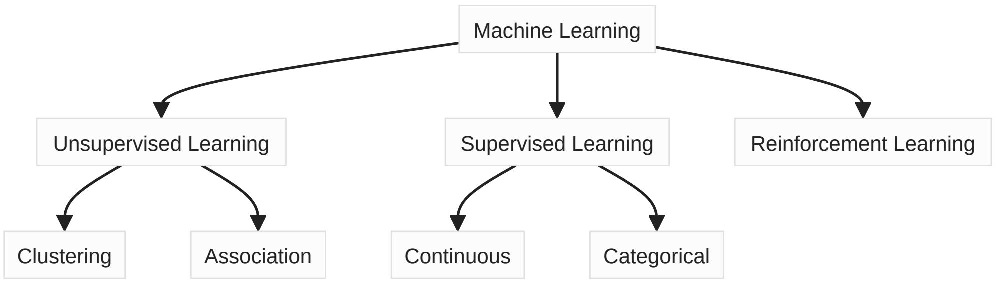
```yaml
id: 1726149824-ZTNE
aliases:
  - ML U1
Author: vortex
```

ML U1

> While *Traditional Programs* take the data and rules as input and apply the rules on all the data to get the answers, *Machine learning* takes the data and answers as input and tries to learn the *rules*.

- Learning is a process by which system improves performance from experience.
- ML is a study of algorithms that improves performance(P) at from task(T) with experience(E).



Supervised Learning

- Input data and the expected output to that data.
- Algorithm learns the relationship between them.

1. Regression
   - Used for Continuous Data.
   - Learns relationship between input and output data.
   - Predict y when given x.
2. Classification
   - When output is Categorical.
   - Given x predict y where y is categorical.

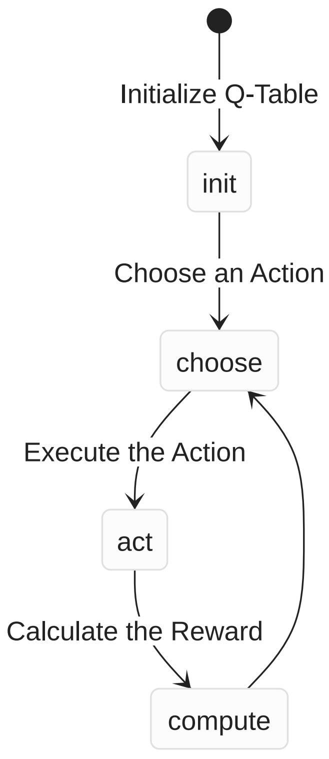
Unsupervised Learning

- Trained using unlabeled data.
- Goal is to detect Underlying patterns in the data.
- Grouped based on similarities and differences.

1. Clustering
    - Grouping method
    - Objects with similarities go togeather, others go separately.
2. Association
    - Relationships between variables.
    - Predicts set of items that occur togeather in a dataset.

Reinforcement Learning

- Learn what to do and how to map situations to actions.
- `Agents` interact with its `Environment` to learn how to best act within it.
- No Training Dataset.
- Highly Compute intensive.



Terminologies

- Agent: Learner
- Environment: The external System
- State: Current situation in the environment
- Action: Choices available to the Agent
- Policy: Rules to be followed while selecting actions.
- Reward: Scalar feedback to quantify favourable outcomes.
- Value Function: Predicts Expected cumulative reward.

Q-Table

- Look-up table where maximum expected future rewards for an action are computed at each state.
- Tells us which action is best at each state.
- Q-Learning Function
  $Q'(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots |(s_t, a_t)]$
  - R : Reward
  - $Q'(s_t, a_t)$: Q-value at time t.
  - γ : discount rate

Semi Supervised Learning

- Only Some of the data is labelled.
- Useful when there is large amount of unlabelled data but not feasible to label them all.

Generative & Discriminative Models

- Discriminative: Conditional probability, $p(y|x)$
- Generative: Joint Probability, $p(x, y)$

> Discriminative Models draw a decision boundary on the data, while Generative Models show how the data is placed throughout the space.

Concept Learning

- Concept: A subset of objects/events defined over a larger set.
- In ML, models must gain the ability to take an object and determine if it belongs to a concepts.
- Must look at the *feature space* and learn the concept in a general way.
- Basically approximating a boolean valued function.

> A *claim* made about the concept is called the *hypothesis*.
> Set of all the hypotheses is called the *hypothesis space*.
> Set of all instances is called the *instance space*.

- If there are d binary attributes, there are 2^d instances and 2^{2^d} concepts.
- To reduce concept space we introduce *bias*.
  - Bias can be introduced by making some assumption about the concept.
  1. Use Conjunction of the features.
  2. No disjunction allowed.
  - Such reduction of concept space is called *inductive bias*.
  - $?$ in this notation stands for any acceptable value, ϕ stands for *reject all*.
  - $? \wedge ? \wedge ? \wedge ?$ is called the most *general* hypothesis.
  - $\phi \wedge \phi \wedge \phi \wedge \phi$ is called the most *specific* hypothesis.

