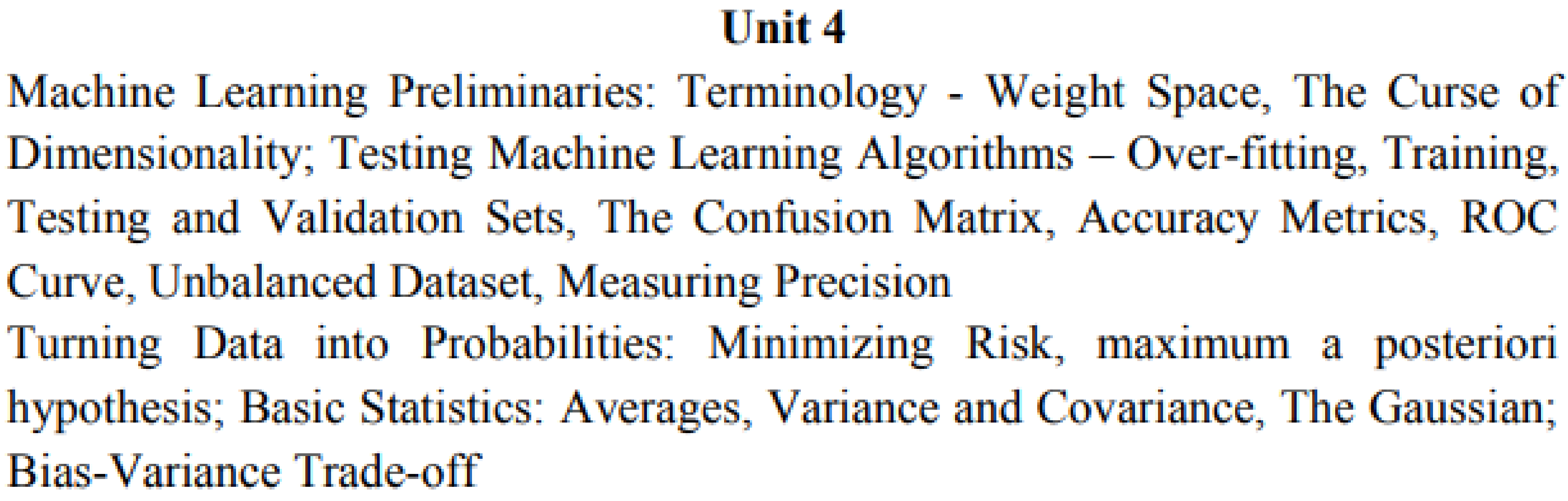
Machine Learning Preliminaries (Unit – 4)

Syllabus



Unit 4

Machine Learning Preliminaries: Terminology - Weight Space, The Curse of Dimensionality; Testing Machine Learning Algorithms – Over-fitting, Training, Testing and Validation Sets, The Confusion Matrix, Accuracy Metrics, ROC Curve, Unbalanced Dataset, Measuring Precision

Turning Data into Probabilities: Minimizing Risk, maximum a posteriori hypothesis; Basic Statistics: Averages, Variance and Covariance, The Gaussian; Bias-Variance Trade-off

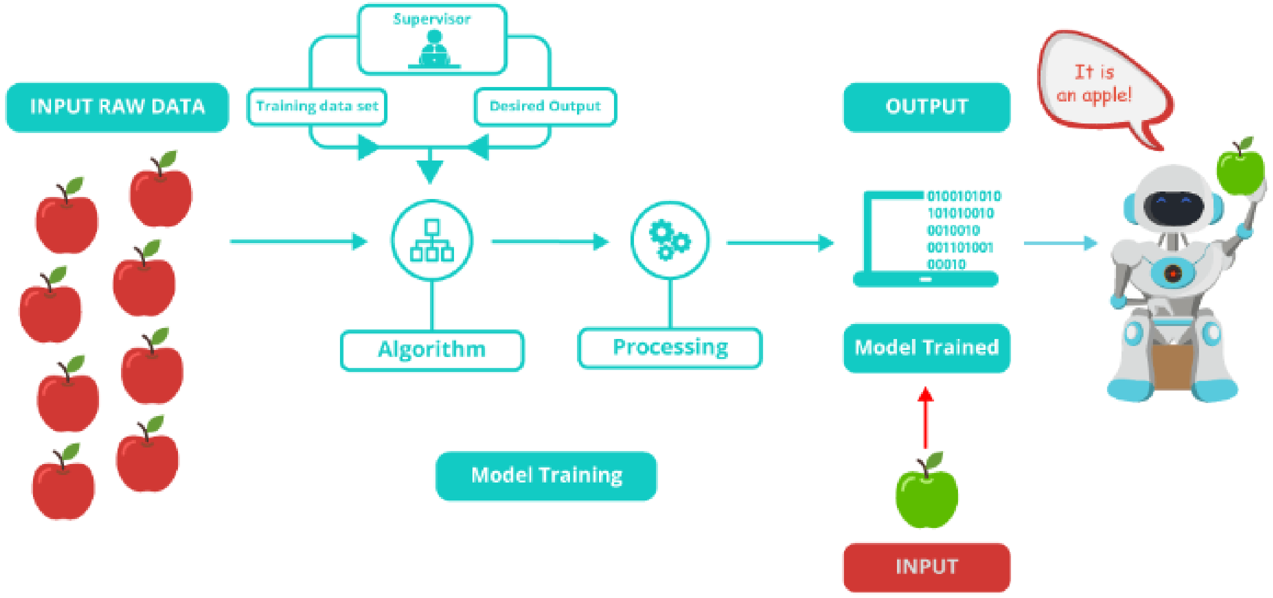
Introduction

- **Topics covered in this Unit**
  - To present important concepts of Machine Learning.
  - Basic ideas of data processing and statistics in Machine Learning
  - Bias and Variance

Types of Machine Learning

1. Supervised Learning

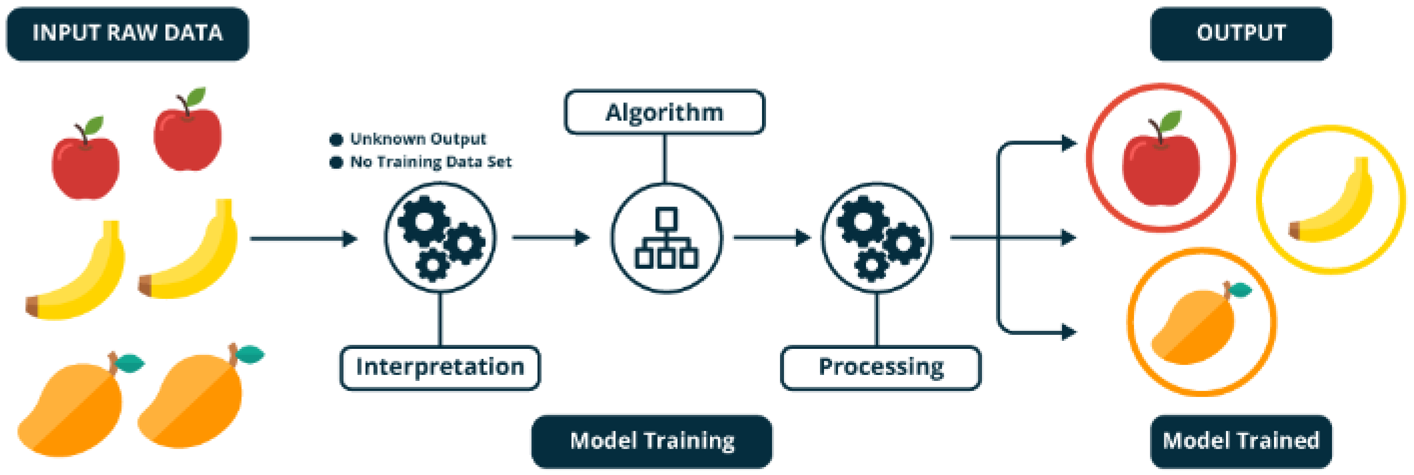
- A training set of examples with the correct responses (targets) is provided and, based on this training set, the algorithm generalizes to respond correctly to all possible inputs.
- This is also called learning from exemplars.



Types of Machine Learning

2. Unsupervised Learning

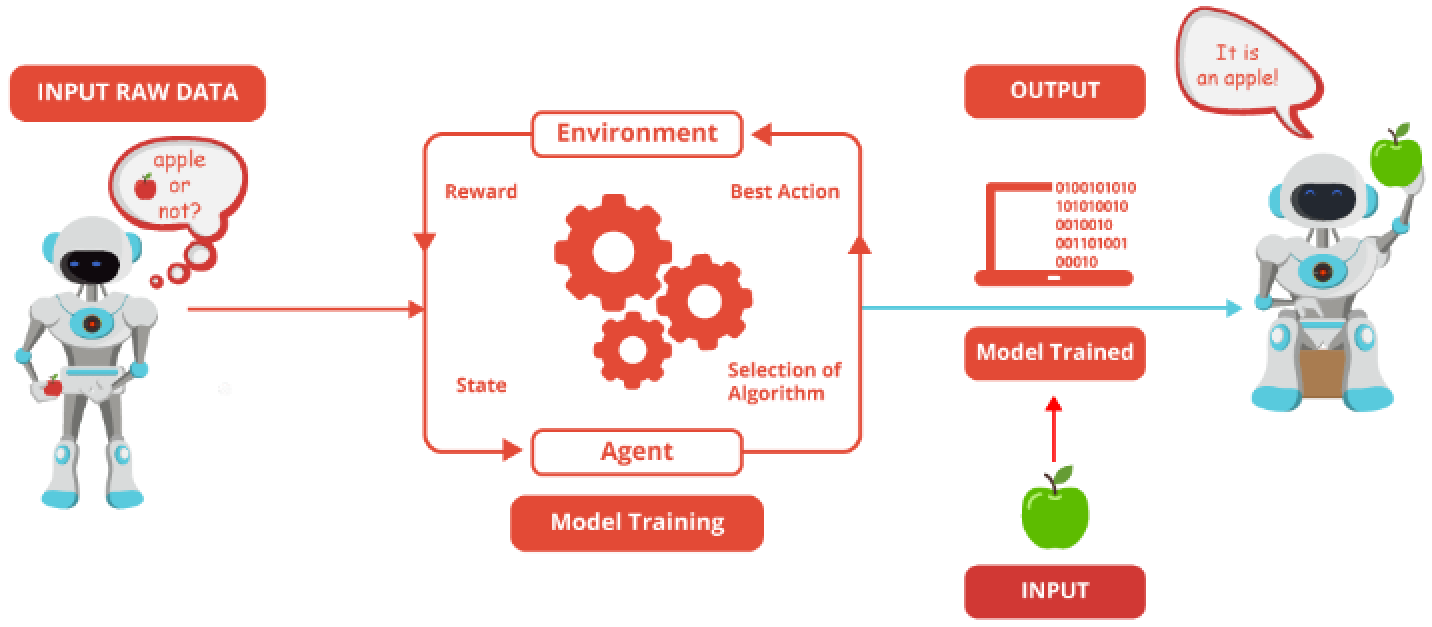
- Correct responses are not provided, but instead the algorithm tries to identify similarities between the inputs so that inputs that have something in common are categorized together.
- The statistical approach to unsupervised learning is known as density estimation.



Types of Machine Learning

3. Reinforcement Learning

- This is between supervised and unsupervised learning.
- The algorithm gets told when the answer is wrong, but does not get told how to correct it.



Types of Machine Learning

3. Reinforcement Learning

- It has to explore and try out different possibilities until it works out how to get the answer right.
- Reinforcement learning is sometime called learning with a critic because of this monitor that scores the answer, but does not suggest improvements.

Types of Machine Learning

4. Evolutionary Learning

- Biological evolution can be seen as a learning process.
- Biological life forms adapt to improve their survival rates and chance of having offspring in their environment.
- Model this in a computer using an idea of fitness, which corresponds to a score for how good the current solution is.

Machine Learning Process

- Gathering data
- Preparing that data
- Choosing a model
- Training
- Evaluation
- Hyperparameter tuning
- Prediction

Machine Learning Process

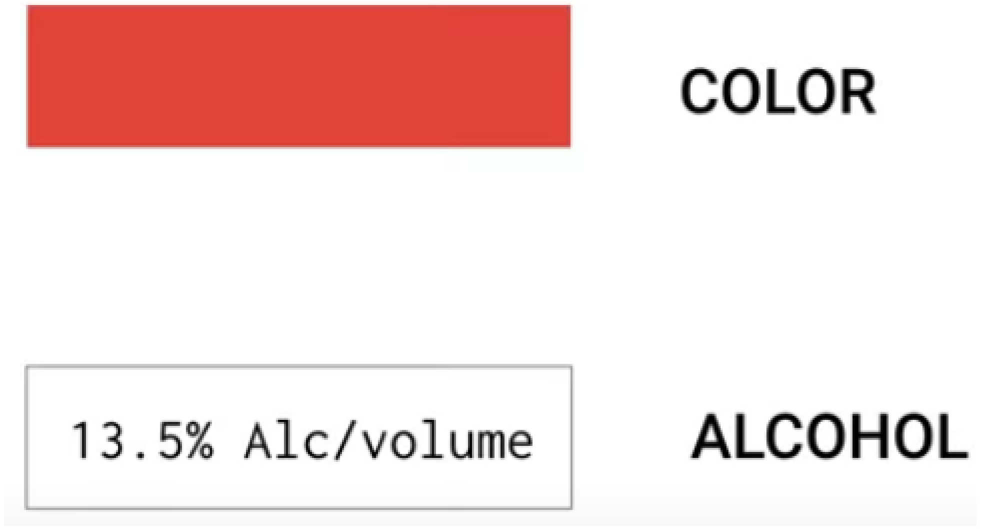
- Create a system that answers the question of whether a drink is **wine or beer**.
- This question answering system that we build is called a “**model**”.
- This model is created via a process called “**training**”.
- The goal of training is to create an **accurate model** that answers our questions correctly most of the time.
- But in order to train a model, we need to **collect data** to train on.

Machine Learning Process

- Our data will be collected from glasses of wine and beer.
- There are many aspects of the drinks that we could collect data on, everything from the amount of foam, to the shape of the glass.

Machine Learning Process

- We’ll pick just 2 simple ones: The **colo**r and the **alcohol content** (as a percentage).
- The hope is that we can split our two types of drinks along these two factors alone.
- We’ll call these our “**features**” from now on: **color, and alcohol**.



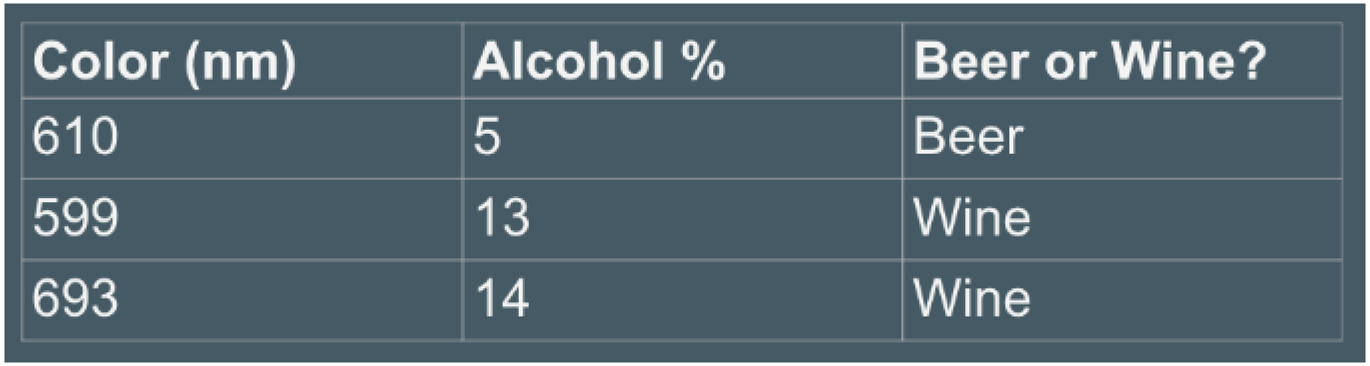
Machine Learning Process

- The first step to our process will be to buy up a bunch of different beers and wine, as well as get some equipment to do our measurements — a **spectrometer** for measuring the color, and a **hydrometer** to measure the alcohol content.

Machine Learning Process

- **1. Gathering Data**
- Once we have our equipment and drinks, it's time for our first real step of machine learning: gathering data.
- Very important step because the quality and quantity of data that gathered will directly determine how good the predictive model can be.
- In this case, the data we collect will be the color and the alcohol content of each drink.

Machine Learning Process



Color (nm)	Alcohol %	Beer or Wine?
610	5	Beer
599	13	Wine
693	14	Wine

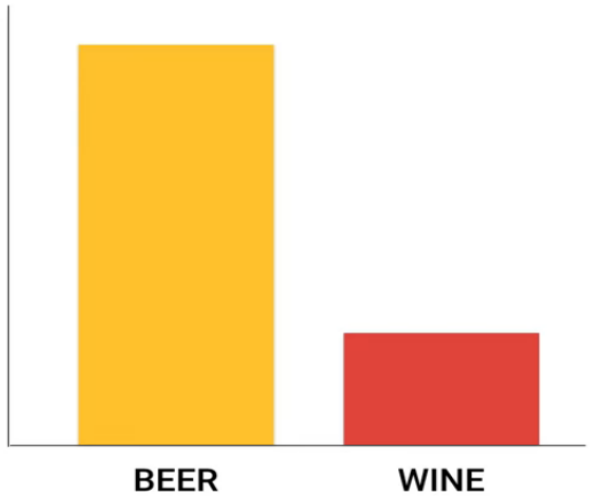
- This will yield a table of color, alcohol%, and whether it’s beer or wine.
- This will be our training data.

Machine Learning Process

- **2. Preparing that Data**
- Next step of machine learning: **Data preparation**, where we load our data into a suitable place and prepare it for use in our machine learning training.
- We'll first put all our data together, and then randomize the ordering.
- We make a determination of what a drink is, independent of what drink came before or after it.

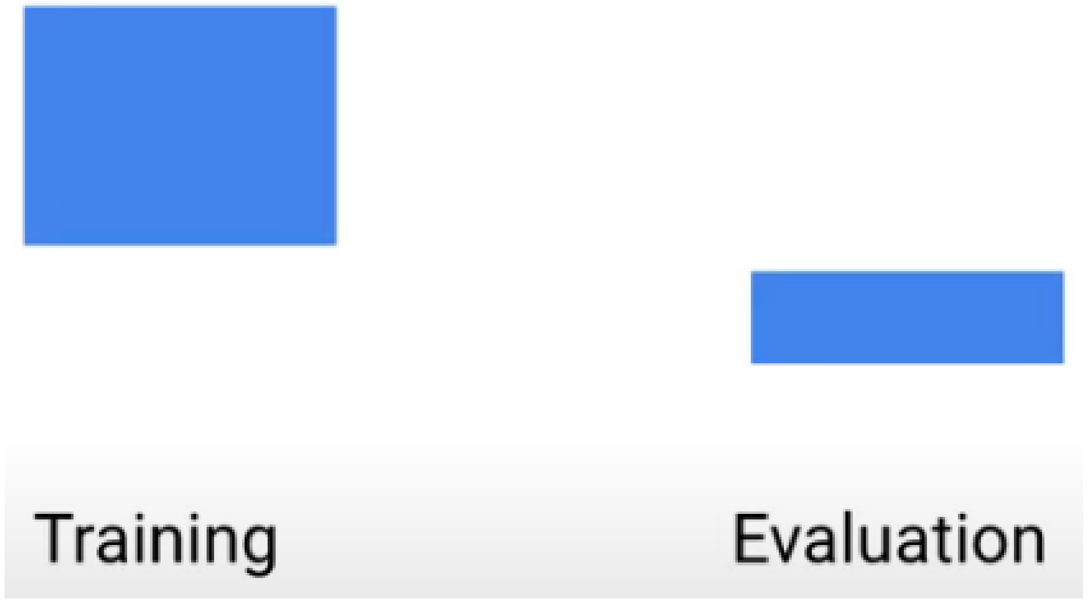
Machine Learning Process

- Good time to do any visualizations of the data, to help see if there are any relevant relationships between different variables and also if there are any data imbalances.
- If more data points about beer than wine are collected, then model will be biased toward guessing that everything that it sees is beer.



Machine Learning Process

- Need to **split the data in two parts**.
- The first part, used in **training our model**, will be the majority of the dataset.
- The second part will be used for **evaluating our trained model's performance**.

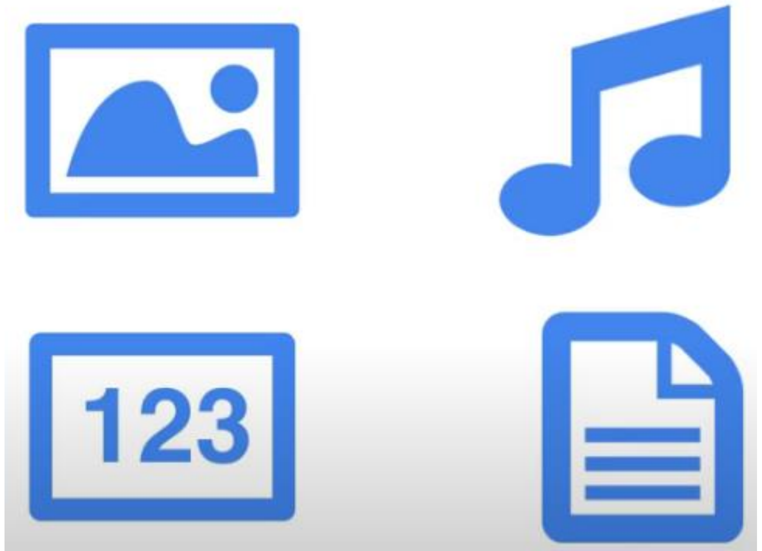


Machine Learning Process

- Sometimes the data we collect needs other forms of adjusting and manipulation.
- Things like duplication, normalization, error correction, and more.
- These would all happen at the data preparation step.

