



# LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

(AUTONOMOUS)

Accredited by NAAC with 'A' Grade & NBA (Under Tier - I), ISO 9001:2015 Certified Institution

Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada

L.B. REDDY NAGAR, MYLAVARAM, KRISHNA DIST., A.P.-521 230.

<http://lbrce.ac.in/it/index.php>, [hodit@lbrce.ac.in](mailto:hodit@lbrce.ac.in), Phone: 08659-222933, Fax: 08659-222931

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**Course Name : MACHINE LEARNING**

**Course Code : 20AD04**

**Course Instructor : M. Rajesh Reddy**

**Semester : V**

**Regulation : R20**

**Unit: 5**



## **UNIT-5: SYLLABUS**

**Ensemble Learning-** Bagging, Boosting, Stacking and its impact on bias and variance, AdaBoost, Gradient Boosting Machines, XGBoost.

**Reinforcement Learning** -Introduction, Q Learning



## 5.1 ENSEMBLE METHODS

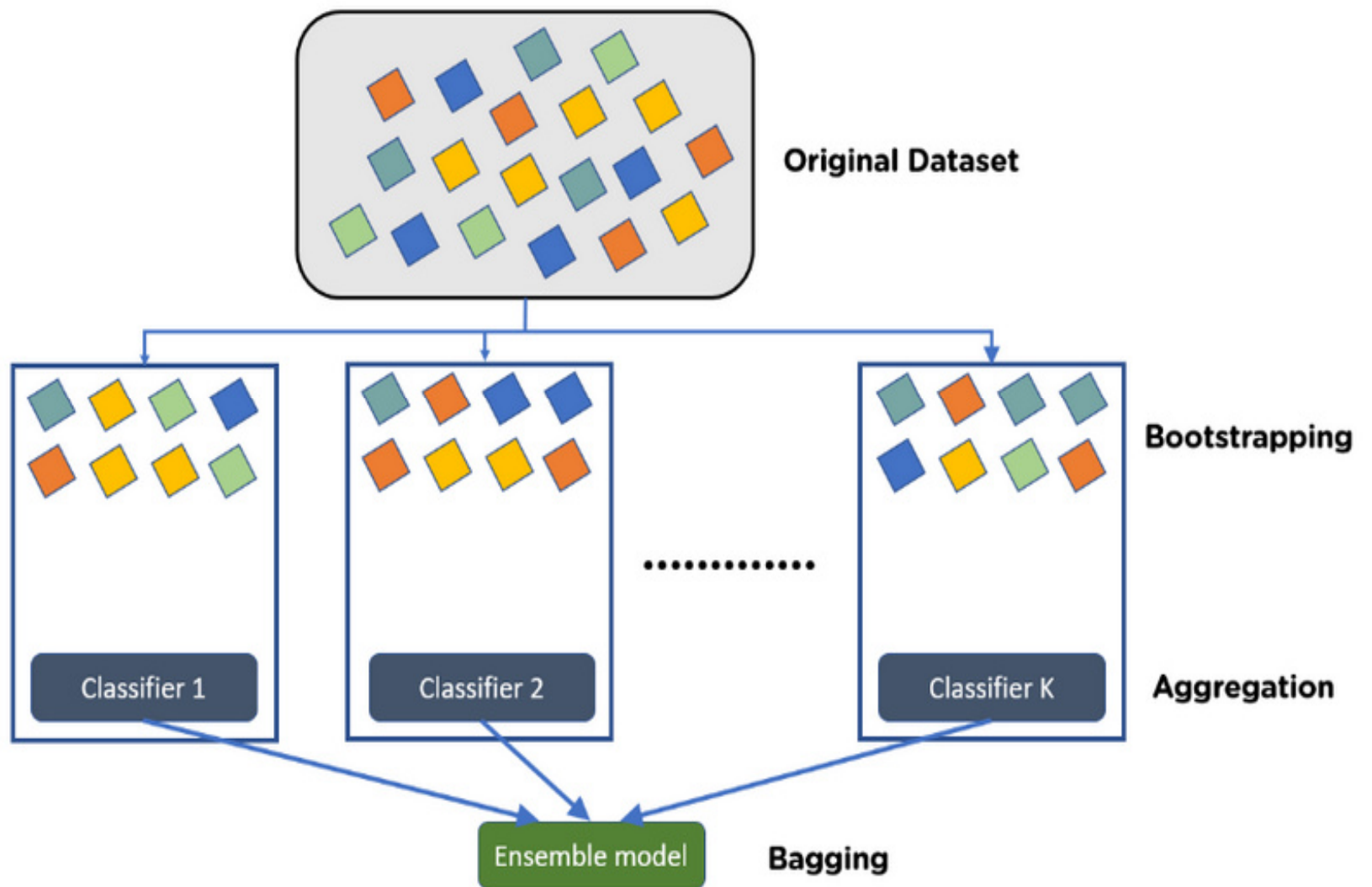
- Ensemble learning is a widely-used machine learning technique in which multiple individual models, called **base models**, are combined to produce an effective optimal prediction model.
- Combine the predicted output of each model by using model averaging techniques like max voting, variance, etc.
- There are two types of ensemble models: **homogeneous model and heterogeneous model**.
- In the homogeneous model, all weak learners are trained homogeneously with the same base algorithms, while in the heterogeneous model, all weak learners are trained with multiple base algorithms.
- Two homogeneous models:
  - **Bagging (Bootstrap Aggregation)**
  - **Boosting**



## 5.1 ENSEMBLE METHODS

### 5.1.1 Bagging

- Bagging, also known as **Bootstrap aggregating**, is an ensemble learning technique that helps to improve the performance and accuracy of machine learning algorithms.
- It is used to deal with bias-variance trade-offs and **reduces the variance of a prediction model**.
- Bagging avoids overfitting of data and is used for both regression and classification models, specifically for decision tree algorithms.
- In Bagging, make multiple subsets by sampling and replacement methods from the training datasets.
- For each subset, a base algorithm, like a Decision Tree, is used to predict the output.
- Combine the predictions of all subset models using model averaging techniques to predict the outcome.
- For example, the Random Forest Model uses Bagging; which uses multiple decision trees to grow trees, resulting in a complete random forest.





