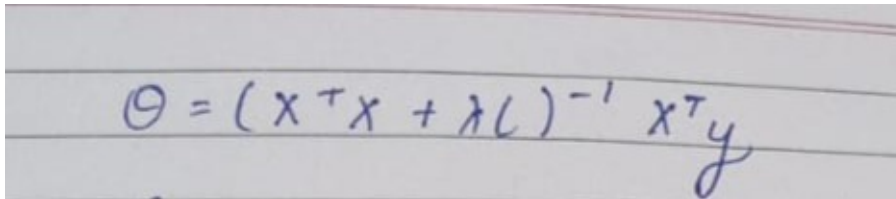


DSW_ASSIGNMENT-5

Untitled Attachment

1 ANS-The closed form solution for θ in Ridge Regression is given by:


$$\theta = (X^T X + \lambda I)^{-1} X^T y$$

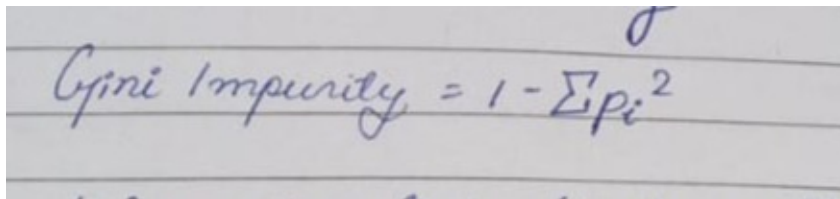
Here,

- \mathbf{X} is the matrix of input features,
- \mathbf{y} is the vector of target values,
- λ is the regularization parameter, and
- \mathbf{I} is the identity matrix.

This formula looks very similar to the normal equation of linear regression, but with a slight modification in the term $(X^T X + \lambda I)$. This term is always invertible, which solves one of the potential issues with the normal equation (where $X^T X$ may not be invertible). The regularization term λI shrinks the coefficients and helps to reduce the complexity of the model, thereby preventing overfitting. This is the essence of Ridge Regression.

2. Explain Gini impurity with an example.

Answer-Gini Impurity is a score that evaluates how accurate a split is among the classified groups. The Gini Impurity evaluates a score in the range between 0 and 1, where 0 is when all observations belong to one class, and 1 is a random distribution of the elements within classes. In this case, we want to have a Gini index score as low as possible. Gini Index is the evaluation metric we shall use to evaluate our Decision Tree Model.


$$\text{Gini Impurity} = 1 - \sum p_i^2$$

```
from collections import Counter
def gini_impurity(y):
    counts=Counter(y)
    impurity=1-sum([(count/len(y))**2 for count in counts.values()])
    return impurity
gini_impurity(['M', 'M', 'M', 'F', 'F', 'F'])

0.5
```

3. Explain Entropy of a dataset with an example.

Answer-

Entropy:

Entropy is the measure of the degree of randomness or uncertainty in the dataset.

```
import numpy as np
import pandas as pd
```

```

df=pd.read_csv("C:\\Users\\abham\\Downloads\\data.csv")
df

if 'Index' in df.columns:
df['obese'] = df['Index'].apply(lambda x: 1 if x > 3 else 0)
df.drop(columns=['Index'], inplace=True)
else:
print("Error: 'index' column not found in DataFrame.")

def compute_entropy(data,feature_col,split_value):
left_branch=data[data[feature_col]<=split_value]
right_branch=data[data[feature_col]>split_value]
total_samples=len(data)
def entropy(branch):
if len(branch)==0:
return 0
counts=Counter(branch['obese'])
impurity=-sum([(count/len(branch))*math.log2(count/len(branch)) for count
in counts.values()])
return impurity
gini_left=entropy(left_branch)
gini_right=entropy(right_branch)
print(gini_left,gini_right)
weight_left=len(left_branch)/total_samples
weight_right=len(right_branch)/total_samples
weight_avg_gini=(weight_left*gini_left)+(weight_right*gini_right)
return weight_avg_gini

feature_column='Gender'
split_value='Female'
entropy_impurity=compute_entropy(df,feature_column,split_value)

0.9400481779747517 0.9155506778147324

```

4. Is a Node's Entropy is generally higher than it's parent's entropy? Justify your answer

Answer-No, a node's entropy is not generally higher than its parent's entropy. In fact, it's usually the opposite in the context of decision trees.