- For an n featured problem, number of conjunctive concepts are $3^n + 1$
- Hence the search-space is shrunken.
- When a hypothesis can be applied correctly to atleast one instance in the training data, it is said to be *satisfied*. Such hypotheses are not correct in all cases.
- When a hypothesis can be applied on all instances in the training data correctly, its said to be *consistent*. This is correct in all cases.
- A subset of the Hypothesis space where every entry is consistent is called the *version space*. It is a set of *all* consistent hypothesis.

Find S Algorithm

1. Start with a highly specific hypothesis$(\phi, \phi, \phi ...)$
2. Proceed to the next instance in the training data.
    - If it is negative, no change to the current hypothesis.
    - If it is positive, any mismatched feature is replaced by $?$. This makes the hypothesis more general.
3. Keep iterating through the examples until either hypothesis is completely general or examples get over.

> Syntactically and Semantically different hypotheses

Concept	Hypothesis
True underlying pattern	candidate pattern that the model has proposed
Model tries to learn the concept	Simplified representation of the concept
Decision boundary is the true ideal boundary	The learned boundary learned by the model to estimate the ideal boundary

Performance Metrics

Confusion Matrix

- x-axis: predicted values.
- y-axis: actual values.

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

- Number of correct predictions made over all types of predictions made.
- Used when training data is *balanced*. If we have skewed data this isn't an appropriate measure.
- If TP<FP and we change the algo to classify -ve classes, the accuracy improves. This is called the *Accuracy Paradox*.

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

- Number of positive cases caught.

- Used when cost of false positive is high.
- Reduces *Type-1* Errors(false alarms).

$Recall(Sensitivity) = \frac{TP}{TP+FN}$

- Number of positive cases that weren't missed
- Sensitivity: what proportion of +ve class for correctly classified.
- When cost of false negatives is high.
- Reduces *Type-2* Errors(failure to detect existing event).

$FalseNegativeRate(FNR) = \frac{FN}{TP+FN}$

- Must try to maximize TPR and minimize FNR.

$FalsePositiveRate(FPR) = \frac{FP}{TN+FP}$

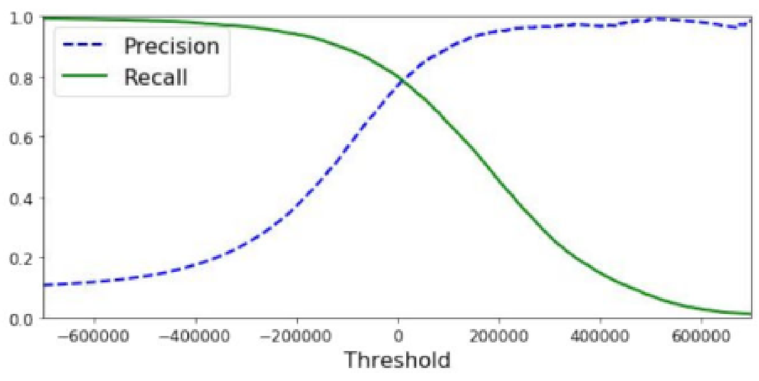
- True Negative Rate(Specificity) = 1-FPR
- Higher TNR and Lower FPR are desirable.

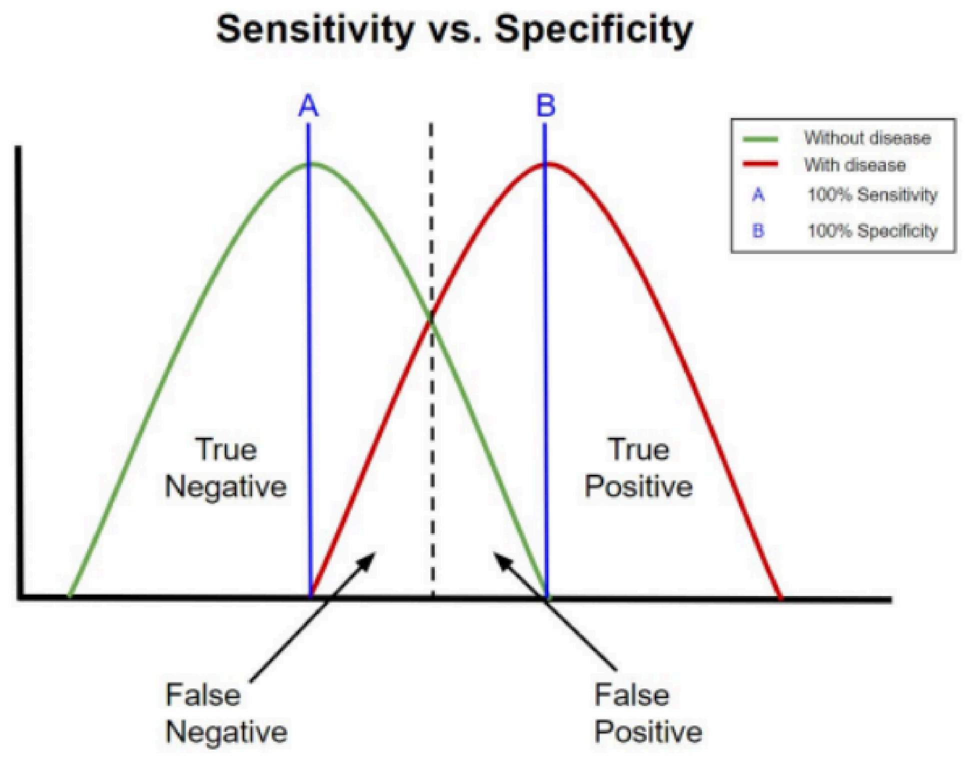
$F1score = \frac{2*recall*precision}{recall+precision} = \frac{2TP}{2TP+FP+FN}$

Precision-Recall Tradeoff

Precision	Recall
Reliability of the positive values predicted.	Ability to detect positive samples.

