Artificial Intelligence Diploma

Machine Learning Session 3 & 4

NEUROTECH

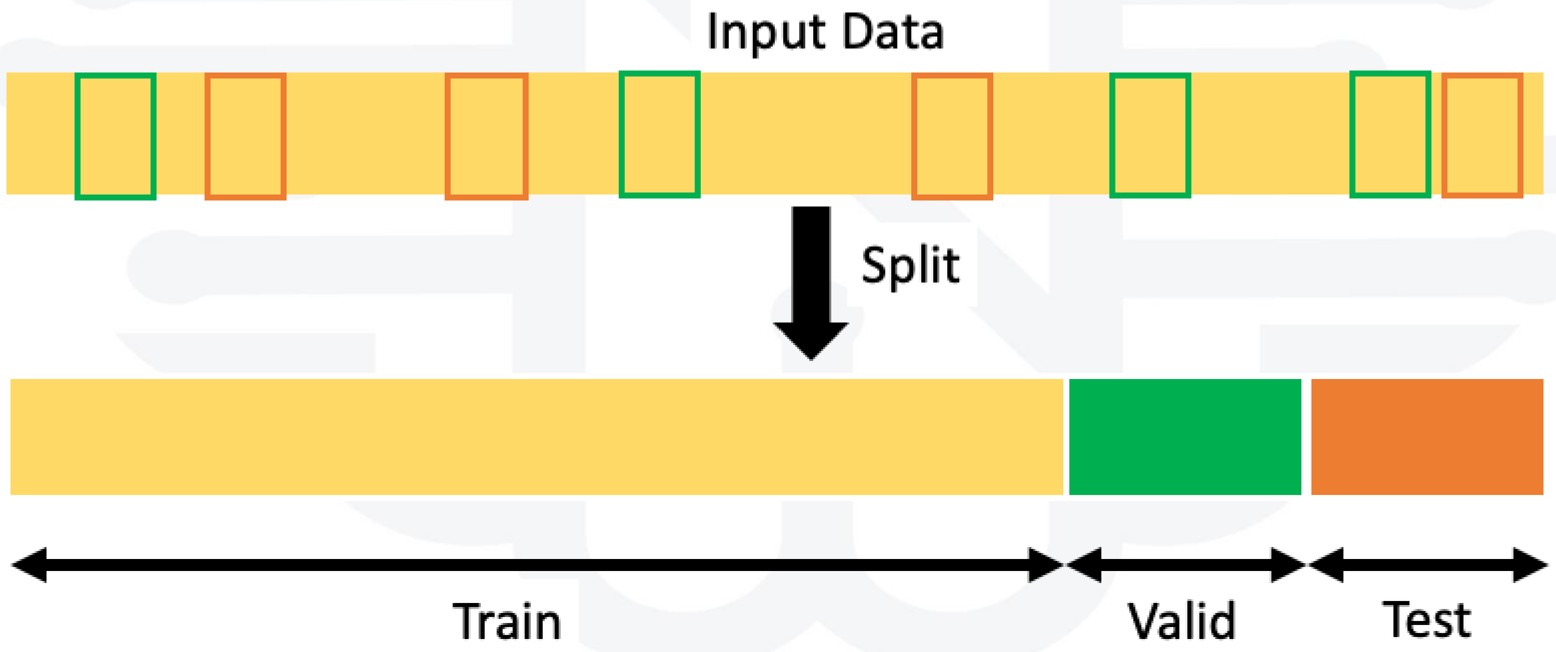
Agenda

Error in Machine Learning

NeuroTech

Error in Machine Learning

In machine learning, data is typically divided into three main sets: the training set, the validation set, and the test set. Each of these sets serves a specific purpose in the model development and evaluation process



NEUROTECH

Error in Machine Learning

- **Training Set:**
  - The training set is the largest portion of the dataset and is used to train the machine learning model. It consists of input data (features) and their corresponding target values (labels) for supervised learning tasks.
  - The model learns from the training set by optimizing its parameters to minimize the error between its predictions and the actual target values.
  - Training set examples are used to update the model's internal parameters or weights through optimization algorithms such as gradient descent.

Error in Machine Learning

❑ **Validation Set (or Development Set):**

- The validation set is used to fine-tune the model's hyperparameters and to monitor its performance during training. It's essentially a smaller portion of the data that the model doesn't directly learn from.
- After each training iteration (or epoch), the model's performance is evaluated on the validation set using metrics like accuracy, mean squared error, or others relevant to the problem.
- Hyperparameters, such as learning rates or the depth of decision trees, are adjusted based on the validation set's performance to improve the model's generalization.
- The validation set helps in detecting overfitting (high variance) and underfitting (high bias) by assessing the model's performance on unseen data.

Error in Machine Learning

- **Test Set:**

  - The test set is a completely independent dataset that the model has never seen during training or validation. It is used to provide an unbiased evaluation of the model's generalization to new, unseen data.
  - The model's final performance is assessed on the test set using the same evaluation metrics used for the validation set.
  - The test set helps estimate how the model is likely to perform in real-world applications and is crucial for making decisions about model deployment.

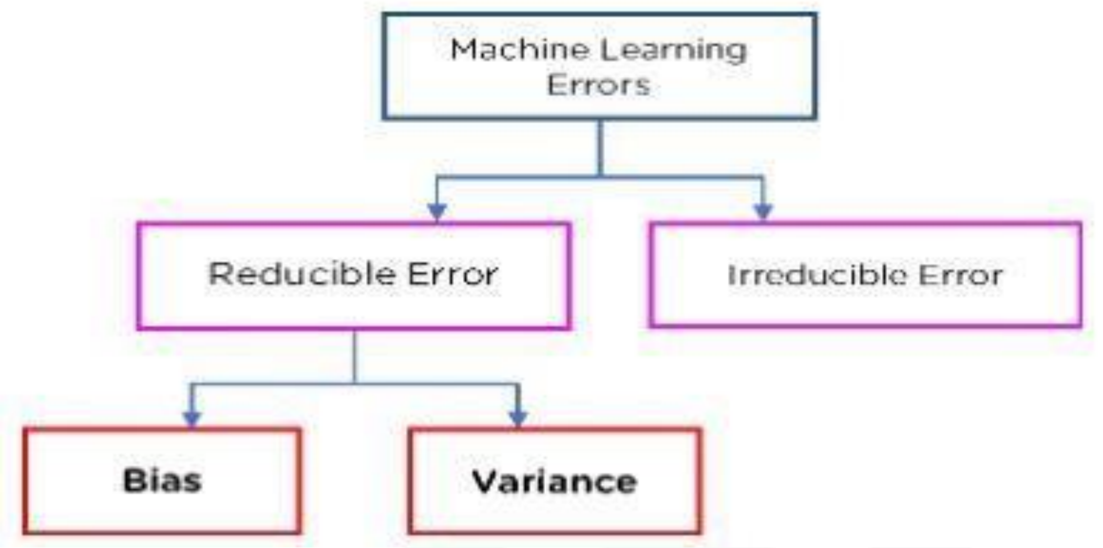
Error in Machine Learning

- **There are two main types of errors present in any machine learning model. They are Reducible Errors and Irreducible Errors.**

1. Irreducible errors are errors which will always be present in a machine learning model, because of unknown variables, and whose values cannot be reduced.
2. Reducible errors are those errors whose values can be further reduced to improve a model. They are caused because our model's output function does not match the desired output function and can be optimized.

Error in Machine Learning

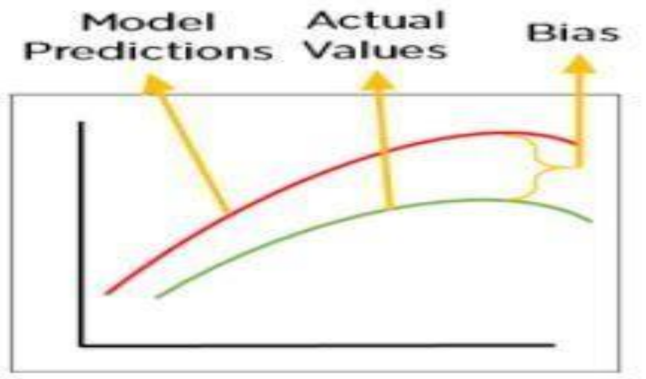
- We can further divide reducible errors into two: Bias and Variance.



Error in Machine Learning

What is Bias?

- To make predictions, our model will analyze our data and find patterns in it. Using these patterns, we can make generalizations about certain instances in our data. Our model after training learns these patterns and applies them to the test set to predict them.
- Bias is the difference between our actual and predicted values. Bias is the simple assumptions that our model makes about our data to be able to predict new data.



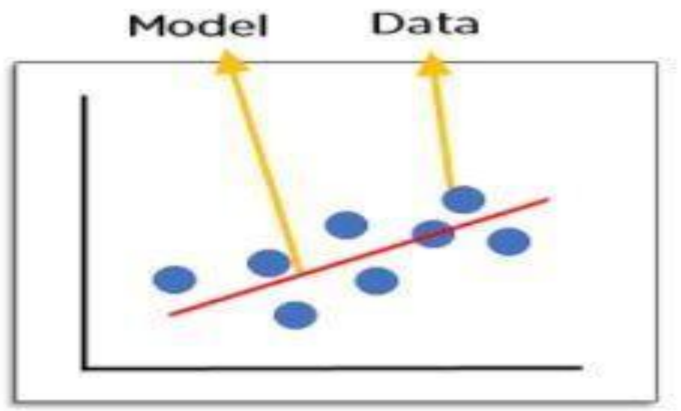
Error in Machine Learning

When the Bias is high, assumptions made by our model are too basic, the model can’t capture the important features of our data. This means that our model hasn’t captured patterns in the training data and hence cannot perform well on the testing data too. If this is the case, our model cannot perform on new data and cannot be sent into production.

