Variance -StatQuest with Josh Starmer)

Intuition: Linear Regression

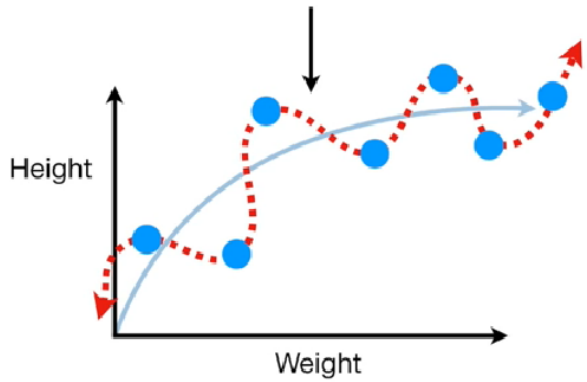
- Fitting Linear Regression model on the training data (blue dots)
- It fits a line, whatever the underlying data is.
- But it finds best line that minimizes Least Square error.
- Anyway, it can not bend, so said to have HIGH bias. It is not very accurate.
- Inability of Machine Learning model to capture the TRUE relationship is called as **Bias**.



(Ref: Machine Learning Fundamentals: Bias and Variance -StatQuest with Josh Starmer)

Intuition: Non-Linear (Tree) Regression

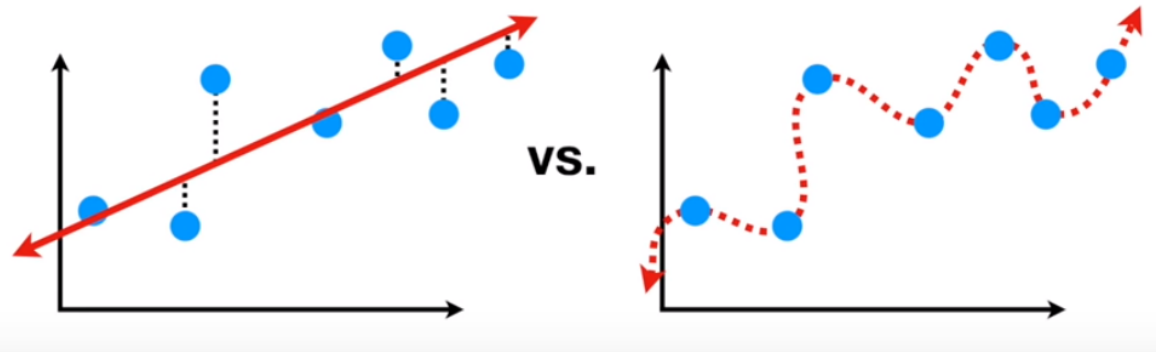
- Some other Machine Learning model can give high degree and flexible model.
- As it passes through points, its very accurate (as there is just no error)
- Being flexible, it has ability to represent true relationship, so very little bias.



(Ref: Machine Learning Fundamentals: Bias and Variance -StatQuest with Josh Starmer)

Intuition: Training set Comparison

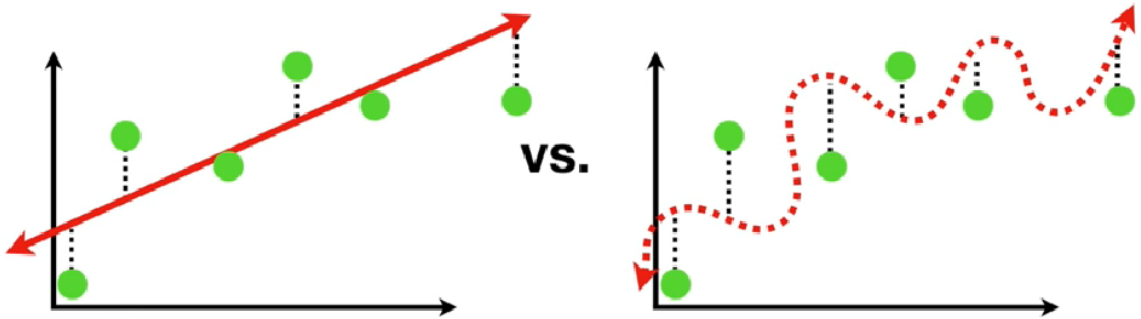
- Linear model has error on Training set. High Bias.
- Non-linear model has almost no error on Training set. Low Bias.



