XGBoost

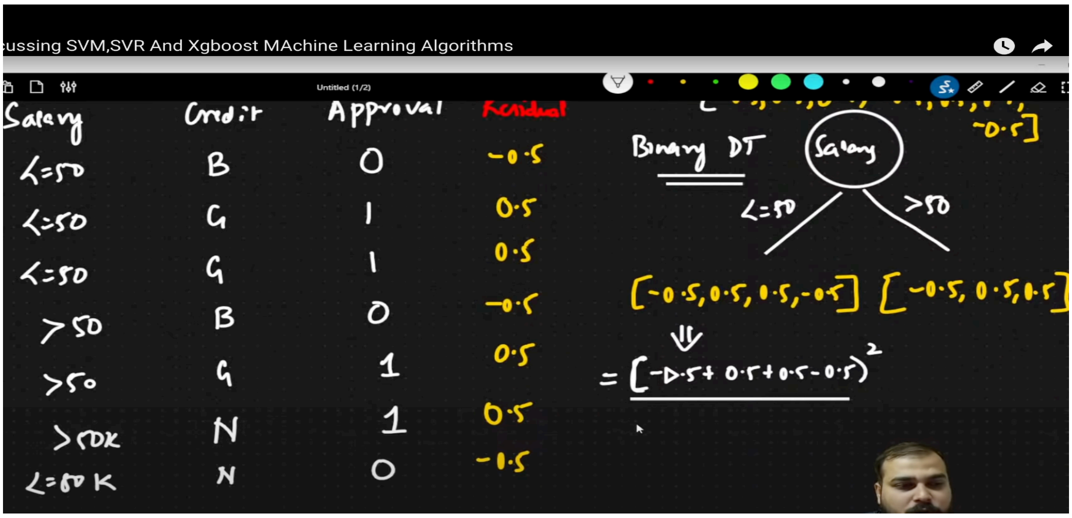
- As an example, we have a bank loan dataset in which fields are **salary, credit, approval.** Approval is our target variable in which **0** means **rejected** and **1** means **Approved.**
- First of all, we will create a base model Which is weak learner, and it will give an output of probability of 0.5 in the case of classification.
- After then, we wil create one more field which is **Residual,** what we do that we will **subtract** the **approval** value from the **output of probability**.
- After that There are some steps we will perform.

Steps:

1. First we create a binary tree using the features.
2. We calculate similarity weight

$$= \frac{\Sigma(Residual)^2}{\Sigma(pr(1-pr)) + \lambda}$$

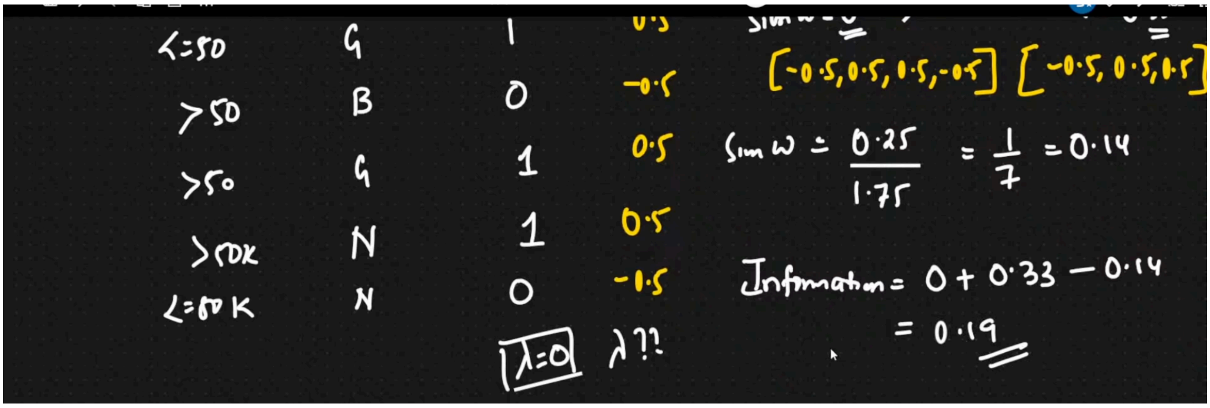
3. Calculate Information Gain



First step

In the **second step** , we will calculate the similarity weight, here pr means Probability which we give 0.5 as default, it might be different, then λ is the **Hyper parameter.**