Entropy is a measure of impurity, uncertainty, or disorder. In the context of decision trees in machine learning, entropy is used as a criterion that the model uses to split the data. The goal of the model is to reduce the entropy, i.e., reduce the uncertainty or disorder.

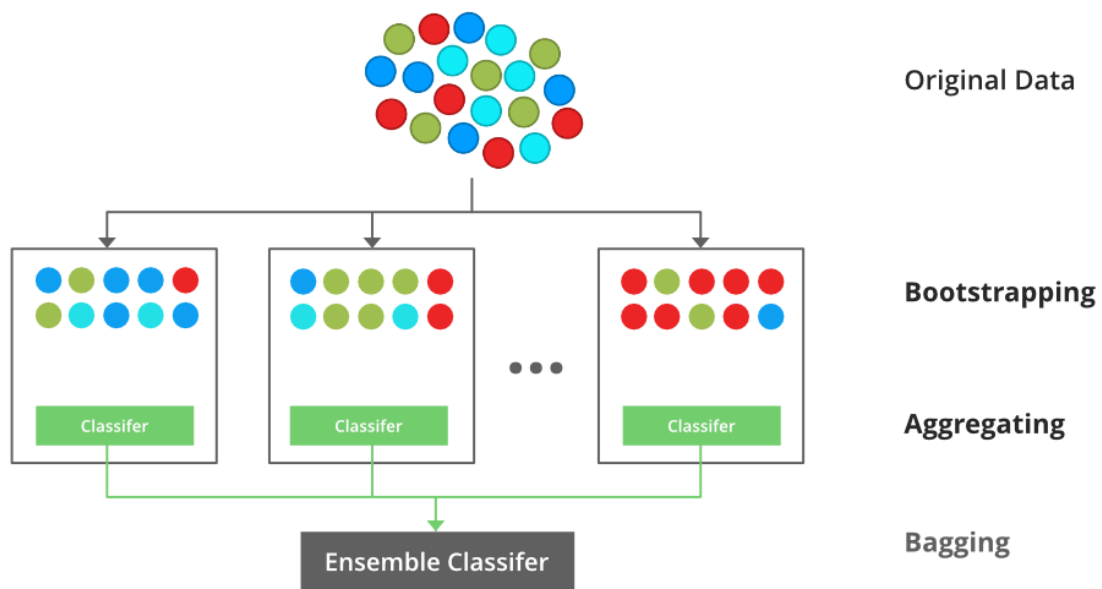
When a parent node is split into multiple child nodes, the aim is to organize the data in such a way that increases the homogeneity of the child nodes. This means that each child node should ideally have less entropy (less disorder) than the parent node.

Therefore, while there might be exceptions, in general, a node's entropy is not higher than its parent's entropy in the context of decision trees. This is a fundamental aspect of how decision trees work.