(Ref: Machine Learning Fundamentals: Bias and Variance -StatQuest with Josh Starmer)

Intuition: Testing set Comparison

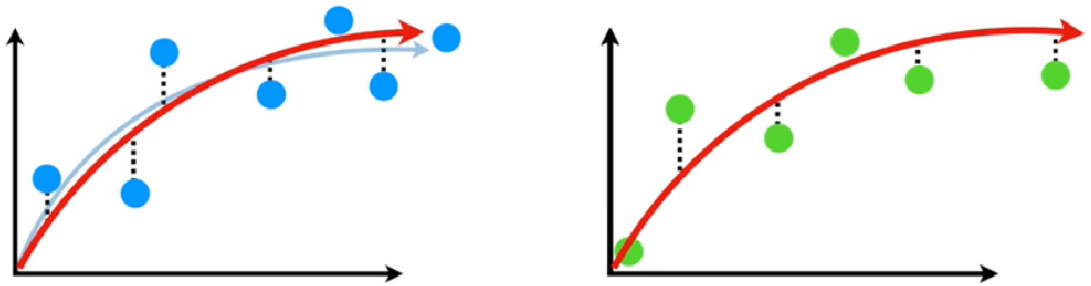
- When same model is applied on the testing set, how does it fair?
- Which model looks better? Linear or Non-Linear?
- Linear is not accurate, but the Non-linear is far too bad.
- This is called as **Variance**.
- Difference in fits on different datasets is the Variance.
- Linear model has less Variance and Non Linear model has High Variance.



(Ref: Machine Learning Fundamentals: Bias and Variance -StatQuest with Josh Starmer)

Intuition: Ideal

- Linear model gives good predictions but not great predictions, but its consistent.
- Non linear model gives great predictions but not consistent many a times.
- Ideal model captures underlying true relationship. So, low Bias and low Variance.
- It is done by finding a Sweet spot between simple model and complex model.
- 3 such methods of finding sweet spot are : regularization, bagging and boosting.



(Ref: Machine Learning Fundamentals: Bias and Variance -StatQuest with Josh Starmer)

Bias and Variance (Recap)

- Bias: the error introduced by modeling for complex situation by a much simpler model
- **The more flexible (complex) a method is, the less bias it will generally have.**
- Variance: how much the learned model will change if the training set was different
- Does changing a few observations in the training set, dramatically affect the model?
- **Generally, the more flexible (complex) a method is, the more variance it has.**

Learning Method Bias

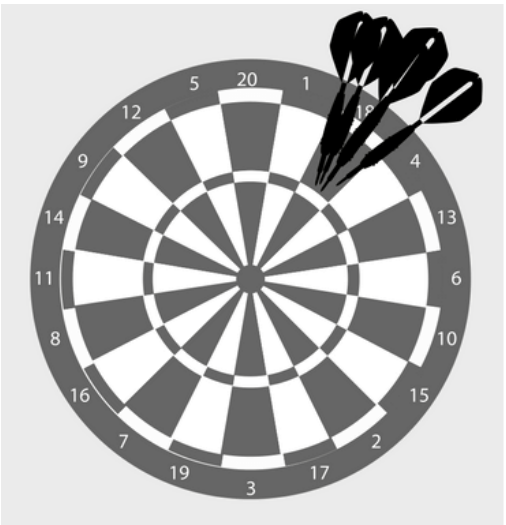
- Bias: the error introduced by modeling a real-life problem (usually extremely complicated) by a much simpler model.
- High bias is a strong view of modeling with simplistic solution.
  - Example: linear regression assumes a linear relationship between the target variable Y and the predictor variables X
  - It's unlikely that the relationship is exactly linear, so some bias will be present in the model
- The more flexible (complex) a method is, the less bias it will generally have.