This instance, where the model cannot find patterns in our training set and hence fails for both seen and unseen data, is called Underfitting.

Error in Machine Learning

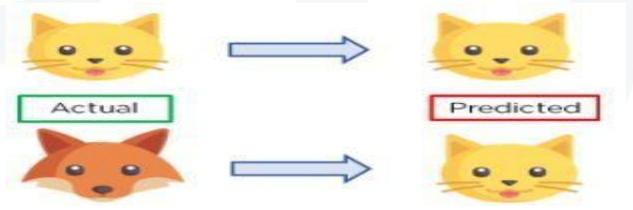
The below figure shows an example of Underfitting. As we can see, the model has found no patterns in our data and the line of best fit is a straight line that does not pass through any of the data points. The model has failed to train properly on the data given and cannot predict new data either.



Error in Machine Learning

What is Variance?

- Variance is the very opposite of Bias. During training, it allows our model to ‘see’ the data a certain number of times to find patterns in it. If it does not work on the data for long enough, it will not find patterns and bias occurs. On the other hand, if our model is allowed to view the data too many times, it will learn very well for only that data. It will capture most patterns in the data, but it will also learn from the unnecessary data present, or from the noise.
- We can define variance as the model’s sensitivity to fluctuations in the data. Our model may learn from noise. This will cause our model to consider trivial features as important.



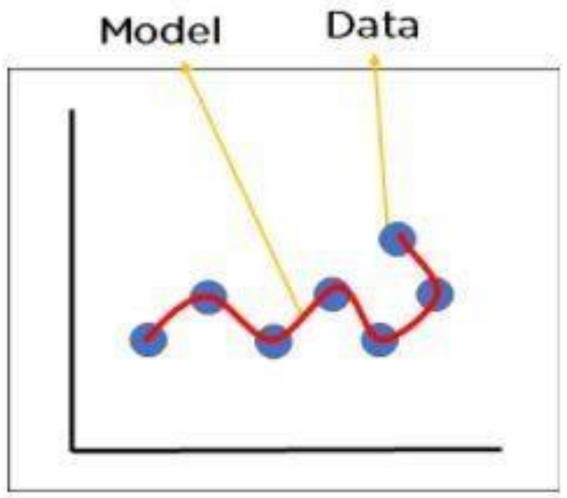
NeuroTech

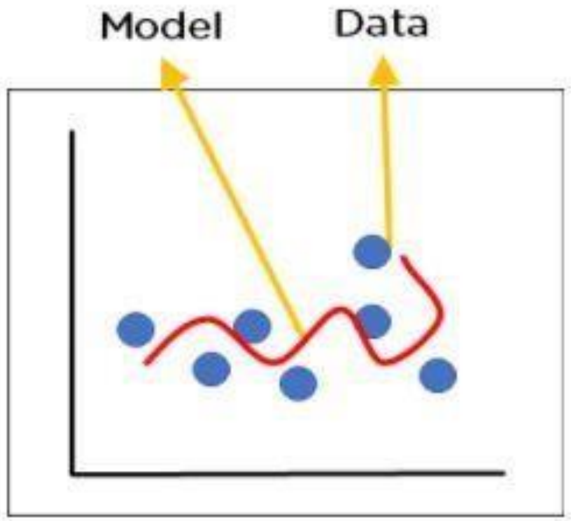
Error in Machine Learning

In the Previous figure, we can see that our model has learned extremely well for our training data, which has taught it to identify cats. But when given new data, such as the picture of a fox, our model predicts it as a cat, as that is what it has learned. This happens when the Variance is high, our model will capture all the features of the data given to it, including the noise, will tune itself to the data, and predict it very well but when given new data, it cannot predict on it as it is too specific to training data.

Hence, our model will perform really well on testing data and get high accuracy but will fail to perform on new, unseen data. New data may not have the exact same features and the model won’t be able to predict it very well. This is called Overfitting.

Error in Machine Learning





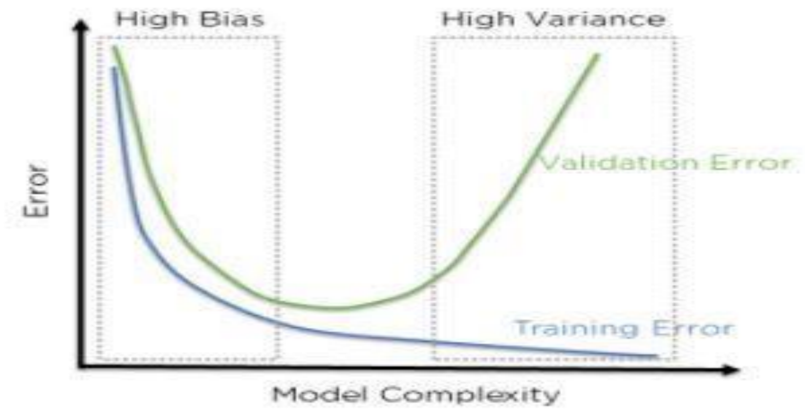
Over-fitted model where we see model performance on, a) training data b) new data

NEUROTECH

Error in Machine Learning

Bias-Variance Tradeoff

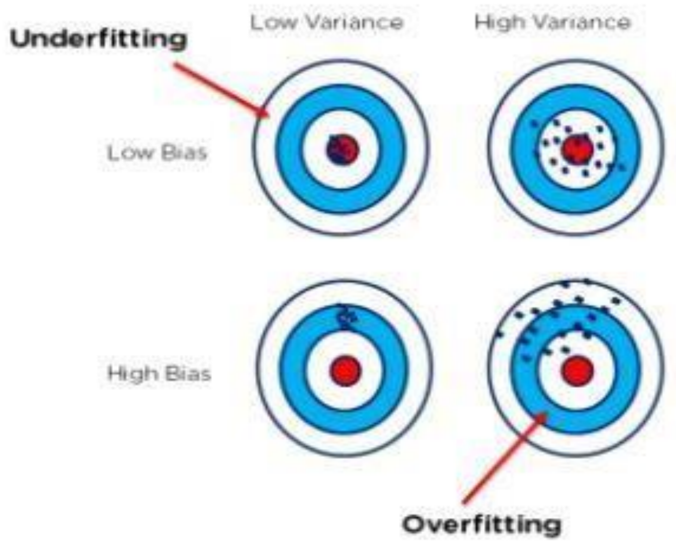
- For any model, we have to find the perfect balance between Bias and Variance. This just ensures that we capture the essential patterns in our model while ignoring the noise present it in. This is called Bias-Variance Tradeoff. It helps optimize the error in our model and keeps it as low as possible.
- An optimized model will be sensitive to the patterns in our data, but at the same time will be able to generalize to new data. In this, both the bias and variance should be low so as to prevent overfitting and underfitting.



NEUROTECH

Error in Machine Learning

In the Previous figure, we can see that when bias is high, the error in both testing and training set is also high.If we have a high variance, the model performs well on the testing set, we can see that the error is low, but gives high error on the training set. We can see that there is a region in the middle, where the error in both training and testing set is low and the bias and variance is in perfect balance.



NeuroTech

Underfitting

Underfitting

Underfitting, also known as high bias, happens when a machine learning model is too simplistic to represent the underlying structure of the data. The model is not complex enough to capture the relationships between the features and the target variable, resulting in poor predictive performance.

Characteristics and Causes of Underfitting:

1. **High Training Error:** The model performs poorly on the training data itself. The training error is high because the model cannot adequately fit the training data.
2. **High Testing Error:** Underfitting is often characterized by high testing error as well. The model fails to generalize to new, unseen data, leading to poor out-of-sample performance.
3. **Simple Models:** Underfit models are typically simple, with low complexity. For example, a linear regression model applied to a highly non-linear dataset can result in underfitting.
4. **Inadequate Feature Representation:** Underfitting can occur when the feature representation of the data is insufficient to capture the underlying relationships. This can be addressed by adding more relevant features or performing better feature engineering.
5. **Model Complexity:** If the chosen model is too simple for the problem at hand, it may not be able to represent the data adequately. In such cases, more complex models may be needed.

Underfitting

Mitigating Underfitting:

To mitigate underfitting and improve model performance:

1. **Increase Model Complexity:** Use more complex models that have the capacity to capture intricate patterns in the data. For example, consider using decision trees with greater depth, neural networks with more layers, or more sophisticated algorithms.
2. **Feature Engineering:** Enhance the feature representation of the data by adding relevant features, interactions, or transformations. Proper feature engineering can make a simple model more effective.
3. **Optimize Hyperparameters:** Adjust hyperparameters, such as the learning rate, regularization strength, or tree depth, to find the right balance between model simplicity and complexity.
4. **Collect More Data:** In some cases, underfitting can be addressed by collecting more data, especially if the model is data-hungry and the existing dataset is small.
5. **Consider Ensemble Methods:** Ensemble methods like Random Forest or Gradient Boosting can be more resilient to underfitting because they combine multiple models.

Underfitting

Signs of Underfitting:

- High training error
- High testing error
- Poor performance on new data
- Overly simplistic model
- Inadequate feature representation

Summary:

- Underfitting is a common challenge in machine learning, where a model is too simple to capture the underlying patterns in the data. Addressing underfitting often involves increasing model complexity, improving feature engineering, optimizing hyperparameters, or using ensemble methods to achieve better predictive performance.

Overfitting

Overfitting

Overfitting occurs when a machine learning model is excessively complex, fitting the training data closely, but failing to generalize well to new, unseen data. It essentially "memorizes" the training data, including the noise, rather than capturing the underlying patterns.