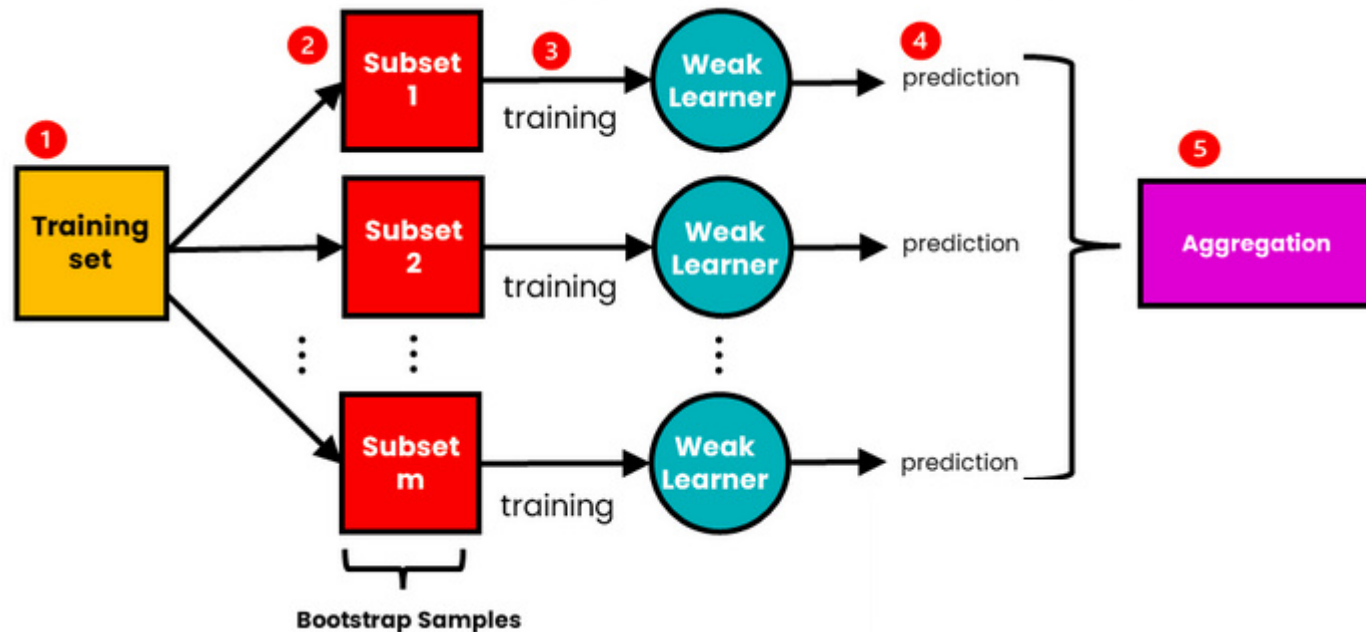
## 5.1 ENSEMBLE METHODS

### 5.1.1 Bagging

#### Steps in Bagging:

- Sample equal-sized subsets with replacement
- Train weak models on each of the subsets independently and in parallel
- Combine the results from each of the weak models by averaging or voting to get a final result

#### The Process of Bagging (Bootstrap Aggregation)





## 5.1 ENSEMBLE METHODS

### 5.1.2 Boosting

- Boosting aims to produce a model with a **lower bias** than that of the individual models.
- In boosting, we train a sequence of models.
- Each model is trained on a weighted training set.
- We assign weights based on the errors of the previous models in the sequence.
- The main idea behind sequential training is to have each model correct the errors of its predecessor.
- This goes on until the predefined number of models has been trained or some other criteria are met.
- During training, instances that are classified incorrectly are assigned higher weights to give some form of priority when trained with the next model.
- Weaker models are assigned lower weights than strong models when combining their predictions into the final output.

## 5.1 ENSEMBLE METHODS

### 5.1.2 Boosting

#### Steps in Boosting:

Initialize data weights to the same value and then perform the following steps iteratively:

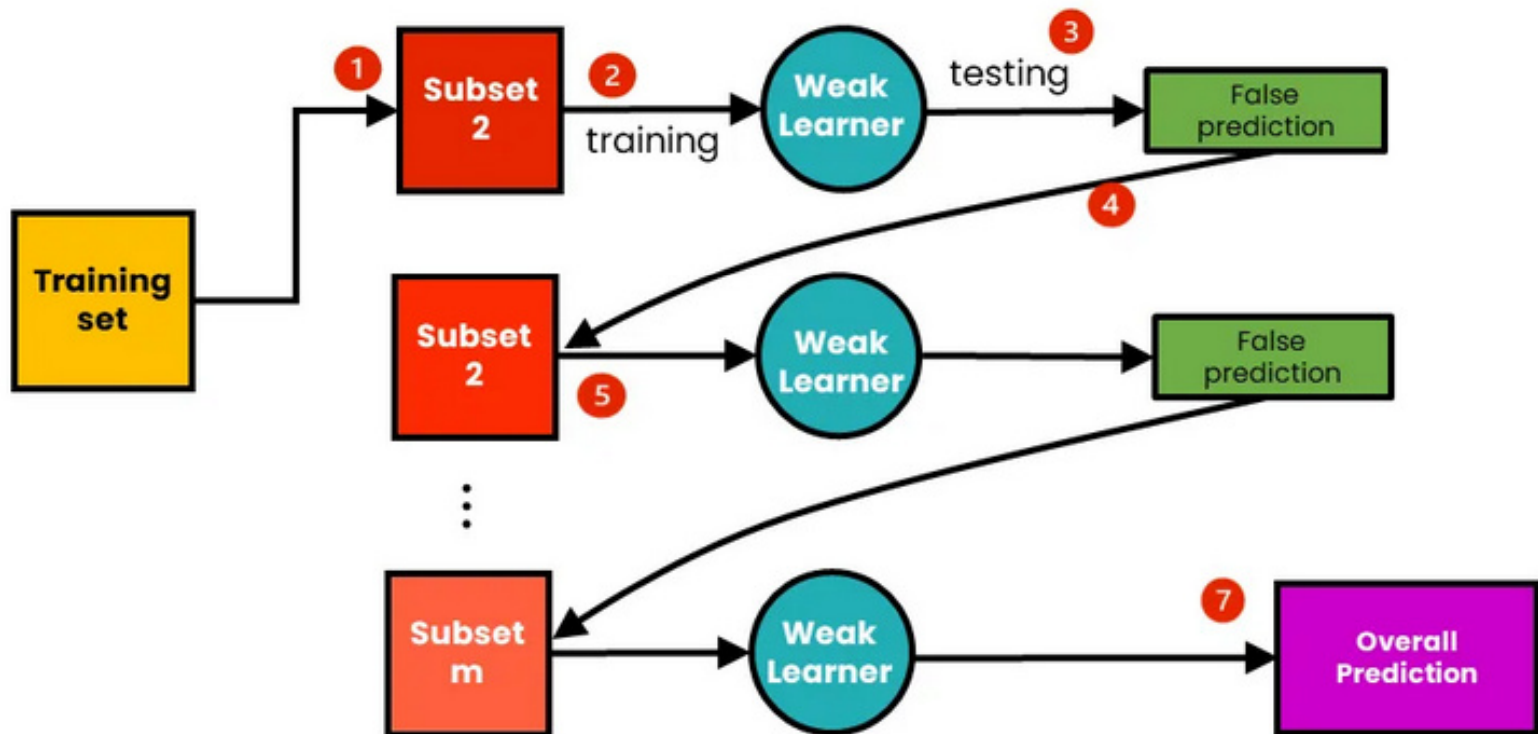
- Train a model on all instances.
  - Calculate the error on model output over all instances
  - Assign a weight to the model (high for good performance and vice-versa)
  - Update data weights: give higher weights to samples with high errors
  - Repeat the previous steps if the performance isn't satisfactory or other stopping conditions are met.
- Finally, combine the models into the one that can use for prediction.