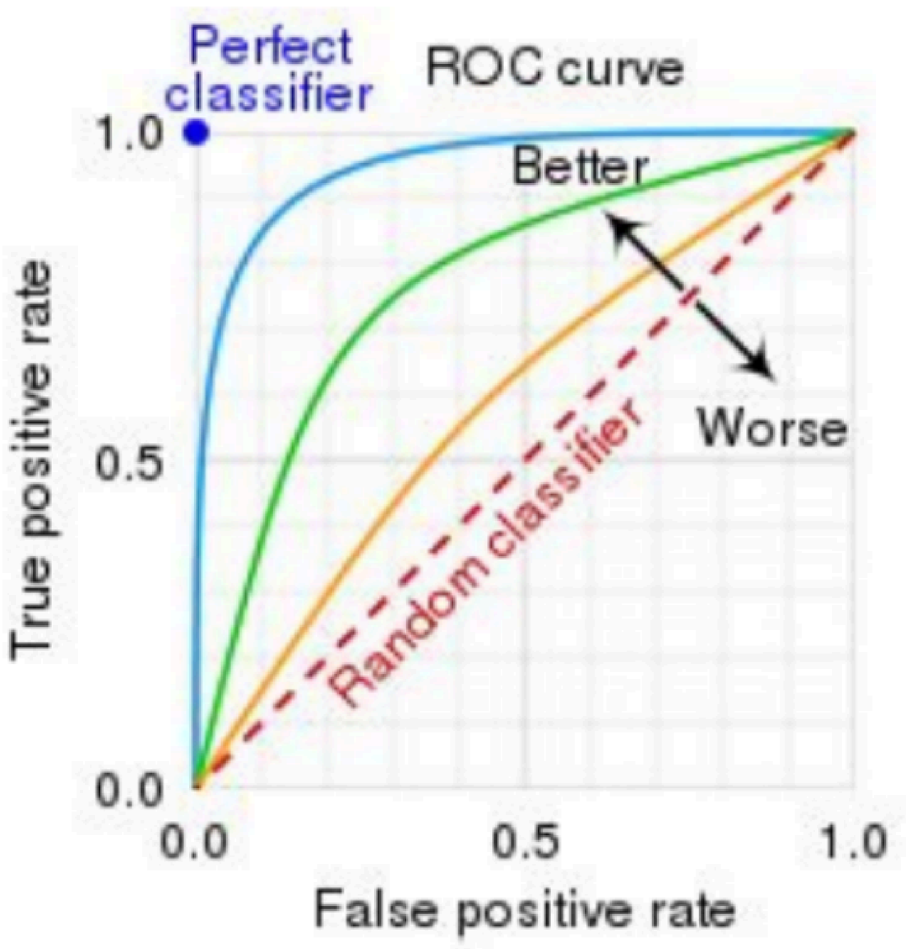
- Usually precision and recall are inversely proportional.



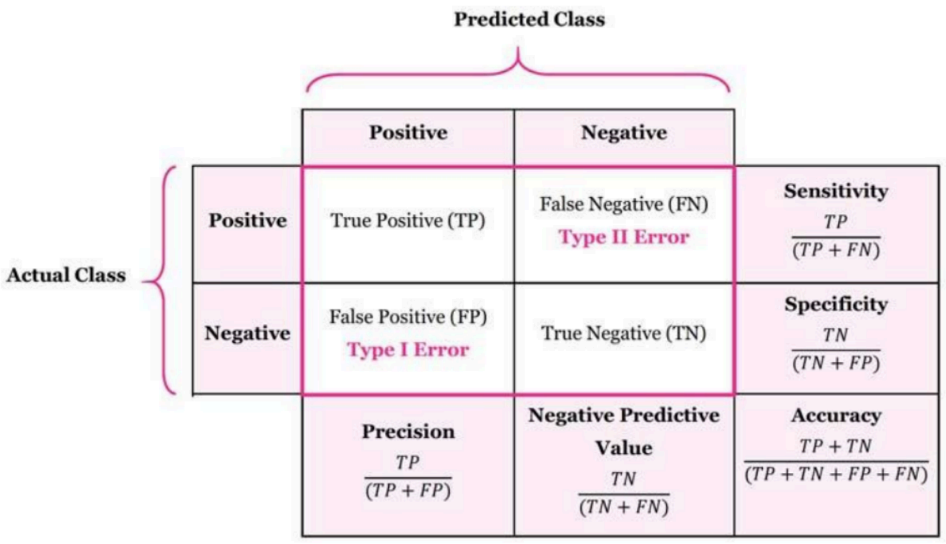


Relationship between sensitivity and specificity – inversely proportional

- ROC(Receiver Operating Characteristics): TPR vs FPR plot for different classification thresholds
- Lowering Threshold increases number of positives.
- Logistic regression is inefficient for this, we use a different algorithm called Area Under Curve(AUC)