Signs of Overfitting:

- Low training error
- High testing error
- Poor performance on new data
- Complex model with many parameters
- Fitting noise and fluctuations

Overfitting

Characteristics and Causes of Overfitting:

1. **Low Training Error:** Overfit models typically have low training error because they fit the training data extremely well, often achieving near-perfect accuracy.
2. **High Testing Error:** High testing error is the hallmark of overfitting. The model's performance on new data is significantly worse than on the training data.
3. **Complex Models:** Overfit models are usually complex, with a high number of parameters. Examples include deep neural networks with many layers or decision trees with great depth.
4. **Inadequate Amount of Data:** Overfitting is more likely when the amount of training data is small relative to the model's complexity. In such cases, the model may find it easier to memorize the data rather than learn general patterns.
5. **High Model Capacity:** Models with high capacity can represent intricate relationships but are also more prone to overfitting. When the model capacity greatly exceeds the complexity of the problem, overfitting can occur.
6. **Noisy Data:** If the training data contains noise or outliers, complex models are more likely to fit this noise, leading to overfitting.

Overfitting

Mitigating Overfitting:

To mitigate overfitting and improve model generalization:

1. **Simplify the Model:** Reduce the complexity of the model by reducing the number of parameters or layers. For example, in the case of decision trees, you can limit tree depth.
2. **Regularization:** Apply regularization techniques, such as L1 (Lasso) or L2 (Ridge) regularization for linear models or dropout for neural networks, to penalize large model weights.
3. **More Data:** Collect additional data to provide the model with more examples to learn from. More data can help the model generalize better.
4. **Feature Selection:** Carefully select relevant features and reduce the dimensionality of the data to focus on the most important information.
5. **Cross-Validation:** Use techniques like k-fold cross-validation to assess the model's performance on different subsets of the data, helping identify overfitting.
6. **Early Stopping:** In iterative training algorithms like gradient descent, stop training when the model's performance on a validation set starts to degrade.
7. **Ensemble Methods:** Ensemble methods like Random Forest, which combine multiple models, can help reduce overfitting.

Overfitting

Summary:

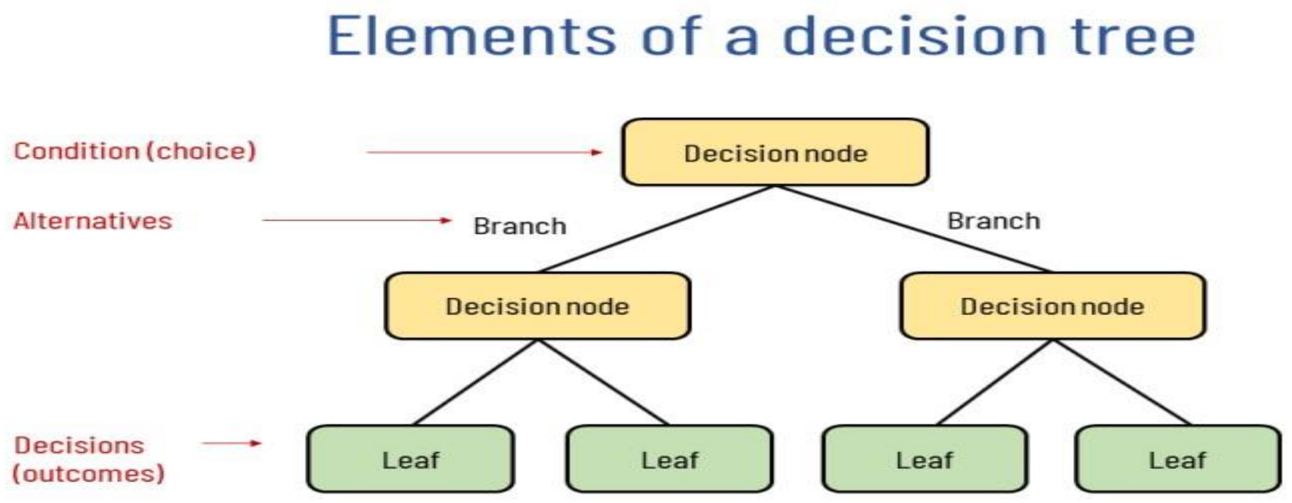
- Overfitting is a common problem in machine learning, where a model learns the training data too well, leading to poor performance on new data. To address overfitting, it's essential to simplify the model, apply regularization, collect more data, perform feature selection, use cross-validation, employ early stopping, or consider ensemble methods. The goal is to strike a balance between model complexity and generalization.

Decision Tree

NeuroTech

Introduction

A Decision tree is a popular machine learning model used for both classification and regression tasks. It is a supervised learning algorithm that is particularly known for its simplicity, interpretability, and versatility. Decision trees are used to make decisions or predictions by recursively splitting the data into subsets based on the values of the features, and the final outcome is determined by the path taken through the tree.



Introduction

Components and characteristics of decision trees:

- **Nodes**: Decision trees consist of nodes that represent a specific feature or attribute. There are two main types of nodes:
  - **Root Node**: This is the top node of the tree and represents the initial dataset. It is the starting point for making decisions.
  - **Internal Nodes**: These nodes represent feature attributes and are used to split the data into subsets based on specific conditions or rules.
  - **Leaf Nodes**: Leaf nodes are the endpoints of the tree, and they represent the final decisions or outcomes.
- **Edges**: Edges connect nodes and represent the path taken through the tree based on specific feature values.

- **Splitting**: The process of dividing the dataset at each internal node is known as "splitting." The decision tree algorithm selects the best feature and a splitting criterion (e.g., Gini impurity for classification or mean squared error for regression) to create two or more child nodes based on the feature's values.

Decision Tree - Regression

NeuroTech

Decision Tree - Regression

- Decision tree builds regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.

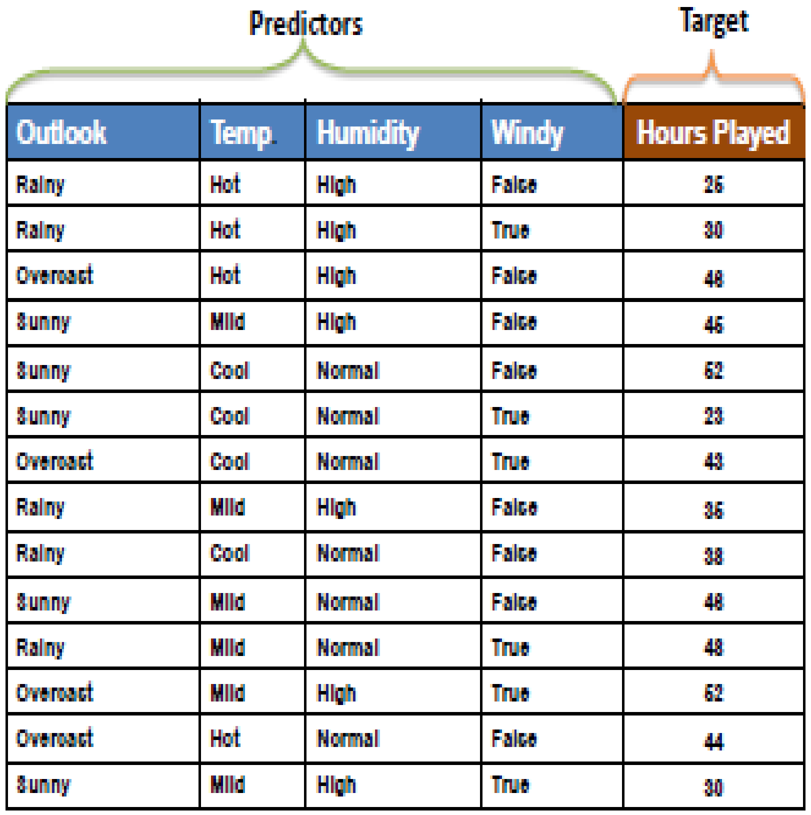
- The final result is a tree with **decision nodes** and **leaf nodes**.

- A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target.

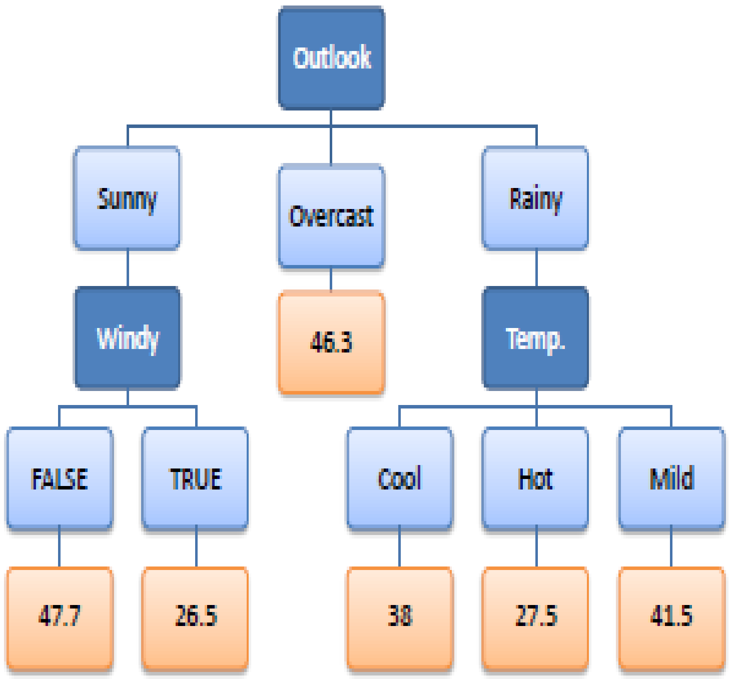
- The topmost decision node in a tree which corresponds to the best predictor called **root node**.