# 5.1 ENSEMBLE METHODS

## 5.1.2 Boosting

### The Process of Boosting





## 5.1 ENSEMBLE METHODS

### Bagging vs Boosting

#### Bagging

The original dataset is divided into multiple subsets, selecting observations with replacement.

This method combines predictions that belong to the same type.

Bagging decreases variance.

Base classifiers are trained parallelly.

The models are created independently.

#### Boosting

The new subset contains the components mistrained by the previous model.

This method combines predictions that belong to the different types.

Boosting decreases bias.

Base classifiers are trained sequentially.

The model creation is dependent on the previous ones.

## 5.1 ENSEMBLE METHODS

### 5.1.3 Stacking

- Stacking aims to create a single robust model from multiple **heterogeneous strong learners**.
- Stacking differs from bagging and boosting in that:
  - It combines strong learners
  - It combines heterogeneous models
  - It consists of creating a **Metamodel**. A metamodel is a model created using a new dataset.
- Individual heterogeneous models are trained using an initial dataset.
- These models make predictions and form a single new dataset using those predictions.
- This new data set is used to train the metamodel, which makes the final prediction. The prediction is combined using weighted averaging.

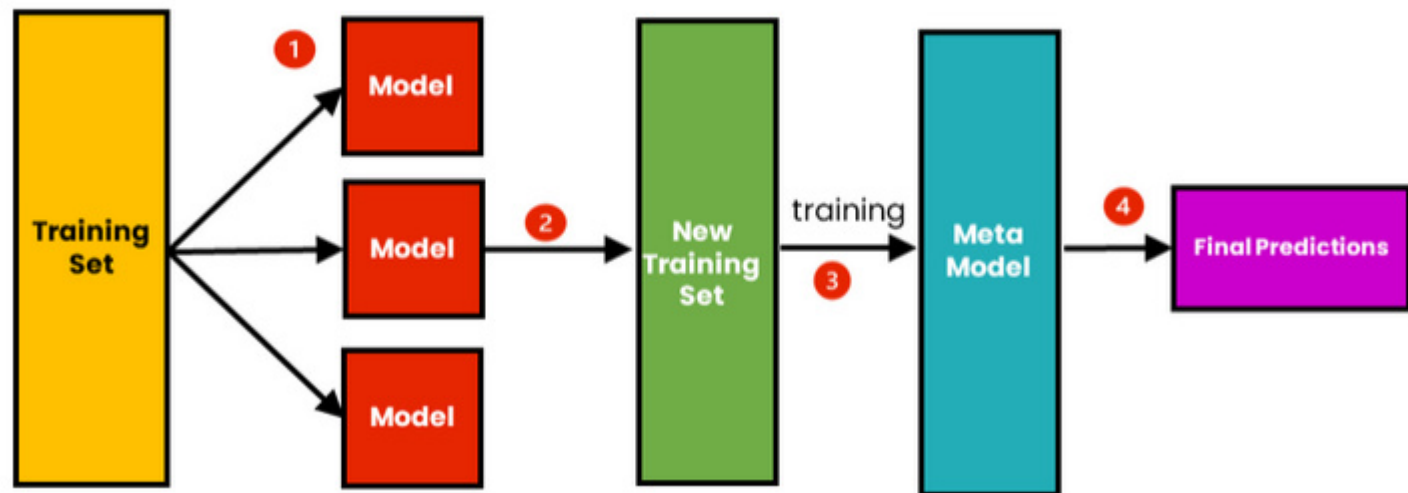
# 5.1 ENSEMBLE METHODS

## 5.1.3 Stacking

### Steps in Stacking

- We use initial training data to train m-number of algorithms.
- Using the output of each algorithm, we create a new training set.
- Using the new training set, we create a meta-model algorithm.
- Using the results of the meta-model, we make the final prediction. The results are combined using weighted averaging.

### The Process of Stacking



## 5.2 AdaBoost Algorithm

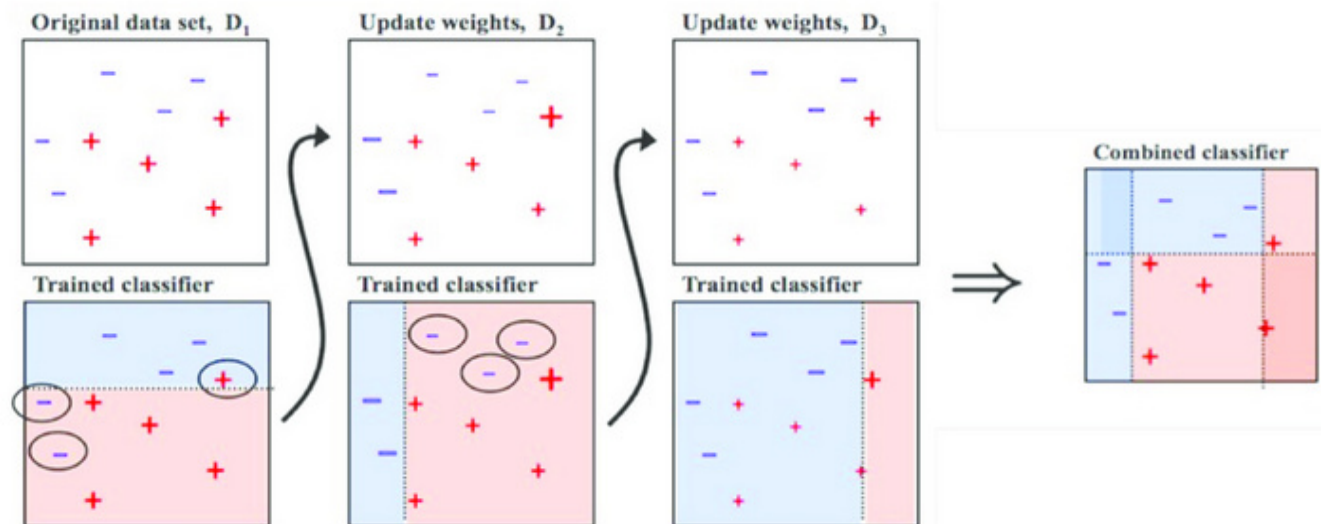
- AdaBoost algorithm, short for **Adaptive Boosting**, is a Boosting technique used as an **iterative Ensemble Method** in Machine Learning.
- AdaBoost classifier builds a strong classifier by **combining multiple poorly performing classifiers** so that you will get **high accuracy strong classifier**
- It is called Adaptive Boosting as the **weights are re-assigned to each instance**, with **higher weights assigned to incorrectly classified instances**.
- AdaBoost's most commonly used estimator is **decision trees with one level**, which is decision trees with **just one split**. These trees are often referred to as **Decision Stumps**.