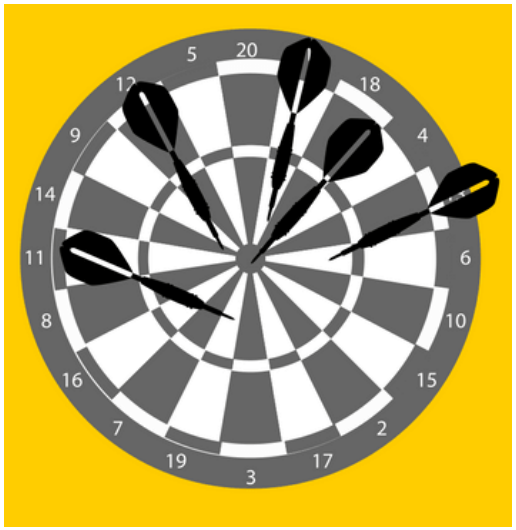
Learning Method Variance

- Variance: how much the learned model will change if the training set was different
  - Does changing a few observations in the training set, dramatically affect the model?
  - Ideally, answer is no.
- Generally, the more flexible (complex) a method is, the more variance it has.

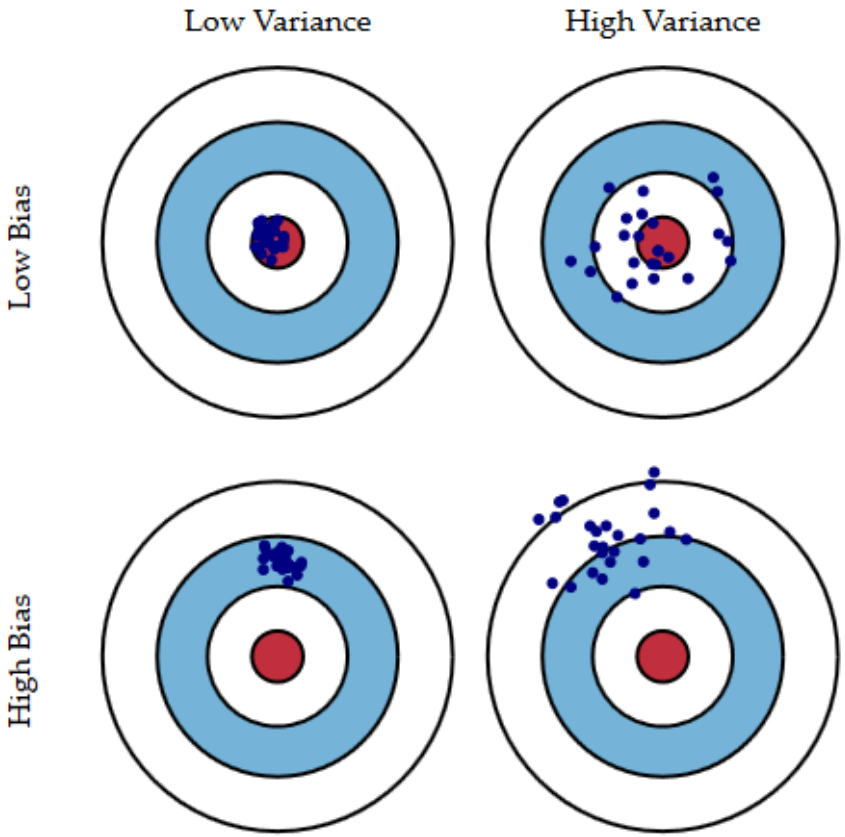
Example

- Imagine you have 5 training datasets of the same problem
- Imagine using an machine learning algo on these sets, so we have 5 models
- High bias algo gives off the mark result
- Low variance gives converged results
- Low bias algo gives towards the mark result
- High variance gives spread results