- Decision trees can handle both categorical and numerical data.

Decision Tree - Regression



Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30



How Decision Tree Work ?

NeuroTech

Decision Tree - Regression

Decision Tree Algorithm

- The core algorithm for building decision trees called **ID3** by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with Standard Deviation Reduction.

Standard Deviation

- A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). We use standard deviation to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous its standard deviation is zero.

Decision Tree - Regression

a) Standard deviation for **one** attribute:

Hours Played
25
30
46
45
52
23
43
35
38
46
48
52
44
30

$$Count = n = 14$$

$$Average = \bar{x} = \frac{\sum x}{n} = 39.8$$

$$Standard\ Deviation = S = \sqrt{\frac{\sum(x-\bar{x})^2}{n}} = 9.32$$

$$Coeffeicient\ of\ Variation = CV = \frac{S}{\bar{x}} * 100\% = 23\%$$

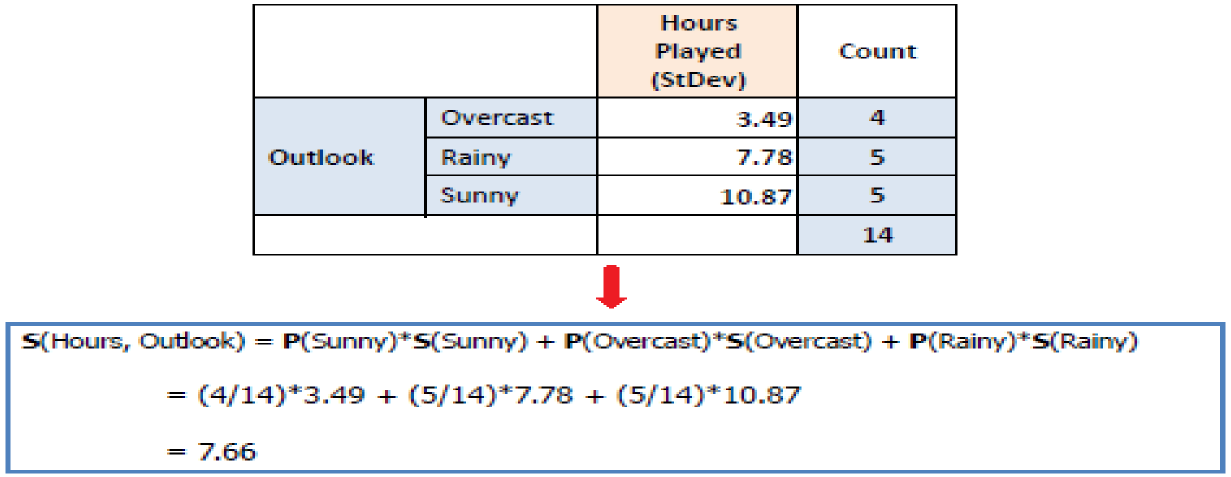
- Standard Deviation (**S**) is for tree building (branching).
- Coefficient of Deviation (**CV**) is used to decide when to stop branching. We can use Count (**n**) as well.
- Average (**Avg**) is the value in the leaf nodes.

NEUROTECH

Decision Tree - Regression

b) Standard deviation for **two** attributes (target and predictor):

$$S(T,X) = \sum_{c \in X} P(c)S(c)$$



		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

S(Hours, Outlook) = **P**(Sunny)***S**(Sunny) + **P**(Overcast)***S**(Overcast) + **P**(Rainy)***S**(Rainy)

= (4/14)*3.49 + (5/14)*7.78 + (5/14)*10.87

= 7.66

Decision Tree - Regression

Standard Deviation Reduction

- The standard deviation reduction is based on the decrease in standard deviation after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest standard deviation reduction (i.e., the most homogeneous branches).

Step 1: The standard deviation of the target is calculated.

Standard deviation (Hours Played) = 9.32

Decision Tree - Regression

- **Step 2**: The dataset is then split on the different attributes. The standard deviation for each branch is calculated. The resulting standard deviation is subtracted from the standard deviation before the split. The result is the standard deviation reduction.

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR= 0.48		

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

$$SDR(T,X) = S(T) - S(T,X)$$

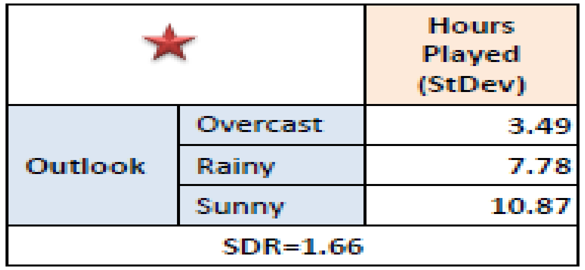
SDR(Hours , Outlook) = **S**(Hours) – **S**(Hours, Outlook)

= 9.32 – 7.66 = 1.66

NEUROTECH

Decision Tree - Regression

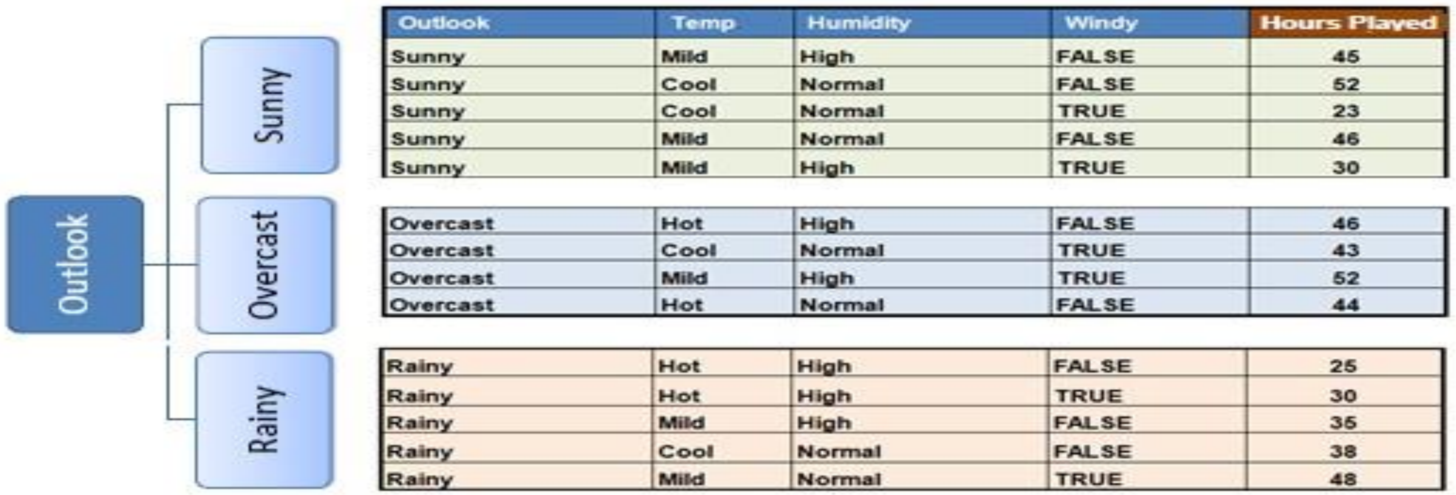
- **Step 3**: The attribute with the largest standard deviation reduction is chosen for the decision node.



★		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

Decision Tree - Regression

- **Step 4a**: The dataset is divided based on the values of the selected attribute. This process is run recursively on the non-leaf branches, until all data is processed.



Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30

Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44

Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48

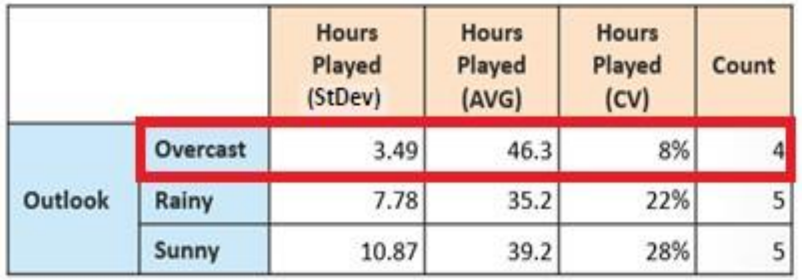
- In practice, we need some termination criteria. For example, when coefficient of deviation (**CV**) for a branch becomes smaller than a certain threshold (e.g., 10%) and/or when too few instances (**n**) remain in the branch (e.g., 3).

NeuroTech

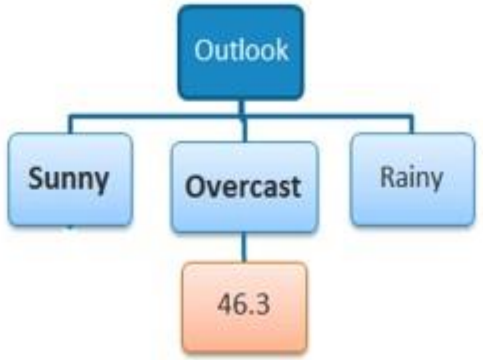
Decision Tree - Regression

- **Step 4b**: "Overcast" subset does not need any further splitting because its CV (8%) is less than the threshold (10%). The related leaf node gets the average of the "Overcast" subset.