## 5.2 AdaBoost Algorithm

### Steps in AdaBoost Algorithm:

1. Initially, Adaboost selects a training subset randomly.
2. It iteratively trains the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training.
3. It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification.
4. Also, It assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.
5. This process iterate until the complete training data fits without any error or until reached to the specified maximum number of estimators.
6. To classify, perform a "vote" across all of the learning algorithms you built.





## 5.2.1 AdaBoost Algorithm Example

### Step 1: Assigning Weights

- The Image shown below is the actual representation of our dataset. Since the target column is binary, it is a classification problem. First of all, these data points will be assigned some weights. **Initially, all the weights will be equal.**

ROW NO.	GENDER	AGE	INCOME	ILLNESS	SAMPLE WEIGHTS
1	Male	41	40000	Yes	1/5
2	Male	54	30000	No	1/5
3	Female	42	25000	No	1/5
4	Female	40	60000	Yes	1/5
5	Male	46	50000	Yes	1/5

The formula to calculate the sample weights is:  $w(x_i, y_i) = \frac{1}{N}$ ,  $i = 1, 2, \dots, n$

Where N is the total number of data points

Here since we have 5 data points, the sample weights assigned will be 1/5.

## 5.2.1 AdaBoost Algorithm Example

### Step 2: Classify the Samples

- We start by seeing how well “*Gender*” classifies the samples and will see how the variables (Age, Income) classify the samples.
- We’ll create a decision stump for each of the features and then calculate the ***Gini Index*** of each tree. The tree with the lowest Gini Index will be our first stump.
- Here in our dataset, let’s say ***Gender*** has the lowest gini index, so it will be our **first stump**.

### Step 3: Calculate the Influence

- We’ll now calculate the “**Amount of Say**” or “**Importance**” or “**Influence**” for this classifier in classifying the data points using this formula:

$$\alpha = \frac{1}{2} \log \frac{1 - \text{Total Error}}{\text{Total Error}}$$

The total error is nothing but the summation of all the sample weights of misclassified data points.

## 5.2.1 AdaBoost Algorithm Example

### Step 3: Calculate the Influence

- Here in our dataset, let's assume there is 1 wrong output, so our total error will be  $1/5$ , and the alpha (performance of the stump) will be:

$$\alpha = \frac{1}{2} \log_e \left( \frac{1 - \frac{1}{5}}{\frac{1}{5}} \right)$$

$$\alpha = \frac{1}{2} \log_e \left( \frac{0.8}{0.2} \right)$$

$$\alpha = \frac{1}{2} \log_e(4) = \frac{1}{2} * (1.38)$$

$$\alpha = 0.69$$

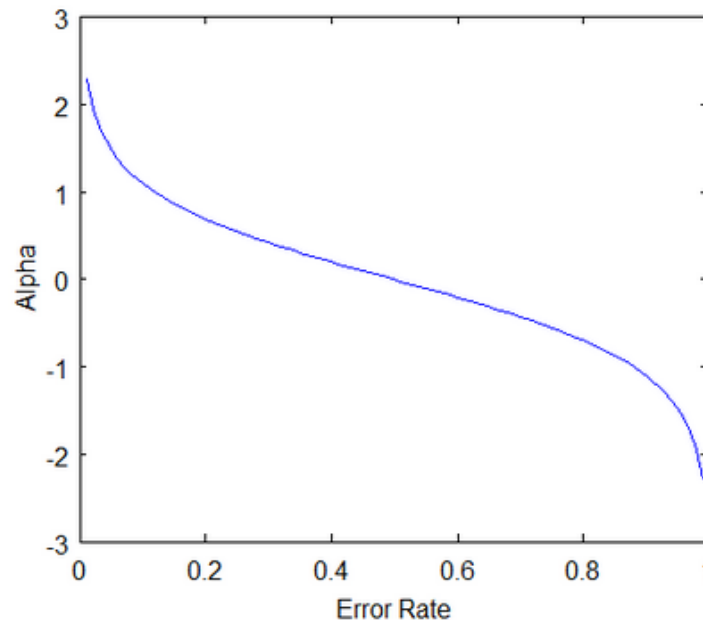
Total error will always be between 0 and 1.

**0 Indicates perfect stump, and 1 indicates horrible stump.**

## 5.2.1 AdaBoost Algorithm Example

### Step 3: Calculate the Influence

- From the graph we can see that **when there is no misclassification**, then we have no error (Total Error = 0), so the **“influence (alpha)” will be a large number**.
- When the classifier predicts **half right and half wrong**, then the Total Error = 0.5, and the **“influence (alpha)” will be 0**.
- If all the samples have been **incorrectly classified**, then the error will be very high (approx. to 1), and hence our **alpha value will be a negative integer**



## 5.2.1 AdaBoost Algorithm Example

### Step 4: Calculate TE and Performance

- The wrong predictions will be given more weight, whereas the correct predictions weights will be decreased.
- When we build our next model after updating the weights, more preference will be given to the points with higher weights.

$$\text{New sample weight} = \text{old weight} * e^{\pm \text{Amount of say } (\alpha)}$$

- There are four correctly classified samples and 1 wrong. Here, the sample weight of that datapoint is  $1/5$ , and the influence of the stump of Gender is 0.69.