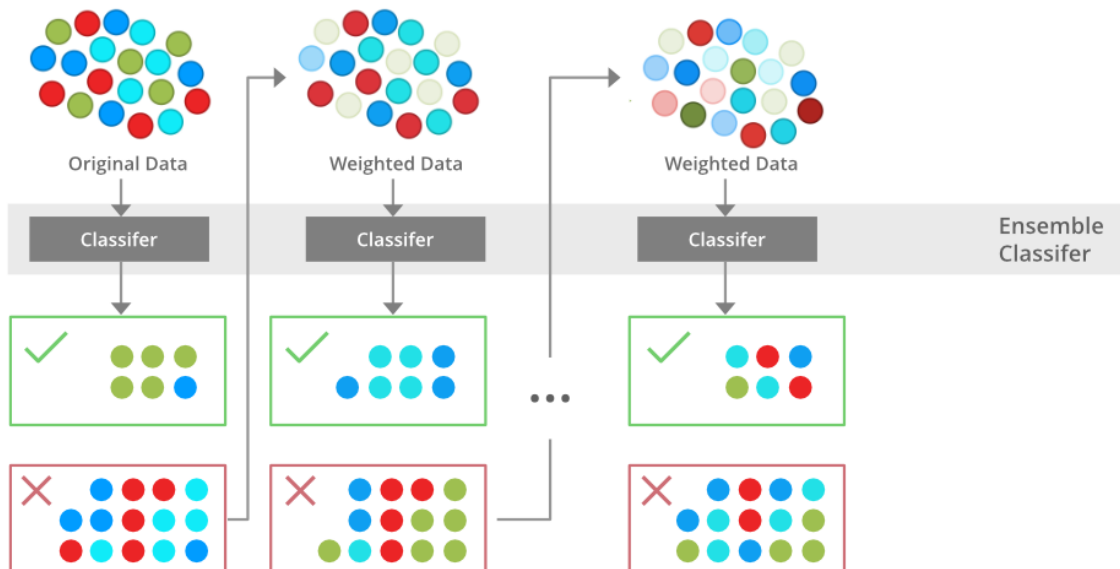
5. Difference between the followings

- Bagging and Boosting

Bagging	Boosting
<p>Bootstrap Aggregating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.</p>	<p>Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series.</p>
<p>It decreases the <u>variance</u> and helps to avoid <u>overfitting</u>. It is usually applied to <u>decision tree methods</u>. Bagging is a special case of the model averaging approach.</p>	<p>Aim to decrease bias, not variance. If the classifier is stable and simple (high bias) then apply boosting. Boosting tries to reduce bias.</p>



BAGGING



BOOSTING

– Random Forest and Bagged decision tree

Random Forest:	Bagged Decision Trees:
<ul style="list-style-type: none">• Random Forest is an extension of Bagged Decision Trees. But, in addition to constructing each tree from a different bootstrap sample of the data, Random Forest changes how the classification or regression trees are constructed by splitting nodes on a subset of the features that are selected at random.	<ul style="list-style-type: none">• Bagging (Bootstrap Aggregating) is a method that involves manipulating the training set by resampling. We create subsets of the original dataset (with replacement), train a model on each, and average the predictions.

<p>very best feature when splitting a node, it searches for the best feature among a random subset of features. This results in a greater tree diversity, which trades a higher bias for a lower variance, generally yielding an overall better model.</p>	<ul style="list-style-type: none"> • Bagging is typically used to reduce the variance of a decision tree. Each model is built independently which means bagging helps to decrease the model's variance, not its bias. Hence, it works well with high variance, low bias models (like decision trees).
<ul style="list-style-type: none"> • In Random Forest, each decision tree is trained with a random subset of features, not all features like Bagged Decision Trees. 	<ul style="list-style-type: none"> • In Bagged Decision Trees, every decision tree uses all features for splitting the nodes.

– Splitting and pruning

Splitting:	Pruning:
<ul style="list-style-type: none"> Splitting is the process of dividing a node into two or more sub-nodes to create more branches in a decision tree. When we split a node, we move from a more general hypothesis to a more specific one, which is why this process is also known as "specialization" 	<ul style="list-style-type: none"> Pruning is the process of removing the unnecessary structure (branches) from a decision tree. We do this in order to prevent overfitting, which happens when the model learns the training data too well and performs poorly on unseen data.
<ul style="list-style-type: none"> The decision of which attribute to choose as the splitting criterion is determined by a measure like Information Gain, Gain Ratio, or Gini 	

<p>Index. The goal is to choose the attribute that best separates the data into distinct classes.</p>	<ul style="list-style-type: none"> • There are two main types of pruning: pre-pruning (also known as early stopping) and post-pruning. Pre-pruning stops the growth of the decision tree at an earlier stage, while post-pruning allows the tree to grow fully first and then prunes it.
<ul style="list-style-type: none"> • Splitting helps to make the model more complex and fit the training data better, while pruning helps to simplify the model and make it more robust to unseen data. 	<ul style="list-style-type: none"> • Pruning works by merging the leaves of the tree, going from a more specific hypothesis to a more general one, which is why this process is also known as “generalization”.

