------------------------------------Decision Tree------------------------------------

1. **Random_State :** This is to check and validate the data when running the code multiple times. Setting **random_state** a fixed value will guarantee that same sequence of random numbers are generated each time you run the code. And unless there is some other randomness present in the process, the results produced will be same as always. This helps in verifying the output.
   If you don't mention the random_state in the code, then whenever you execute your code a new random value is generated and the train and test datasets would have different values each time.

   Note : There are couple of algorithms there to build a decision tree , we only talk about a few which are

   CART (Classification and Regression Trees) → uses Gini Index(Classification) as metric.
   ID3 (Iterative Dichotomiser 3) → uses Entropy function and Information gain as metrics.
   And In DT we are performing top-down, greedy search through the space of possible decision trees.

   okay so how do we choose the best attribute?
   Answer: use the attribute with the highest ***information gain*** in ***ID3*** for that we needed "entropy" measure which is a measure of randomness in the data.
   For every feature we need to calculate the IG.
   (https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1)

2. **Gini_Index (DT) :**

$$= 1 - \sum_{t=0}^{t=1} P_t^2$$

Out of 14 instances ,
yes=9,no=5
1 - (9/14)^2- (5/14)^2

1-0.413-0.127= 0.46
Gini = 0.46

Steps:

1.compute the gini index for data-set
2.for every attribute/feature:
  1.calculate gini index for all categorical values
  2.take average information entropy for the current attribute
  3.calculate the gini gain
3. pick the best gini gain attribute.
4. Repeat until we get the tree we desired.

3. Entropy (DT) :

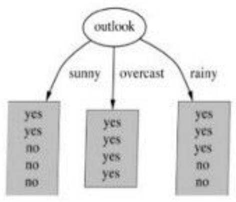
$$H(S) = \sum_{c \in C} -p(c) \log_2 p(c)$$

C ={yes,no}

Out of 14 instances, 9 are classified as yes,
and 5 as no
pyes = −(9/14)*log2(9/14) = 0.41
pno = −(5/14)*log2(5/14) = 0.53

H (S) = pyes + pno = 0.94

For every feature calculate the entropy and information gain :

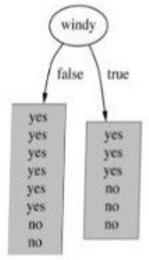


$$\left.\begin{array}{l}\text{E (Outlook=sunny)} = -\frac{2}{5}\log\left(\frac{2}{5}\right)-\frac{3}{5}\log\left(\frac{3}{5}\right)=0.971\\ \text{E (Outlook=overcast)} = -1\log(1)-0\log(0)=0\\ \text{E (Outlook=rainy)} = -\frac{3}{5}\log\left(\frac{3}{5}\right)-\frac{2}{5}\log\left(\frac{2}{5}\right)=0.971\end{array}\right\}\ \text{H(S,Outlook)}$$

Average Entropy information for Outlook

$$\left. I\text{ (Outlook)} = \frac{5}{14}*0.971+\frac{4}{14}*0+\frac{5}{14}*0.971=0.693 \right\}\ \sum_{t\in T} p(t)H(t)$$

$$\text{Gain (Outlook)} = \text{E(S)} - \text{I (outlook)} = 0.94\text{-}.693 = 0.247 \Rightarrow IG(A,S)=H(S)-\sum_{t\in T}p(t)H(t)$$



$$\text{E (Windy=false)} = -\frac{6}{8}\log\left(\frac{6}{8}\right)-\frac{2}{8}\log\left(\frac{2}{8}\right)=0.811$$

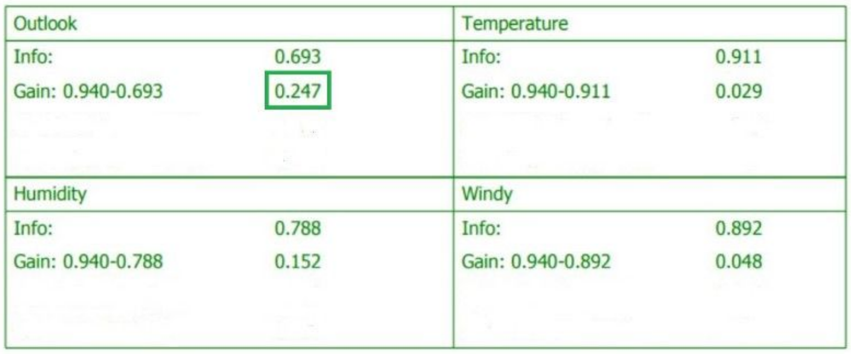
$$\text{E (Windy=true)} = -\frac{3}{6}\log\left(\frac{3}{6}\right)-\frac{3}{6}\log\left(\frac{3}{6}\right)=1$$

Average entropy information for Windy

$$I\text{ (Windy)} = \frac{8}{14}*0.811+\frac{6}{14}*1=0.892$$

$$\text{Gain (Windy)} = \text{E(S)} - I\text{ (Windy)} = 0.94\text{-}0.892\text{=}0.048$$

Pick the highest gain attribute for first split :



Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	$\boxed{0.247}$	Gain: 0.940-0.911	0.029
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048

Repeat the same thing for sub-trees till we get the tree.