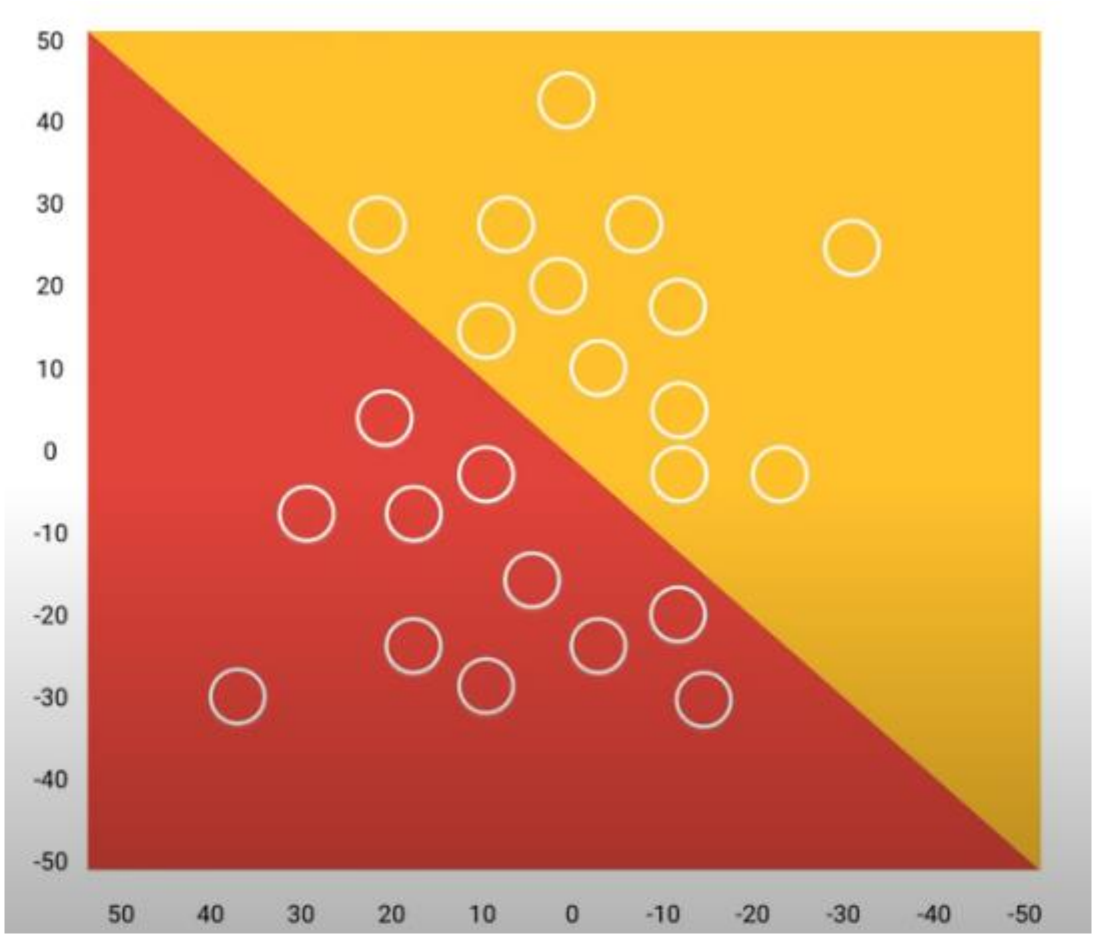
Machine Learning Process

- **3. Choosing a model**
- There are many models that researchers and data scientists have created over the years.
- Some are well suited for image data, others for sequences (like text, or music), some for numerical data, others for text-based data.



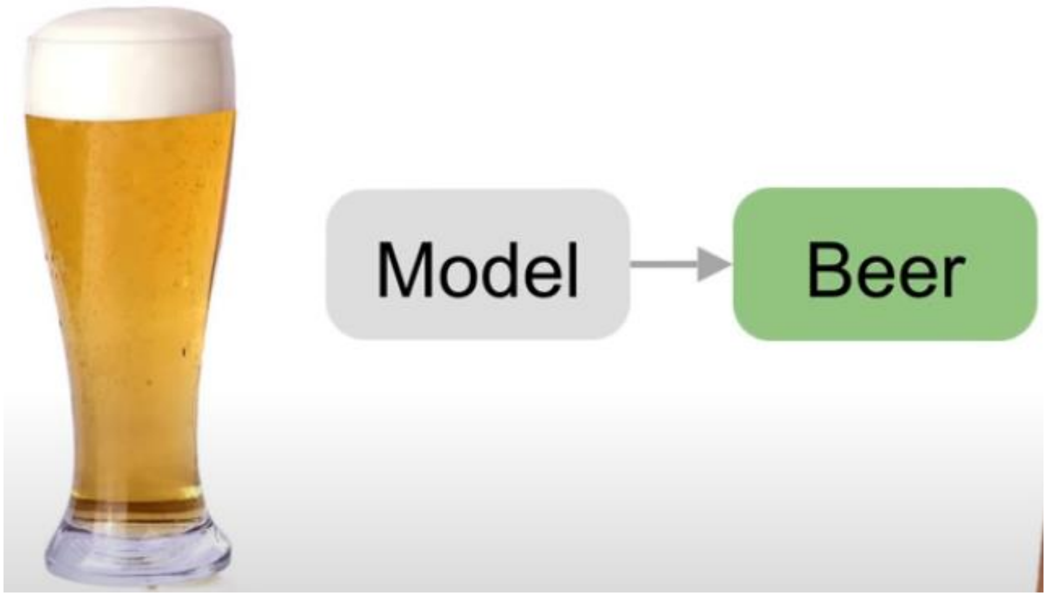
Machine Learning Process

- Since we have 2 features, color and alcohol%, we can use a small linear model.



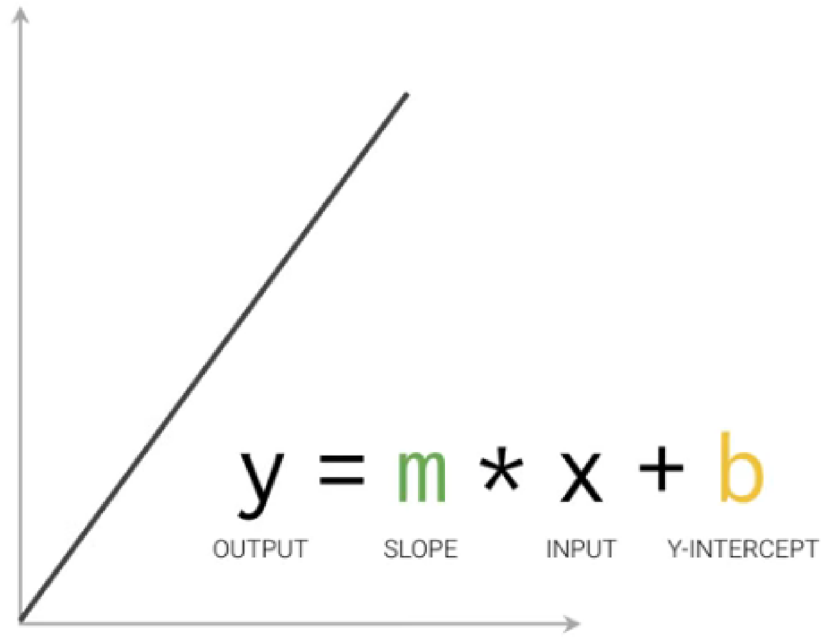
Machine Learning Process

- **4. Training**
- The training step is called as the bulk of machine learning.
- In this step, we will use our data to incrementally improve our model's ability to predict whether a given drink is wine or beer.



Machine Learning Process

- The formula for a straight line is y=m*x+b, where **x is input**, **m is slope**, **b is y-intercept**, and **y is the value of the line at the position x**.
- Values used for adjusting(training) are **m & b**.
- Other variables are x, our input, and y, our output.



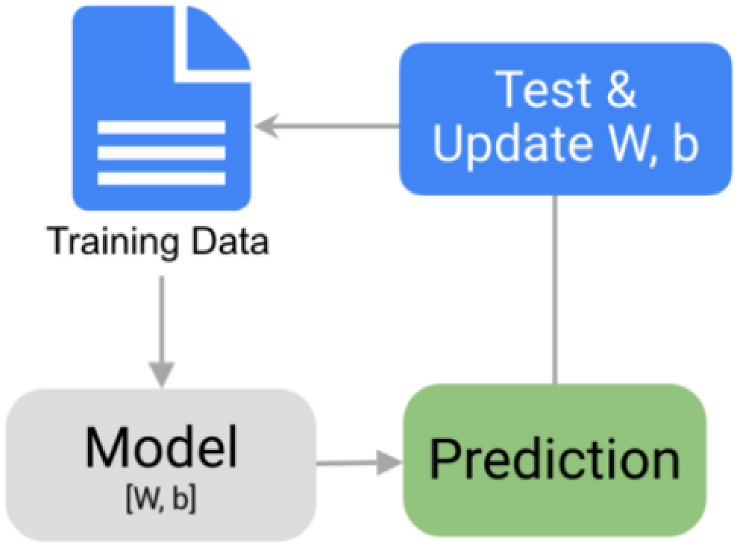
Machine Learning Process

- In machine learning, there are many m’s since there may be many features.
- The collection of these m values is usually formed into a matrix, that we will denote W, for the “weights” matrix.
- Similarly for b, we arrange them together and call that the biases.

$$\text{WEIGHTS} = \begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \\ m_{3,1} & m_{3,2} \end{bmatrix} \qquad \text{BIASES} = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix}$$

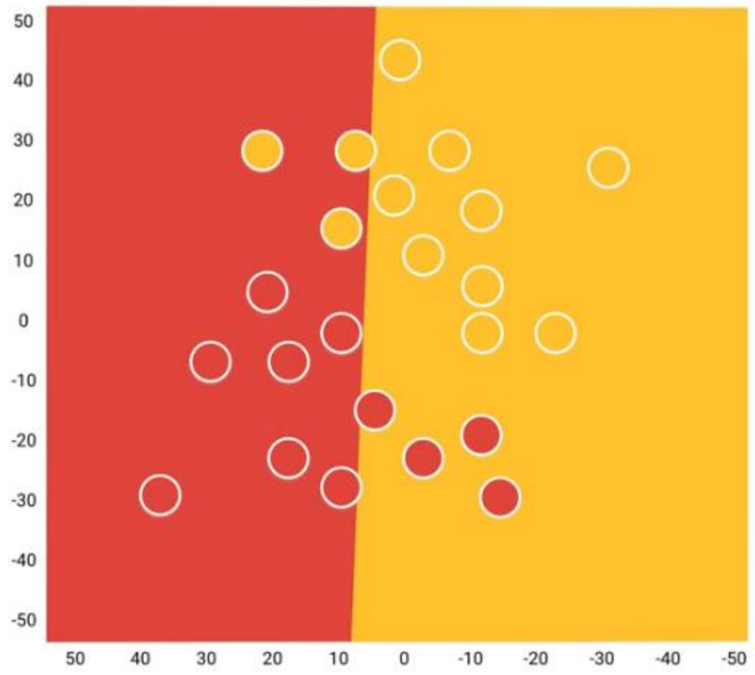
Machine Learning Process

- Training process involves initializing random values for W and b and predict the output.
- Initially it does pretty poorly. But we can compare our model's predictions with the output, and adjust the values in W and b such that we will have more correct predictions.



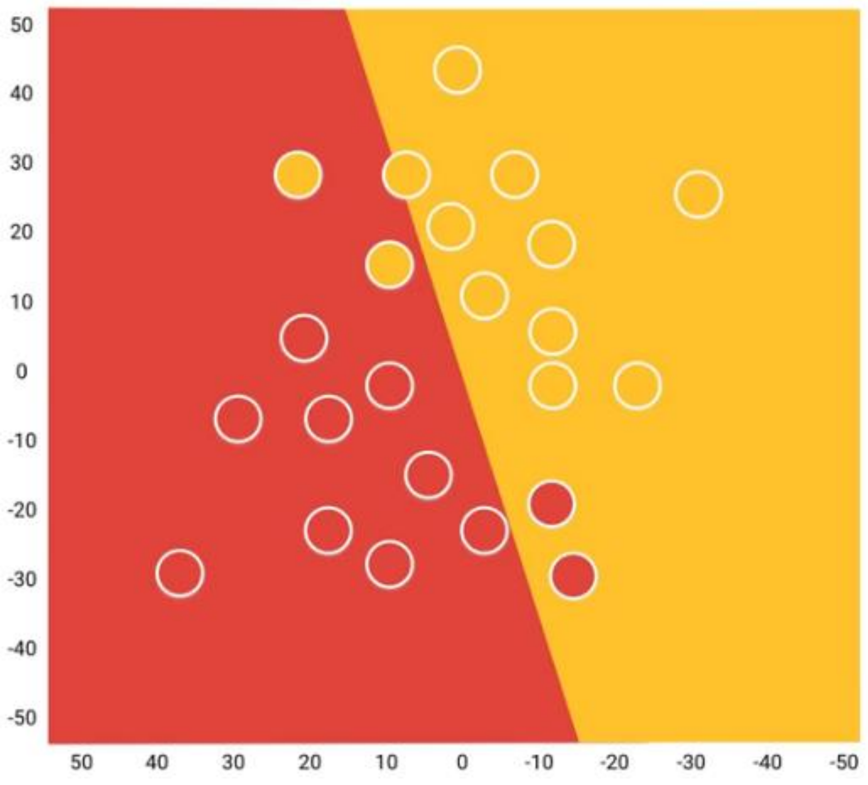
Machine Learning Process

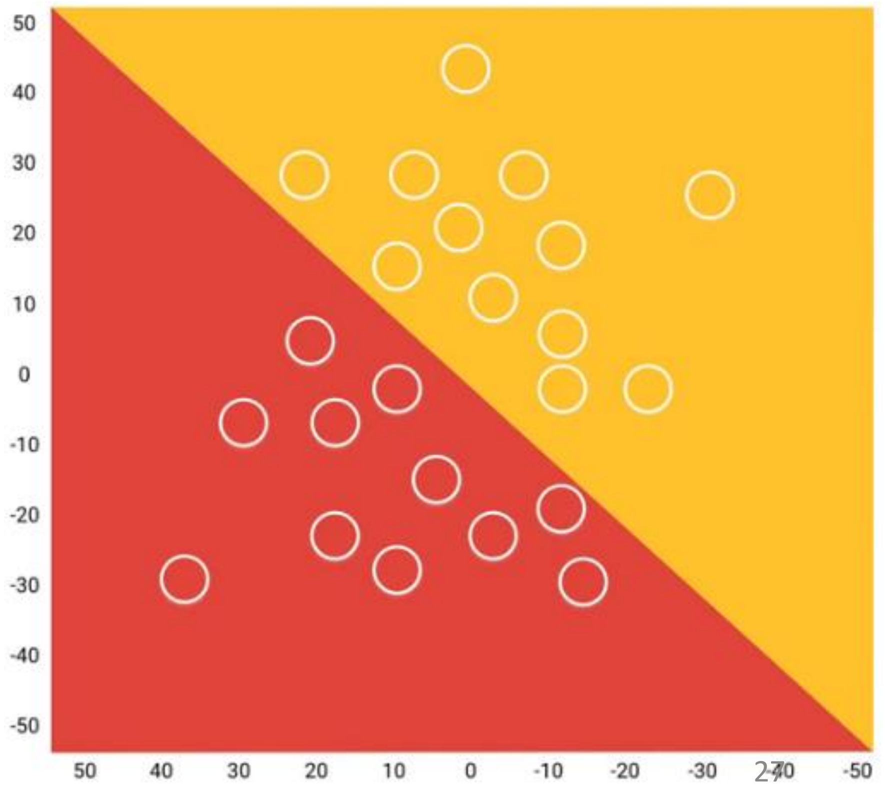
- This process then repeats.
- Each iteration or cycle of updating the weights and biases is called one training “step”.
- When we first start the training, it’s like we drew a random line through the data.



Machine Learning Process

- Then as each step of the training progresses, the line moves, step by step, closer to an ideal separation of the wine and beer.



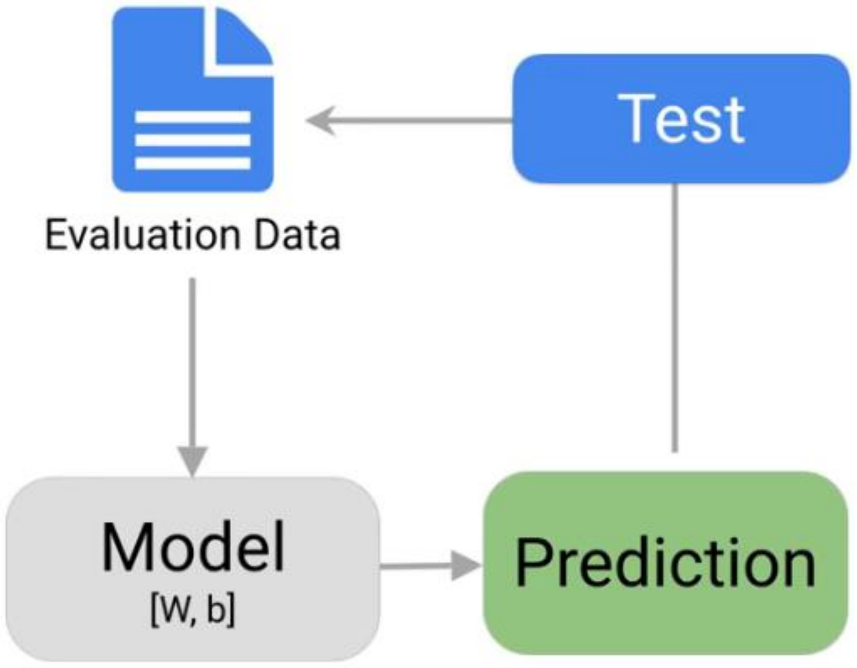


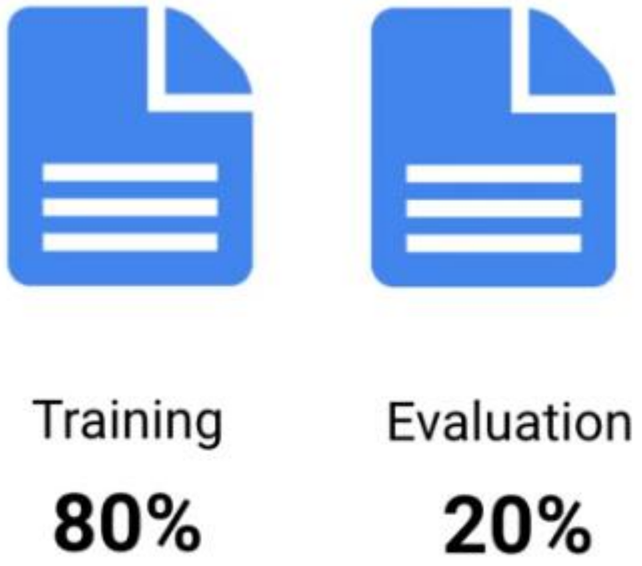
Machine Learning Process

- **5. Evaluation**
- Once training is complete, it’s time to see if the model is any good, using Evaluation.
- This is where that dataset that we set aside earlier comes into play.
- Evaluation allows us to test our model against data that has never been used for training.
- This metric allows us to see how the model might perform against data that it has not yet seen.

Machine Learning Process

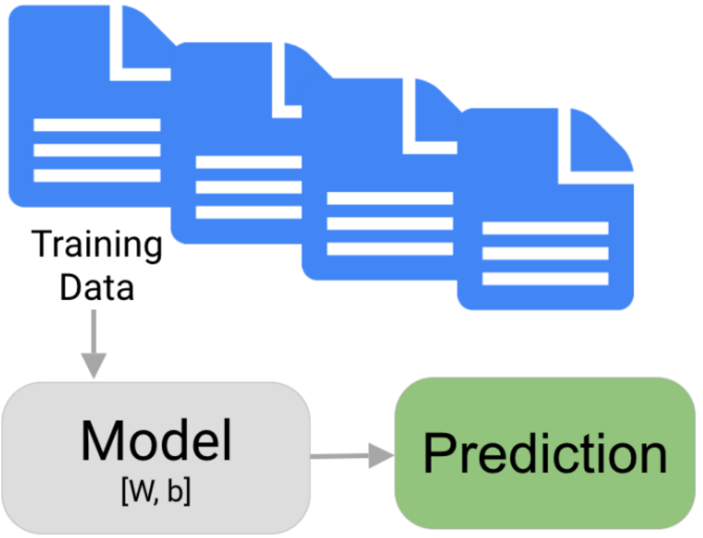
- **Evaluation**
- For training-evaluation split ideally should be 80/20 or 70/30.
- Much of this depends on the size of the original source dataset.