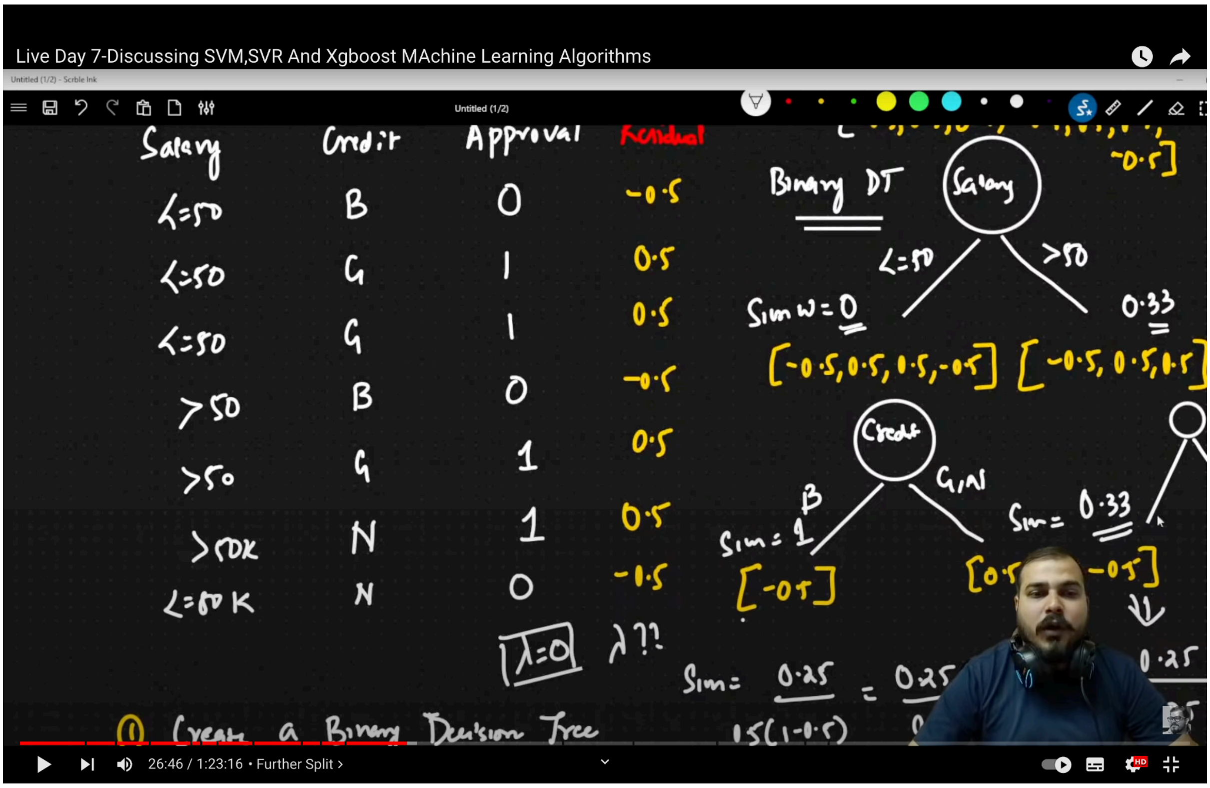
We get the similarity weight for both leaf after then we get the similarity weight for node(**Salary**).
In the third step, we will calculate information gain.



Calculating information gain

We use this formula which is **Information gain = similarity weight of left leaf + similarity weight of right leaf - similarity of Node leaf**