– Hard Voting and soft voting

Hard Voting:	Soft Voting:
<ul style="list-style-type: none"> • Hard voting is the simplest way of combining the predictions from multiple machine learning algorithms. 	<ul style="list-style-type: none"> • Soft voting can only be done when all your classifiers can calculate probabilities for the outcomes. Soft voting predicts the class label based on the averaged probabilities.
<ul style="list-style-type: none"> • It involves summing up the predictions made by each classifier for a given input and then selecting the prediction that has the most votes (i.e., the mode). 	<ul style="list-style-type: none"> • This method can be used when the classifiers are able to estimate class probabilities, i.e., they have a <code>predict_proba()</code> method.

prediction made by the majority will therefore be the correct one.

- Instead of looking at the most common class in hard voting, we compute the average predicted class probabilities for each class and predict the class with the highest probability.

– K Means clustering and K nearest neighbors

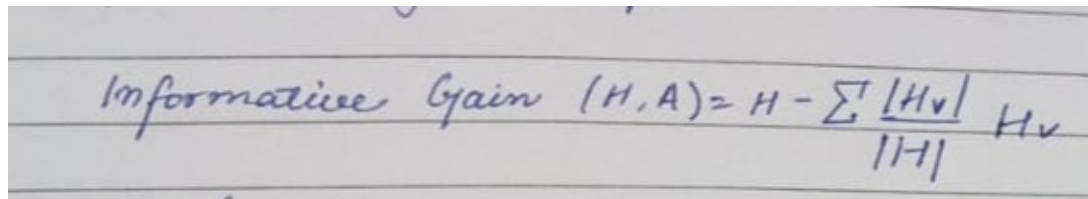
K-Means Clustering:	K-Nearest Neighbors (KNN):
<ul style="list-style-type: none"> • K-Means is an unsupervised learning algorithm used for clustering problem where we try to group similar instances together into clusters. 	<ul style="list-style-type: none"> • KNN is a supervised learning algorithm used for classification (and regression) problems. The output is a class membership.
<ul style="list-style-type: none"> • The algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible. 	<ul style="list-style-type: none"> • KNN makes predictions based on how similar training examples are to the example being predicted. It uses a distance metric such as Euclidean distance to identify the 'k' training examples that are closest to the new example.

<p>which are used as the beginning points for every cluster, and then performs iterative calculations to optimize the positions of the centroids.</p>	<ul style="list-style-type: none"> It's a lazy learning algorithm, meaning it does not use the training data points to do any generalization. In other words, there is no explicit training phase or it is zero: the data is only needed at the time of making real-time predictions.
<ul style="list-style-type: none"> The 'means' in the K-means refers to averaging of the data; that is, finding the centroid. 	<ul style="list-style-type: none"> An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

6. What is Information gain in Decision tree? How is it related to Entropy?

Information gain measures the reduction in entropy or variance that results from splitting a dataset based on a

specific property. It is used in decision tree algorithms to determine the usefulness of a feature by partitioning the dataset into more homogeneous subsets with respect to the class labels or target variable. The higher the information gain, the more valuable the feature is in predicting the target variable.



A photograph of a handwritten formula on lined paper. The text reads: 'Informative Gain (H, A) = H - \sum \frac{|H_v|}{|H|} H_v'. The formula is written in blue ink.

$$\text{Informative Gain } (H, A) = H - \sum \frac{|H_v|}{|H|} H_v$$

where

- A is the specific attribute or class label
- $|H|$ is the entropy of dataset sample S
- $|H_v|$ is the number of instances in the subset S that have the value v for attribute A

Information gain measures the reduction in entropy or variance achieved by partitioning the dataset on attribute A. The attribute that maximizes information gain is chosen as the splitting criterion for building the decision tree.

7. What are the disadvantages of K-means clustering over other unsupervised algorithms?

K-Means clustering has several disadvantages: it requires the number of clusters to be specified in advance, it's

sensitive to initial centroids and outliers, assumes clusters are spherical and equally sized, and can struggle with different densities and large datasets. It's also not suitable for binary data and doesn't guarantee a global optima. Other algorithms like DBSCAN or Hierarchical Clustering can sometimes overcome these limitations.