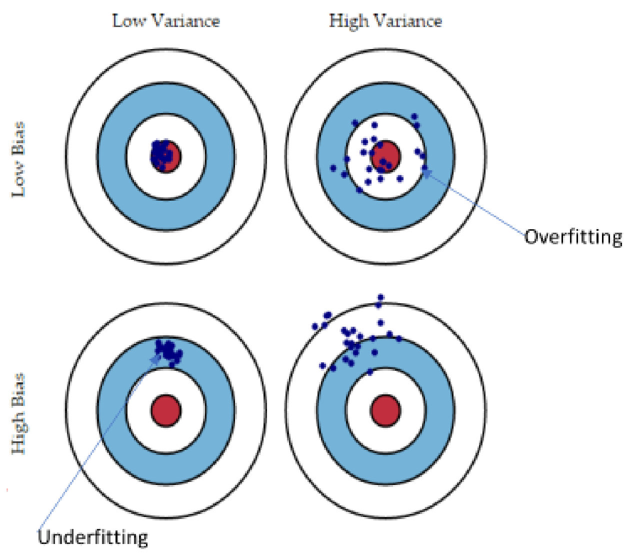
- More the AUC, better the model.

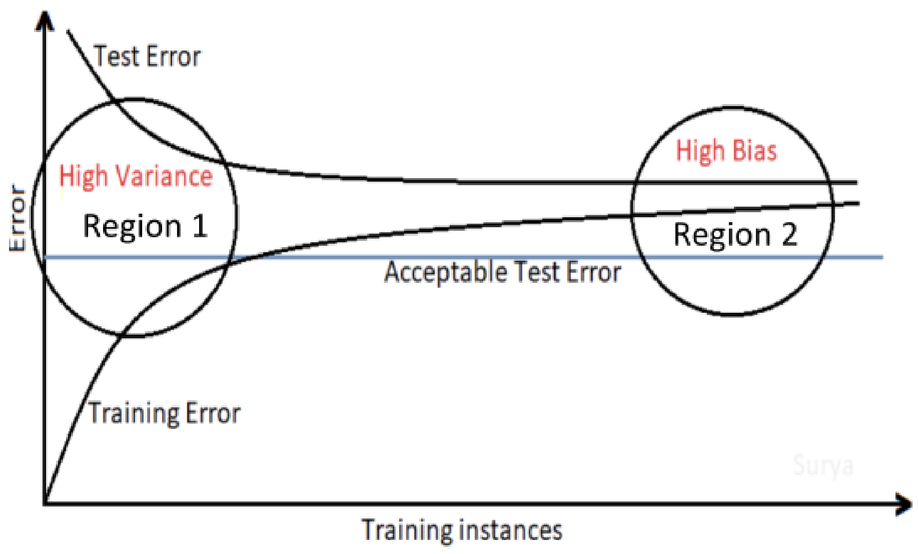


		Predicted Class: Positive	Predicted Class: Negative	
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
Actual Class	**Negative**	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$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

Bias Variance Tradeoff

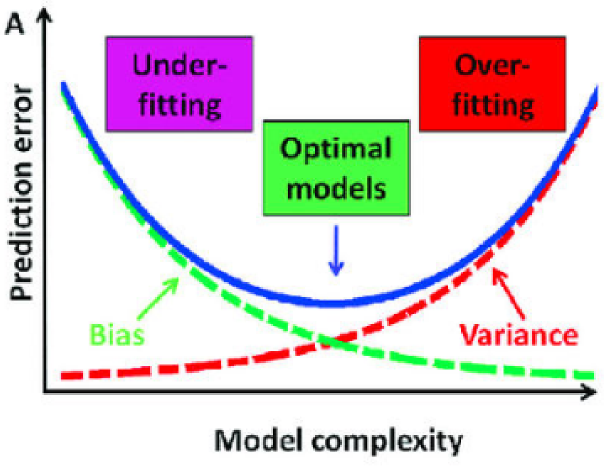
- $Err(x) = Bias^2 + Variance + IrreducibleError$
  $= (E[f(x)] - f(x))^2 + E[f(x) - E[f(x)]]^2 + {\sigma_e}^2$
- Irreducible error is noise inherently present in the data.
- Bias is the difference between average prediction and correct value.
- Variance is error from sensitivity to small fluctuations in the training set.





- The ultimate goal of any ML algorithm is to acheive **Low Bias and Low Variance**.
- As model complexity increases, variance increases, as it decreases bias increases.
- High Variance <-> Overfitting
- High Bias <-> Underfitting

- A good model must acheive balance between these 2 and that is called Bias-Variance Tradeoff.