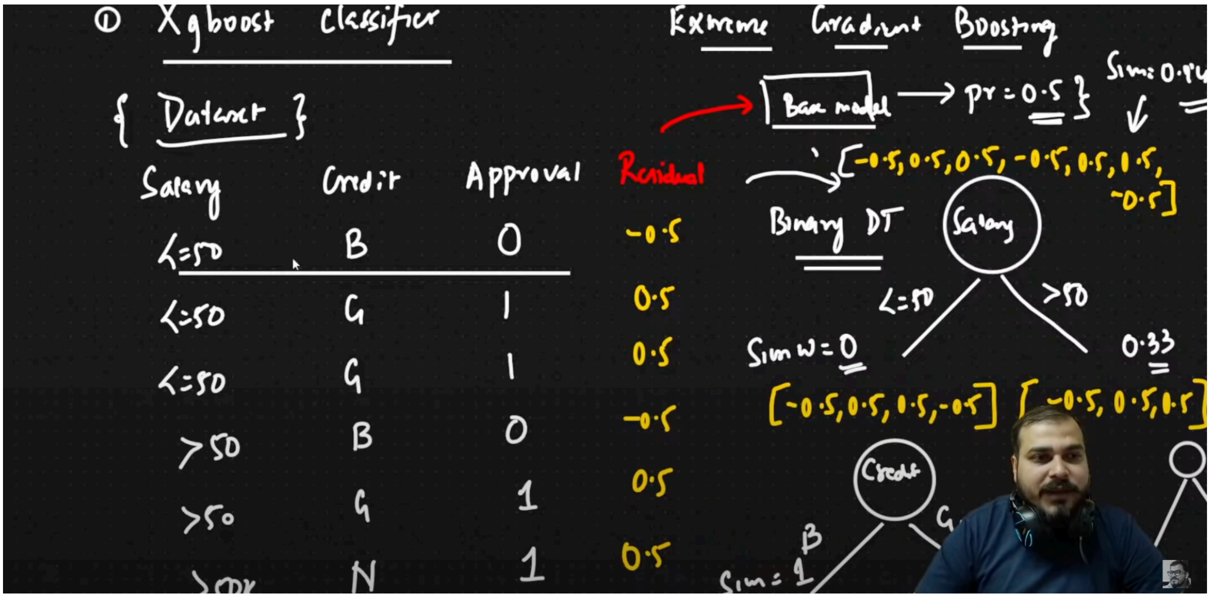
Now, our decision tree will split from salary and then we add another feature which is credit. We will do binary split, but we have three class in credit **Bed, Good, Normal. So,** what do we do now?
We will take bed credit to left side and good, normal to right side. Then we split ahead. As shown in below image.



Now, let's see how **inferencing** will happen,
We have base model and as per it we have default probability, now we take first record into base model an get the real probability using **log** formula which is,

$$\log\left(\frac{p}{1-p}\right)$$

We apply this only the case of base model and if we give default values we got log(1) which is 0.

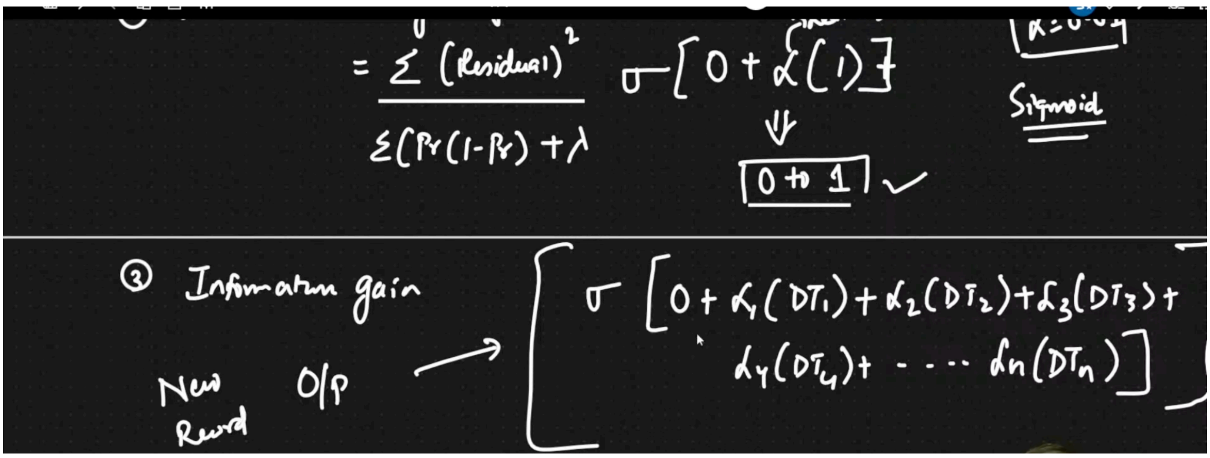


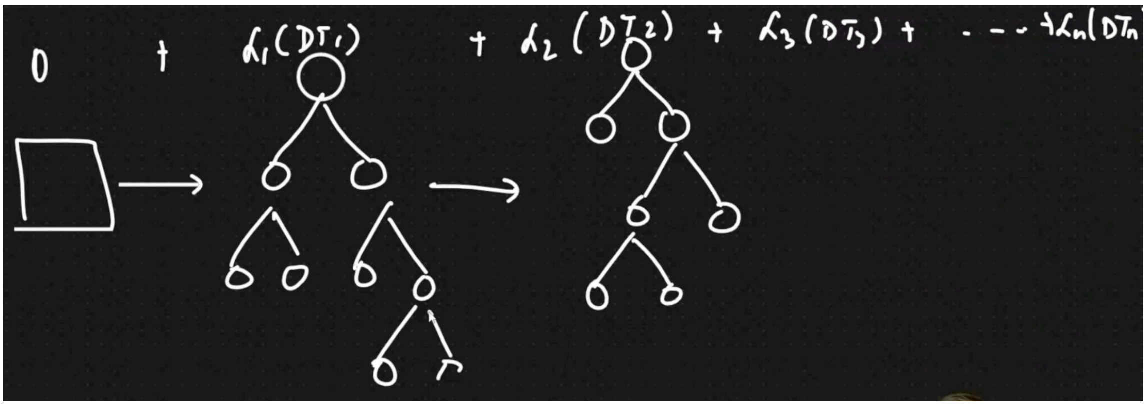
Now that value getting add in binary decision tree