8. Write an python program to calculate the entropy of a dataset if the dataset, a particular feature in the dataset and a value in that feature is given to you.

```
import pandas as pd
import numpy as np

def calculate_entropy(dataset, feature, value):
    # Filter the dataset for rows where the feature equals the given value
    subset = dataset[dataset[feature] == value]

    # Calculate the proportions of each class in the subset
    class_counts = subset['obese'].value_counts().values
    class_proportions = class_counts / class_counts.sum()

    entropy = -np.sum(class_proportions * np.log2(class_proportions))

    return entropy

feature_column='Gender'
split_value='Female'
en=calculate_entropy(df, feature_column, split_value)
print(en)

0.9400481779747517
```

9. Do you think decision tree model can lead to over fitting? If yes, then how to handle over fitting? If no, why?

Yes, decision tree models can indeed lead to overfitting.

Overfitting occurs when the model is too complex and captures the noise along with the underlying pattern in data. In the case of decision trees, overfitting can happen when the tree is too deep, causing it to fit not only the actual data but also the noise.

Here are some strategies to handle overfitting in decision trees:

1. Pruning: Pruning is one of the techniques to handle overfitting. It involves removing the branches that make use of features having low importance. This can be done by setting a minimum number of samples required at a leaf node or setting the maximum depth of the tree.
2. Random Forests: Random Forests is another method where overfitting can be mitigated. It involves creating multiple decision trees on subsets of the data and averaging their predictions. This reduces the model's variance and improves generalization.

3. **Regularization:**Regularization methods like Lasso and Ridge can also be used to prevent overfitting in decision trees. These methods add a penalty term to the loss function, which forces the model to keep the weights as small as possible.
4. **Cross-Validation:**Cross-validation is a powerful preventative measure against overfitting. The idea is to split the training data into subsets and train the model on each subset. The average error rate across all subsets is used as the overall error rate of the model.

10. What do you mean by Ensemble learning? What are the benefits of ensemble learning? Discuss three other ensemble learning techniques other than Random forest. Ensemble Learning is a machine learning paradigm where multiple models (often called "weak learners") are trained to solve the same problem and combined to get better results. The main hypothesis behind ensemble methods is that when weak models are correctly combined, we can obtain more accurate and/or robust models.

Benefits of Ensemble Learning:

1. **Improved Accuracy:**Combining predictions from multiple models can often lead to better accuracy than could be obtained from any of the individual models.

2. **Robustness:** Even if some of the models make errors or fail, the other models can compensate, leading to more robust overall predictions.
3. **Reduced Overfitting:** By averaging multiple models, ensemble methods can also help to reduce overfitting.

Three other ensemble learning techniques other than Random Forest are:

1. **Bagging (Bootstrap Aggregating):** Bagging is a method that involves manipulating the training set by resampling. We create subsets of the original dataset (with replacement), train a model on each, and average the predictions. Bagging is typically used to reduce the variance of a decision tree. Each model is built independently which means bagging helps to decrease the model's variance, not its bias. Hence, it works well with high variance, low bias models (like decision trees).

2. **Boosting:** Boosting is an iterative technique that adjusts the weight of an observation based on the last classification. If an observation was classified incorrectly, it tries to increase the weight of this observation and vice versa. Boosting in general decreases the bias error and builds strong predictive models. Unlike bagging, in boosting, the trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the weights of the data points.
3. **Stacking (Stacked Generalization):** Stacking involves training a learning algorithm to combine the predictions of several other learning algorithms. First, all sub-models are trained based on the complete training set, then the meta-model is fitted based on the outputs, or the predictions, of those sub-models. The sub-models are trained based on the complete training set, then the meta-model is fitted based on the outputs — predictions — of the sub-models. The meta-model can either be trained on the predicted class labels or probabilities from the sub-models.