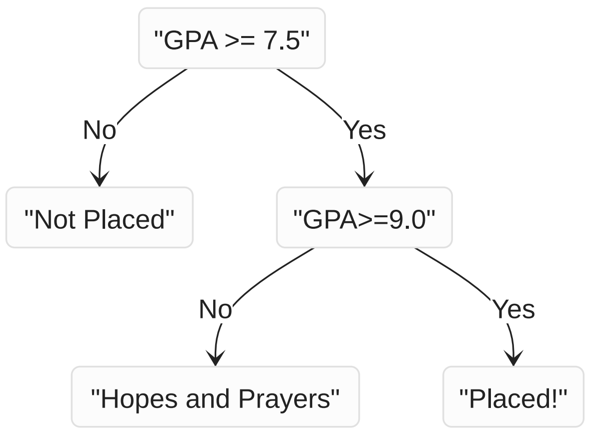


- Remedies:
  - Bagging: to combat high variance
  - Boosting: to combat High bias
- Examples:
  - High Variance, Low Bias: Decision Trees,KNN,SVM
  - Low Variancem, High Bias: Linear Regression, Linear Discriminant Analysis, Logistic Regression.

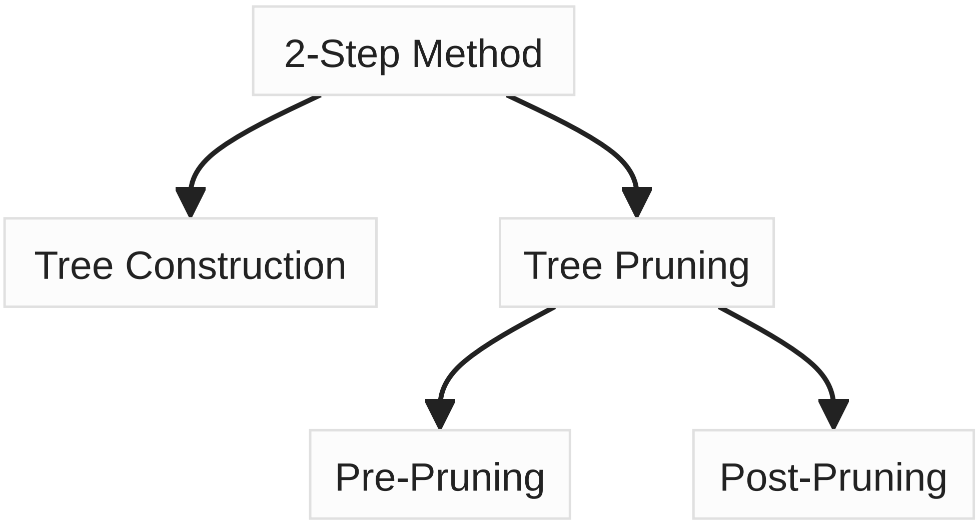
> [Some Questions Regarding the Bias-Variance Tradeoff]

Decision Tree - The ID3 Algorithm

- Consists of Decision nodes and Leaf nodes.
- Leaf nodes are the final classified categories.
- Best predictor is taken as *root* node.
- DTs can handle both *categorical* and *numerical* data.
- Graphical representation of all possible solutions of a particular decision.



- A decision tree that handles both classification and regression is called **CART**.
- Scikit-Learn uses CART to train Decision trees.
- CART uses the *Gini Index* to split the data into decision tree.o
- Characteristics:
    1. Looks for Short&Fat trees.
    2. Disjunction of Conjunctions
    3. Uses heusterics for best split