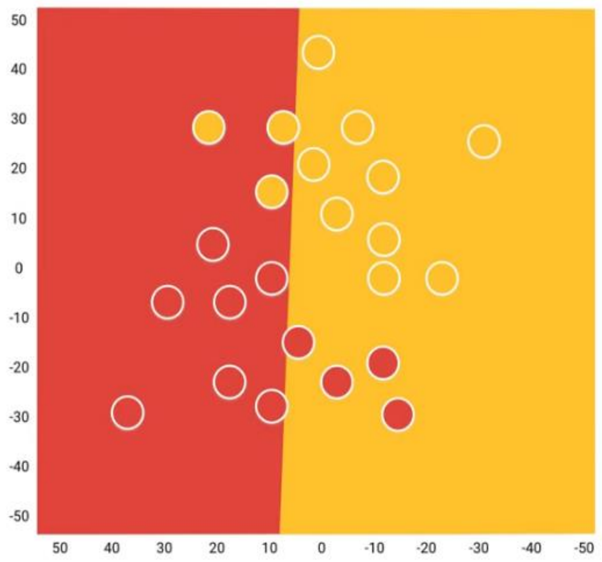
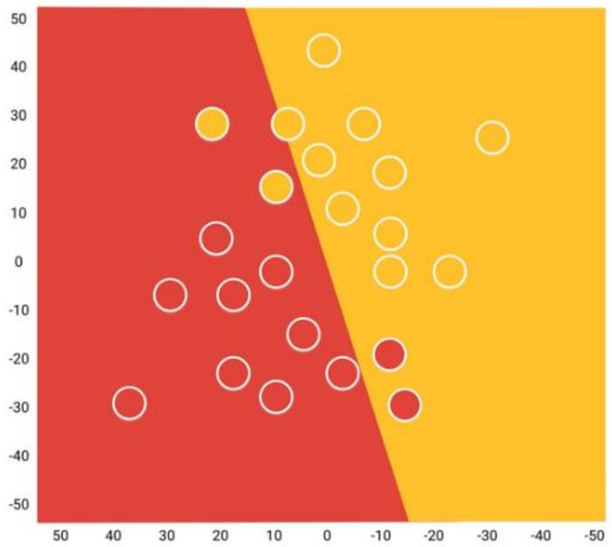
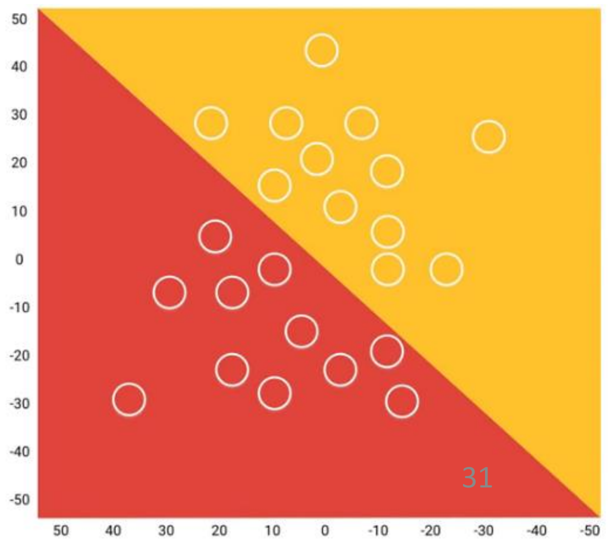
Machine Learning Process

- **6. Parameter Tuning**
- How to improve your training in any way.
- We can do this by tuning our parameters.
- One example is to train the model on the full dataset multiple times, rather than just once.
- This can sometimes lead to higher accuracies.



Machine Learning Process

- Another parameter is “learning rate”.
- This defines how far we shift the line during each step, based on the information from the previous training step.
- These values all play a role in how accurate model can become, and how long the training takes.

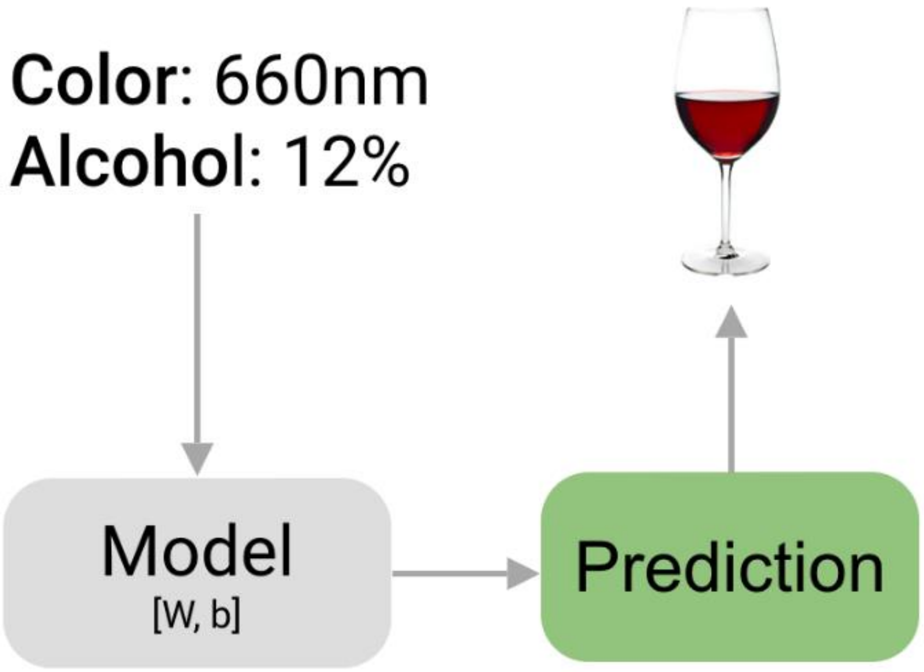




Machine Learning Process

- These parameters are typically referred to as “hyperparameters”.
- The adjustment, or tuning, of these hyperparameters, remains a bit of an art, and is more of an experimental process that heavily depends on the specifics of your dataset, model, and training process.
- Once you’re happy with your training and hyperparameters, guided by the evaluation step, it’s time to finally use your model to do something useful!

Machine Learning Process

- **7. Prediction**
- Machine learning is using data to answer questions.
- So Prediction, or inference, is the step where we get to answer some questions.
- This is the point of all this work, where the value of machine learning is realized.



Machine Learning Process

- **The big picture**
- The power of machine learning is that we were able to determine how to differentiate between wine and beer using our model, rather than using human judgment and manual rules.

1. **Gathering data**
2. **Preparing that data**
3. **Choosing a model**
4. **Training**
5. **Evaluation**
6. **Hyperparameter tuning**
7. **Prediction.**

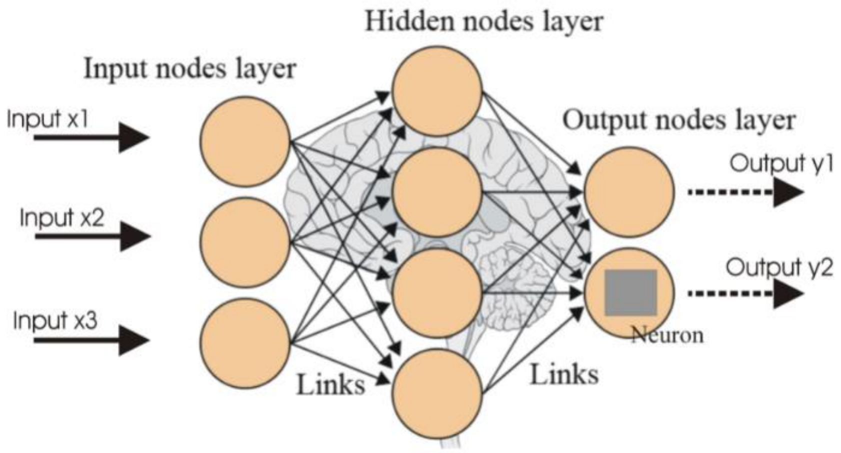
Machine Learning Terminology

- **Inputs:**
- An input vector is the data given as one input to the algorithm.
- Written as x, with elements xi, where i runs from 1 to the number of input dimensions, m.

- **Weights wij** , are the weighted connections between nodes i and j.
- For neural networks these weights are analogous to the synapses in the brain.
- They are arranged into a matrix W.

Machine Learning Terminology

- **Outputs:**
- The output vector is y, with elements yj , where j runs from 1 to the number of output dimensions, n.
- We can write y(x,W) as the output depends on the inputs to the algorithm and the current set of weights of the network.



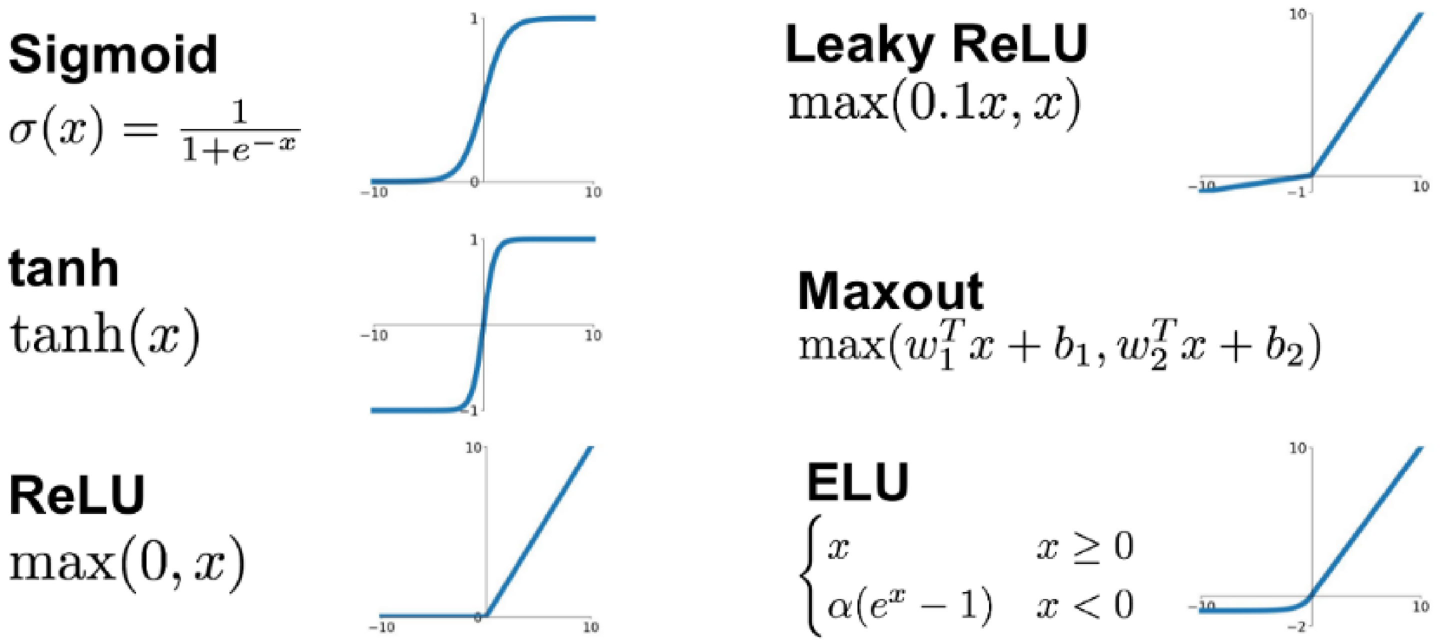
Machine Learning Terminology

- **Targets:**
- The target vector t, with elements tj , where j runs from 1 to the number of output dimensions, n, are the extra data that we need for supervised learning, since they provide the ‘correct’ answers that the algorithm is learning about.

- **Error E,** a function that computes the inaccuracies of the network as a function of the outputs y and targets t.

Machine Learning Terminology

- **Activation Function:**
- For neural networks, g(·) is a mathematical function that describes the firing of the neuron as a response to the weighted inputs, such as the threshold function.



Machine Learning Terminology

- **Weight Space:**
- When working with data it is good to plot it and look at the data.
- If the data has 2 or 3 input dimensions, then use the x-axis for feature-1, the y-axis for feature-2, and the z-axis for feature-3.
- Plot the input vectors on these axes.
- If we have 200 input dimensions (ie, 200 elements in each of input vectors) then we can try to imagine it plotted by using 200 axes that are all mutually orthogonal.

Machine Learning Terminology

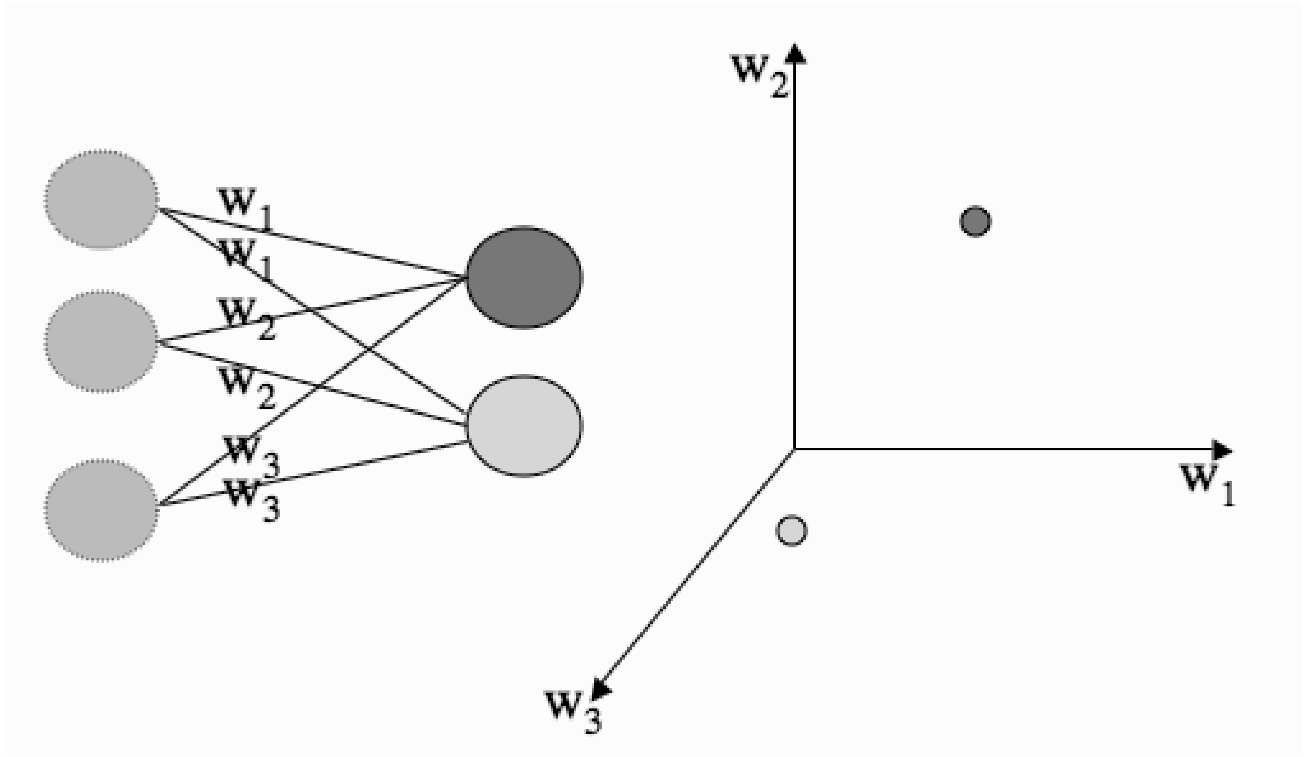
- **Weight Space:**
- We can look at projections of the data into our 3D world by plotting just three of the features against each other.
- Points can look very close together in the chosen three axes, but can be a very long way apart in the full set.

Machine Learning Terminology

- **Weight Space:**
- We can plot some of the parameters of a machine learning algorithm.
- Particularly useful for neural networks since the parameters of a neural network are the values of a set of weights that connect the neurons to the inputs.

Machine Learning Terminology

- **Weight Space:**
- Schematic of a neural network on the left of the following figure showing the inputs on the left, and the neurons on the right.



Machine Learning Terminology

- **Weight Space:**
- If we treat the weights that get fed into one of the neurons as a set of coordinates in what is known as weight space, then we can plot them.
- We think about the weights that connect into a particular neuron, and plot the strengths of the weights by using one axis for each weight that comes into the neuron, and plotting the position of the neuron as the location, using the value of w1 as the position on the 1st axis, the value of w2 on the 2nd axis, etc

Machine Learning Terminology

- **Weight Space:**
- By changing the weights we are changing the location of the neurons in this weight space.
- We can measure distances between inputs and neurons by computing the Euclidean distance, which in 2-dimensions can be written as:

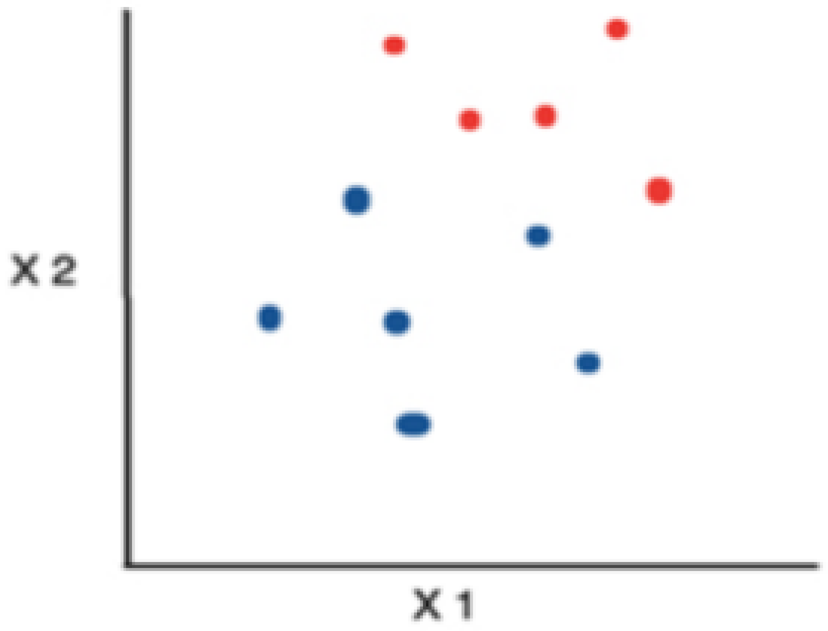
$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Machine Learning Terminology

- **Weight Space:**
- If the neuron is close to the input then it should fire, and if it is not close then it shouldn’t.
- This picture of weight space can be helpful for understanding what effect the number of input dimensions can have.
- Include all of the information that we can get, and let the algorithm sort out for itself what it needs.

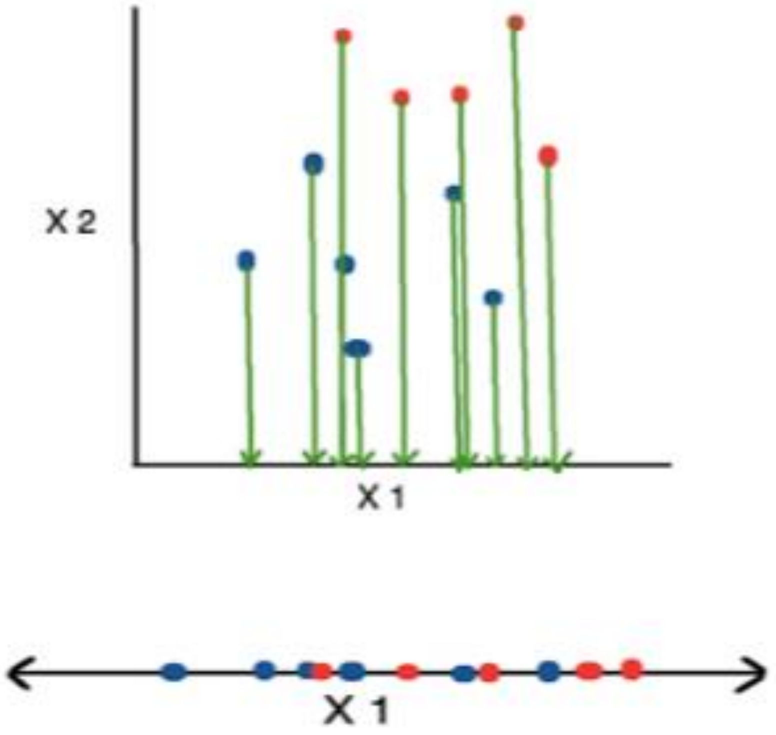
Machine Learning Terminology

- **Dimensionality Reduction:**
- Relationship between 2 variables are plotted where each color represents a different class.



Machine Learning Terminology

- **Dimensionality Reduction:**
- To reduce the number of dimensions to 1, just project everything to the x-axis.

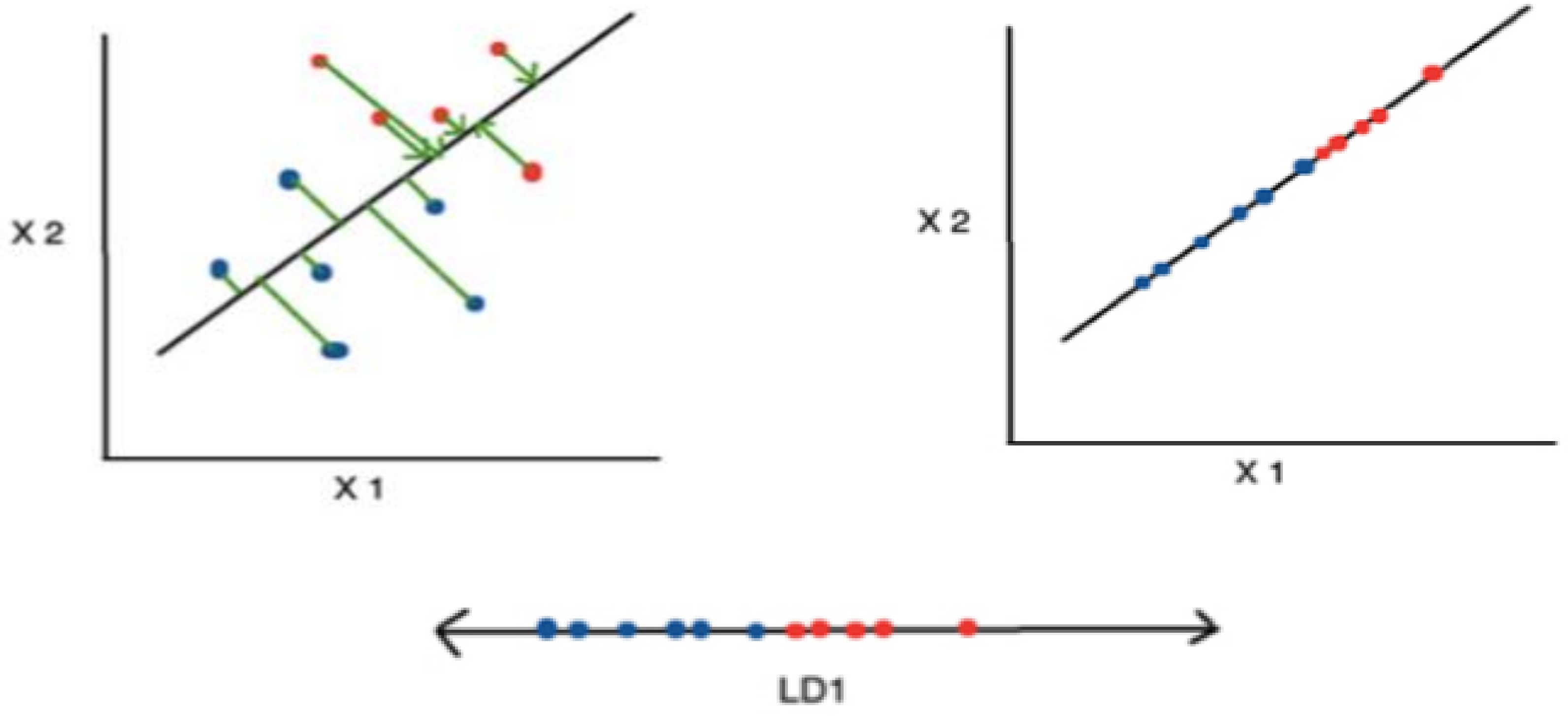


Machine Learning Terminology

- **Dimensionality Reduction : Linear Discriminant Analysis (LDA)**
- This approach neglects any helpful information provided by the second feature.
- Use LDA to plot it.
- The advantage of LDA is that it uses information from both the features to create a new axis which in turn minimizes the variance and maximizes the class distance of the two variables.