New weights for correctly classified samples are:

$$\text{New sample weight} = \frac{1}{5} * \exp(-0.69)$$

$$\text{New sample weight} = 0.2 * 0.502 = 0.1004$$

For *wrongly classified* samples, the updated weights will be

$$\text{New sample weight} = \frac{1}{5} * \exp(0.69)$$

$$\text{New sample weight} = 0.2 * 1.994 = 0.3988$$

## 5.2.1 AdaBoost Algorithm Example

### Step 4: Calculate TE and Performance

- The **alpha value is negative** when the data point is correctly classified, and this ***decreases the sample weight*** from 0.2 to 0.1004.
- It is **positive** when there is **misclassification**, and this will ***increase the sample weight*** from 0.2 to 0.3988

Row No.	Gender	Age	Income	Illness	Sample Weights	New Sample Weights
1	Male	41	40000	Yes	1/5	0.1004
2	Male	54	30000	No	1/5	0.1004
3	Female	42	25000	No	1/5	0.1004
4	Female	40	60000	Yes	1/5	0.3988
5	Male	46	50000	Yes	1/5	0.1004



## 5.2.1 AdaBoost Algorithm Example

### Step 4: Calculate TE and Performance

- The total sum of the sample weights must be equal to 1, but here if we sum up all the new sample weights, we will get 0.8004.
- To bring this sum equal to 1, we will normalize these weights by dividing all the weights by the total sum of updated weights, which is 0.8004. So, after normalizing the sample weights, we get this dataset, and now the sum is equal to 1.

Row No.	Gender	Age	Income	Illness	Sample Weights	New Sample Weights
1	Male	41	40000	Yes	1/5	$0.1004/0.8004 = 0.1254$
2	Male	54	30000	No	1/5	$0.1004/0.8004 = 0.1254$
3	Female	42	25000	No	1/5	$0.1004/0.8004 = 0.1254$
4	Female	40	60000	Yes	1/5	$0.3988/0.8004 = 0.4982$
5	Male	46	50000	Yes	1/5	$0.1004/0.8004 = 0.1254$

## 5.2.1 AdaBoost Algorithm Example

### Step 5: Decrease the errors

- We need to make a new dataset to see if the errors decreased or not.
- For this, we will remove the “sample weights” and “new sample weights” columns and then, based on the “new sample weights,” divide our data points into buckets.

Row No.	Gender	Age	Income	Illness	New Sample Weights	Buckets
1	Male	41	40000	Yes	$0.1004/0.8004=0.1254$	0 to 0.1254
2	Male	54	30000	No	$0.1004/0.8004=0.1254$	0.1254 to 0.2508
3	Female	42	25000	No	$0.1004/0.8004=0.1254$	0.2508 to 0.3762
4	Female	40	60000	Yes	$0.3988/0.8004=0.4982$	0.3762 to 0.8744
5	Male	46	50000	Yes	$0.1004/0.8004=0.1254$	0.8744 to 0.9998

## 5.2.1 AdaBoost Algorithm Example

### Step 6: New Dataset

- The algorithm selects random numbers from 0-1.
- Since incorrectly classified records have higher sample weights, the probability of selecting those records is very high.
- Suppose the 5 random numbers our algorithm take is 0.38,0.26,0.98,0.40,0.55.
- We will see where these random numbers fall in the bucket, and according to it, we'll make our new dataset shown below.

Row No.	Gender	Age	Income	Illness
1	Female	40	60000	Yes
2	Male	54	30000	No
3	Female	42	25000	No
4	Female	40	60000	Yes
5	Female	40	60000	Yes

We see the data point, which was wrongly classified, has been selected 3 times because it has a higher weight



## 5.2.1 AdaBoost Algorithm Example

### Step 6: Repeat Previous Steps

1. Assign equal weights to all the data points.
2. Find the stump that does the best job classifying the new collection of samples by finding their Gini Index and selecting the one with the lowest Gini index.
3. Calculate the “Influence” and “Total error” to update the previous sample weights.
4. Normalize the new sample weights.

Iterate through these steps until and unless a low training error is achieved.



## 5.3 Gradient Boosting Machine (GBM)

- We use gradient boosting to solve classification and regression problems.
- It is a sequential ensemble learning technique where the performance of the model improves over iterations.
- This method creates the model in a stage-wise fashion. It infers the model by enabling the **optimization of an absolute differentiable loss function**.
- As we add each weak learner, a new model is created that gives a more precise estimation of the response variable.
- The gradient boosting algorithm requires the below components to function:
  - **Loss Function**
  - **Weak Learner**
  - **Additive Model**





## 5.3 Gradient Boosting Machine (GBM)

1. **Loss function:** To reduce errors in prediction, we need to optimize the loss function. Unlike in AdaBoost, the incorrect result is not given a higher weightage in gradient boosting. It tries to reduce the loss function by averaging the outputs from weak learners.
2. **Weak learner:** In gradient boosting, we require weak learners to make predictions. To get real values as output, we use regression trees. To get the most suitable split point, we create trees in a greedy manner, due to this the model overfits the dataset.
3. **Additive model:** In gradient boosting, we try to reduce the loss by adding decision trees. Also, we can minimize the error rate by cutting down the parameters. So, in this case, we design the model in such a way that the addition of a tree does not change the existing tree.



## 5.3.1 Gradient Boosting Machine (GBM) Example

- Consider the below data set. Here **Age, Sft., Location** are **independent variables** and **Price** is **Target variable**.

Age	Sft.	Location	Price
5	1500	5	480
11	2030	12	1090
14	1442	6	350
8	2501	4	1310
12	1300	9	400
16	1789	11	500

## 5.3.1 Gradient Boosting Machine (GBM)

### Example

**Step 1:** Calculate the average/mean of the target variable and consider it as predicted value for all instances.

$$\frac{480+1090+350+1310+400+500}{6} = 688$$

Age	Sft.	Location	Price	Average_Price
5	1500	5	480	688
11	2030	12	1090	688
14	1442	6	350	688
8	2501	4	1310	688
12	1300	9	400	688
16	1789	11	500	688