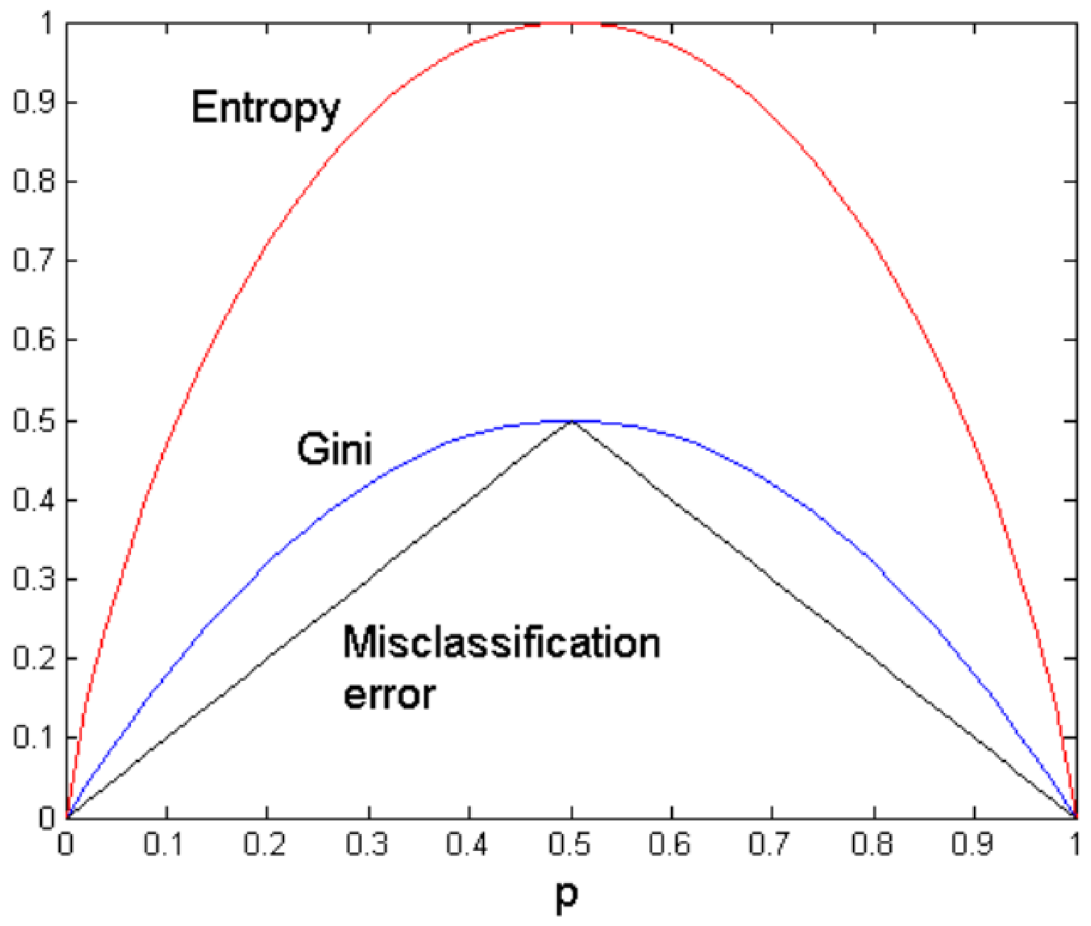


- DT Splits are decided based on node *purity*.

> Here *purity* refers to Pure class distribution.

- Impurity Measures
    - Entropy
      $Entropy(t) = -\Sigma_j((p(j|t))log_2p(j|t))$
    - Gini Index
      $Gini(t) = 1 - \Sigma_j[p(j|t)]^2$
    - Misclassification Error
      $Error(t) = 1 - max(p(j|t))$
- $p(j|t)$: fraction of examples that belong to class j at node t.

$$\text{Entropy(S)} = -\underbrace{\frac{p}{p+n}log_2\frac{p}{p+n}}_{\text{uncertainty due to positive examples in data set}} - \underbrace{\frac{n}{p+n}log_2\frac{n}{p+n}}_{\text{uncertainty due to negative examples in data set}}$$



- Entropy maxes at p=0.5 .
- This isn't really useful for splitting data.
- Hence we Use *information gain* which measures the reduction in entropy which is a much better metric.

$$InformationGain = Entropy(S) - \Sigma_{i=1}^{k}\frac{N(v_j)}{N}Entropy(v_j)$$

Issues with decision trees

- Determining how deeply to grow the tree
- Overfitting
- Handling continuous attributes
- Handling missing data
- Handling Attributes with differing costs and improving computational efficiency

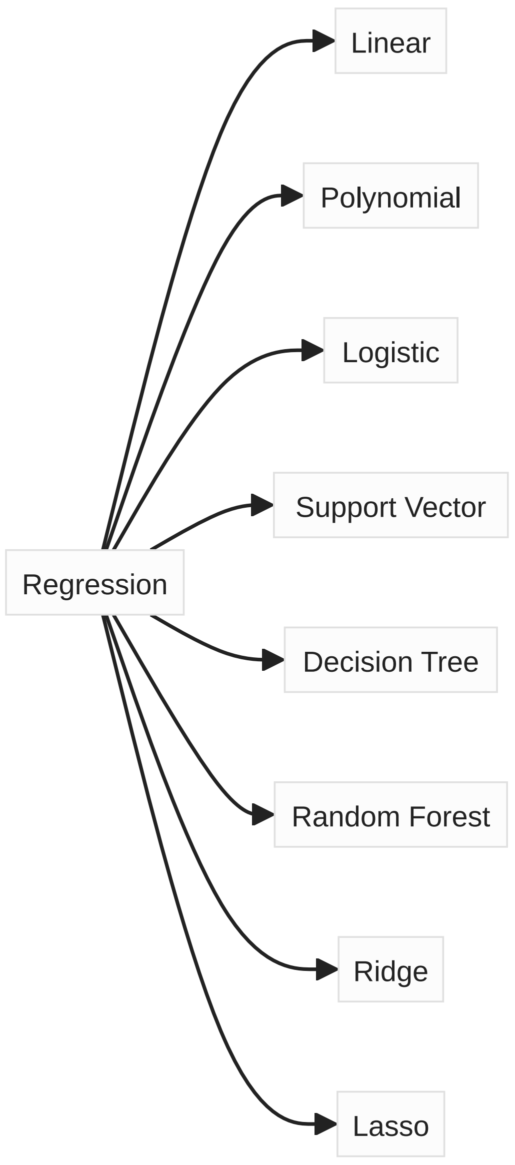
Decision Tree Induction

- Non parametric approach to building class model
- No prior assumptions needed.