Outlook - Overcast

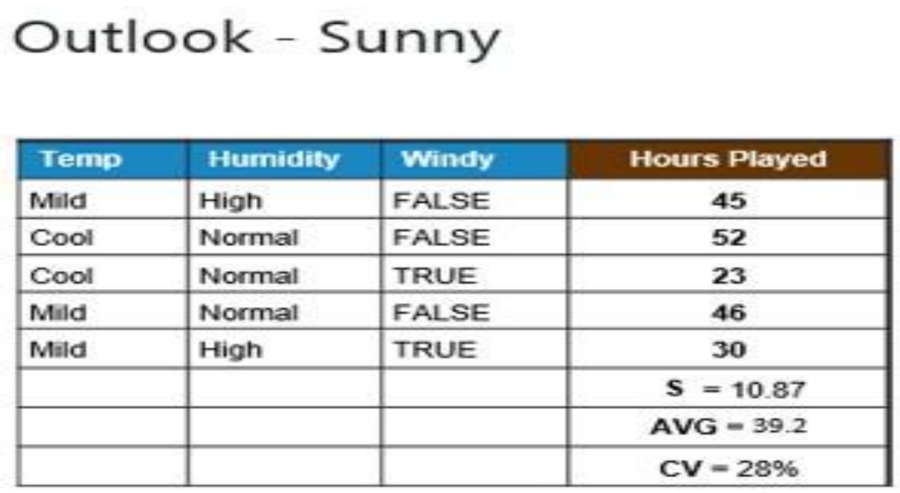


		Hours Played (StDev)	Hours Played (AVG)	Hours Played (CV)	Count
Outlook	Overcast	3.49	46.3	8%	4
	Rainy	7.78	35.2	22%	5
	Sunny	10.87	39.2	28%	5



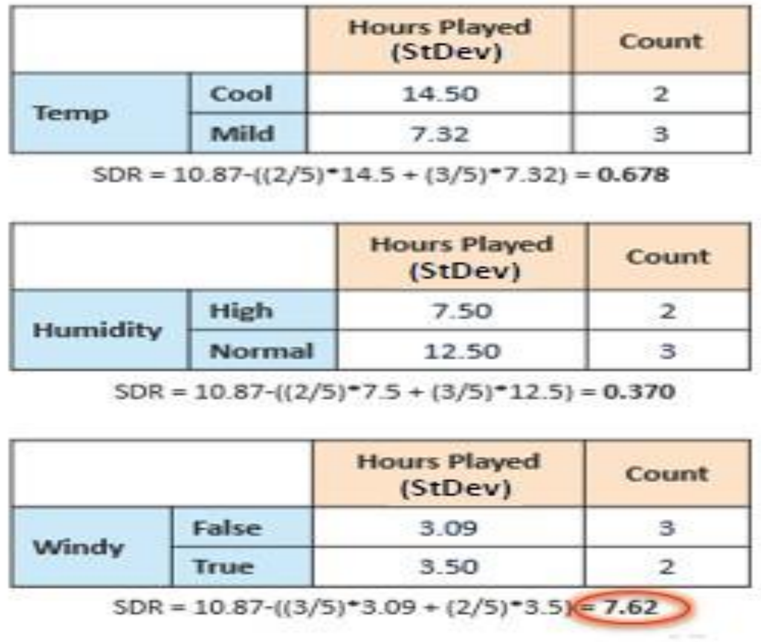
Decision Tree - Regression

- **Step 4c**: However, the "Sunny" branch has an CV (28%) more than the threshold (10%) which needs further splitting. We select "Temp" as the best best node after "Outlook" because it has the largest SDR.



Outlook - Sunny

Temp	Humidity	Windy	Hours Played
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Cool	Normal	TRUE	23
Mild	Normal	FALSE	46
Mild	High	TRUE	30
			S = 10.87
			AVG = 39.2
			CV = 28%



		Hours Played (StDev)	Count
Temp	Cool	14.50	2
	Mild	7.32	3

SDR = 10.87-((2/5)*14.5 + (3/5)*7.32) = **0.678**

		Hours Played (StDev)	Count
Humidity	High	7.50	2
	Normal	12.50	3

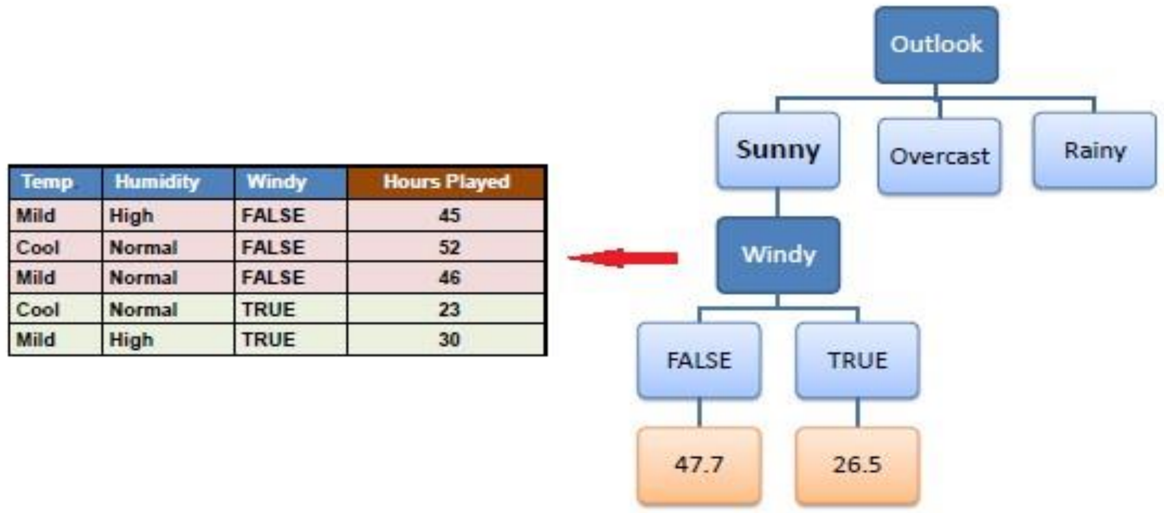
SDR = 10.87-((2/5)*7.5 + (3/5)*12.5) = **0.370**

		Hours Played (StDev)	Count
Windy	False	3.09	3
	True	3.50	2

SDR = 10.87-((3/5)*3.09 + (2/5)*3.5) = **7.62**

Decision Tree - Regression

- Because the number of data points for both branches (FALSE and TRUE) is equal or less than 3 we stop further branching and assign the average of each branch to the related leaf node.

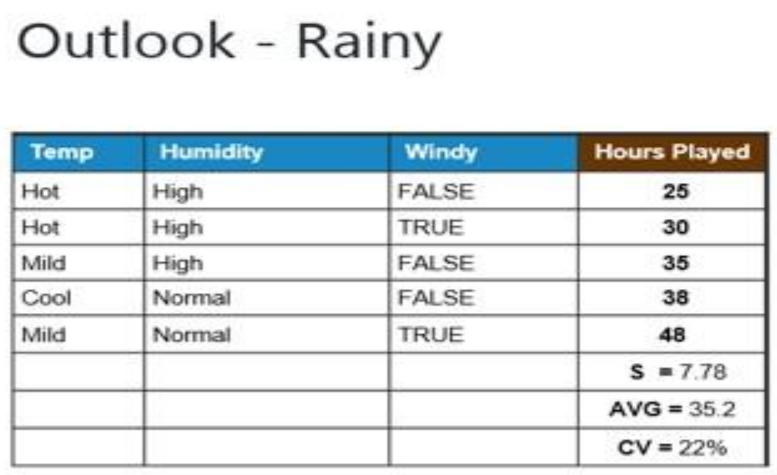


Temp	Humidity	Windy	Hours Played
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Mild	Normal	FALSE	46
Cool	Normal	TRUE	23
Mild	High	TRUE	30

NEUROTECH

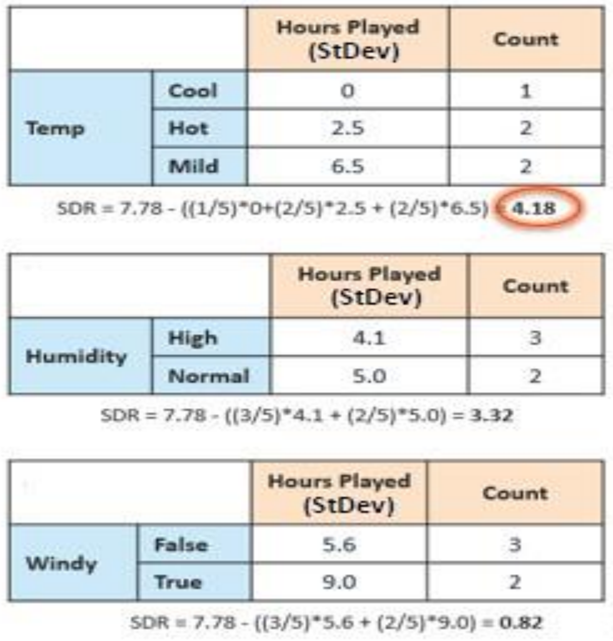
Decision Tree - Regression

- **Step 4d**: Moreover, the "rainy" branch has an CV (22%) which is more than the threshold (10%). This branch needs further splitting. We select "Temp" as the best best node because it has the largest SDR.