11. Using the following dataset, find the final Cluster with initial centroid [1, 1] and [3, 3]

x1	x2
1	1
1	2
2	2
2	3
3	4
4	4
5	1
5	2
5	3

```
import numpy as np

def kmeans(X, k, initial_centroids, max_iters):
    centroids = initial_centroids
    for _ in range(max_iters):
        labels = np.argmin(np.sqrt(((X - centroids[:,
            np.newaxis])**2).sum(axis=2)), axis=0)
        new_centroids = np.array([X[labels==i].mean(axis=0) for i in range(k)])
        if np.all(centroids == new_centroids):
            break
        centroids = new_centroids
    return centroids, labels
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3], [3, 4], [4, 4], [5, 1], [5,
2], [5, 3]])
initial_centroids = np.array([[1, 1], [3, 3]])
```

```
centroids, labels = kmeans(X, 2, initial_centroids, 100)
print("Final centroids:", centroids)
print("Cluster assignments:", labels)
```

```
Final centroids: [[1.5 2. ]
 [4.4 2.8]]
Cluster assignments: [0 0 0 0 1 1 1 1 1]
```

12. Explain the following clustering algorithms.

– DB Scan

1. Find all the neighbor points within ϵ and identify the core points or visited with more than MinPts neighbors.
2. For each core point if it is not already assigned to a cluster, create a new cluster.
3. Find recursively all its density-connected points and assign them to the same cluster as the core point.
A point a and b are said to be density connected if there exists a point c which has a sufficient number of points in its neighbors and both points a and b are within the ϵ distance. This is a chaining process. So, if b is a neighbor of c , c is a neighbor of d , and d is a neighbor of e , which in turn is neighbor of a implying that b is a neighbor of a .
4. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

– K-modes clustering

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

13. Consider the following dataset.

Using the given dataset predict whether a B. Tech CSE student will get the job or not using Naive Bays algorithm, If the evaluation parameters are as follows: CGPA= High; Communication= Bad; Aptitude= High; Programming skills= Bad

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
```



```

data = {
'CGPA': ['High', 'Medium', 'Low', 'Low', 'High', 'High', 'Medium',
'Medium', 'High', 'High', 'High'],
'Communication': ['Good', 'Good', 'Bad', 'Good', 'Good', 'Good', 'Bad',
'Bad', 'Good', 'Good', 'Good'],
'Aptitude': ['High', 'High', 'Low', 'Low', 'High', 'High', 'Low', 'Low',
'High', 'High', 'Low'],
'Programming_skill': ['Good', 'Good', 'Good', 'Bad', 'Bad', 'Good', 'Bad',
'Good', 'Good', 'Good', 'Good'],
'Job_offered': ['Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'No',
'Yes', 'No', 'Yes']
}

df = pd.DataFrame(data)

le = LabelEncoder()
df = df.apply(le.fit_transform)

model = GaussianNB()

features = df.drop('Job_offered', axis=1)
target = df['Job_offered']
model.fit(features, target)

student = pd.DataFrame([{'CGPA': 'High', 'Communication': 'Bad',
'Aptitude': 'High', 'Programming_skill': 'Bad'}])
student = student.apply(le.fit_transform)
prediction = model.predict(student)

print('Job offered:', 'Yes' if prediction[0] == 1 else 'No')