- Construction of DT is very inexpensive computationally and hence makes it possible to construct models very fast even with large datasets.
- Smaller DTs are easier to interpret.
- Not thrown off by noisy data.
- Redundant attributes don't affect accuracy of the model.

Regression

- Method to understand relationship between independent variable X(features) and dependent variable y(outcome)
- Predict y values from the independent variable by establishing relations between the data progression.



> Use scatter plots to determine which regression method is the best suited to the data.

Linear Regression

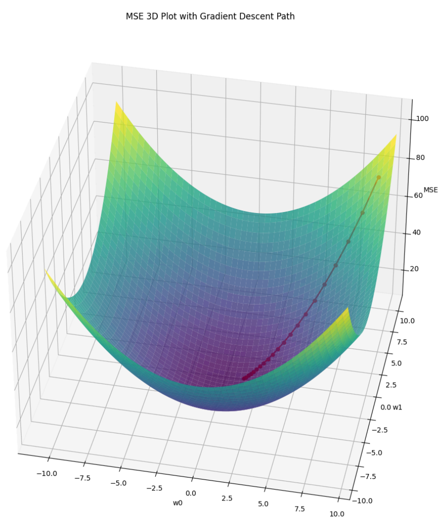
- Useful when there is a linear progression of relation between the dependent and independent variables.
- Types:
    - Simple: one input variable
    - Multiple: More than one input variable

Simple Linear Regression

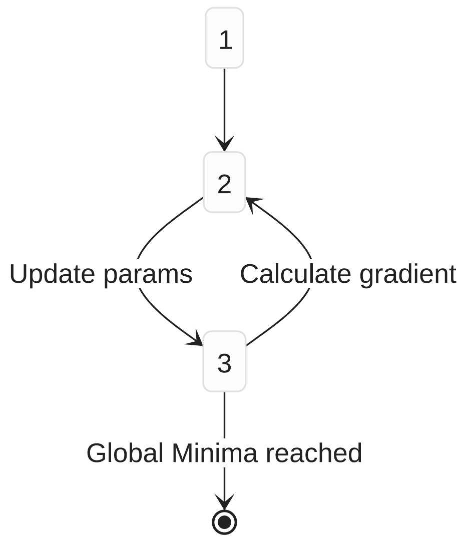
- $y = mx + b$
    - y : dependent variable
    - x : independent variable
    - m : slope
    - b : y-axis intercept
- The aim of regression is to obtain the **Best Fit Line** which is a line in the dataplot that minimizes the vertical distance between all the datapoints.
- To optimize this line to obtain the best fit, we use the **Gradient Descent** algorithm.

Some terms to know

1. Error = $y_i - (mx_i + b)$
2. Squared Error = $(y_i - (mx_i + b))^2$
3. Sum of Squared Errors(SSE) = $\Sigma_{i=1}^{n}(y_i - (mx_i + b))^2$
4. Mean Squared Error(MSE) = $\frac{1}{n}\Sigma_{i=1}^{n}(y_i - (mx_i + b))^2$



- Gradient descent starts out with arbitrary weights in order to minimize the cost function.
- The final goal is to bring the Error function to a global minimum in the hypothesis space.
- Each step is taken as per the learning rate(α).
- Steps:
    1. Init m and b randomly.
    2. Calculate Gradient:
        - $\frac{\partial MSE}{\partial m} = \frac{-2}{n}\Sigma_{i=1}^{n} x_i(y_i - (mx_i + b))$
        - $\frac{\partial MSE}{\partial b} = \frac{-2}{n}\Sigma_{i=1}^{n}(y_i - (mx_i + b))$
    3. Update the params:
        - $m = m - \alpha\frac{\partial MSE}{\partial m}$
        - $b = b - \alpha\frac{\partial MSE}{\partial b}$



R-Squared Method

- A way to evaluate model performance.
- Varies between 0-1. If it is 1, model has fit *perfectly* and if it is 0, model has not learnt any of the relationships correctly.

$$R^2 = 1 - \frac{SSR}{SST} = 1 - \frac{\Sigma(y_i - \hat{y}_i)^2}{\Sigma(y_i - \bar{y}_i)^2} = \frac{ExplainedVariation}{TotalVariation}$$

> Also called the "Coefficient of determination"

Advantages	**Disadvantages**
Simple to implement	Outliers can throw it off
When Linearity is known, its less-complex	Assumes a linear relationship.
Susceptible to overfitting	Establishes a relationship between the mean of the dependent variable and the independent variable. This might not be a true measure.