## 5.3.1 Gradient Boosting Machine (GBM) Example

**Step 2:** Calculate the residuals for each sample.

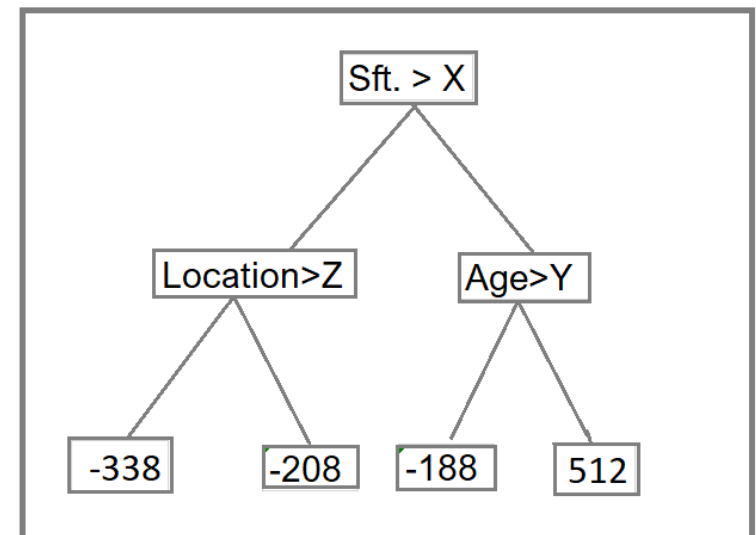
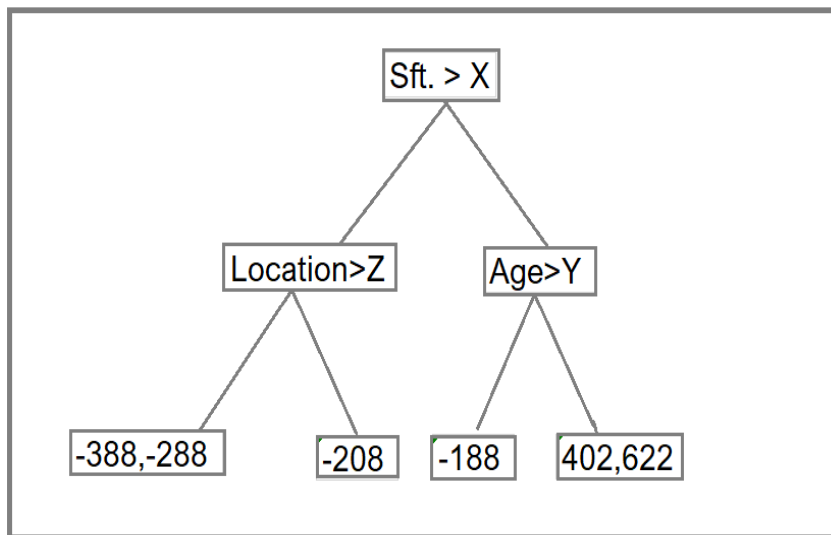
$$\text{Residual} = \text{Actual Value} - \text{Predicted Value}$$

Age	Sft.	Location	Price	Average_Price	Residual
5	1500	5	480	688	-208
11	2030	12	1090	688	402
14	1442	6	350	688	-338
8	2501	4	1310	688	622
12	1300	9	400	688	-288
16	1789	11	500	688	-188

## 5.3.1 Gradient Boosting Machine (GBM)

### Example

**Step 3:** Construct a decision tree. We build a tree with the goal of predicting the Residuals.



If there are more residuals than leaf nodes, we compute their average and place that inside the leaf.

## 5.3.1 Gradient Boosting Machine (GBM)

### Example

**Step 4:** Predict the target label using all the trees within the ensemble.

- Each sample passes through the decision nodes of the newly formed tree until it reaches a given leaf.
- The residual in the said leaf is used to predict the house price.

**Predicted Price = Average Price + Learning Rate \* Residual  
Predicted by decision tree**

We have initially taken 0.1 as learning rate.

Eg: For instance 3, Predicted Price =  $688 + (0.1 * (-338)) = 654.2$

For instance 1, Predicted Price =  $688 + (0.1 * (-208)) = 667.2$



## 5.3.1 Gradient Boosting Machine (GBM) Example

**Step 5 :** Compute the new residuals

Residual=Actual Value - Predicted Value

$$\Rightarrow 350 - 654.2 = -304.2$$

$$\Rightarrow 480 - 667.2 = -187.2$$

**When Price is 350 and 480 Respectively**

Age	Sft.	Location	Price	Average_Price	Residual	New Residual
5	1500	5	480	688	-208	-187.2
11	2030	12	1090	688	402	350.8
14	1442	6	350	688	-338	-304.2
8	2501	4	1310	688	622	570.8
12	1300	9	400	688	-288	-254.1
10	1789	11	500	688	-188	-169.2



## 5.3.1 Gradient Boosting Machine (GBM)

### Example

**Step 6 :** Repeat steps 3 to 5 until the number of iterations matches the number specified by the hyper parameter(numbers of estimators)

**Step 7 :**

- Once trained, use all of the trees in the ensemble to make a final prediction as to value of the target variable.
- The final prediction will be equal to the mean we computed in Step 1 plus all the residuals predicted by the trees that make up the forest multiplied by the learning rate.

=>Average Price + LR \* Residual Predicted by DT 1 + LR \* Residual Predicted by DT 2 + .....+LR\*Residual Predicted by DT N

**Where**

**LR :** Learning Rate

**DT:** Decision Tree

Eg: for instance 6, Final Predicted Price=

$688 + 0.1 * (-188) + 0.1 * (-169.2) + \dots$



# AdaBoost vs Gradient Boost Machine

Features	Gradient boosting	Adaboost
<b>Model</b>	It identifies complex observations by huge residuals calculated in prior iterations	The shift is made by up-weighting the observations that are miscalculated prior
<b>Trees</b>	The trees with weak learners are constructed using a greedy algorithm based on split points and purity scores. The trees are grown deeper with eight to thirty-two terminal nodes. The weak learners should stay a week in terms of nodes, layers, leaf nodes, and splits	The trees are called decision stumps.
<b>Classifier</b>	The classifiers are weighted precisely and their prediction capacity is constrained to learning rate and increasing accuracy	Every classifier has different weight assumptions to its final prediction that depend on the performance.

# AdaBoost vs Gradient Boost Machine

<b>Prediction</b>	<p>It develops a tree with help of previous classifier residuals by capturing variances in data.</p> <p>The final prediction depends on the maximum vote of the week learners and is weighted by its accuracy.</p>	<p>It gives values to classifiers by observing determined variance with data. Here all the week learners possess equal weight and it is usually fixed as the rate for learning which is too minimum in magnitude.</p>
<b>Short-comings</b>	<p>Here, the gradients themselves identify the shortcomings.</p>	<p>Maximum weighted data points are used to identify the shortcomings.</p>
<b>Loss value</b>	<p>Gradient boosting cut down the error components to provide clear explanations and its concepts are easier to adapt and understand</p>	<p>The exponential loss provides maximum weights for the samples which are fitted in worse conditions.</p>



# AdaBoost vs Gradient Boost Machine

## Applications

This method trains the learners and depends on reducing the loss functions of that week learner by training the residues of the model

Its focus on training the prior miscalculated observations and it alters the distribution of the dataset to enhance the weight on sample values which are hard for classification

## 5.4 XGBoost

- XGBoost (**Ext**reme **G**radient **B**oosting) is an optimized distributed gradient boosting library.
- It uses gradient boosting (GBM) framework at core and performs better than GBM framework alone.
- It is used for supervised ML problems

### Features of XGBoost:

1. **Parallel Computing:** It is enabled with parallel processing (using OpenMP); i.e., when you run xgboost, by default, it would use all the cores of your laptop/machine.
2. **Regularization:** GBM has no provision for regularization. Regularization is a technique used to avoid overfitting in linear and tree-based models.
3. **Enabled Cross Validation:** xgboost is enabled with internal CV function.
4. **Missing Values:** XGBoost is designed to handle missing values internally. The missing values are treated in such a manner that if there exists any trend in missing values, it is captured by the model.