Machine Learning Terminology

- **Dimensionality Reduction : Linear Discriminant Analysis (LDA)**
- This approach neglects any helpful information provided by the second feature.



Machine Learning Terminology

- **The Curse of Dimensionality:**
- Phenomena of strange/weird things happening during analyses of the data in high-dimensional spaces.
- Building several machine learning models to analyze the performance of a Formula One (F1) driver.

Machine Learning Terminology

- **The Curse of Dimensionality:**
- i) Model-1 consists of only two features say the circuit name and the country name.
- ii) Model-2 consists of 4 features say weather and max speed of the car plus the above two.
- iii) Model-3 consists of 8 features say driver's experience, number of wins, car condition, and driver's physical fitness including all the above features.

Machine Learning Terminology

- **The Curse of Dimensionality:**
- iv) Model-4 consists of 16 features say driver's age, latitude, longitude, driver's height, hair color, car color, the car company, and driver's marital status including all the above features.
- v) Model-5 consists of 32 features.
- vi) Model-6 consists of 64 features.
- vii) Model-7 consists of 128 features.
- viii) Model-8 consists of 256 features.
- ix) Model-9 consists of 512 features.
- x) Model-10 consists of 1024 features

Machine Learning Terminology

- **The Curse of Dimensionality:**
- Assuming the training data remains constant, it is observed that on increasing the number of features the accuracy tends to increase until a certain threshold value and after that, it starts to decrease.
- From the above example the accuracy of Model_1 < accuracy of Model_2 < accuracy of Model_3 but if we try to extrapolate this trend it doesn’t hold true for all the models having more than 8 features.

Machine Learning Terminology

- **The Curse of Dimensionality:**
- The essence of the curse is the realization that as the number of dimensions increases, the volume of the unit hypersphere does not increase with it.
- The unit hypersphere is the region we get if we start at the origin and draw all the points that are distance 1 away from the origin.
- In 2 dimensions we get a circle of radius 1 around (0, 0), and in 3D we get a sphere around (0, 0, 0).

Machine Learning Terminology

- **The Curse of Dimensionality:**
- In higher dimensions, the sphere becomes a hypersphere.

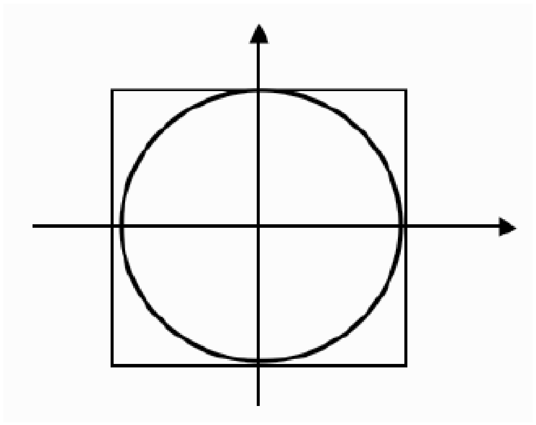


FIGURE 2.2 The unit circle in 2D with its bounding box.

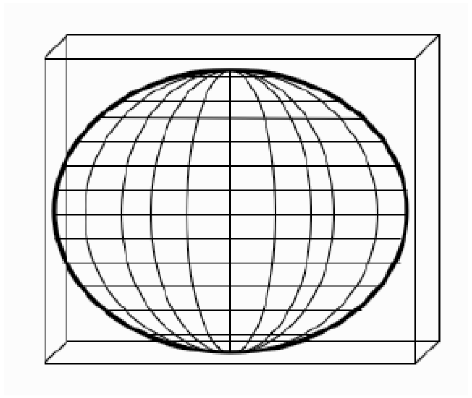
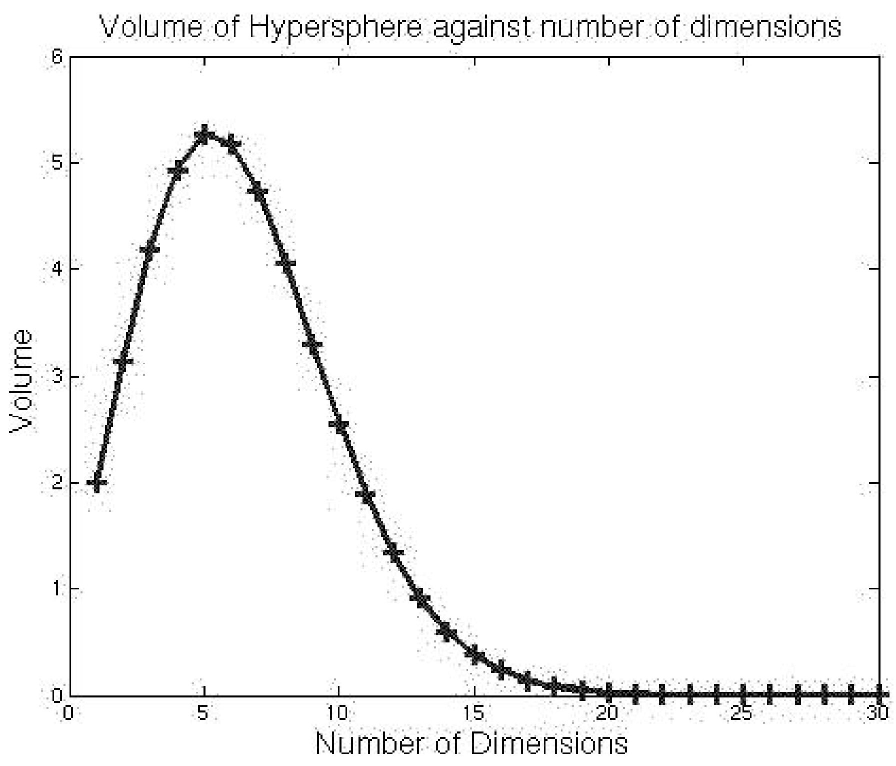


FIGURE 2.3 The unit sphere in 3D with its bounding cube. The sphere does not reach as far into the corners as the circle does, and this gets more noticeable as the number of dimensions increases.

Machine Learning Terminology

- **The Curse of Dimensionality:**
- As the number of dimensions tends to infinity, so the volume of the hypersphere tends to zero.

Dimension	Volume
1	2.0000
2	3.1416
3	4.1888
4	4.9348
5	5.2636
6	5.1677
7	4.7248
8	4.0587
9	3.2985
10	2.5502



Machine Learning Terminology

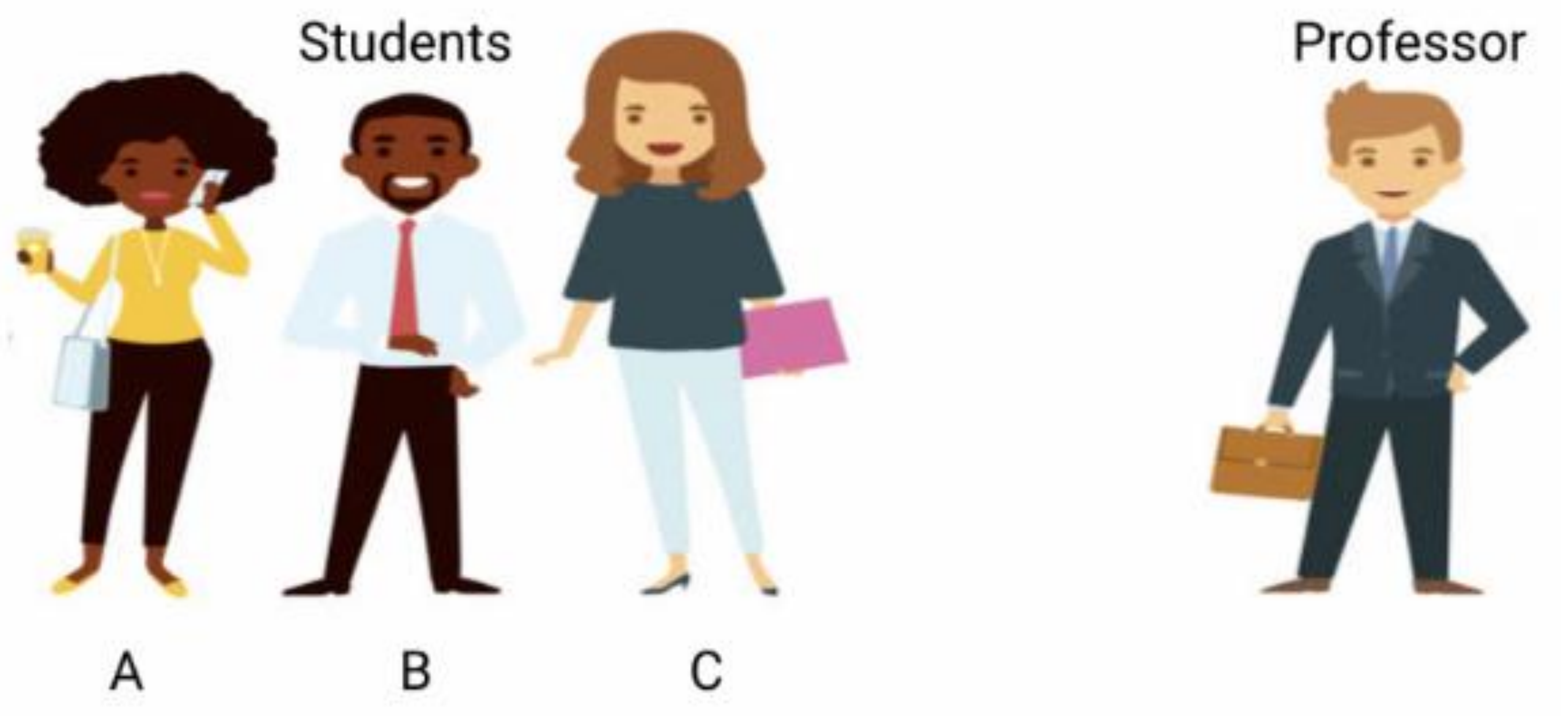
- **The Curse of Dimensionality:**
- The curse of dimensionality will apply to ML algorithms because as the number of input dimensions gets larger, we will need more data to enable the algorithm to generalize sufficiently well.
- The ML algorithms try to separate data into classes based on the features.
- Therefore as the number of features increases, so will the number of data points.

Testing ML Algorithms

- The purpose of learning is to get better at predicting the outputs, be the class labels or continuous regression values.
- The only real way to know how successfully the algorithm has learnt is to compare the predictions with known target labels, which is how the training is done for supervised learning.
- Training Set and Test Set
- Use test set of (input, target) pairs and compare the predicted output with the target to decide how well the algorithm has learnt.

Testing ML Algorithms

- **Underfitting vs Overfitting vs Bestfit:**



Testing ML Algorithms

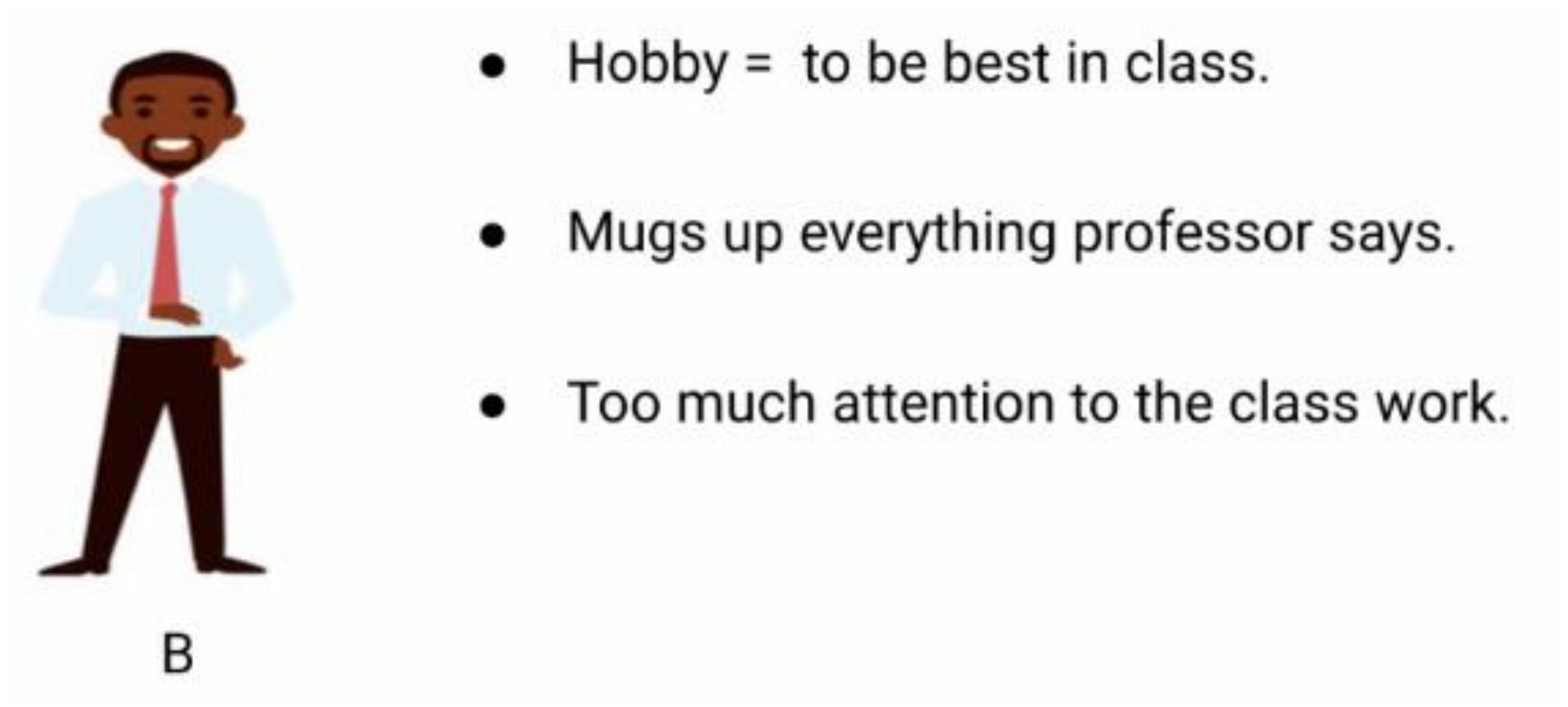
- **Underfitting vs Overfitting vs Bestfit:**

A

- Hobby = chating
- Not interested in class
- Doesn’t pay much attention to professor

Testing ML Algorithms

- **Underfitting vs Overfitting vs Bestfit:**



- Hobby = to be best in class.
- Mugs up everything professor says.
- Too much attention to the class work.

Testing ML Algorithms

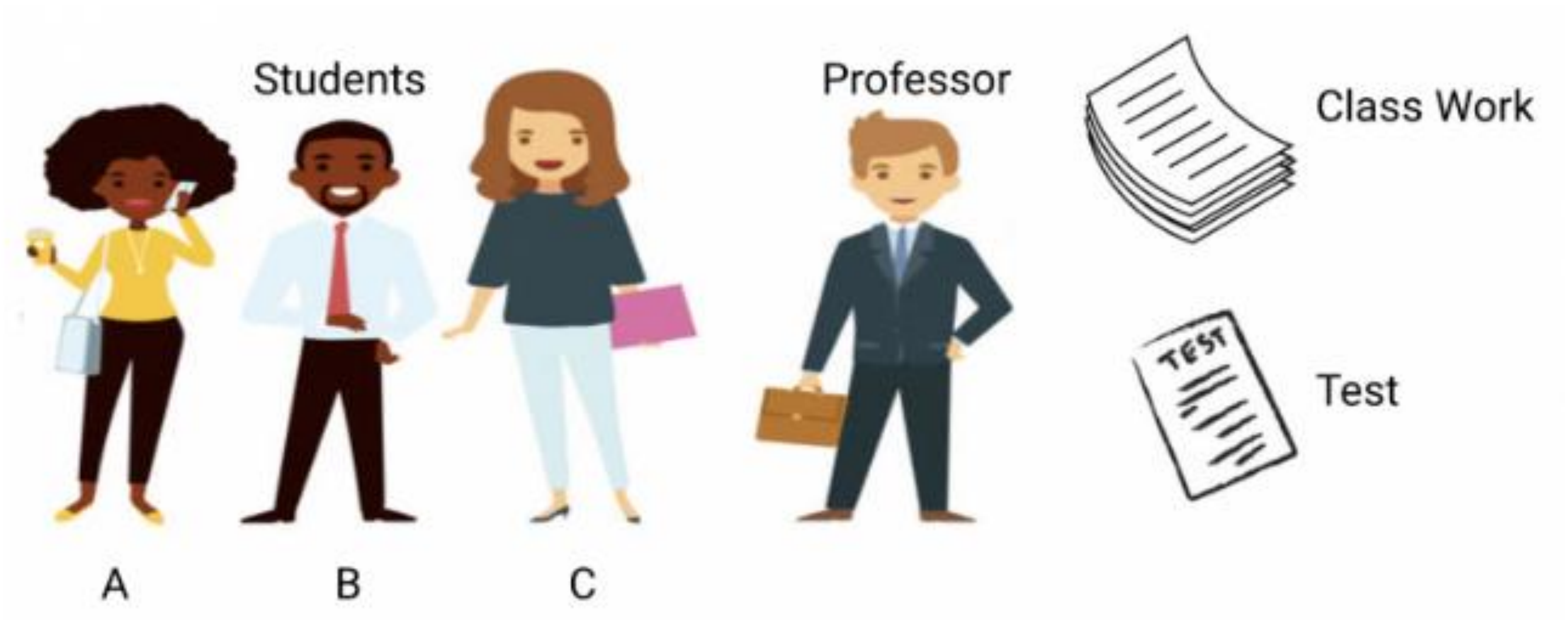
- **Underfitting vs Overfitting vs Bestfit:**

C

- Hobby = learning new things
- Eager to learn concepts.
- Pays attention to class and learns the idea behind solving a problem.