Outlook - Rainy

Temp	Humidity	Windy	Hours Played
Hot	High	FALSE	25
Hot	High	TRUE	30
Mild	High	FALSE	35
Cool	Normal	FALSE	38
Mild	Normal	TRUE	48
			S = 7.78
			AVG = 35.2
			CV = 22%



		Hours Played (StDev)	Count
Temp	Cool	0	1
	Hot	2.5	2
	Mild	6.5	2

SDR = 7.78 - ((1/5)*0+(2/5)*2.5 + (2/5)*6.5) = 4.18

		Hours Played (StDev)	Count
Humidity	High	4.1	3
	Normal	5.0	2

SDR = 7.78 - ((3/5)*4.1 + (2/5)*5.0) = 3.32

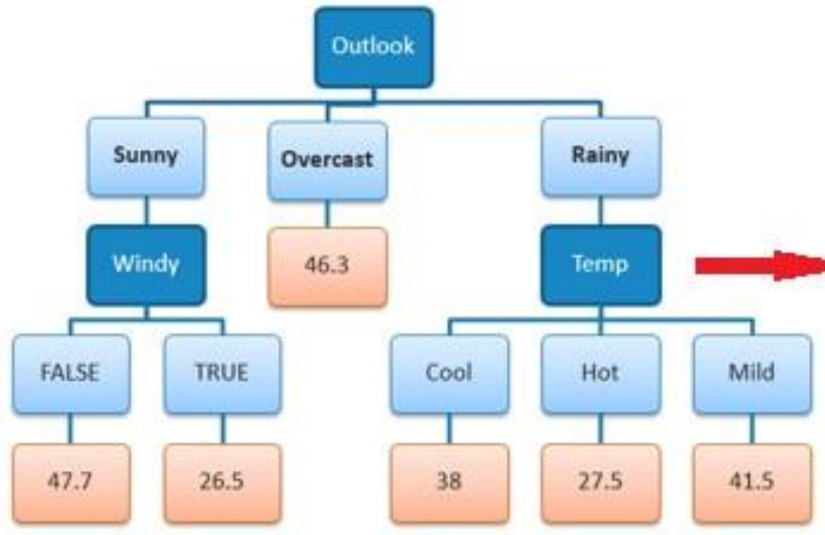
		Hours Played (StDev)	Count
Windy	False	5.6	3
	True	9.0	2

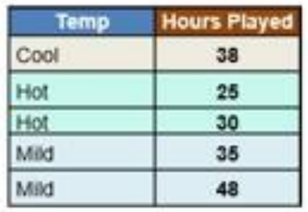
SDR = 7.78 - ((3/5)*5.6 + (2/5)*9.0) = 0.82

NeuroTech

Decision Tree - Regression

- Because the number of data points for all three branches (Cool, Hot and Mild) is equal or less than 3 we stop further branching and assign the average of each branch to the related leaf node.





Temp	Hours Played
Cool	38
Hot	25
Hot	30
Mild	35
Mild	48

- When the number of instances is more than one at a *leaf node* we calculate the *average* as the final value for the target.

Decision Tree - Regression

Advantages of Decision Trees:

1. **Interpretability:** Decision trees are easy to understand and interpret. The model's decision-making process can be visualized, which is particularly useful for non-technical stakeholders.
2. **No Assumptions About Data:** Decision trees do not make strong assumptions about the data distribution. They can work well with both linear and non-linear relationships in the data.
3. **Handling Non-Numeric Data:** Decision trees can handle both numeric and categorical data without the need for one-hot encoding. This simplifies data preprocessing.
4. **Feature Selection:** Decision trees can implicitly perform feature selection by giving more importance to the most informative features in the tree structure.
5. **Robust to Outliers:** Decision trees are not affected by outliers in the data, as they can isolate them in leaf nodes.
6. **Suitable for Small to Medium Datasets:** Decision trees can perform well on small to medium-sized datasets.

Decision Tree - Regression

Disadvantages of Decision Trees:

1. **Overfitting:** Decision trees are prone to overfitting, especially if they are deep and not pruned. They can capture noise in the data and perform poorly on new, unseen data.
2. **High Variance:** Decision trees tend to have high variance, which means they might not generalize well to unseen data. This issue can be mitigated by using ensemble methods like Random Forest or Gradient Boosting.
3. **Instability:** Small changes in the training data can lead to significantly different tree structures. This makes them less stable and robust.
4. **Bias Toward Features with Many Values:** Decision trees tend to favor features with more values, which can lead to biased splits.
5. **Limited Expressiveness:** Decision trees may not be suitable for modeling complex relationships in data compared to more advanced algorithms like neural networks or gradient boosting.
6. **Inefficient on Large Datasets:** Decision trees can be inefficient on large datasets with many features and examples because constructing the tree can be computationally expensive.

Go to Notebook

NeuroTech

Ensemble Models

Ensemble Models

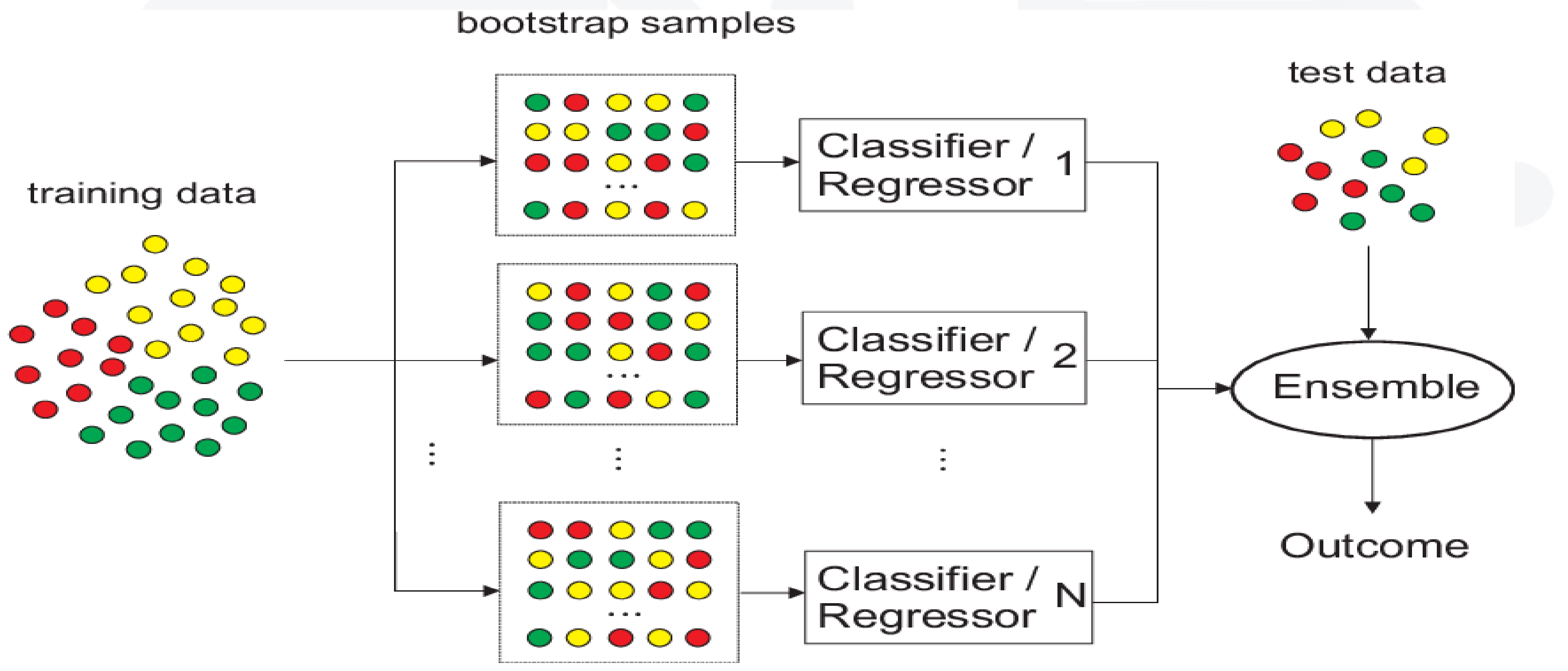
Ensemble models in machine learning are techniques that combine the predictions of multiple individual models to improve overall predictive performance. The idea behind ensemble modeling is that by aggregating the results of multiple models, you can often achieve better accuracy, generalization, and robustness compared to using a single model. Ensemble methods are widely used in various machine learning tasks, such as classification and regression.

Some popular ensemble methods include:

1. **Bagging (Bootstrap Aggregating):** Bagging involves creating multiple subsets of the training data through resampling (with replacement), training separate models on each subset, and then averaging their predictions. The most well-known algorithm in this category is Random Forest.

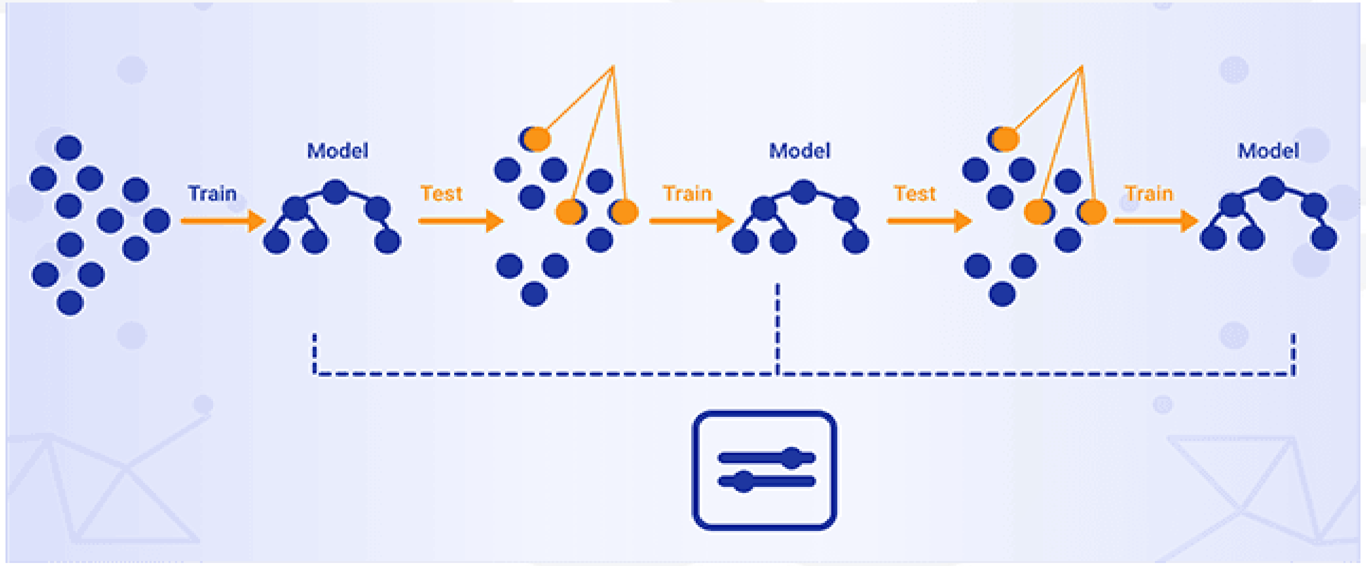
Ensemble Models

Bagging (Bootstrap Aggregating):



Ensemble Models

2. Boosting: Boosting is an ensemble technique that combines the predictions of multiple weak learners to create a strong learner. Weak learners are models that perform slightly better than random guessing. Popular boosting algorithms include AdaBoost, Gradient Boosting, and XGBoost.