After then, we add real probability and learning rate by multiply with similarity weight and apply sigmoid function, and this will give us a value between 0 and 1 when we use learning rate too small.

$$\sigma(0 \ + \ \alpha(1))$$

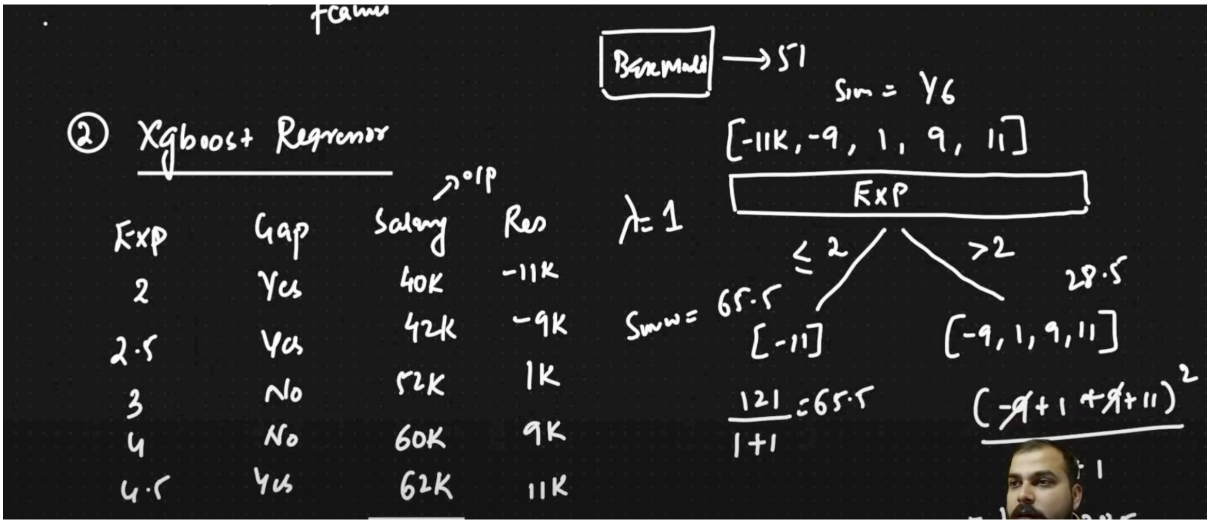
If we check first raw in above image we got real probability from there and then go depth in binary tree for 'bed' credit we got simialrtiy weight **1.**





Above image shows , how we get our output by adding all decision tree parallel.**That’s how we will find inference for new records.** This way **XGBoost** work. It is quite hard to calculate all this things manually on paper,so thats why **XGBoost** is **BlackBox Model**

XGBoost Regresser



Here, as per the above image salary is our output, So as usual we will make base model and in regresser, we will get the average of output field, after then we will get residuals by subtracting salary to average of salary.

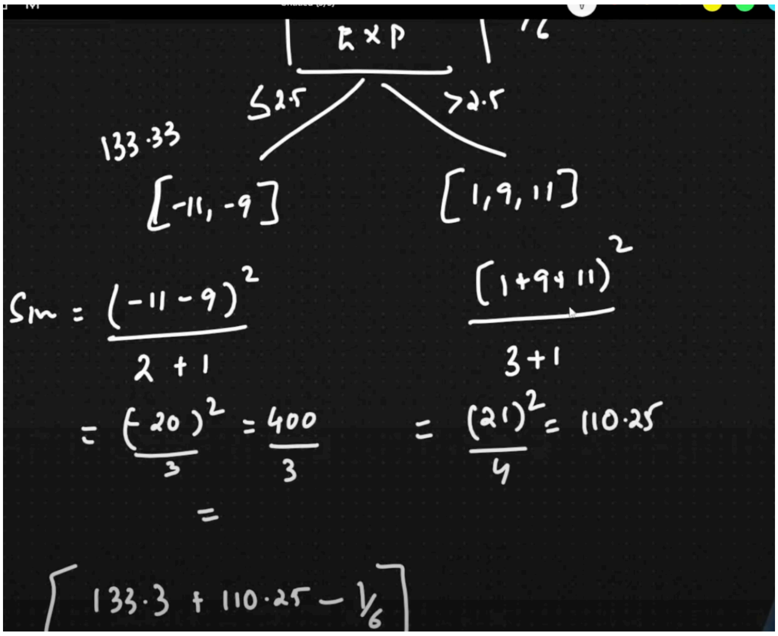
After then, we make decision tree as we did in **Classification, however,** the formula in order to calculate SImilarty weight is different which is as follow,