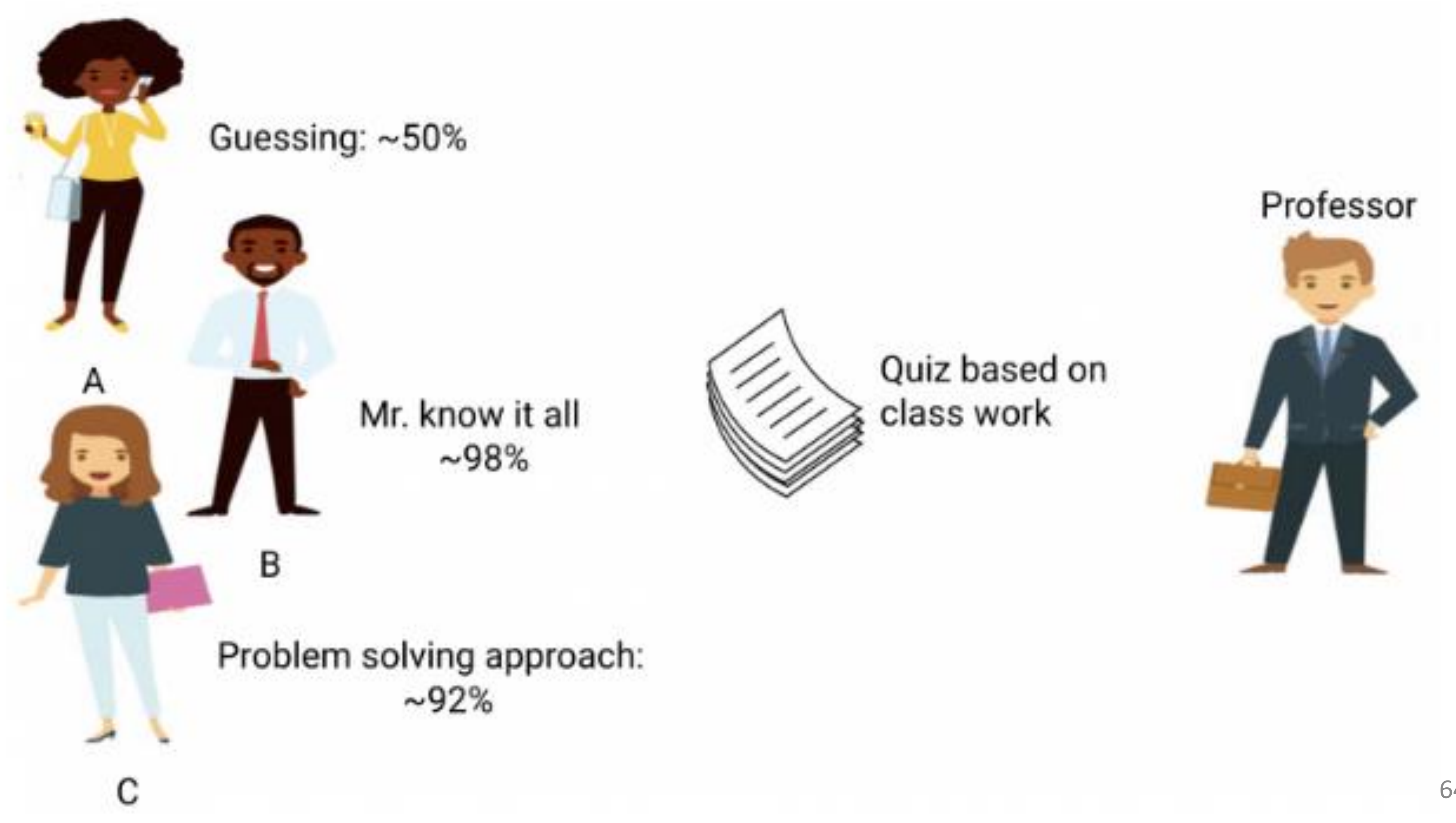
Testing ML Algorithms

- **Underfitting vs Overfitting vs Bestfit:**



Testing ML Algorithms

- **Underfitting vs Overfitting vs Bestfit:**
- **Class Test**

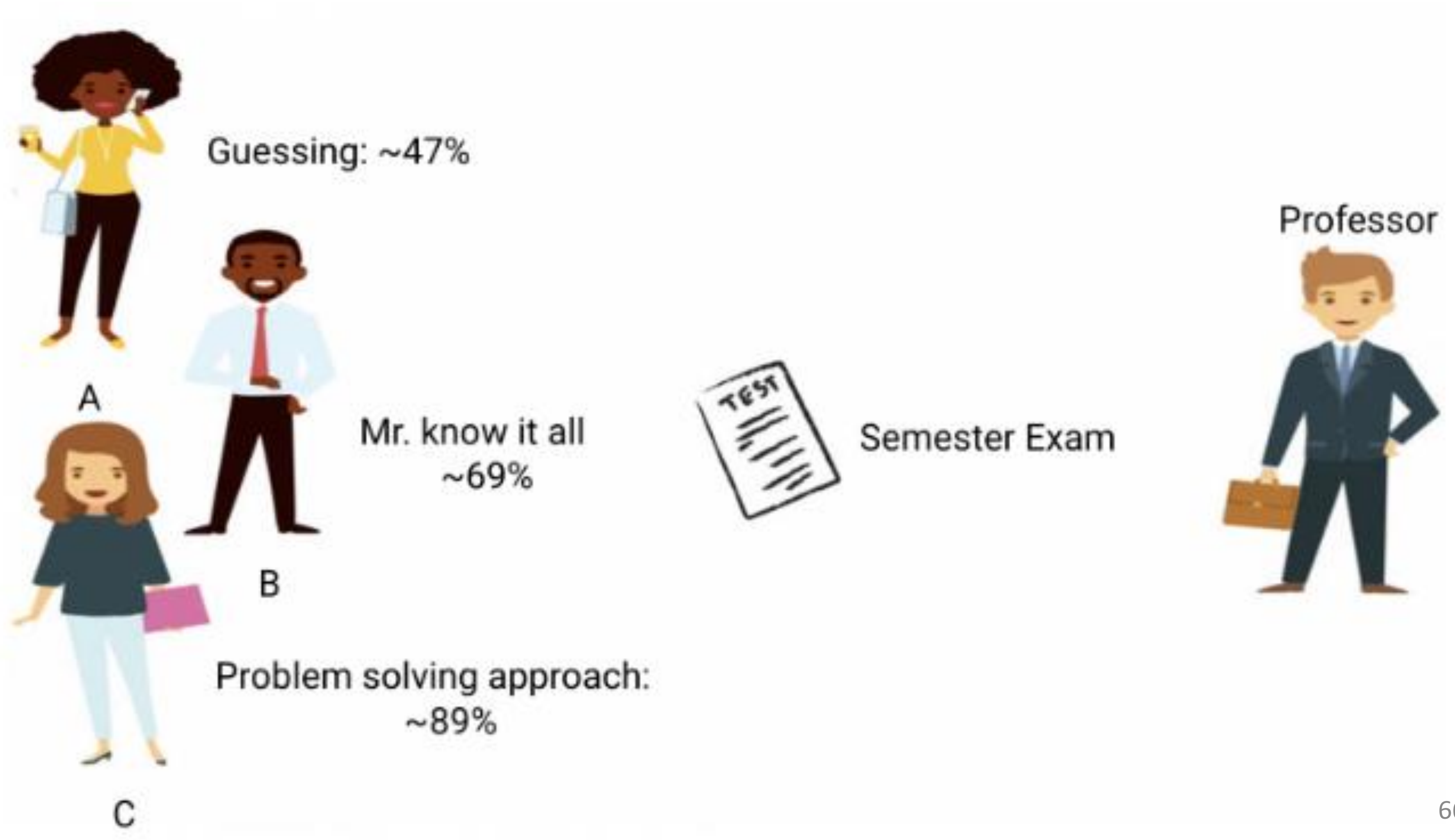


Testing ML Algorithms

- **Underfitting vs Overfitting vs Bestfit:**
- Student A, who was distracted in her own world, simply guessed the answers and got approximately 50% marks in the test.
- Student B, who memorized each and every question taught in the classroom was able to answer almost every question by memory and therefore obtained 98% marks in the class test.
- Student C actually solved all the questions using the problem-solving approach she learned in the classroom and scored 92%.

Testing ML Algorithms

- **Underfitting vs Overfitting vs Bestfit:**
- Semester End Examination

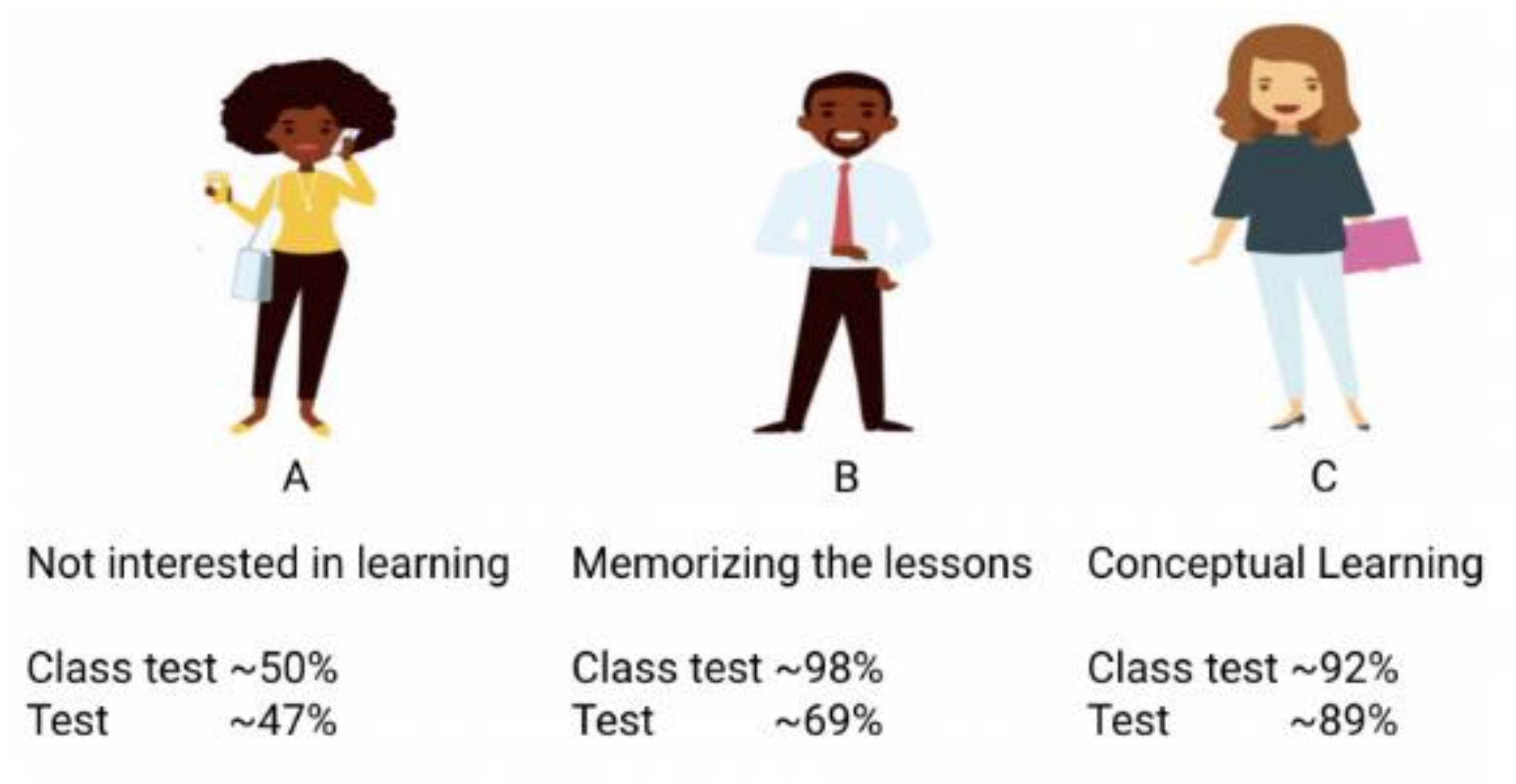


Testing ML Algorithms

- **Underfitting vs Overfitting vs Bestfit:**
- In the case of student A, things did not change much and she still randomly answers questions correctly ~50% of the time.
- Student B performed poorly in the SEE. Because he always memorized the problems that were taught in the class but this SEE test contained questions which he has never seen before.
- Student C scored more or less the same. This is because she focused on learning the problem-solving approach and therefore was able to apply the concepts she learned to solve the unknown questions.

Testing ML Algorithms

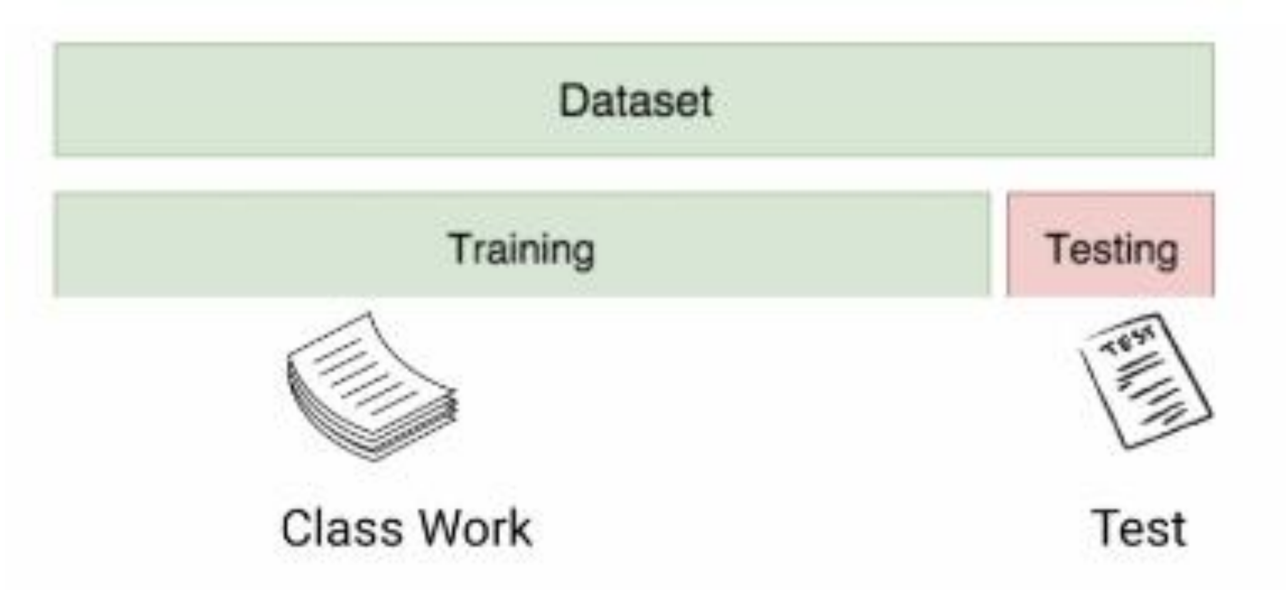
- **Underfitting vs Overfitting vs Bestfit:**



A	B	C
Not interested in learning	Memorizing the lessons	Conceptual Learning
Class test ~50%	Class test ~98%	Class test ~92%
Test ~47%	Test ~69%	Test ~89%

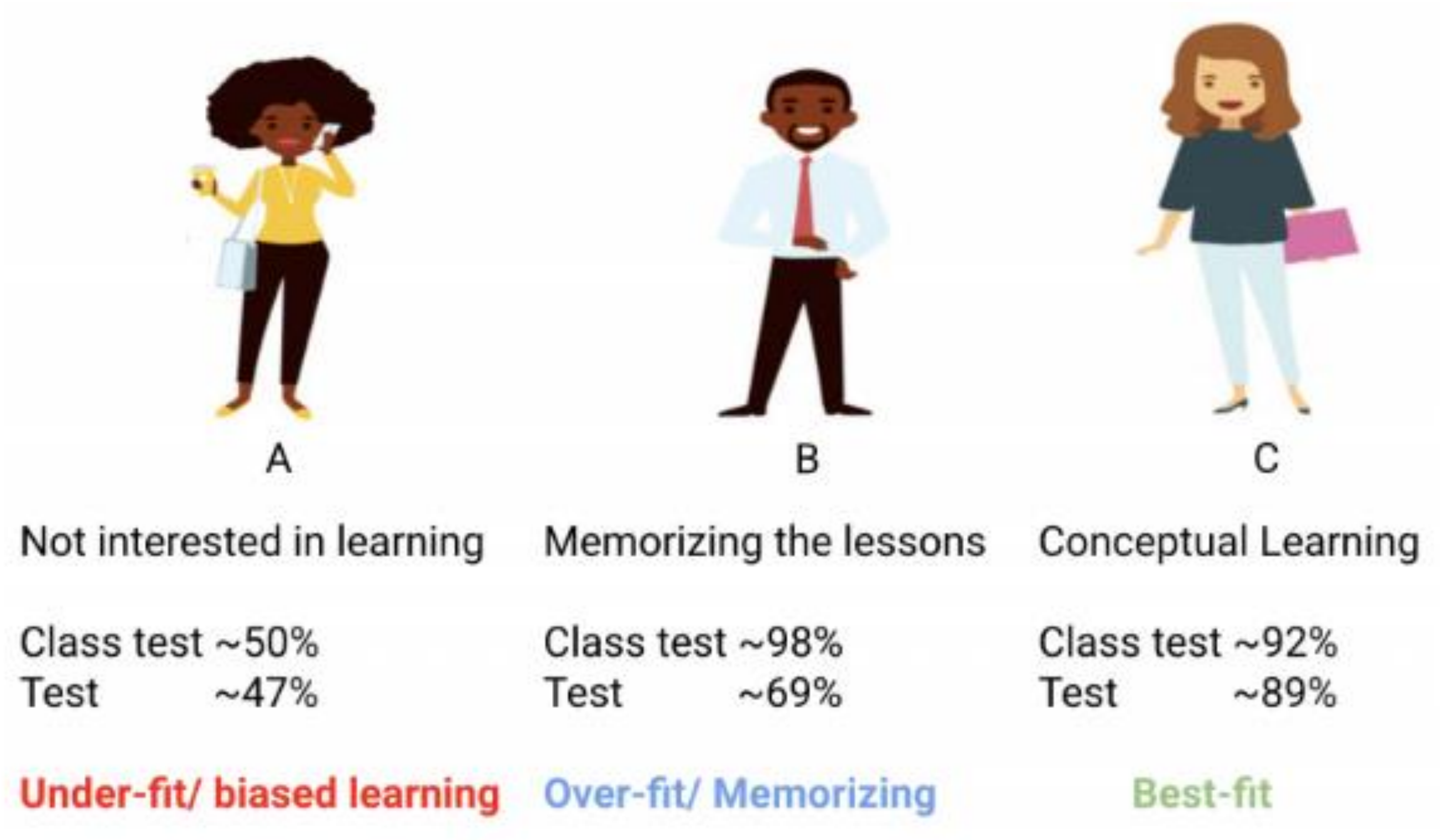
Testing ML Algorithms

- **Underfitting vs Overfitting vs Bestfit:**



Testing ML Algorithms

- **Underfitting vs Overfitting vs Bestfit:**



A	B	C
Not interested in learning	Memorizing the lessons	Conceptual Learning
Class test ~50% Test ~47%	Class test ~98% Test ~69%	Class test ~92% Test ~89%
Under-fit/ biased learning	Over-fit/ Memorizing	Best-fit

- https://www.analyticsvidhya.com/blog/2020/02/underfitting-overfitting-best-fitting-machine-learning/

Testing ML Algorithms

- **Underfitting vs Overfitting vs Bestfit:**
- If a given model is performing too well on the training data but the performance drops significantly over the test set is called an **overfitting model**.

- If the model is performing poorly over the test and the train set, then we call that an **underfitting model**.

Testing ML Algorithms

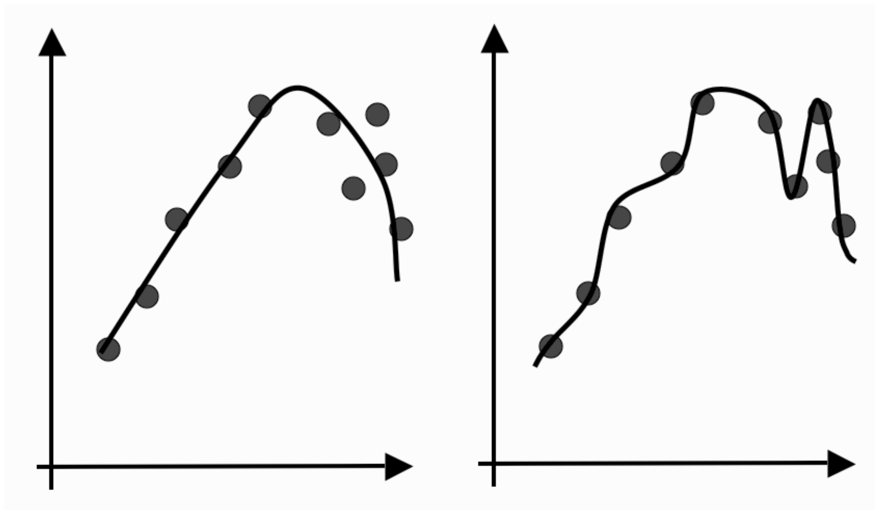
- **Underfitting vs Overfitting vs Bestfit:**

Testing ML Algorithms

- **Overfitting:**
- Enough training is performed so that the algorithm generalizes well.
- There is at least as much danger in over-training as there is in under-training.
- If algorithm is trained for too long, then it will overfit the data.
- It means that algorithm has learnt about the noise and inaccuracies in the data as well as the actual function.

Testing ML Algorithms

- **Overfitting:**
- Effect of overfitting is that rather than finding the generating function, the neural network matches the inputs perfectly, including the noise in them.
- This reduces the generalization capabilities of the network.



Testing ML Algorithms

- **Overfitting:**
- On the left of the figure the curve fits the overall trend of the data well, but the training error would still not be that close to zero since it passes near, but not through, the training data.
- As the network continues to learn, it will produce a complex model that has a lower training error (close to zero).
- Meaning that it has memorized the training examples, including any noise component of them, so that is has overfitted the training data.

Testing ML Algorithms

- **Overfitting:**
- Stop the learning process before the algorithm overfits.
- Need to know how well the algorithm is generalizing at each time step.
- Can’t use the training data for this, because we wouldn’t detect overfitting, but we can’t use the testing data either, because we’re saving that for the final tests.

Testing ML Algorithms

- **Overfitting:**
- Need a third set of data to use for this purpose called as the validation set.
- This set is used to validate the learning.
- It is part of model selection: choosing the right parameters for the model so that it generalizes as well as possible.

Testing ML Algorithms

- **Training, Testing, and Validation Sets:**
- 3 sets of data:
- The **training set** to actually train the algorithm.
- The **validation set** to keep track of how well the algorithm is doing as it learns.
- The **test set** to produce the final results.
- Supervised learning need target values.
- For unsupervised learning, the validation & test sets need targets for comparison of results.
- It is not always easy to get accurate labels.

Testing ML Algorithms

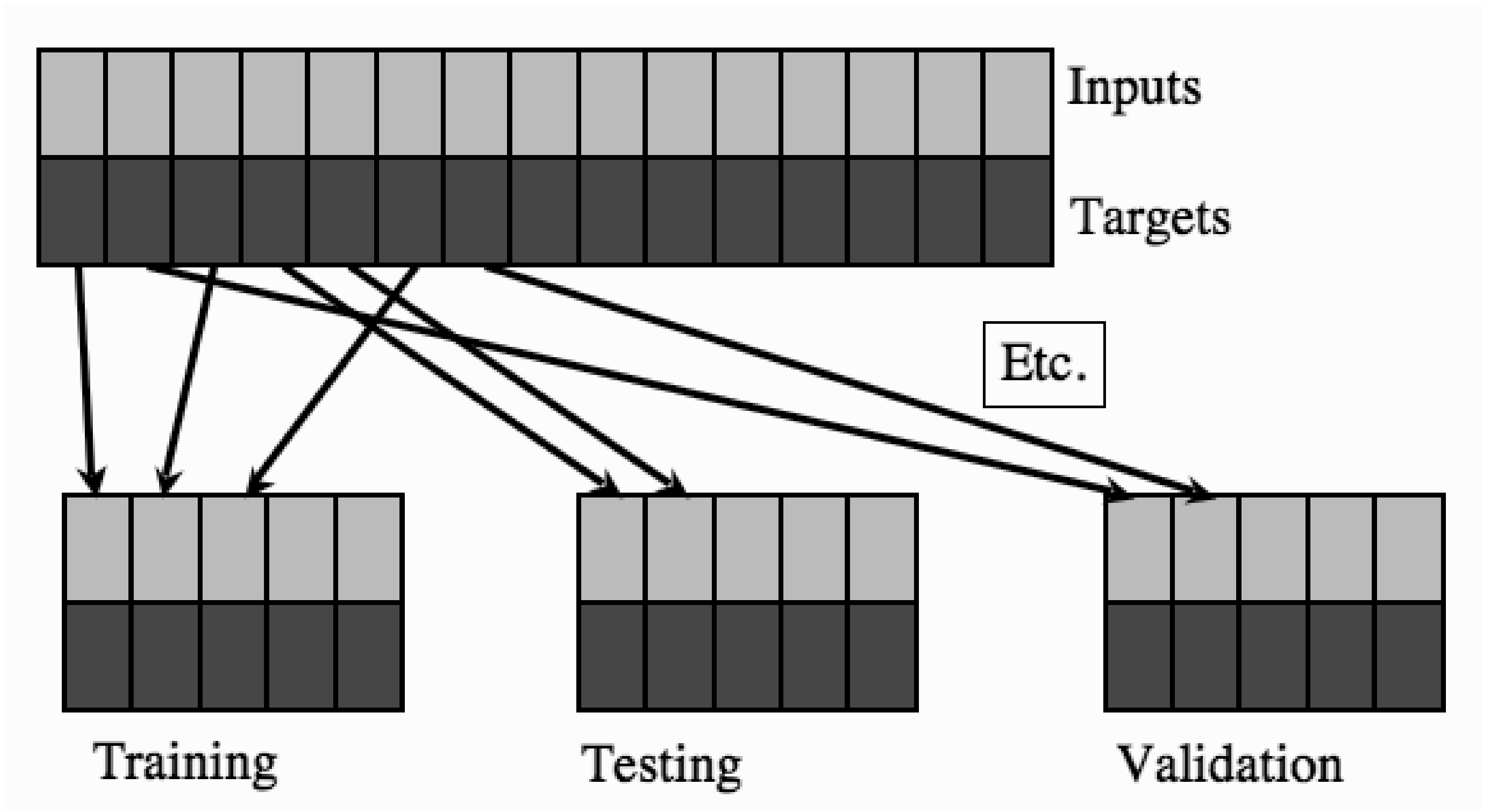
- **Training, Testing, and Validation Sets:**
- More data the algorithm sees, the more likely it is to have seen examples of each possible type of input.
- But more data also increases the computational time to learn.
- Generally, the exact proportion of training to testing to validation data is up to you.
- But it is typical to do something like 50:25:25 if you have plenty of data, and 60:20:20 if you don’t.