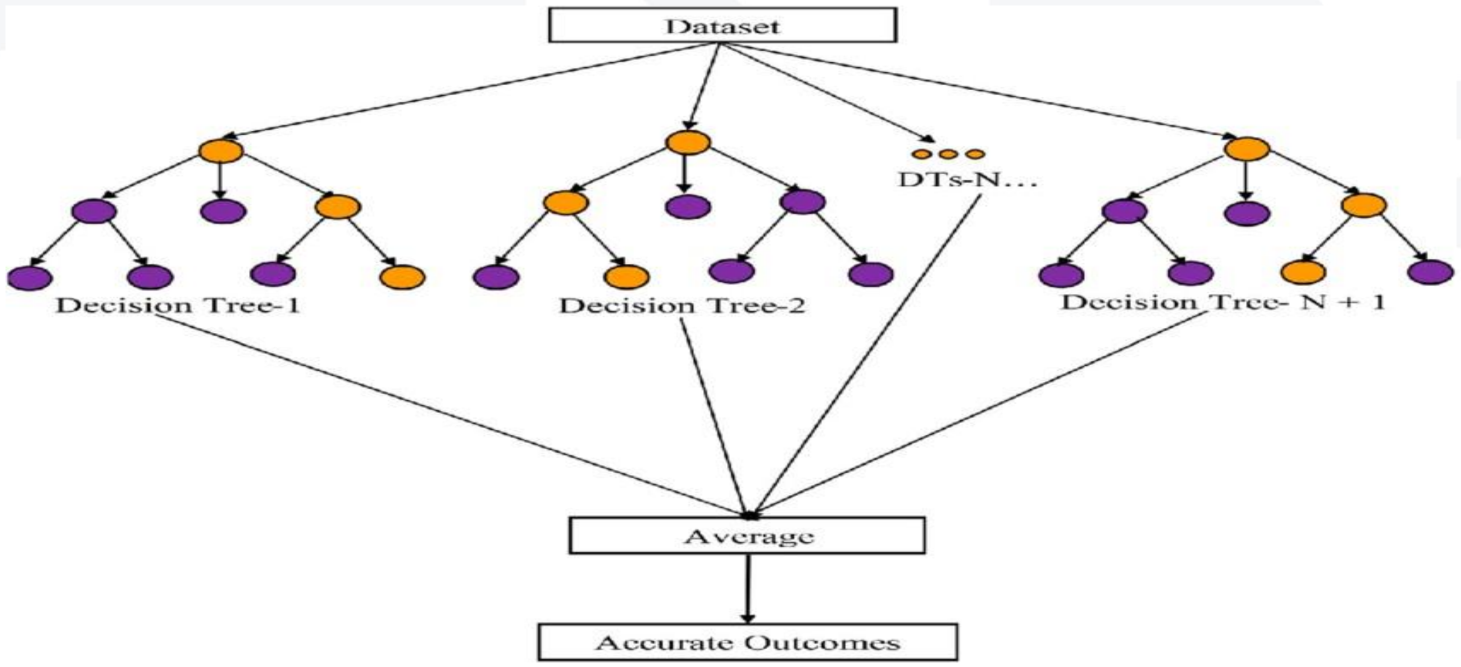


Bagging (Bootstrap Aggregation)

NeuroTech

Random Forest - Regression

- **Random Forest** is an ensemble learning method in machine learning used for both classification and regression tasks. It is a versatile and powerful algorithm that combines the concepts of bagging (Bootstrap Aggregating) and decision trees to create a robust and accurate predictive model.



NEUROTECH

Random Forest - Regression

How Random Forest Works?

1. **Bootstrapping (Data Sampling):**
    1. Random Forest starts by creating multiple bootstrap samples from the original training data. This means it randomly selects a subset of the data with replacement. As a result, each sample may contain duplicate instances and may not include some original data points.
2. **Random Feature Selection:**
    1. For each decision tree it creates, Random Forest randomly selects a subset of features to consider at each node in the tree. This feature selection adds an element of randomness to the modeling process and reduces the chance of overfitting.
3. **Multiple Decision Trees:**
    1. Random Forest creates a specified number of decision trees, typically in the range of hundreds to thousands. Each decision tree is constructed using the bootstrapped data and the randomly selected subset of features.
4. **Averaging :**
    1. For regression tasks, the predictions are averaged to get the final prediction.

Random Forest - Regression

Advantages of Random Forest:

1. **High Accuracy:** Random Forest generally produces highly accurate predictions due to the ensemble of many decision trees.
2. **Robustness:** Random Forest is robust against overfitting, thanks to the use of bootstrap samples and random feature selection.
3. **Implicit Feature Selection:** The feature selection process within each tree can serve as an implicit method for identifying important features in the data.
4. **Works for Classification and Regression:** Random Forest can be used for both classification and regression problems.
5. **Can Handle Large Datasets:** It can efficiently handle large datasets with many features and examples.
6. **Outlier Robustness:** Random Forest is less sensitive to outliers compared to some other algorithms.
7. **Interpretability:** While Random Forest models can be somewhat challenging to interpret, it's possible to examine feature importance scores to understand which features are more influential in the predictions.

Random Forest - Regression

Disadvantages of Random Forest:

1. **Complexity:** Random Forest models can be computationally intensive and may require more time for training and prediction compared to simpler models.
2. **Difficulty in Interpretation:** Despite providing feature importance scores, the individual decision trees within the ensemble can be hard to interpret. It's challenging to understand the logic behind the predictions.
3. **Overhead:** The use of multiple decision trees can lead to higher memory consumption, especially when you have a large number of trees.
4. **Hyperparameter Tuning:** Proper tuning of hyperparameters, such as the number of trees, the depth of trees, and feature selection, is necessary to get the best performance.

Go to Notebook

NeuroTech

Boosting - Regression

NeuroTech

Gradient Boosting

- **Gradient Boosting** is an ensemble machine learning technique that's widely used for both regression and classification tasks. It is particularly known for its high predictive accuracy and robustness. Gradient Boosting builds an ensemble of decision trees in a sequential and adaptive manner, where each tree corrects the errors made by the previous ones. One of the most popular implementations of Gradient Boosting is the Gradient Boosting Machine (GBM), which serves as the foundation for other variants like XGBoost and LightGBM.

Gradient Boosting

How Gradient Boosting Work?

- In the following example, Age is the Target variable whereas LikesExercising, GotoGym, DrivesCar are independent variables. As in this example, the target variable is continuous, GradientBoostingRegressor is used here.

LikesExercising	GotoGym	DrivesCar	Age
False	True	True	14
False	True	False	15
False	True	False	16
True	True	True	26
False	True	True	36
True	False	False	50
True	True	True	69
True	False	False	72
True	False	True	73

Gradient Boosting

1st-Estimator

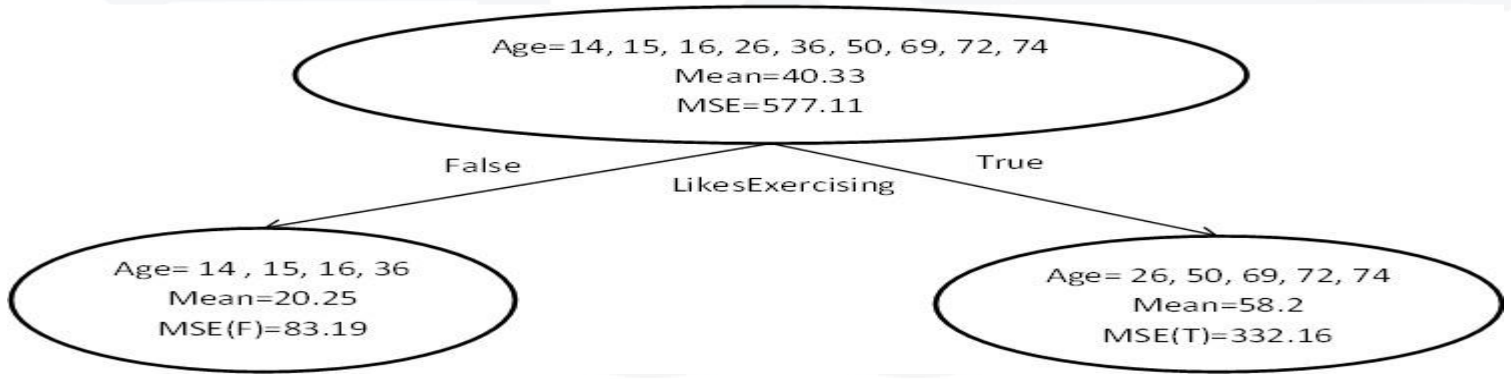
- For estimator-1, the root level (level 0) will consist of all the records. The predicted age at this level is equal to the mean of the entire Age column i.e. 41.33(addition of all values in Age column divided by a number of records i.e. 9). Let us find out what is the MSE for this level. MSE is calculated as the mean of the square of errors. Here error is equal to actual age-predicted age. The predicted age for the particular node is always equal to the mean of age records of that node. So, the MSE of the root node of the 1st estimator is calculated as given below.