## 5.4 XGBoost

### Features of XGBoost:

5. **Flexibility:** In addition to regression, classification, and ranking problems, it supports user-defined objective functions also. An objective function is used to measure the performance of the model given a certain set of parameters. Furthermore, it supports user defined evaluation metrics as well.
6. **Availability:** Currently, it is available for programming languages such as R, Python, Java, Julia, and Scala.
7. **Save and Reload:** XGBoost gives us a feature to save our data matrix and model and reload it later. Suppose, we have a large data set, we can simply save the model and use it in future instead of wasting time redoing the computation.
8. **Tree Pruning:** Unlike GBM, where tree pruning stops once a negative loss is encountered, XGBoost grows the tree upto `max_depth` and then prune backward until the improvement in loss function is below a threshold



## 5.5 Reinforcement Learning

- Reinforcement learning addresses the question of **how an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals.**
- Each time the agent performs an action in its environment, a trainer may provide a **reward or penalty** to indicate the desirability of the resulting state.
- The task of the agent is to learn from this indirect, delayed reward, to choose sequences of actions that produce the greatest cumulative reward.
- **For example**, when training an agent to play a game the trainer might provide a positive reward when the game is won, negative reward when it is lost, and zero reward in all other states.
- Reinforcement learning algorithms are related to dynamic programming algorithms frequently used to solve optimization problems.

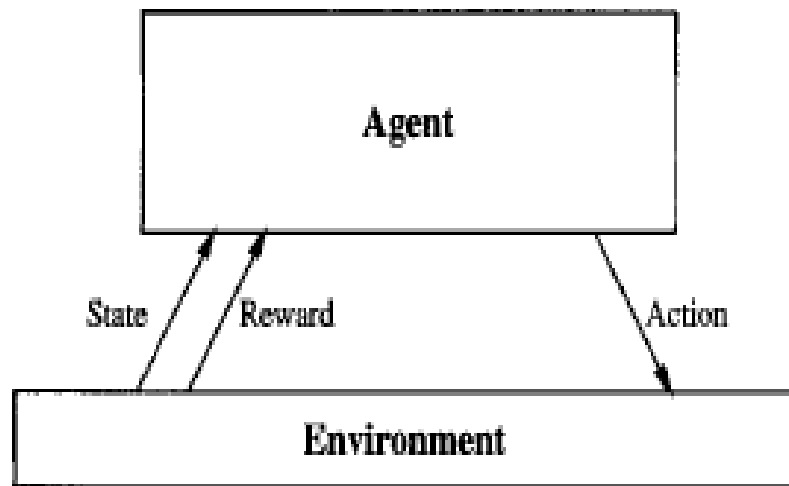


## 5.5 Reinforcement Learning

### 5.5.1 Introduction

- Consider building a learning robot. The robot, or agent, has a set of sensors to observe the **state** of its environment, and a set of **actions** it can perform to alter this state.
- Its task is to learn a control strategy, or **policy**, for choosing actions that achieve its goals.
- The task of the robot is to perform sequences of actions, observe their consequences, and learn a control policy.
- The control policy we desire is one that, from any initial state, chooses actions that maximize the reward accumulated over time by the agent.
- **For example**, the robot may have a goal of docking onto its battery charger whenever its battery level is low.
- We assume that the goals of the agent can be defined by a **reward** function that assigns a numerical value—an immediate payoff—to each distinct action the agent may take from each distinct state.

## 5.5 Reinforcement Learning



Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \dots, \text{ where } 0 \leq \gamma < 1$$

- An agent interacting with its environment.
- The agent exists in an environment described by some set of possible states  $S$ . *It can* perform any of a set of possible actions  $A$ .
- Each time it performs an action  $a$ , in some state  $s$  the agent receives a real-valued reward  $r$ , that indicates the immediate value of this state-action transition.
- This produces a sequence of states  $s_i$ , actions  $a_i$ , and immediate rewards  $r_i$  as shown in the figure.
- The agent's task is to learn a control policy,  $\pi : S \rightarrow A$ , that maximizes the expected sum of these rewards.

## 5.5 Reinforcement Learning

Reinforcement learning problem differs from other function approximation tasks in several important respects.

### 1. **Delayed reward:**

- The task of the agent is to learn a target function  $n$  that maps from the current state  $s$  to the optimal action  $a = \pi(s)$ .
- In reinforcement learning, the trainer provides only a sequence of immediate reward values as the agent executes its sequence of actions.
- The agent, therefore, faces the problem of **temporal credit assignment**: *determining which of the actions in its sequence are to be credited with producing the eventual rewards.*

### 2. **Exploration:**

- *In reinforcement learning, the agent influences the distribution of training examples by the action sequence it chooses.*
- *The learner faces a tradeoff in choosing whether to favor exploration of unknown states and actions or states and actions that it has already learned will yield high reward*





## 5.5 Reinforcement Learning

### 3. **Partially observable states:**

- In many practical situations, agents sensors provide only partial information.
- For example, a robot with a forward-pointing camera cannot see what is behind it.
- In such cases, it may be necessary for the agent to consider its previous observations together with its current sensor data when choosing actions.

### 4. ***Life-long learning:***

- Unlike isolated function approximation tasks, robot learning often requires that the robot learn several related tasks within the same environment, using the same sensors
- For example, a mobile robot may need to learn how to dock on its battery charger, how to navigate through narrow corridors, and how to pick up output from laser printers.

## 5.5 Reinforcement Learning

### Approaches to implement Reinforcement Learning

There are mainly three ways to implement reinforcement-learning in ML:

- The **model-based** algorithms use transition and reward functions to estimate the optimal policy and create the model. In contrast, **model-free** algorithms learn the consequences of their actions through the experience without transition and reward function.
- The **value-based** method trains the value function to learn which state is more valuable and take action. On the other hand, **policy-based** methods train the policy directly to learn which action to take in a given state.
- In the **off-policy**, the algorithm evaluates and updates a policy that differs from the policy used to take an action. Conversely, the **on-policy** algorithm evaluates and improves the same policy used to take an action.

# 5.5 Reinforcement Learning

## 5.2 Q-Learning

- Q-learning is a **model-free, value-based, off-policy** algorithm that will find the best series of actions based on the agent's current state.
- Depending on where the agent is in the environment, it will decide the next action to be taken.
- The “Q” stands for quality. Quality represents how valuable the action is in maximizing future rewards.