Testing ML Algorithms

- **Training, Testing, and Validation Sets:**
- How splitting is done also matter.
- Many datasets have first set of data points being in class 1, the next in class 2, and so on.
- If we pick the first few points to be the training set, the next the test set, etc., then the results are going to be pretty bad, since the training did not see all the classes.
- Randomly reordering the data first, or by assigning each data point randomly to one of the sets.

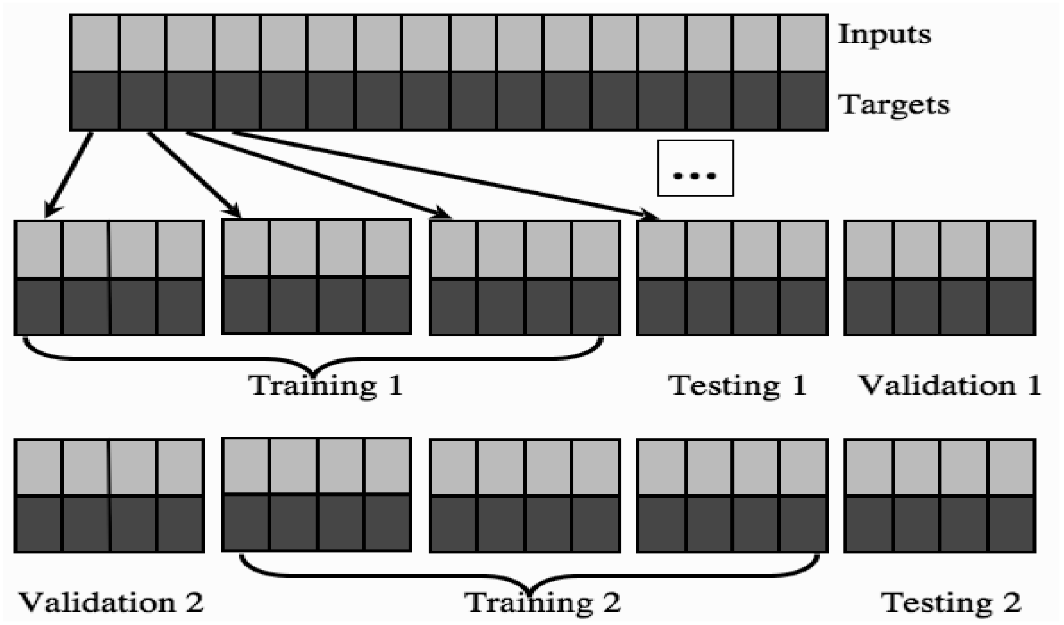
Testing ML Algorithms

- **Training, Testing, and Validation Sets:**



Testing ML Algorithms

- **Training, Testing, and Validation Sets:**
- If training data is less, and has a separate validation set there is a worry that the algorithm won't be sufficiently trained.
- Leave-some-out, multi-fold cross-validation.



Testing ML Algorithms

- **Training, Testing, and Validation Sets:**
- The dataset is randomly partitioned into K subsets, and one subset is used as a validation set, while the algorithm is trained on all of the others.
- A different subset is then left out and a new model is trained on that subset, repeating the same process for all of the different subsets.
- Finally, the model that produced the lowest validation error is tested and used.

Testing ML Algorithms

- **Training, Testing, and Validation Sets:**
- We’ve traded off data for computation time, since we’ve had to train K different models instead of just one.
- In the most extreme case of this there is leave-one-out cross-validation, where the algorithm is validated on just one piece of data, training on all of the rest.

Testing ML Algorithms

- **The Confusion Matrix:**
- Regardless of how much data we use to test the trained algorithm, we still need to work out whether or not the result is good.
- A method that is suitable for classification problems, known as the **confusion matrix**.
- For regression problems things are more complicated because the results are continuous, and so the most common thing to use is the **sum-of-squares error** that we will use to drive the training.

Testing ML Algorithms

- **The Confusion Matrix:**
- The confusion matrix is a nice simple idea.
- Make a square matrix that contains all the possible classes in both the horizontal and vertical directions.
- List the classes along the top of a table as the predicted outputs, and then down the left-hand side as the targets.

Testing ML Algorithms

- **The Confusion Matrix:**
- So for example, the element of the matrix at (i, j) tells us how many input patterns were put into class i in the targets, but class j by the algorithm.
- Anything on the leading diagonal is a correct answer.

	Outputs		
	C_1	C_2	C_3
C_1	5	1	0
C_2	1	4	1
C_3	2	0	4

Testing ML Algorithms

- **The Confusion Matrix:**
- For a small number of classes this is a nice way to look at the outputs.
- If you just want one number, then it is possible to divide the sum of the elements on the leading diagonal by the sum of all of the elements in the matrix, which gives the fraction of correct responses.
- This is known as the accuracy.

Testing ML Algorithms

- **Accuracy Metrics:**
- More analyses of the results can be done apart from accuracy.
- If there are 2 class labels then
  - True Positive is an observation correctly put into class 1.
  - False Positive is an observation incorrectly put into class 1.
  - Negative Examples both true and false are those put into class 2.

Testing ML Algorithms

- **Accuracy Metrics:**
- The entries on the leading diagonal are correct and those off the diagonal are wrong.
- Just like confusion matrix.
- This chart and the concepts of false positives, etc., are based on binary classification.

	PREDICTED		
ACTUAL		**Positive**	**Negative**
	Positive	**TP**	**FN**
	Negative	**FP**	**TN**

Testing ML Algorithms

- **Accuracy Metrics:**
- More analyses of the results can be done apart from accuracy.
- Accuracy is defined as the sum of the number of True Positives and True Negatives divided by the total number of examples.

$$\text{Accuracy} = \frac{\#TP + \#TN}{\#TP + \#FP + \#TN + \#FN}$$

Testing ML Algorithms

- **Accuracy Metrics:**
- There are two complementary pairs of measurements that can help us to interpret the performance of a classifier, namely sensitivity and specificity, and precision and recall.

$$\text{Sensitivity} = \frac{\#TP}{\#TP + \#FN}$$

$$\text{Specificity} = \frac{\#TN}{\#TN + \#FP}$$

$$\text{Precision} = \frac{\#TP}{\#TP + \#FP}$$

$$\text{Recall} = \frac{\#TP}{\#TP + \#FN}$$

Testing ML Algorithms

- **Accuracy Metrics:**
- **Sensitivity** (true positive rate) is the ratio of the number of correct positive examples to the number classified as positive.
- **Specificity** is the same ratio for negative examples.

$$\text{Sensitivity} = \frac{\#TP}{\#TP + \#FN}$$

$$\text{Specificity} = \frac{\#TN}{\#TN + \#FP}$$

Testing ML Algorithms

- **Accuracy Metrics:**
- **Precision** is the ratio of correct positive examples to the number of actual positive examples.
- **Recall** is the ratio of the number of correct positive examples out of those that were classified as positive, which is the same as sensitivity.

$$\text{Precision} = \frac{\#TP}{\#TP + \#FP}$$

$$\text{Recall} = \frac{\#TP}{\#TP + \#FN}$$

Testing ML Algorithms

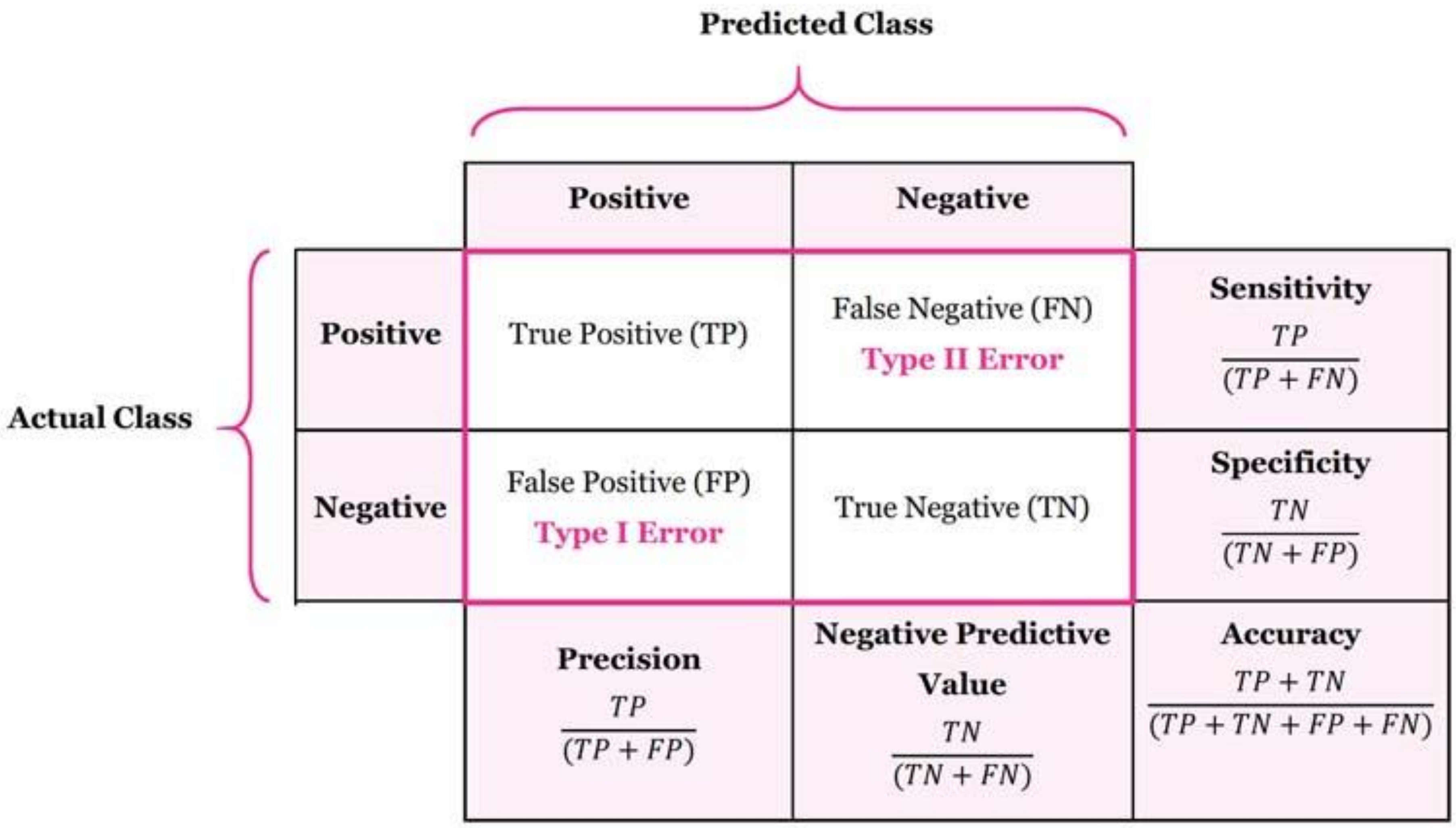
- **Accuracy Metrics:**
- **F1 Measure** is written in terms of precision and recall as.

$$F_1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

$$F_1 = \frac{\#TP}{\#TP + (\#FN + \#FP)/2}.$$

Testing ML Algorithms

- **Accuracy Metrics:**



		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	**Negative Predictive Value** $\frac{TN}{(TN + FN)}$	**Accuracy** $\frac{TP + TN}{(TP + TN + FP + FN)}$

Testing ML Algorithms

- **Accuracy Metrics:**
- **Example:**

n = 165	Prediction = Yes	Prediction = No
Actual = Yes	100	5
Actual = No	10	50

$$\text{Accuracy} = \frac{\#TP + \#TN}{\#TP + \#FP + \#TN + \#FN}.$$

$$\text{Precision} = \frac{\#TP}{\#TP + \#FP}$$

$$\text{Recall} = \frac{\#TP}{\#TP + \#FN}$$

$$F_1 = \frac{\#TP}{\#TP + (\#FN + \#FP)/2}.$$

Testing ML Algorithms

- **Accuracy Metrics:**
- **Example:**

n = 165	Prediction = Yes	Prediction = No
Actual = Yes	100	5
Actual = No	10	50

- **Accuracy: (100+50) / 165 = 0.91**
- **Precision: 100 / (100+10) = 0.90**
- **Recall: 100 / (100+5) = 0.95**
- **F1: 100 /(100 + (10 + 5) / 2) = 0.93**

Testing ML Algorithms

- Q: Suppose 10,000 patients get tested for flu; out of them, 9,000 are actually healthy and 1,000 are actually sick. For the sick people, a test was positive for 620 and negative for 380. For the healthy people, the same test was positive for 180 and negative for 8,820.

- Create a Confusion Matrix.
- **Calculate Accuracy, Precision, Recall and F1 Score.**

Testing ML Algorithms

- **Accuracy Metrics:**
- Q: Suppose 10,000 patients get tested for flu; out of them, 9,000 are actually healthy and 1,000 are actually sick. For the sick people, a test was positive for 620 and negative for 380. For the healthy people, the same test was positive for 180 and negative for 8,820.

n = 10000	Prediction = Yes	Prediction = No
Actual = Yes	**620**	**380**
Actual = No	**180**	**8820**

Testing ML Algorithms

- **Accuracy: (620+8820)/10000 = 0.944**
- **Precision: 620/(620+180) = 0.775**
- **Recall: 620/(620+380) = 0.62**
- **F1: 620 / (620 + (380+180)/2) = 0.68**

n = 10000	Prediction = Yes	Prediction = No
Actual = Yes	**620**	**380**
Actual = No	**180**	**8820**

Testing ML Algorithms

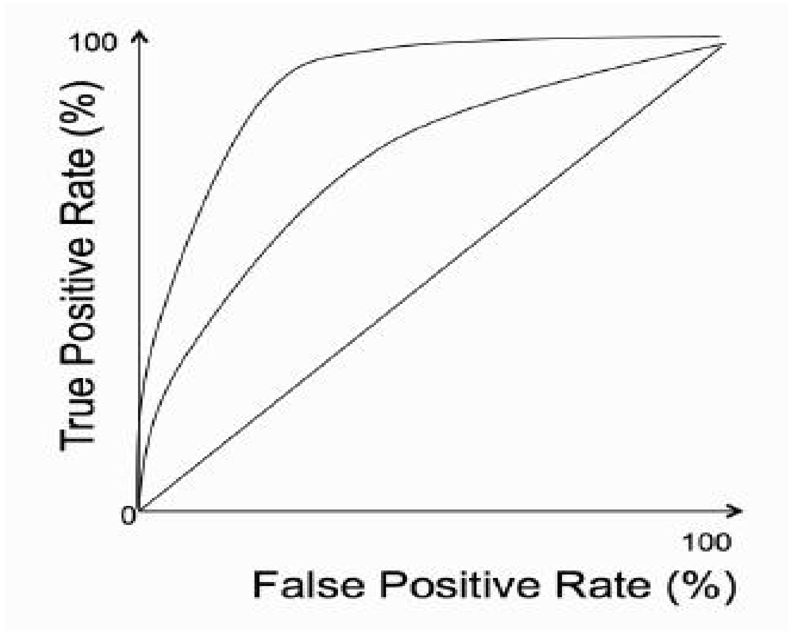
- **Receiver Operator Characteristic (ROC) Curve:**
- An ROC curve is a graph showing the performance of a classification model at all classification thresholds.
- This curve plots two parameters:
- **True Positive Rate** $TPR = \frac{TP}{TP + FN}$
- **False Positive Rate.** $FPR = \frac{FP}{FP + TN}$

Testing ML Algorithms

- **Receiver Operator Characteristic (ROC) Curve:**
- Receiver Operator Characteristic curve
  - Compare different classifiers
  - Same classifier with different learning parameters.
- A Plot of the percentage of True Positives on the y axis against False Positives on the x axis.

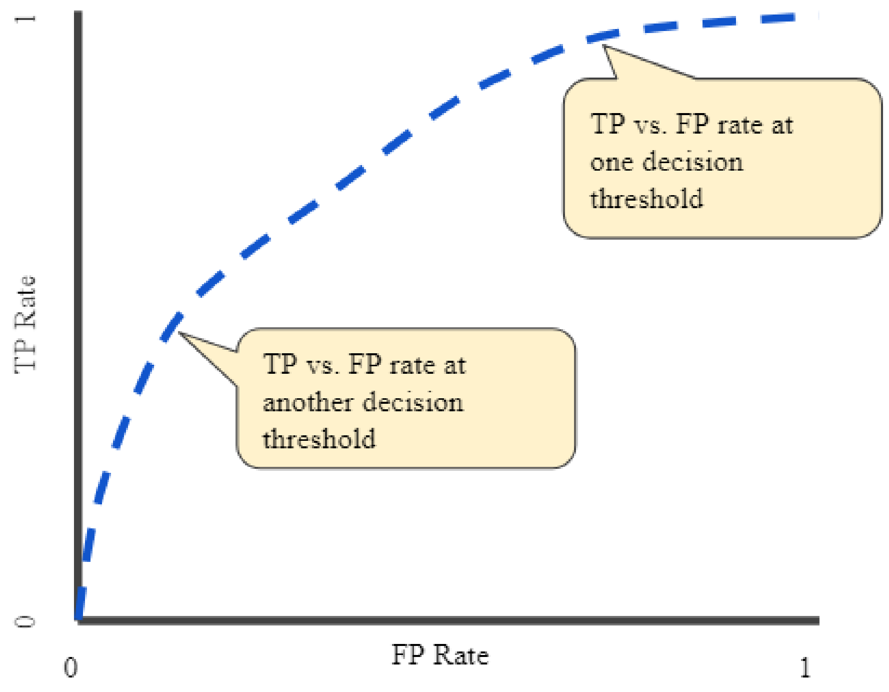
Testing ML Algorithms

- **Receiver Operator Characteristic (ROC) Curve:**
- The diagonal line represents exactly chance, so anything above the line is better than chance, and the further from the line, the better



Testing ML Algorithms

- **ROC Curve:**
- An ROC curve plots TPR vs. FPR at different classification thresholds.
- Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.



Testing ML Algorithms

- **ROC Curve:**
- A single run of a classifier produces a single point on the ROC plot, and a perfect classifier would be a point at (0, 1) (100% true positives, 0% false positives).
- The anti-classifier that got everything wrong would be at (1,0).
- Closer to the top-left-hand corner the result of a classifier is, the better the classifier has performed.

Testing ML Algorithms

- **ROC Curve:**
- Any classifier that sits on the diagonal line from (0,0) to (1,1) behaves exactly at the chance level.
- To compare classifiers, or choices of parameters settings for the same classifier, compute the point that is furthest from the 'chance' line along the diagonal.
- Compute the area under the curve (AUC) instead.

Testing ML Algorithms

- **ROC Curve:**
- If you only have one point for each classifier, the curve is the trapezoid that runs from (0,0) up to the point and then from there to (1,1).
- If there are more points (based on more runs of the classifier, such as trained and/or tested on different datasets), then they are just included in order along the diagonal line.
- The key to getting a curve rather than a point on the ROC curve is to use cross validation.

Testing ML Algorithms

- **ROC Curve:**
- If you use 10-fold cross-validation, then you have 10 classifiers, with 10 different test sets.
- By producing an ROC curve for each classifier it is possible to compare their results.

Testing ML Algorithms

- **Unbalanced Datasets:**
- For the accuracy, we have implicitly assumed that there are the same number of positive and negative examples in the dataset (balanced dataset).
- Can compute the balanced accuracy as the sum of sensitivity and specificity divided by 2.
- However, a more correct measure is Matthew's Correlation Coefficient, which is computed as.

$$MCC = \frac{\#TP \times \#TN - \#FP \times \#FN}{\sqrt{(\#TP + \#FP)(\#TP + \#FN)(\#TN + \#FP)(\#TN + \#FN)}}$$

Testing ML Algorithms

- **Unbalanced Datasets:**
- If any of the brackets in the denominator are 0, then the whole of the denominator is set to 1.
- This provides a balanced accuracy computation.
- If there are more than two classes then the calculations get a little more complicated, since instead of one set of false positives and one set of false negatives, you have some for each class.

Testing ML Algorithms

- **Unbalanced Datasets:**
- In this case, specificity and recall are not the same.
- However, it is possible to create a set of results, where you use one class as the positives and everything else as the negatives, and repeat this for each of the different classes.