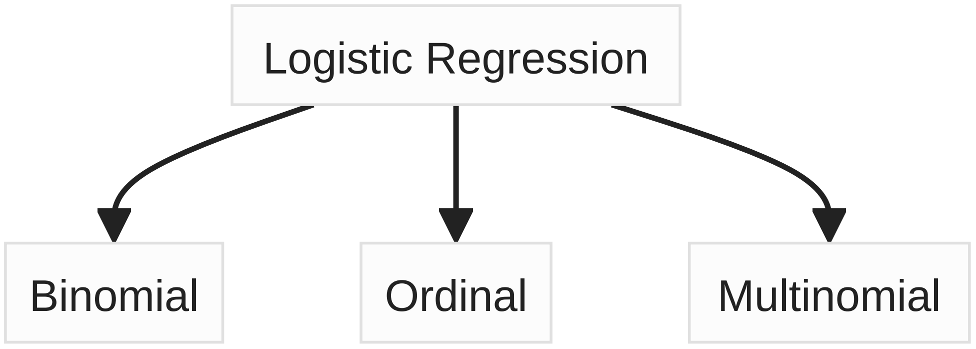
Logistic Regression

- Used for classification tasks
- Takes the output of linear regression as input to estimate the probability values.

> Comparison between Linear and Logistic Regression

- Logistic Regression is more favourable to outliers due to the use of the sigmoid function.

Linear	Logistic
Continuous Dependent Variable	Categorical Dependent Variable
Regression Problem	Classification Problem
Best fit line	S-Curve
MSE is used	MLE is used
Linear Relationship	Linear relationship not required



Some terms to know

1. Probability: $p = \frac{EventofInterest}{AllEventsinSampleSpace}$
2. Odds: $Odds = \frac{p}{1-p}$
3. Odds Ratio: Ratio of 2 odds

> Output of the logistic regression follows Bernoulli Distribution(2 possible outcomes)

- The **Logit** function is a function that links the linear combination of input to an output that follows Bernoulli Distribution.
- $Logit(p) = ln(odds) = ln(\frac{p}{1-p})$

> Sigmoid is the inverse of the Logit function.

Useful Links

1. Derivative Of Sigmoid
2. Training a Logistic Regression Model

Cost Function for Logistic Regression

- Logistic Regression uses Minimum Liklihood Estimation(MLE) as the metric to estimate the coefficients.
- The cost function used it Negative Log Likelihood
- $E(w) = \frac{-1}{m}\Sigma_{i=1}^{n} y_i log(\sigma(z)) + (1-y_i) log(1-\sigma(z))$

$w_i = w_i - \alpha\frac{\partial E(w)}{\partial w_i}$

Instance Based Learning

- One way of classifying types of learning is:
  - Generalized
  - Memorized(Instance based learning falls here)
- In this type of learning, the system doesn't try to find a general pattern in the data by "training".
- This is also called **Lazy-Learning**.
- One algorithm that uses such a process is called K-Nearest Neighbours(KNN).
- It groups data according to similarities(closeness) to the training data.
- The distance between the datapoints is taken as the metric for similarity.
- Flow of KNN:
  1. K = Number of neighbours
  2. Calculate the distance between the neighbours
  3. Take the `K` nearest of the neighbours
  4. The category that is found to be max in these `K` neighbours is finalized.
- Target can be :
  - Discrete: returns category that is max
  - Real-Valued: returns mean of the `k` closest datapoints.

> Argmax is used to find the class with the highest predicted probability.

$\hat{f}(x) = argmax\Sigma(\delta(v, f(x_i)))$

- Making Class Boundaries
  1. Join closest points between classes.
  2. Draw Perpendicular bisectors on them.
- Factors to consider while starting KNN:
  - Value of K.
  - Is normalization needed for that data?
  - Relavent Attributes
  - Distance Method
- Distance Methods:
  - Euclidean: $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

    - Manhattan: $|x_2 - x_1| + |y_2 - y_1|$
    - Minkowski: $\Sigma_{i=1}^{k}((|x_i - y_i|)^q)^{1/q}$
- Choosing `K`
    - If model ties, decrease k by 1.
    - Usually good practice to start with $k = \sqrt{n}$ where n is number of datapoints.
    - $k = n + 1$
    - Elbow Method: plot the error of different `k` values and choose the one at the elbow point.
    - "Thumb" rule: $k = \sqrt{n/2}$

> Low `k` : Overfitting
> High `k` : Underfitting

Weighted KNN

- Vanilla KNN algorithms just plainly trust the distance measure.
- Every neighbouring point has the same impact on the prediction.
- This increases the dependence on the `k` value.
- Hence we weigh the importance of different data points like so:
  $w_i = \frac{1}{d(x_q,x_i)^2}$

$$\hat{f}(x) = argmax\Sigma(w_i\delta(v, f(x_i)))$$

Issues with KNN

1. Speed drops fast as size of dataset increases.
2. **Curse of Dimensionality** : as number of attributes grow, algorithm struggles more.
    - **Radius of Influence** of each attribute becomes smaller.
3. Can't handle imbalanced data.
4. Very sensitive to outliers.
5. Features need to be scaled(normalization)
    - $x_i = \frac{x_i - \mu}{\sigma}$

> Complexity : $O(knd) to find k-nearest neighbours d : dimension of data, n : number of datapoints, k : k$
> O(d)$ for computing distance to one example
> $O(nd)$ for finding one nearest neighbour

Handling Attributes

- Boolean: Convert to `0` or `1`.
- Natural Progressions: small, medium, large : 0,1,2
- Random Unordered data: One-hot encode