Bias vs Variations



http://scott.fortmann-roe.com/docs/BiasVariance.html http://scott.fortmann-roe.com/docs/BiasVariance.html

On a Lighter Note (Don't start imagining the individuals)

A person with high bias is someone who starts to answer before you can even finish asking. A person with high variance is someone who can think of all sorts of crazy answers. Combining these gives you different personalities:

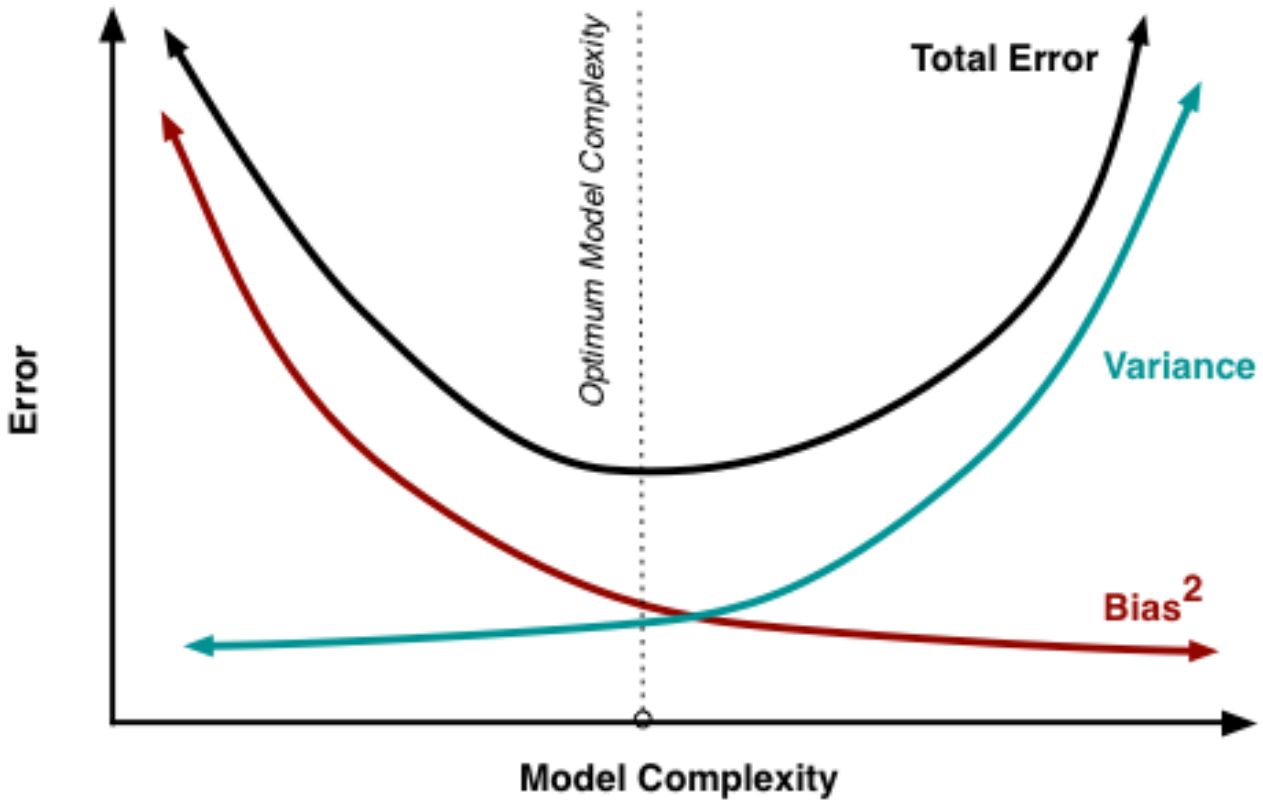
- High bias/low variance: this is someone who usually gives you the same answer (low variance), no matter what you ask, and is usually wrong about it (high bias)
- High bias/high variance: someone who takes wild guesses (high variance), all of which are sort of wrong; (high bias)
- Low bias/high variance: a person who listens to you and tries to answer the best they can (low bias, looks at lots of data), but that daydreams a lot and may say something totally crazy; (high variance)
- Low bias/low variance: a person who listens to you very carefully and gives you good answers pretty much all the time.