4. Overfitting and Underfitting in Decision tree :

Underfitting:

A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data. *(It's just like trying to fit undersized pants!)*Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough. It usually happens when we have less data to build an accurate model and also when we try to build a linear model with a non-linear data. In such cases the rules of the machine learning model are too easy and flexible to be applied on such a minimal data and therefore the model will probably make a lot of wrong predictions. Underfitting can be avoided by using more data and also reducing the features by feature selection.

Overfitting:

A statistical model is said to be overfitted, when we train it with a lot of data *(just like fitting ourselves in an oversized pants!)*. When a model gets trained with so much of data, it starts learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too much of details and noise. The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models. A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.

High Bias : underfit
High Variance : Overfit

How to avoid Overfitting/Underfitting:

1) Can use Random Forest(Overfitting) :

There are two important techniques that you can use when evaluating machine learning algorithms to limit overfitting:

1. Use a resampling technique to estimate model accuracy.
2. Hold back a validation dataset.

The most popular resampling technique is k-fold cross validation. It allows you to train and test your model k-times on different subsets of training data and build up an estimate of the performance of a machine learning model on unseen data.

A validation dataset is simply a subset of your training data that you hold back from your machine learning algorithms until the very end of your project. After you have selected and tuned your machine learning algorithms on your training dataset you can evaluate the learned models on the validation dataset to get a final objective idea of how the models might perform on unseen data.

Using cross validation is a gold standard in applied machine learning for estimating model accuracy on unseen data. If you have the data, using a validation dataset is also an excellent practice.

------------------------------Linear Regression------------------------------

1. The objective of a linear regression model is to find a relationship between one or more features(independent variables) and a continuous target variable(dependent variable)

$$Y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$$

- Y is the predicted value
- θ_0 is the bias term

How do we determine the best fit line?
The line for which the the *error* between the predicted values and the observed values is minimum is called the best fit line or the regression line. These errors are also called as ***residuals.***

Line of Best Fit (Least Square Method)

A line of best fit is a straight line that is the best approximation of the given set of data.

It is used to study the nature of the relation between two variables. (We're only considering the two-dimensional case, here.)

A line of best fit can be roughly determined using an eyeball method by drawing a straight line on a scatter plot so that the number of points above the line and below the line is about equal (and the line passes through as many points as possible).

A more accurate way of finding the line of best fit is the least square method .

Use the following steps to find the equation of line of best fit for a set of ordered pairs (x1,y1),(x2,y2),...(xn,yn)(x1,y1),(x2,y2),...(xn,yn) .

Step 1: Calculate the mean of the xx -values and the mean of the yy -values.

X‾‾‾ =∑i=1 to n (xi / n)

Y‾‾‾ =∑i=1 to n (yi / n)

Step 2: The following formula gives the slope of the line of best fit:

m=∑i=1 to n ((xi−X‾‾‾)(yi−Y‾‾‾)) / (∑i=1 to n (xi−X‾)^2)

Step 3: Compute the yy -intercept of the line by using the formula:

b=Y‾‾‾−mX‾‾‾ b=Y‾−mX‾

Step 4: Use the slope mm and the yy -intercept bb to form the equation of the line (for best fit).

-----------------------------Logistic Regression--------------------

- Uses sigmoid function whose value will be between 0 and 1 (gives the probability)
- Used for Discrete/Categorical target values (Supervised->Classification Problems)
- Sigmoid function : p(x) = 1 / (1 + e^ - (b0+bx))

TP, TN, FP, FN

------------------------------SVM--------------------------

- print hyperplane in SVM:

```
w = clf.coef_[0]
xx = np.linspace(0,20)
a = -w[0] / w[1]
yy = a * xx - clf.intercept_[0] / w[1]
ho = plt.plot(xx, yy, 'k-', label="non weighted div")
plt.scatter(X[:,0], X[:,1], c=y)
plt.legend()
plt.show()
```