Testing ML Algorithms

- **Measurement Precision:**
- Different way to evaluate the accuracy of a learning system, which also uses precision.
- The concept here is to treat the machine learning algorithm as a measurement system.
- We feed in inputs and look at the outputs that we get.
- Even before comparing them to the target values, we can measure something about the algorithm: if we feed in a set of similar inputs, then we would expect to get similar outputs for them.

Testing ML Algorithms

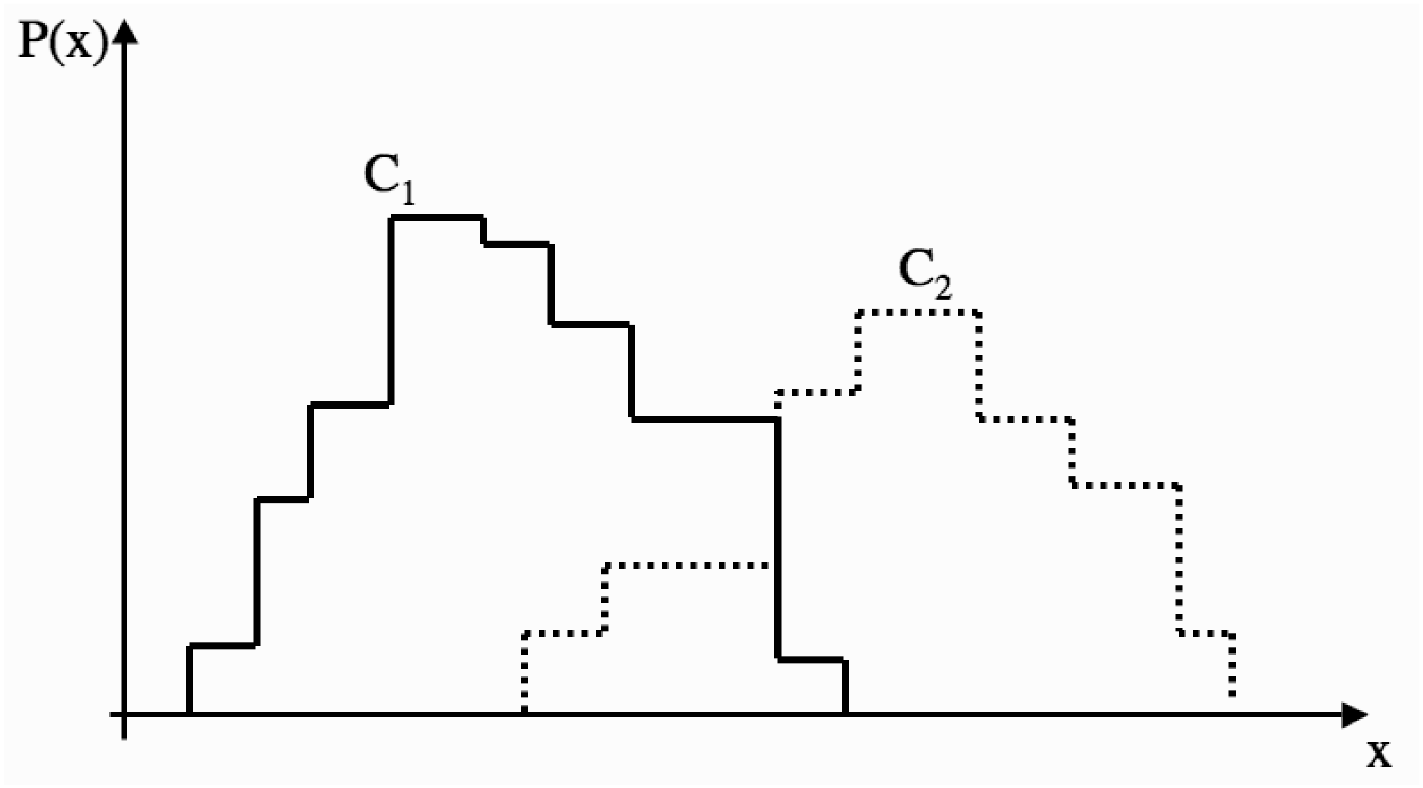
- **Measurement Precision:**
- This measure of the variability of the algorithm is also known as precision, and it tells us how repeatable the predictions that the algorithm makes are.
- It might be useful to think of precision as being something like the variance of a probability distribution: it tells you how much spread around the mean to expect.
- If an algorithm is precise it does not mean that it is accurate.

Testing ML Algorithms

- **Measurement Precision:**
- It can be precisely wrong if it always gives the wrong prediction.
- One measure of how well the algorithm’s predictions match reality is known as trueness, and it can be defined as the average distance between the correct output and the prediction.
- Trueness doesn’t usually make much sense for classification problems unless there is some concept of certain classes being similar to each other.

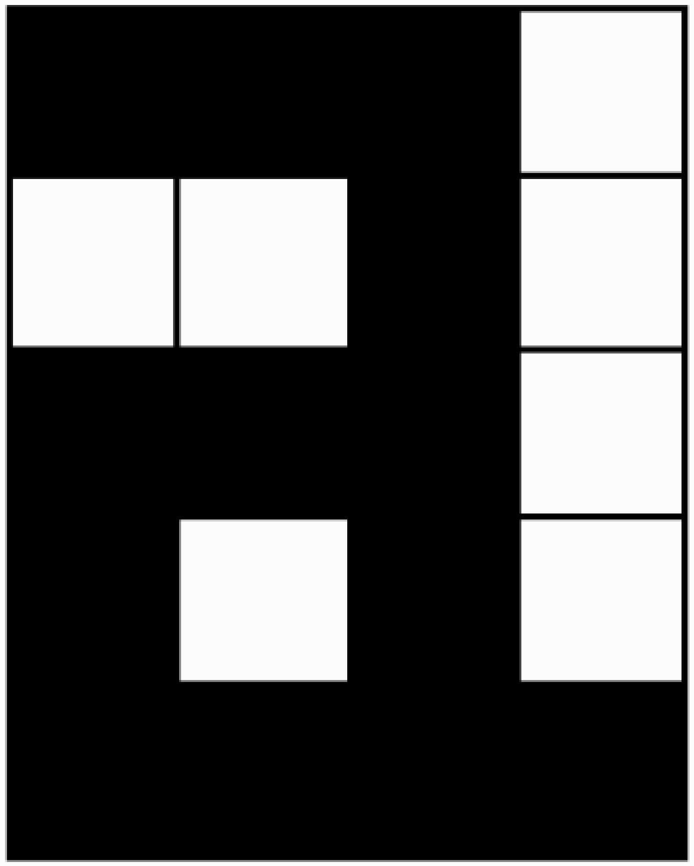
Turning Data Into Probabilities

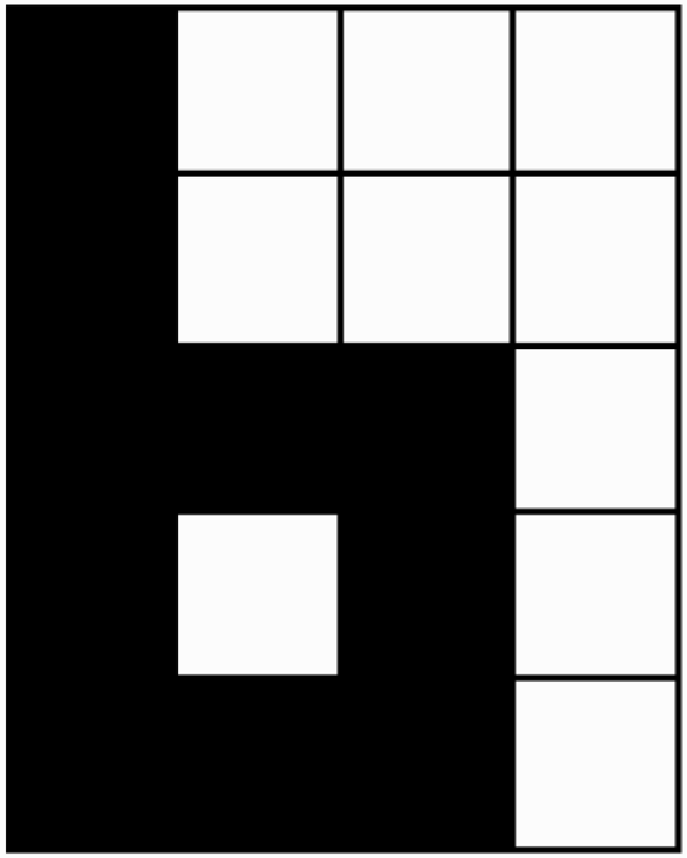
- This plot shows the measurements of some feature x for 2 classes, C1 and C2.
- Easy to predict correct class at the extremes of the range, but what to do in the middle is unclear.



Turning Data Into Probabilities

- Classify writing of the letters ‘a’ and ‘b’ based on their height.





Turning Data Into Probabilities

- The letter ‘a’ is much more common than the letter ‘b’.
- If we see a letter that is either an ‘a’ or a ‘b’ in normal writing, then there is a 75% chance that it is an ‘a.’
- We are using prior knowledge to estimate the probability that the letter is an ‘a’: in this example, P(C1) = 0.75, P(C2) = 0.25.
- P(C1): Count of the total Number of C1 divide by the total number of examples.

Turning Data Into Probabilities

- Joint Probability P(A, B): The probability of event A and event B occurring.
- It is the probability of the intersection of two or more events P(A ∩ B).
- Ex: The probability that a card is a four and red = P(four and red) = 2/52=1/26.
- Conditional Probability P(A|B): It is the probability of event A occurring, given that event B occurs.
- Example: While drawing a red card, what's the probability that it's a four (P(four|red))=2/26=1/13.

Turning Data Into Probabilities

- Now we know Joint Probability P(Ci,Xj) and the conditional probability P(Xj |Ci).
- Since we want to compute P(Ci|Xj), we need to know how to link these things together.
- The answer is Bayes' rule:
- There is a link between the joint probability and the conditional probability.

$$P(C_i, X_j) = P(X_j|C_i)P(C_i),$$

OR

$$P(C_i, X_j) = P(C_i|X_j)P(X_j).$$

Turning Data Into Probabilities

- Clearly, the right-hand side of these two equations must be equal to each other, since they are both equal to P(Ci,Xj), and so with one division we can write:

$$P(C_i|X_j) = \frac{P(X_j|C_i)P(C_i)}{P(X_j)}.$$

- This is Bayes' rule.
- It relates the posterior probability P(Ci|Xj) with the prior probability P(Ci) and class-conditional probability P(Xj |Ci).
- The denominator acts to normalize everything, so that all the probabilities sum to 1.

Turning Data Into Probabilities

- However, if we notice that any observation Xk has to belong to some class Ci, then we can marginalize over the classes to compute:

$$P(X_k) = \sum_i P(X_k|C_i)P(C_i)$$

- Bayes' rule is important that it lets us obtain the posterior probability by calculating things that are much easier to compute.
- Use the posterior probability to assign each new observation to one of the classes by picking the class Ci, where:

$$P(C_i|\mathbf{x}) > P(C_j|\mathbf{x}) \quad \forall \; i \neq j,$$

Turning Data Into Probabilities

- Here, **x** is a vector of feature values instead of just one feature.
- This is known as the **maximum a posteriori or MAP** hypothesis, and it gives us a way to choose which class to choose as the output one.
- If there are 3 possible output classes, and for a particular input the posterior probabilities of the classes are **P(C1|x) = 0.35**, **P(C2|x) = 0.45**, **P(C3|x)=0.2**.
- The MAP hypothesis tells us that this input is in class C2, because that is the class with the highest posterior probability.

Turning Data Into Probabilities

- Based on the class that the data is in, we want to take some decision.
- If the class is C1 or C3 then we do action 1, and if the class is C2 then we do action 2.
- Ex: Suppose that the inputs are the results of a blood test, the three classes are different possible diseases, and the output is whether or not to treat with a particular antibiotic.
- The MAP method has told us that the output is C2, and so we will not treat the disease.

Turning Data Into Probabilities

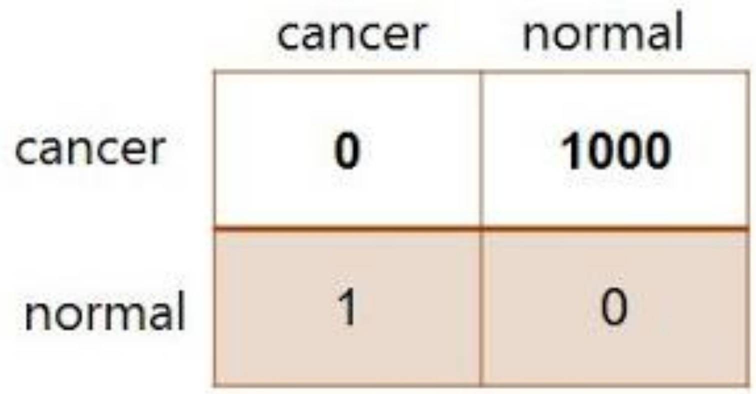
- But what is the probability that it does not belong to class C2, and so should have been treated with the antibiotic?
- It is $1 - P(C2) = 0.55$.
- The MAP prediction seems to be wrong: we should treat with antibiotic, as it is more likely.
- Here we take into account the final outcomes of all of the classes and it is called the **Bayes' Optimal Classification**.
- It minimizes the probability of misclassification, rather than maximizing the posterior probability.

Turning Data Into Probabilities

- **Minimizing Risk:**
- Classify based on minimizing the probability of misclassification.
- We can also consider the risk that is involved in the misclassification.
- The risk from misclassifying someone as unhealthy when they are healthy is usually smaller than the other way around, but not necessarily always.
- Create a loss matrix that specifies the risk involved in classifying an example of class Ci as class Cj.

Turning Data Into Probabilities

- **Minimizing Risk:**
- It looks like the confusion matrix except that a loss matrix always contains zeros on the leading diagonal since there should never be a loss from getting the classification correct.
- Once we have the loss matrix, we just extend our classifier to minimize risk by multiplying each case by the relevant loss number.



	cancer	normal
cancer	**0**	**1000**
normal	1	0

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem.
- It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.
- Using Bayes theorem, we can find the probability of A happening, given that B has occurred.
- Here, B is the evidence and A is the hypothesis.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- We classify whether the day is suitable for playing golf, given the features of the day.
- The columns represent these features and the rows represent individual entries.
- Bayes theorem can be rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

- The variable y is the class variable(play golf), which represents if it is suitable to play golf or not.
- Variable X represent the parameters/features.

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- X is given as,

$$X = (x_1, x_2, x_3, ..., x_n)$$

- Here x1,x2….xn represent the features, i.e they can be mapped to outlook, temperature, humidity and windy.
- By substituting for X and expanding using the chain rule we get,.

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)
- X = (Rainy, Hot, High, False)
- y = No.
- So basically, P(y|X) here means, the probability of "Not playing golf" given that the weather conditions are "Rainy outlook", "Temperature is hot", "high humidity" and "no wind".

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- Now, its time to put a naive assumption to the Bayes' theorem, which is, independence among the features.
- So now, we split evidence into the independent parts.
- Now, if any two events A and B are independent, then,
- P(A,B) = P(A)P(B)

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- Hence, we reach to the result:

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

- which can be expressed as:

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$

- Now, as the denominator remains constant for a given input, we can remove that term:.

$$P(y|x_1, ..., x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- Now, we need to create a classifier model.
- For this, we find the probability of given set of inputs for all possible values of the class variable y and pick up the output with maximum probability.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

- P(y): Class Probability
- P(xi | y): Conditional Probability.

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		**P(Yes)/P(No)**
Yes	9	9/14
No	5	5/14
Total	14	100%

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- We have calculated P(xi | yj) for each xi in X and yj in y manually in the tables 1-4.
- For example, probability of playing golf given that the temperature is cool, i.e P(temp. = cool | play golf = Yes) = 3/9.
- Also, we need to find class probabilities (P(y)) which has been calculated in the table 5.
- For example, P(play golf = Yes) = 9/14.

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- Let us test it on a new set of features (let us call it today):
- today = (Sunny, Hot, Normal, False).
- So, probability of playing golf is given by:

$$P(Yes|today) = \frac{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$

- And probability to not play golf is given by:

$$P(No|today) = \frac{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}{P(today)}$$

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:

$$P(Yes|today) \propto \frac{2}{9} \cdot \frac{2}{9} \cdot \frac{6}{9} \cdot \frac{6}{9} \cdot \frac{9}{14} \approx 0.0141$$

$$P(No|today) \propto \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} \cdot \frac{5}{14} \approx 0.0068$$

- Now, since
- P(Yes | today) + P(No | today) = 1

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- These numbers can be converted into a probability by making the sum equal to 1 (normalization):

$$P(Yes|today) = \frac{0.0141}{0.0141+0.0068} = 0.67$$

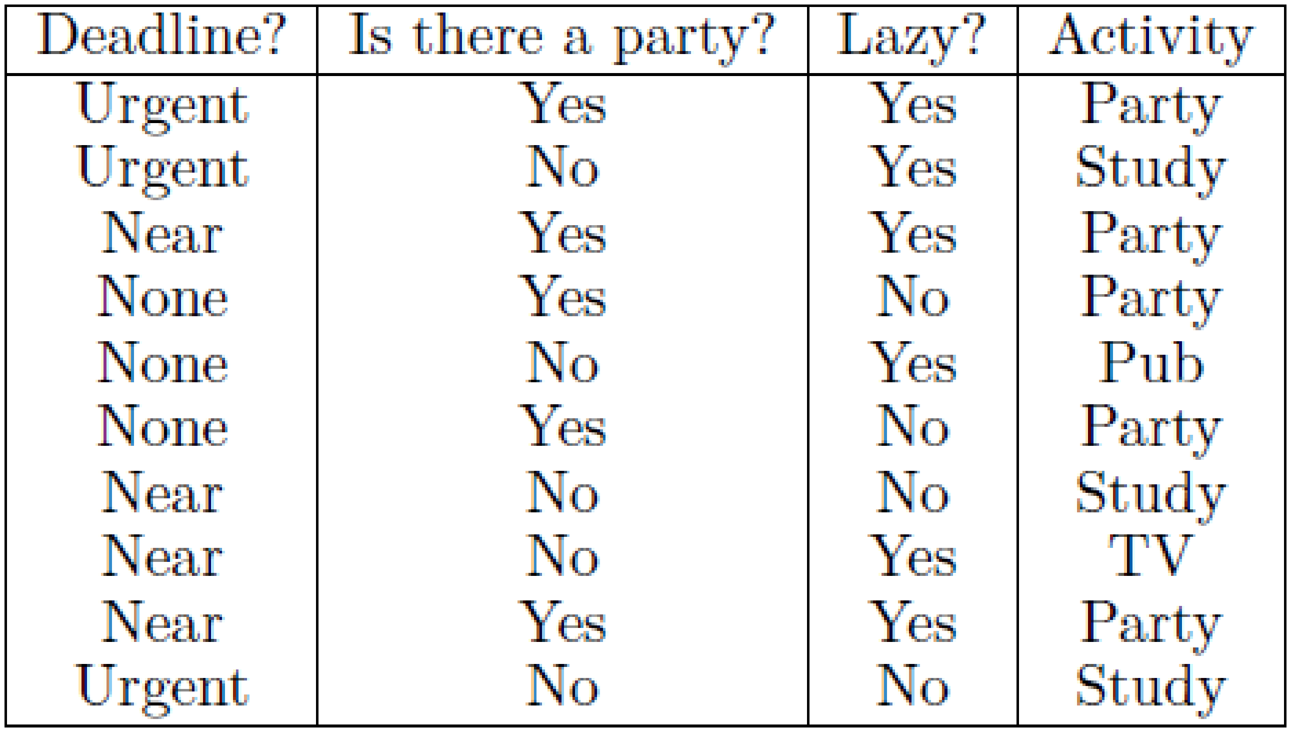
$$P(No|today) = \frac{0.0068}{0.0141+0.0068} = 0.33$$

$$P(Yes|today) > P(No|today)$$

- So, prediction that golf would be played is 'Yes'.

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- What to do in the evening based on whether you have an assignment deadline and what is happening.
- The data, shown below, consists of a set of prior examples from the last few days.



Deadline?	Is there a party?	Lazy?	Activity
Urgent	Yes	Yes	Party
Urgent	No	Yes	Study
Near	Yes	Yes	Party
None	Yes	No	Party
None	No	Yes	Pub
None	Yes	No	Party
Near	No	No	Study
Near	No	Yes	TV
Near	Yes	Yes	Party
Urgent	No	No	Study

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- Feed in the current values for the feature variables (deadline, whether there is a party, etc.) and ask the classifier to compute the probabilities of each of the four possible things that you might do in the evening based on the data in the training set.
- Then pick the most likely class.
- Note that the probabilities will be very small.
- Since we are multiplying lots of probabilities, which are all less than one, the numbers get very small.

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**
- Suppose that you have deadlines looming, but none of them are particularly urgent, that there is no party on, and that you are currently lazy.
- Then the classifier needs to evaluate:

- $\mathrm{P(Party)} \times \mathrm{P(Near} \mid \mathrm{Party)} \times \mathrm{P(No\ Party} \mid \mathrm{Party)} \times \mathrm{P(Lazy} \mid \mathrm{Party)}$
- $\mathrm{P(Study)} \times \mathrm{P(Near} \mid \mathrm{Study)} \times \mathrm{P(No\ Party} \mid \mathrm{Study)} \times \mathrm{P(Lazy} \mid \mathrm{Study)}$
- $\mathrm{P(Pub)} \times \mathrm{P(Near} \mid \mathrm{Pub)} \times \mathrm{P(No\ Party} \mid \mathrm{Pub)} \times \mathrm{P(Lazy} \mid \mathrm{Pub)}$
- $\mathrm{P(TV)} \times \mathrm{P(Near} \mid \mathrm{TV)} \times \mathrm{P(No\ Party} \mid \mathrm{TV)} \times \mathrm{P(Lazy} \mid \mathrm{TV)}$

Turning Data Into Probabilities

- **The Naïve Bayes' Classifier:**

$$\begin{aligned}
P(\text{Party}|\text{near (not urgent) deadline, no party, lazy}) &= \frac{5}{10} \times \frac{2}{5} \times \frac{0}{5} \times \frac{3}{5} \\
&= 0 \\
P(\text{Study}|\text{near (not urgent) deadline, no party, lazy}) &= \frac{3}{10} \times \frac{1}{3} \times \frac{3}{3} \times \frac{1}{3} \\
&= \frac{1}{30} \\
P(\text{Pub}|\text{near (not urgent) deadline, no party, lazy}) &= \frac{1}{10} \times \frac{0}{1} \times \frac{1}{1} \times \frac{1}{1} \\
&= 0 \\
P(\text{TV}|\text{near (not urgent) deadline, no party, lazy}) &= \frac{1}{10} \times \frac{1}{1} \times \frac{1}{1} \times \frac{1}{1} \\
&= \frac{1}{10}
\end{aligned}$$

- So based on this you will be watching TV tonight.