Bias Variance Trade-Off

What is the trade-off?

- Generally, as the model becomes more computational (e.g. when the number of free parameters grows), the variance (dispersion) of the estimate also increases, but bias decreases.
- Due to the fact that the training set is memorized completely instead of generalizing, small changes lead to unexpected results (over-fitting).
- On the other side, if the model is too weak, it will not be able to learn the pattern, resulting in learning something different that is offset with respect to the right solution.

What is the trade-off?



But Why is there a trade-off?

- Low bias algos tend to be more complex and flexible e.g. Decision Tree (non-linear) so that it can fit the data well.
- But if the algo is too flexible, it will fit each training data set differently, and hence have high variance.
- A key characteristic of many supervised learning methods is a built-in way to control the bias-variance trade-off either automatically or by providing a special parameter that the data scientist can adjust.

Summary

- Statistical learning reveals hidden data relationships.
- Relationships between dependent and independent data.

Summary

- Model is the transformation engine.
- Parameters are the ingredients that enable the transformation.
- A model uses the training data to learn.
- A model uses the testing data to evaluate.

Summary

- Bias-variance trade-off is a balancing act.
- Balance to find optimal model.
- Balance to find the sweet spot.

Finally

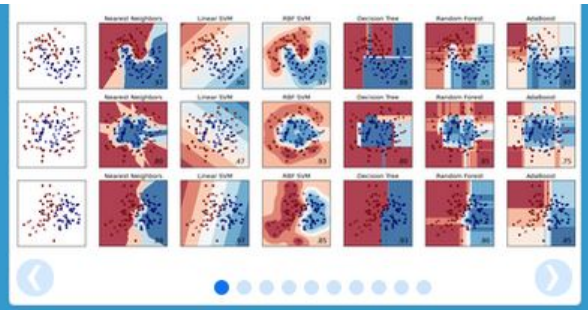
All models are wrong; but some are useful - George E P Box

Introduction to Scikit-Learn aka Sklearn

scikit.learn

- Open source machine learning library for Python.
- Features various classification, regression and clustering algorithms

Machine Learning in Python



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: *SVM, nearest neighbors, random forest, ...* — *Examples*

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: *SVR, ridge regression, Lasso, ...* — *Examples*

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: *k-Means, spectral clustering, mean-shift, ...* — *Examples*

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: *PCA, feature selection, non-negative matrix factorization.* — *Examples*

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: *grid search, cross validation, metrics.* — *Examples*

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: *preprocessing, feature extraction.* — *Examples*

Sklearn Paradigm

Scikit-learn provides an object-oriented interface centered around the concept of an Estimator.

Scikit-Learn's Estimator API

The steps in using the Scikit-Learn estimator API are as follows

- Choose a class of model by importing the appropriate estimator class from Scikit-Learn.
- Choose model hyper-parameters by instantiating this class.
- Arrange data into a features matrix and target vector.
- Fit the model to your data by calling the fit() method of the model instance.
  - For supervised learning, predict labels for unknown data using the predict().
  - For unsupervised learning, transform or infer properties of the data using transform() or predict().

- The `Estimator.fit` method sets the state of the estimator based on the training data.
- Usually, the data is comprised of a two-dimensional numpy array X of shape `(n_samples, n_predictors)` that holds the so-called feature matrix and a one-dimensional numpy array y that holds the responses.
- Some estimators allow the user to control the fitting behavior.