Figure 3: Components of Q-Learning

# 5.5 Reinforcement Learning

## 5.5.2 Q-Learning

### Example: Advertisement recommendation system with Q-Learning

- In a normal ad recommendation system, the ads you get are based on your previous purchases or websites you may have visited. If you've bought a TV, you will get recommended TVs of different brands.
- Using Q-learning, we can optimize the ad recommendation system to recommend products that are frequently bought together. The reward will be if the user clicks on the suggested product..



Figure 4: Ad Recommendation System



Figure 5: Ad Recommendation System with Q-Learning

## 5.5 Reinforcement Learning

### 5.5.2 Q-Learning

#### Key Terminologies in Q-learning

- **States(s)**: the current position of the agent in the environment.
- **Action(a)**: a step taken by the agent in a particular state.
- **Rewards**: for every action, the agent receives a reward and penalty.
- **Episodes**: the end of the stage, where agents can't take new action. It happens when the agent has achieved the goal or failed.
- **$Q(S_{t+1}, a)$** : expected optimal Q-value of doing the action in a particular state.
- **$Q(S_t, A_t)$** : it is the current estimation of  $Q(S_{t+1}, a)$ .
- **Q-Table**: the agent maintains the Q-table of sets of states and actions.
- **Temporal Differences(TD)**: used to estimate the expected value of  $Q(S_{t+1}, a)$  by using the current state and action and previous state and action.



## 5.5 Reinforcement Learning

### 5.5.2 Q-Learning

#### Bellman Equation

- The Bellman Equation determines the value of a particular state and comes to a conclusion on how valuable it is in that state.
- The Q-function uses the Bellman equation and uses two inputs: the state (s) and the action (a).
- We can choose the sequence of actions that generate the best rewards which we can represent as the Q value using the equation:



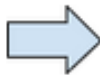

$$Q(s, a) = r(s, a) + \gamma \max_a Q(s', a)$$

- $Q(s, a)$  stands for the Q Value that has been yielded at state 's' and selecting action 'a'.
- This is calculated by  $r(s, a)$  which stands for the immediate reward received + the best Q Value from state 's'.
- $\gamma$  is the discount factor that controls and determines the importance to the current state

## 5.5 Reinforcement Learning

### 5.5.2 Q-Learning

- In order to manage and determine which is the best path, we use **Q-table**.
- **Q-table** is just a simple lookup table used to calculate as well as manage the maximum expected future rewards.
- We can easily identify the best action for each state in the environment.
- Using the Bellman Equation at each state, we get the expected future state and reward, then save it in a table to make comparisons to other states.
- For example:

State:	Action:			
				
0	0	1	0	0
..				
250	-2.07469	-2.34655	-1.99878	-2.03458
..				
500	11.47930	7.23467	13.47290	9.53478

## **5.5 Reinforcement Learning**

### **5.5.2 Q-Learning**

#### **Steps in Q-Learning:**

##### **1. Initialize the Q-Table**

The first step is creating the Q-table

$n$  = number of actions

$m$  = number of states

For example,  $n$  could be left, right, up, or down, while  $m$  could be start, idle, correct move, wrong move, and end in a game.

##### **2. Choose and Perform an Action**

Our Q-table should all have 0's as no action has been performed. We then choose an action and update it in our Q-table in the correct section. This states that the action has been performed.

##### **3. Calculate the Q-Value using Bellman Equation**

Using the Bellman Equation, calculate the value of the actual reward and the Q-value for the action just performed.

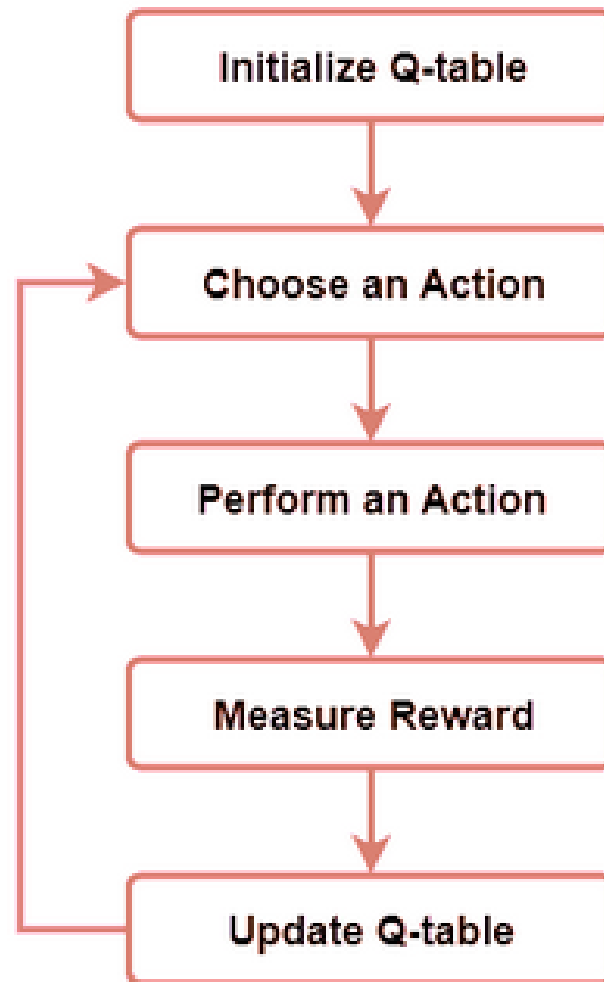
##### **4. Continue Steps 2 and 3**

Repeat Steps 2 and 3 till an episode ends or the Q-table is filled.

## 5.5 Reinforcement Learning

### 5.5.2 Q-Learning

#### Steps in Q-Learning:





## 5.5 Reinforcement Learning

### 5.5.2 Q-Learning

Examples:

<https://youtu.be/J3qX50yyiU0?si=G0w-MIFDPbGDT0go>

<https://blog.floydhub.com/an-introduction-to-q-learning-reinforcement-learning/>

<https://towardsdatascience.com/reinforcement-learning-explained-visually-part-4-q-learning-step-by-step-b65efb731d3e>





## Unit-5 Question Bank

1. Discuss about bagging method with a neat sketch
2. Illustrate about boosting in ensemble learning.
3. Explain about stacking in ensemble learning.
4. Differentiate bagging and boosting.
5. Explain the steps in AdaBoost algorithm with an example.
6. Demonstrate Gradient Boosting Machines with an example.
7. Discuss the features of XGBoost Algorithm.
8. Differentiate AdaBoost and Gradient Boost Machines.
9. Explain various approaches to implement Reinforcement Learning.
10. Discuss in detail Q-Learning algorithm with an example.