Some Basic Statistics

- **Averages:**
- Two numbers that can be used to characterize a dataset are the **Mean** and the **Variance**.
- The mean is easy, it is the most commonly used average of a set of data, and is the value that is found by adding up all the points in the dataset and dividing by the number of points.

$$\bar{x} = \frac{\sum x}{N}$$

Here,

$\sum$ represents the summation

X represents observations

N represents the number of observations

Some Basic Statistics

- **Averages:**
- There are two other averages that are used: the **Median** and the **Mode**.
- The **median** is the middle value, so the most common way to find it is to sort the dataset and then find the point that is in the middle.
- The **mode** is the most common value, so it just requires counting how many times each element appears and picking the most frequent one.

Some Basic Statistics

- **Averages:**
- **The Median**
- If the total number of observations (n) is an odd number, then the formula is given below:

$$Median = \left(\frac{n+1}{2}\right)^{th} observation$$

- If the total number of the observations (n) is an even number, then the formula is given below:

$$Median = \frac{\left(\frac{n}{2}\right)^{th} observation + \left(\frac{n}{2}+1\right)^{th} observation}{2}$$

Some Basic Statistics

- **Variance and Covariance :**
- The variance of the set of numbers is a measure of how spread out the values are.
- It is computed as the sum of the squared distances between each element in the set and the expected value of the set (the mean, μ).

$$\mathrm{var}(\{\mathbf{x}_i\}) = \sigma^2(\{\mathbf{x}_i\}) = E((\{\mathbf{x}_i\} - \mu)^2) = \sum_{i=1}^{N} (\mathbf{x}_i - \mu)^2$$

- The square root of the variance, σ, is known as the standard deviation.
- The variance looks at the variation in one variable compared to its mean.

Some Basic Statistics

- **Variance and Covariance :**
- We can generalize this to look at how two variables vary together, which is known as the covariance.
- It is a measure of how dependent the two variables are (in the statistical sense).
- It is computed by:

$$\operatorname{cov}(\{\mathbf{x}_i\}, \{\mathbf{y}_i\}) = E(\{\mathbf{x}_i\} - \boldsymbol{\mu})E(\{\mathbf{y}_i\} - \boldsymbol{\nu})$$

- where υ is the mean of set {yi}.
- If two variables are independent, then the covariance is 0.

Some Basic Statistics

- **Variance and Covariance :**
- If they both increase and decrease at the same time, then the covariance is positive.
- If one goes up while the other goes down, then the covariance is negative.
- The covariance can be used to look at the correlation between all pairs of variables within a set of data.
- Compute the covariance of each pair to create the Covariance Matrix.

Some Basic Statistics

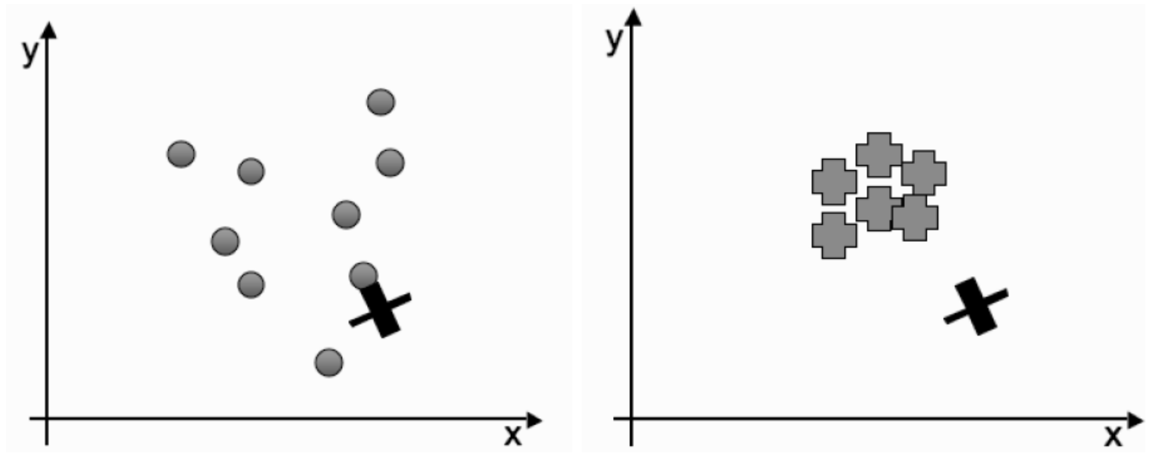
- **Variance and Covariance :**
- Covariance Matrix:

$$\Sigma = \begin{pmatrix} E[(x_1-\mu_1)(x_1-\mu_1)] & E[(x_1-\mu_1)(x_2-\mu_2)] & \dots & E[(x_1-\mu_1)(x_n-\mu_n)] \\ E[(x_2-\mu_2)(x_1-\mu_1)] & E[(x_2-\mu_2)(x_2-\mu_2)] & \dots & E[(x_2-\mu_2)(x_n-\mu_n)] \\ \dots & \dots & \dots & \dots \\ E[(x_n-\mu_n)(x_1-\mu_1)] & E[(x_n-\mu_n)(x_2-\mu_2)] & \dots & E[(x_n-\mu_n)(x_n-\mu_n)] \end{pmatrix}$$

- where xi is a column vector describing the elements of the ith variable, and μi is their mean.
- The covariance matrix is square and that the elements on the leading diagonal of the matrix are equal to the variances, and that it is symmetric since cov(xi, xj) = cov(xj , xi).

Some Basic Statistics

- **Variance and Covariance :**
- Matrix says how the data varies along each data dimension.
- This is useful if we want to think about distances again.
- If we have two datasets and the test point ‘X’. Guess if it is a part of the data.



Some Basic Statistics

- **Variance and Covariance :**
- For the figure on the left you would probably say yes, while for the figure on the right you would say no, even though the two points are the same distance from the centre of the data.
- Considered where the test point lies in relation to the spread of the actual data points.
- If the data is tightly controlled then the test point has to be close to the mean.
- If the data is very spread out, then the distance of the test point from the mean does not matter as much.

Some Basic Statistics

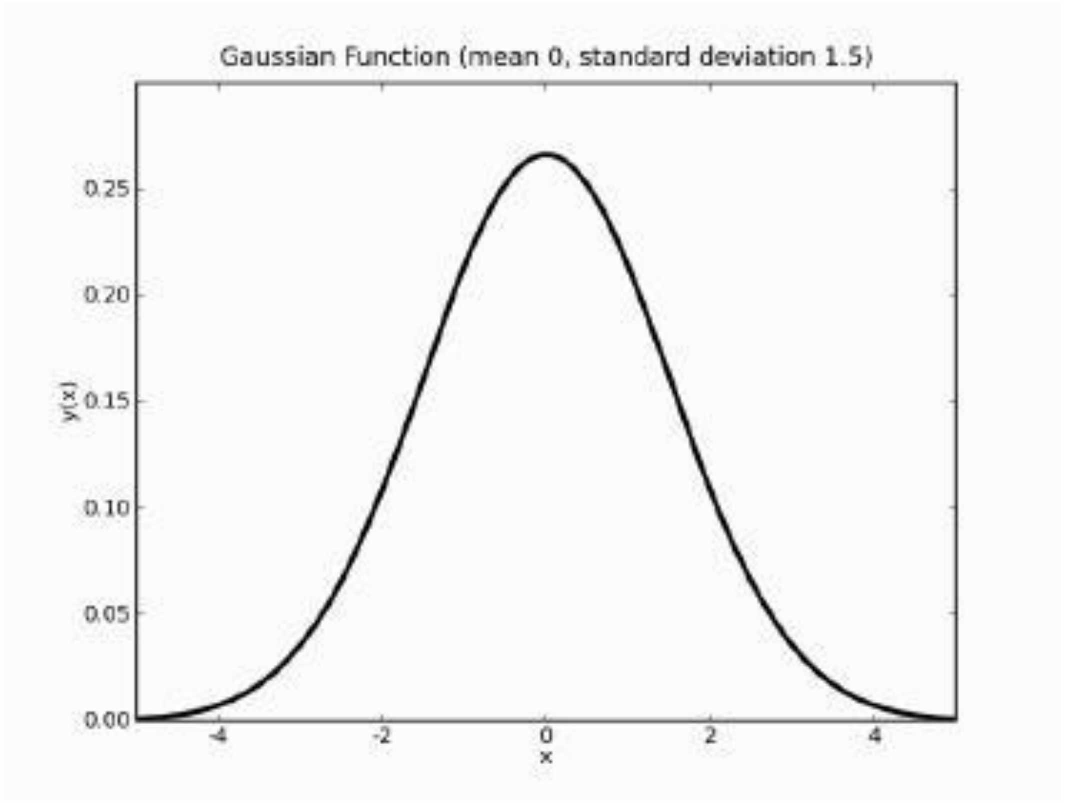
- **Variance and Covariance :**
- We can use this to construct a distance measure called "Mahalanobis distance" that takes this into account.

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$

- where x is the data arranged as a column vector, μ is column vector representing the mean, and Σ^{-1} is the inverse of the covariance matrix.
- If we set the covariance matrix to the identity matrix, then the Mahalanobis distance reduces to the Euclidean distance.

Some Basic Statistics

- **The Gaussian:**
- The probability distribution that is most well known is the Gaussian or normal distribution.
- In one dimension it has the familiar 'bell-shaped'.



Some Basic Statistics

- **The Gaussian:**
- Its equation in one dimension is:

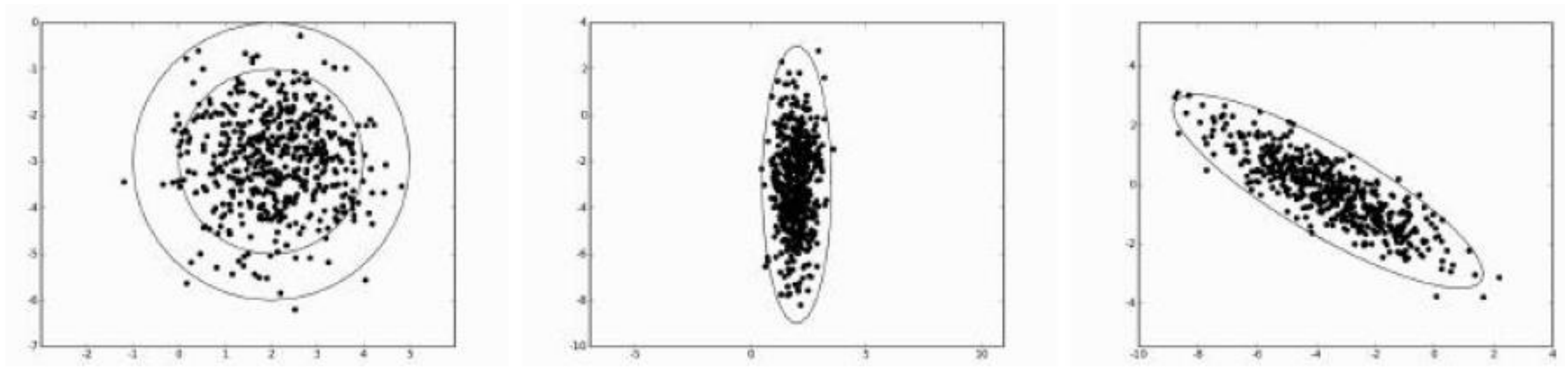
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

- where μ is the mean and σ the standard deviation.
- The Gaussian distribution turns up in many problems because of the Central Limit Theorem, which says that lots of small random numbers will add up to something Gaussian.
- In higher dimensions it looks like:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

Some Basic Statistics

- **The Gaussian:**
- where Σ is the n × n covariance matrix (with | Σ | being its determinant and Σ^{-1} being its inverse).
- The 2-dimensional Gaussian when (left) the covariance matrix is the identity, (centre) the covariance matrix has elements on the leading diagonal only, and (right) the general case.



The Bias-Variance Tradeoff

- During training of any ML algorithm, choices are made about the model to use and parameters of the model.
- The more degrees of freedom the algorithm has, the more complicated the model that can be fitted.
- More complicated models have inherent dangers such as overfitting, need for more training data, need for validation data to ensure that the model does not overfit.
- The bias-variance tradeoff is used to understand that the complex models may not yield better results.

The Bias-Variance Tradeoff

- A model can be bad for two different reasons.
- 1. It is not accurate and doesn't match the data well, or it is not very precise.
- 2. There is a lot of variation in the results.
- The first of these is known as the **Bias**, while the second is the statistical **Variance**.
- More complex classifiers will tend to improve the bias, but the cost of this is higher variance.
- While making the model more specific by reducing the variance will increase the bias.

The Bias-Variance Tradeoff

- Some models are definitely better than others, but choosing the complexity of the model is important for getting good results.
- The most common way to compute the error between the targets and the predicted outputs is to sum up the squares of the difference between the two.
- When looking at this sum-of-squares error function we can split it up into separate pieces that represent the bias and the variance.

The Bias-Variance Tradeoff

- Suppose that the function that we are trying to approximate is y = f(x) + Є, where Є is the noise, which is assumed to be Gaussian with 0 mean and variance σ^2.
- Use machine learning algorithm to fit the following hypothesis (where w is the weight vector) to the data in order to minimize the sum-of-squares error.

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

- To decide whether or not our method is successful we need to consider it on independent data.

The Bias-Variance Tradeoff

- We consider a new input x* and compute the expected value of the sum-of squares error, which we will assume is a random variable.
- Remember that $E[x] = \bar{x}$, the mean value.
- Perform algebraic manipulation where Z is just some random variable

$$\begin{aligned}
E[(Z - \bar{Z})^2] &= E[Z^2 - 2Z\bar{Z} + \bar{Z}^2] \\
&= E[Z^2] - 2E[Z]\bar{Z}] + \bar{Z}^2 \\
&= E[Z^2] - 2\bar{Z}\bar{Z} + \bar{Z}^2 \\
&= E[Z^2] - \bar{Z}^2.
\end{aligned}$$

The Bias-Variance Tradeoff

- Using this, we can compute the expectation of the sum-of-squares error of a new data point:

$$\begin{aligned}
E[(y^* - h(\mathbf{x}^*))^2] &= E[y^{*2} - 2y^* h(\mathbf{x}^*) + h(\mathbf{x}^*)^2] \\
&= E[y^{*2}] - 2E[y^* h(\mathbf{x}^*)] + E[h(\mathbf{x}^*)^2] \\
&= E[(y^{*2} - f(\mathbf{x}^*))^2] + f(\mathbf{x}^*)^2 + E[(h(\mathbf{x}^* - \bar{h}(\mathbf{x}^*))^2] \\
&+ \bar{h}(\mathbf{x}^*)^2 - 2f(\mathbf{x}^*)\bar{h}(\mathbf{x}^*) \\
&= E[(y^{*2} - f(\mathbf{x}^*))^2] + E[(h(\mathbf{x}^*) - \bar{h}(\mathbf{x}^*))^2] + (f(\mathbf{x}^*) + \bar{h}(\mathbf{x}^*))^2 \\
&= \mathrm{noise}^2 + \mathrm{variance} + \mathrm{bias}^2.
\end{aligned}$$

- The first term is the irreducible error and is the variance of the test data.
- The second term is variance, and the third is the square of the bias.

The Bias-Variance Tradeoff

- The variance tells us how much x* changes depending on the particular training set that was used, while the bias tells us about the average error of h(x*).
- It is possible to exchange bias and variance, so that you can have a model with low bias but high variance or vice versa.
- But you can’t make them both zero – for each model there is a tradeoff between them.
- For any particular model and dataset there is some set of parameters that will give the best results for the bias and variance together.

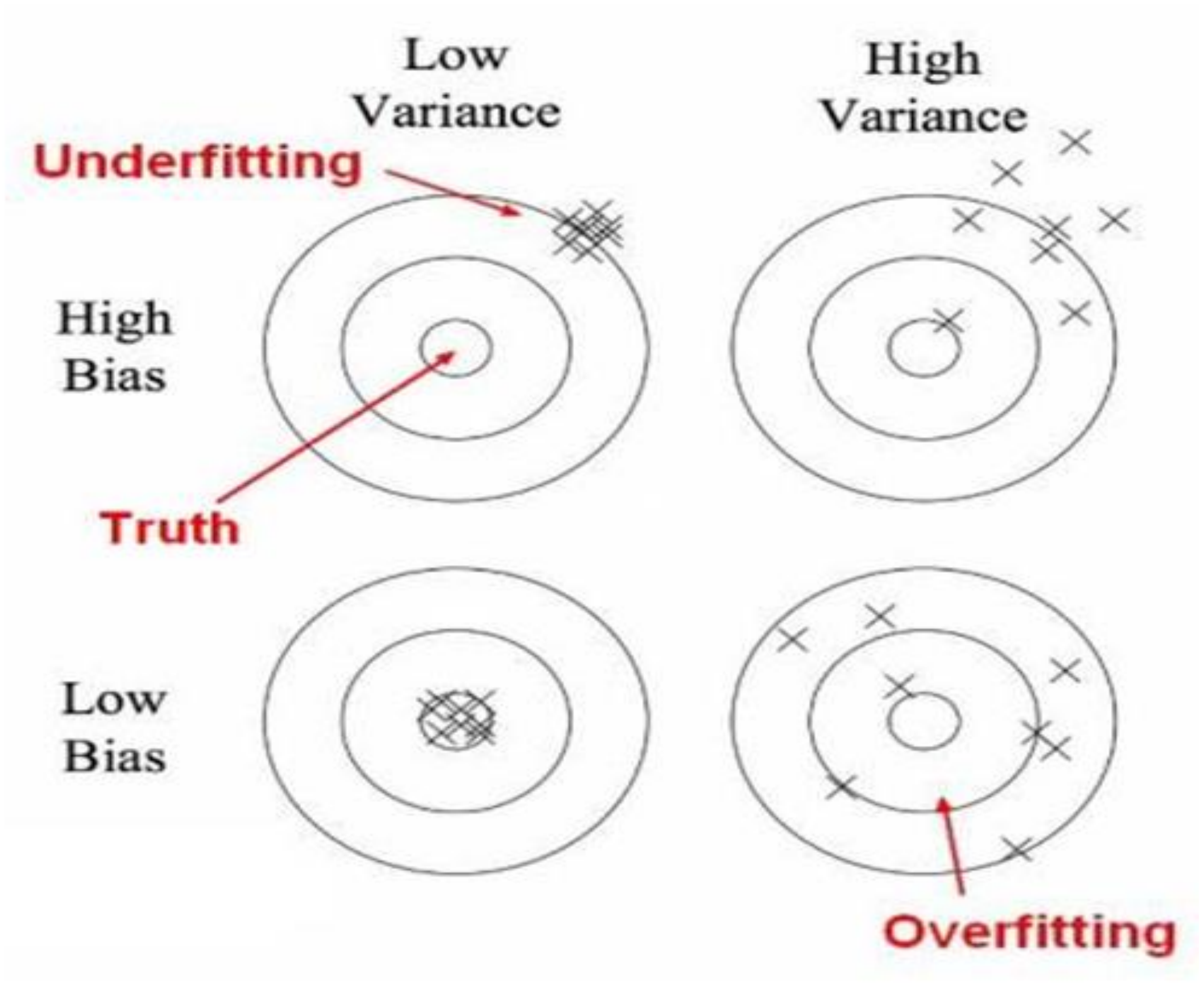
The Bias/Variance Tradeoff

- A model’s generalization error can be expressed as the sum of three very different errors:
- **Bias**
  - Difference between the average prediction of our model and the correct value which we are trying to predict
  - This part of the generalization error is due to wrong assumptions, such as assuming that the data is linear when it is actually quadratic.
  - A high-bias model is most likely to under fit the training data.
  - Model with high bias always leads to high error on training and test data.

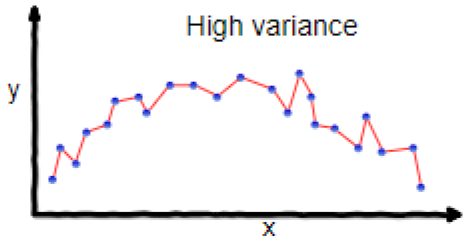
The Bias/Variance Tradeoff

- **Variance**
  - Variance is the amount that the estimate of the target function will change if different training data was used.
  - Excessive sensitivity to the model's small variations in the training data.
  - A model with many degrees of freedom is likely to have high variance, and thus to over fit the training data.
- **Irreducible Error**
  - Due to the noisiness of the data itself.
  - The only way to reduce this part of the error is to clean up the data.

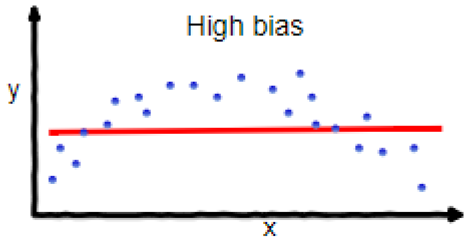
The Bias/Variance Tradeoff



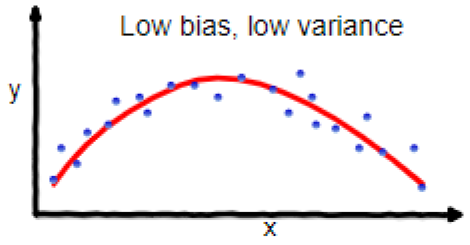
The Bias/Variance Tradeoff



overfitting



underfitting



Good balance

THANK YOU