Estimator

- fit(X,y) sets the state of the estimator.
- X is usually a 2D numpy array of shape (num_samples, num_features).
- y is a 1D array with shape (n_samples,)
- predict(X) returns the class or value

```
class Estimator(object):
    def fit(self, X, y=None):
        """Fits estimator to data. """
        # set state of ''self''
        return self
    def predict(self, X):
        """Predict response of ''X''. """
        # compute predictions ''pred''
        return pred
```

Sklearn: Installations

Scikit Learn: Installation and setup

Install Anaconda distribution having Python 3.5

```
$ conda install numpy scipy matplotlib scikit-learn ipython-notebook
```

Checking your installation

```
import numpy
print('numpy:', numpy.__version__)

import scipy
print('scipy:', scipy.__version__)

import matplotlib
print('matplotlib:', matplotlib.__version__)

import sklearn
print('scikit-learn:', sklearn.__version__)

import seaborn
print('seaborn', seaborn.__version__)
```

Data Preprocessing with Scikit-Learn

Data Preprocessing for Pima-Indians Diabetis Dataset

(Ref: How To Prepare Your Data For Machine Learning in Python with Scikit-Learn - Jason Brownlee)

Read Data

Import Pima Inidans dataset and split into Features (X) and target (Y)

```
import pandas
import scipy
import numpy

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/
     pima-indians-diabetes.data.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]
```

Rescale Data

- Referred to as normalization and 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.

Rescale Data

Rescale your data using scikit-learn using the MinMaxScaler class.

```
# Rescale data (between 0 and 1)
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))
rescaledX = scaler.fit_transform(X)

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

Rescale Data

After rescaling you can see that all of the values are in the range between 0 and 1.

```
[[ 0.353  0.744  0.59   0.354  0.     0.501  0.234  0.483]
 [ 0.059  0.427  0.541  0.293  0.     0.396  0.117  0.167]
 [ 0.471  0.92   0.525  0.     0.     0.347  0.254  0.183]
 [ 0.059  0.447  0.541  0.232  0.111  0.419  0.038  0.   ]
 [ 0.     0.688  0.328  0.354  0.199  0.642  0.944  0.2  ]]
```

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.

Standardize Data

Standardize data using scikit-learn with the StandardScaler class.

```
# Standardize data (0 mean, 1 stdev)
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler().fit(X)
rescaledX = scaler.transform(X)

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

Standardize Data

The values for each attribute now have a mean value of 0 and a standard deviation of 1.

```
[[ 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 ]]
```

Normalize Data

- 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.

Normalize Data

Normalize data in Python with scikit-learn using the Normalizer class.

```
# Normalize data (length of 1)
from sklearn.preprocessing import Normalizer

scaler = Normalizer().fit(X)
normalizedX = scaler.transform(X)
# summarize transformed data
numpy.set_printoptions(precision=3)
print(normalizedX[0:5,:])
```

Normalize Data

The rows are normalized to length 1.

```
[[ 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 ]]
```

Binarize Data (Make Binary)

- 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.

Normalize Data

Create new binary attributes in Python using scikit-learn with the Binarizer class.

```
# binarization
from sklearn.preprocessing import Binarizer

binarizer = Binarizer(threshold=0.0).fit(X)
binaryX = binarizer.transform(X)
# summarize transformed data
numpy.set_printoptions(precision=3)
print(binaryX[0:5,:])
```

Normalize Data

You can see that all values equal or less than 0 are marked 0 and all of those above 0 are marked 1.

```
[[ 1.  1.  1.  1.  0.  1.  1.  1.]
 [ 1.  1.  1.  1.  0.  1.  1.  1.]
 [ 1.  1.  1.  0.  0.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.  1.  1.  1.]
 [ 0.  1.  1.  1.  1.  1.  1.  1.]]
```

Thanks . . .

- Feel free to follow me at:
  - Github (github.com/yogeshhk) for open-sourced Data Science training material, etc.
  - Kaggle (www.kaggle.com/yogeshkulkarni) for Data Science datasets and notebooks.
  - Medium (yogeshharibhaukulkarni.medium.com) and also my Publications:
    - Desi Stack https://medium.com/desi-stack
    - TL;DR,W,L https://medium.com/tl-dr-w-l
- Office Hours: Saturdays, 2 to 5pm (IST); Free-Open to all; email for appointment.
- Email: yogeshkulkarni at yahoo dot com