Job offered: No

```

14. Which attribute will be selected as a root node, while constructing a Decision Tree. If attribute selection measure

(ASM) is Entropy.

In the construction of a decision tree, the attribute that is selected as the root node is the one that provides the most information gain, if the attribute selection measure (ASM) is entropy.

Information Gain is a metric that measures the reduction in entropy achieved because of the split on an attribute. The attribute providing the highest information gain is chosen for the split.

Here's how it works:

1. Calculate the entropy of the target variable before the split (total entropy).
2. For each attribute, calculate the entropy of the target variable after the split (remaining entropy).
3. Subtract the remaining entropy from the total entropy to get the information gain.
4. The attribute with the highest information gain is selected as the root node.

The formula for information gain (IG) is:

where

- A is the specific attribute or class label
- $|H|$ is the entropy of dataset sample S

- $|H_v|$ is the number of instances in the subset S that have the value v for attribute A

This process is repeated recursively for each child node, with the attribute providing the highest information gain at each step chosen for the split. This continues until the tree is fully grown, and then pruning strategies can be applied to avoid overfitting. The entropy-based information gain approach is commonly used in the ID3, C4.5 and CART algorithms for constructing decision trees.

15. Which attribute will be selected as a root node, while constructing a Decision Tree. If attribute selection measure (ASM) is Gini Index

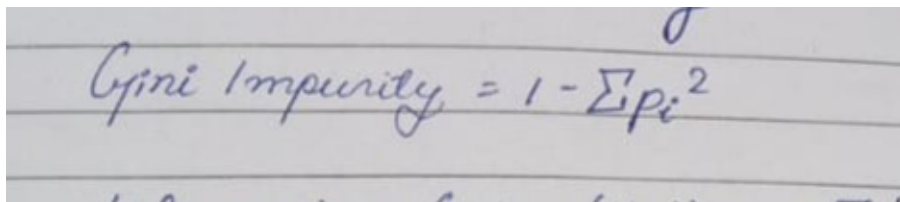
In the construction of a decision tree, if the attribute selection measure (ASM) is the Gini Index, the attribute that is selected as the root node is the one that provides the most reduction in the Gini Index, or in other words, the one that provides the maximum Gini Gain.

The **Gini Index** is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Trees) algorithm. It is a metric to measure how often a randomly chosen element would be incorrectly identified.

Here's how it works:

1. Calculate the Gini Index for the dataset before the split (total Gini Index).
2. For each attribute, calculate the Gini Index for the dataset after the split (remaining Gini Index).
3. Subtract the remaining Gini Index from the total Gini Index to get the Gini Gain.
4. The attribute with the highest Gini Gain is selected as the root node.

The formula for Gini Index (G) is:

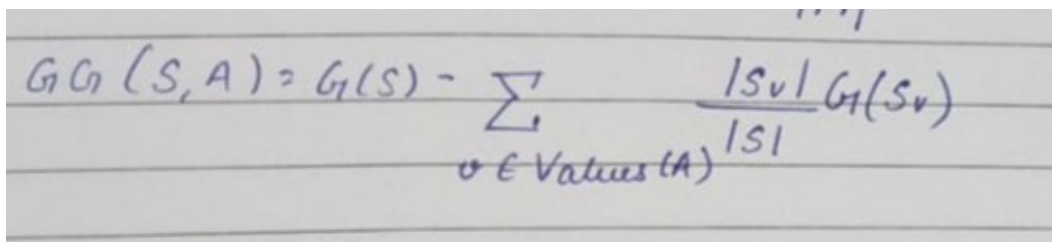


$$\text{Gini Impurity} = 1 - \sum p_i^2$$

where:

- S is the total sample space
- p_i is the proportion of the instances that belong to class i for a particular node.

The Gini Gain is calculated as:



$$G_G(S, A) = G(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} G(S_v)$$

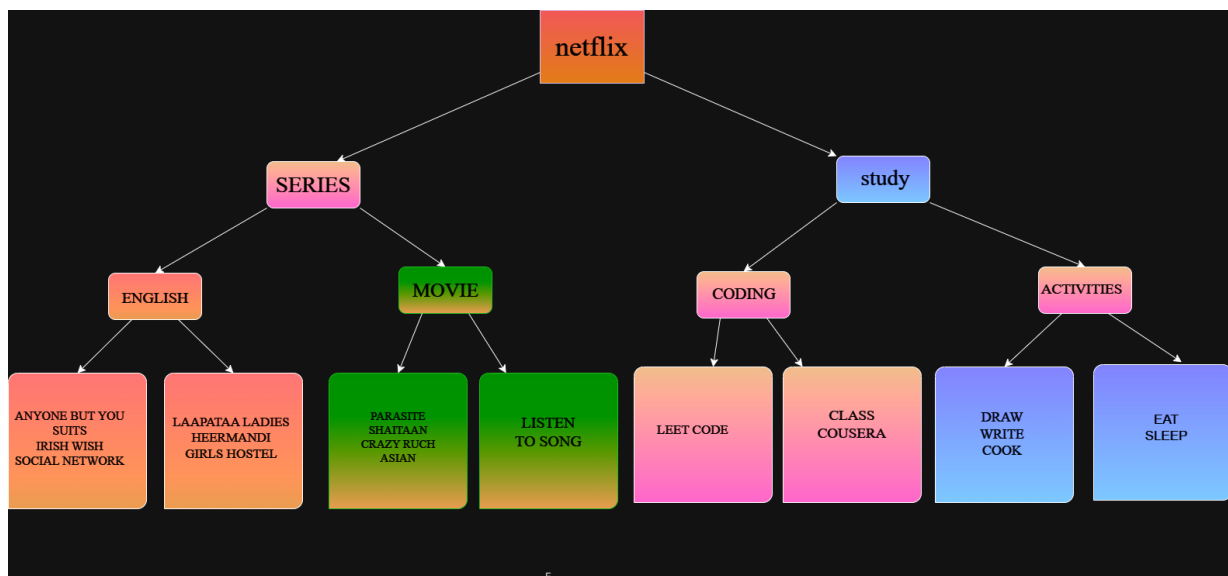
where:

- S is the total sample space
- A is the attribute we split the data on

- $\text{Values}(A)$ are the different values of attribute A
- S_v is the subset of S for which attribute A has value v

This process is repeated recursively for each child node, with the attribute providing the highest Gini Gain at each step chosen for the split. This continues until the tree is fully grown, and then pruning strategies can be applied to avoid overfitting. The Gini Index-based approach is commonly used in the CART algorithm for constructing decision trees.

16. Draw the decision tree by considering max depth = 3.



16. If the data points are: 5, 10, 15, 20, 25, 30, 35. Assume $K = 2$ and initial centroid as 15, 32. Create two clusters with

the given set of centroids and calculate SSE

16. $C_1 = 15, C_2 = 32$
 $D = [5, 10, 15, 20, 25, 30, 35]$

<u>Group 1</u>	<u>Group 2</u>
5 5, 10 15, 20, 25	30, 35

Recalculate centroid

$$\text{New } C_1 = \frac{5 + 10 + 15 + 20 + 25}{5} = 15$$
$$\text{New } C_2 = \frac{30 + 35}{2} = 32.5$$
$$SSE = \sum (observation - mean)^2$$
$$SSE_1$$
$$\text{Group 1} = (5-15)^2 + (10-15)^2 + (15-15)^2 + (20-15)^2 + (25-15)^2$$
$$= 100 + 25 + 0 + 25 + 100 = 250$$
$$SSE_2$$
$$\text{Group 2} = (30-32.5)^2 + (35-32.5)^2 = 6.25 + 6.25 = 12.5$$
$$\text{Total SSE} = 250 + 12.5 = 262.5$$

17. Write the python code for finding closest centroid to a sample in k-means clustering.

```
import numpy as np
```

```
def find_closest_centroid(sample, centroids):  
    distances = np.sqrt(((sample - centroids)**2).sum(axis=1))  
    closest_centroid_index = np.argmin(distances)
```

```
return centroids[closest_centroid_index]
centroids = np.array([[1, 1], [2, 2], [3, 3]])
sample = np.array([2.5, 2.5])
print(find_closest_centroid(sample, centroids))
```

```
[2 2]
```

18. What are the basic assumptions in the case of the Naive Bayes classifier?

The Naive Bayes classifier is based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Here are the basic assumptions:

1. **Feature Independence:** Each feature makes an independent and equal contribution to the outcome.
2. **Equal Importance of Features:** All features are considered equally important.
3. **Prior Probability:** The value of a particular feature is independent of the value of any other feature, given the class variable.
4. **Sufficiency:** Each observation is assumed to be sufficient to classify a new instance correctly.

5. **Zero Conditional Probability Problem:** If a category in the test data set was not observed in the training data set, the model will assign a zero probability to this category and will be unable to make a prediction. This is often known as "Zero Conditional Probability Problem."



Untitled Attachment



Untitled Attachment