- $$\mathbf{MSE=(\Sigma(Age_i - mu)^2\ )/9=577.11}$$

- The cost function hers is MSE and the objective of the algorithm here is to minimize the MSE.

Gradient Boosting

- Now, one of the independent variables will be used by the Gradient boosting to create the Decision Stump.
- Let us suppose that the Likes Exercising is used here for prediction. So, the records with False Likes Exercising will go in one child node, and records with True Likes Exercising will go in another child node as shown below.



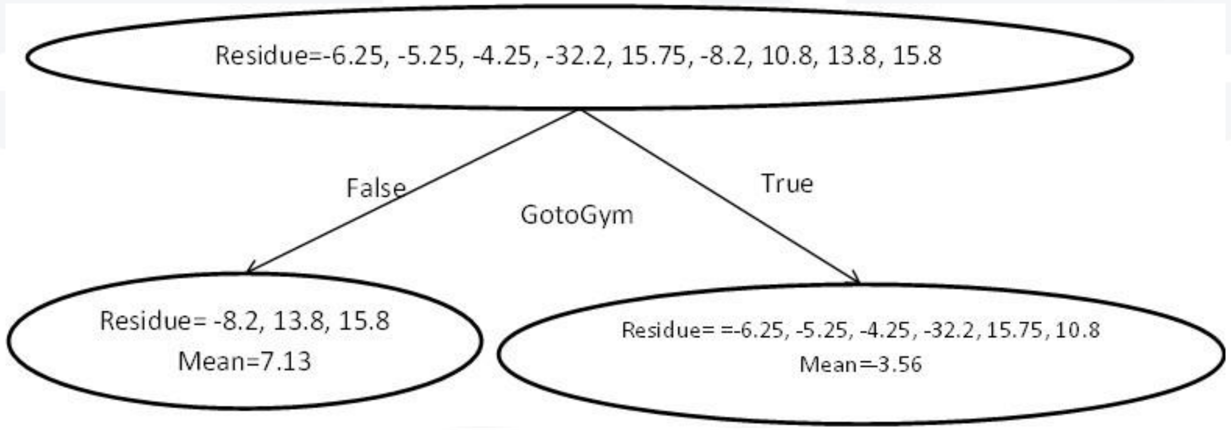
Gradient Boosting

- Let us find the means and MSEs of both the nodes of level 1. For the left node, the mean is equal to 20.25 and MSE is equal to 83.19. Whereas, for the right node, the mean is equal to 58.2 and MSE is equal to 332.16. Total MSE for level 1 will be equal to the addition of all nodes of level 1 i.e. 83.19+332.16=415.35. We can see here, the cost function i.e. MSE of level 1 is better than level 0.

Gradient Boosting

2nd-Estimator:

- Let us now find out the estimator-2. Unlike AdaBoost, in the Gradient boosting algorithm, residues (age_i – mu)of the first estimator are taken as root nodes as shown below. Let us suppose for this estimator another dependent variable is used for prediction. So, the records with False GotoGym will go in one child node, and records with True GotoGym will go in another child node as shown below.


Residue=-6.25, -5.25, -4.25, -32.2, 15.75, -8.2, 10.8, 13.8, 15.8
False
GotoGym
True
Residue= -8.2, 13.8, 15.8
Mean=7.13
Residue= =-6.25, -5.25, -4.25, -32.2, 15.75, 10.8
Mean=3.56

Gradient Boosting

- The prediction of age here is slightly tricky. First, the age will be predicted from estimator 1 as per the value of LikeExercising, and then the mean from the estimator is found out with the help of the value of GotoGym and then that means is added to age-predicted from the first estimator and that is the final prediction of Gradient boosting with two estimators.

- Let us consider if we want to predict the age for the following records:

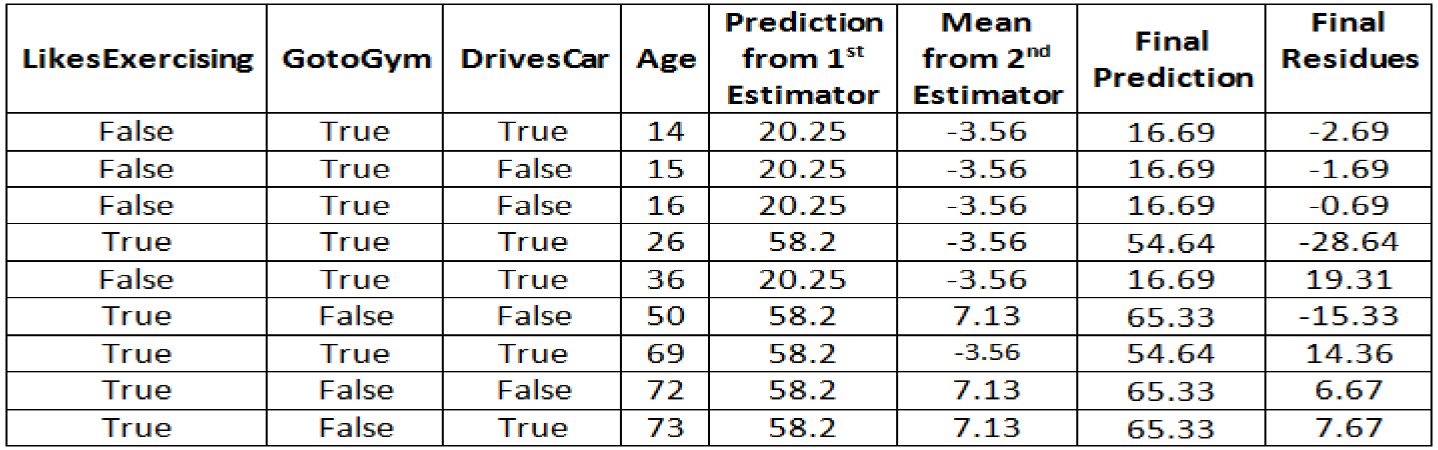
LikesExercising	GotoGym	DrivesCar
False	True	True

- Here, LikesExercising is equal to False. So, the predicted age from the first estimator will be 20.25 (i.e. mean of the left node of the first estimator). Now we need to check what is the value of GotoGym for the second predictor and its value is True. So, the mean of True GotoGym in the 2nd estimator is -3.56. This will be added to the prediction of the first estimator i.e. 20.25. So final prediction of this model will be 20.25+(-3.56) = 16.69.

NEUROTECH

Gradient Boosting

Let us predict of ages of all records we have in the example.



LikesExercising	GotoGym	DrivesCar	Age	Prediction from 1st Estimator	Mean from 2nd Estimator	Final Prediction	Final Residues
False	True	True	14	20.25	-3.56	16.69	-2.69
False	True	False	15	20.25	-3.56	16.69	-1.69
False	True	False	16	20.25	-3.56	16.69	-0.69
True	True	True	26	58.2	-3.56	54.64	-28.64
False	True	True	36	20.25	-3.56	16.69	19.31
True	False	False	50	58.2	7.13	65.33	-15.33
True	True	True	69	58.2	-3.56	54.64	14.36
True	False	False	72	58.2	7.13	65.33	6.67
True	False	True	73	58.2	7.13	65.33	7.67

- Let us now find out the Final MSE for above all 9 records.
- MSE= $((-2.69)^2 + (-1.69)^2 + (-0.69)^2 + (-28.64)^2 + (19.31)^2 + (-15.33)^2 + (14.36)^2 + (6.67)^2 + (7.67)^2)/9$
  $= 194.2478$
- So, we can see that the final MSE is much better than the MSE of the root node of the 1st Estimator. This is only for 2 estimators. There can be n number of estimators in gradient boosting algorithm.

NEUROTECH

Gradient Boosting

Advantages of Gradient Boosting:

1. **High Predictive Accuracy:** Gradient Boosting is known for its high predictive accuracy and often outperforms other machine learning algorithms.
2. **Robustness:** It is robust against overfitting due to its sequential nature and the fact that it corrects errors made by previous models.
3. **Flexibility:** Gradient Boosting can work with various loss functions and weak learners, making it versatile for different tasks.
4. **Feature Importance:** It can provide insights into feature importance, helping identify the most influential features in the model.
5. **Handle Missing Data:** Some implementations, like XGBoost, can handle missing data without the need for preprocessing.

Gradient Boosting

Disadvantages of Gradient Boosting:

1. **Computational Complexity:** Training a Gradient Boosting model can be computationally expensive, especially with a large number of weak learners.
2. **Hyperparameter Tuning:** Proper tuning of hyperparameters is required to get the best performance, which can be challenging and time-consuming.
3. **Black-Box Model:** As an ensemble of many models, Gradient Boosting can be complex and difficult to interpret.
4. **Sensitivity to Noisy Data:** It can be sensitive to noisy data and outliers, potentially leading to overfitting.

Go to Notebook

NeuroTech

Any Question?

NeuroTech

Thanks

NeuroTech