Similarity weight = $\Sigma(residuals)^2$**/ Num. of residents +** λ

Now, we will find information gain,

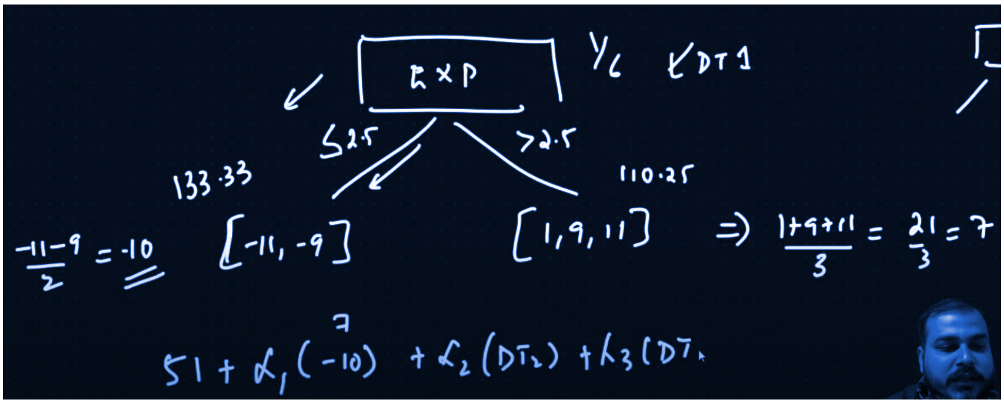
Information gain = 65.5 + 28.8 + ⅙ = 98.34

LIkewise, split each record and use whichever one is better and further we will use that for more split.



We will go for next split for better performance and we got this split is better than we had.

Now, We will see how final Output looks like,

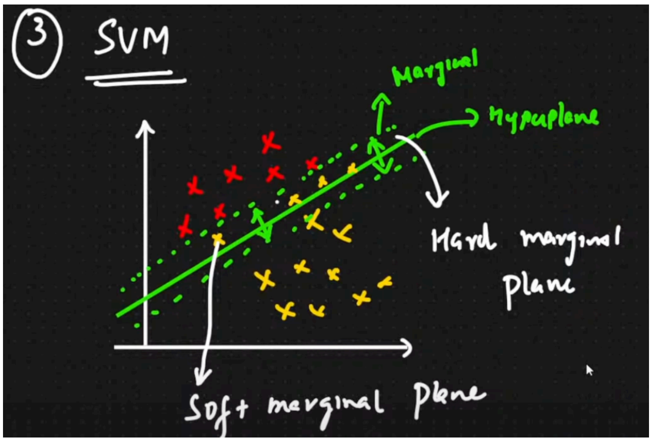


Let's consider than we further go from left leaf then we get, 51 + $\alpha 1(-10)$, here we got 51 from base model, and -10 which is the average, while if we have to go from right leaf then we get, 51 + $\alpha 1(7)$.

This is for one decision tree, **likewise** we add more decision trees parallel like this,

O/P = 51+ alpha1(DT1) + alpha2(DT2) + … + Alpha$_n$(DT$_n$)

SVM



In SVM, we make best-fit line which is known as a hyperplane and also make marginal planes. As the long margin between the marginal plan and hyperplane as we have, our model is more generalized.

Here, in the above image, the hyperplane is a hard marginal plane, and when we have some points that overlap like here yellow points, those are known as soft marginal planes as we will find errors.

Our aim would be that we want more margin between marginal and hyperplane even though we have a soft marginal plane.

When to use SVM?

- Binary target variable
- The feature-to-row ratio is very high
- Very complex relationships
- Lots of outliers

When not to use SVM?

- The feature-to-row ratio is very low
- Transparency is important or interested in the significance of predictors
- Looking for a quick benchmark model