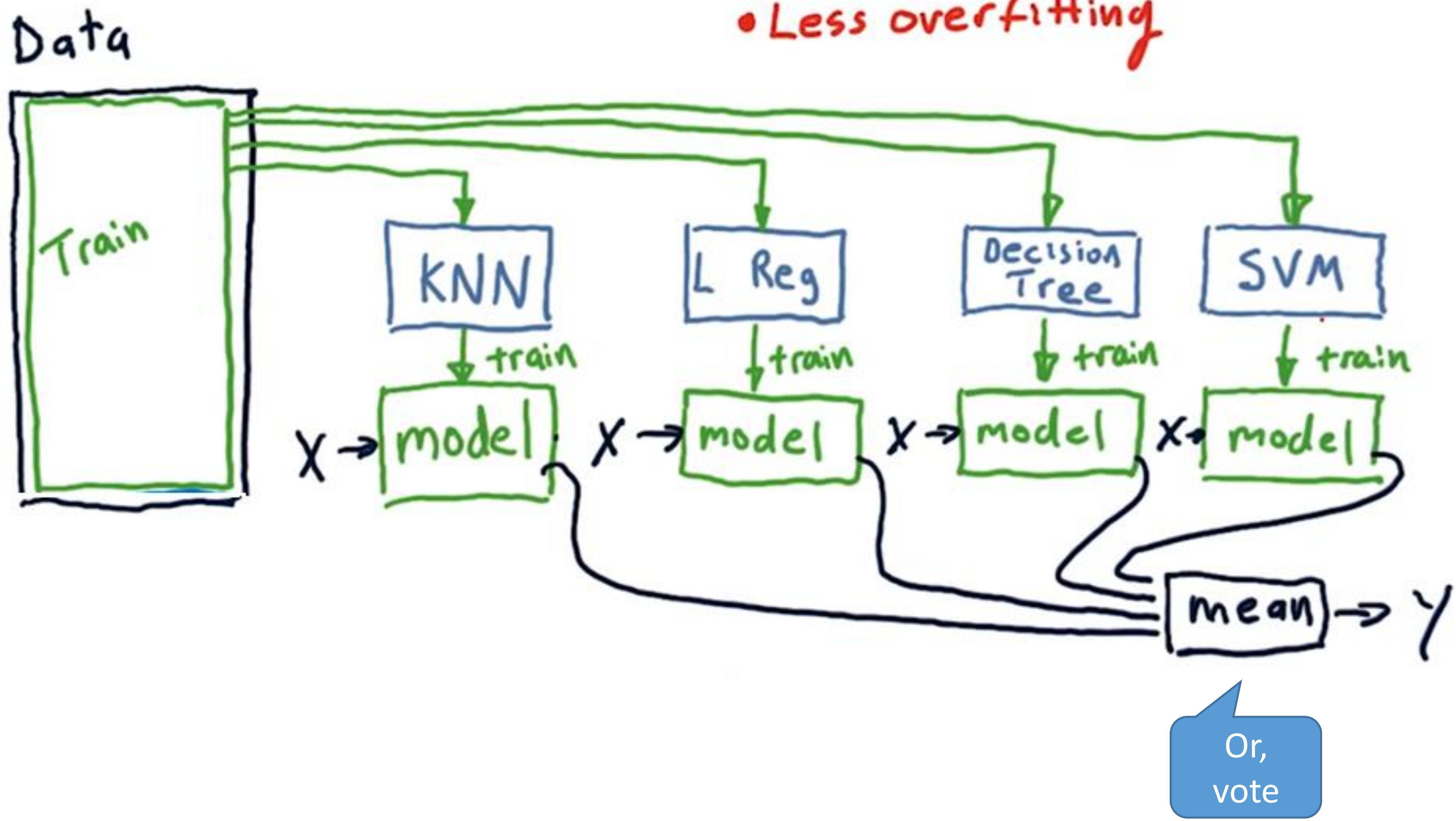
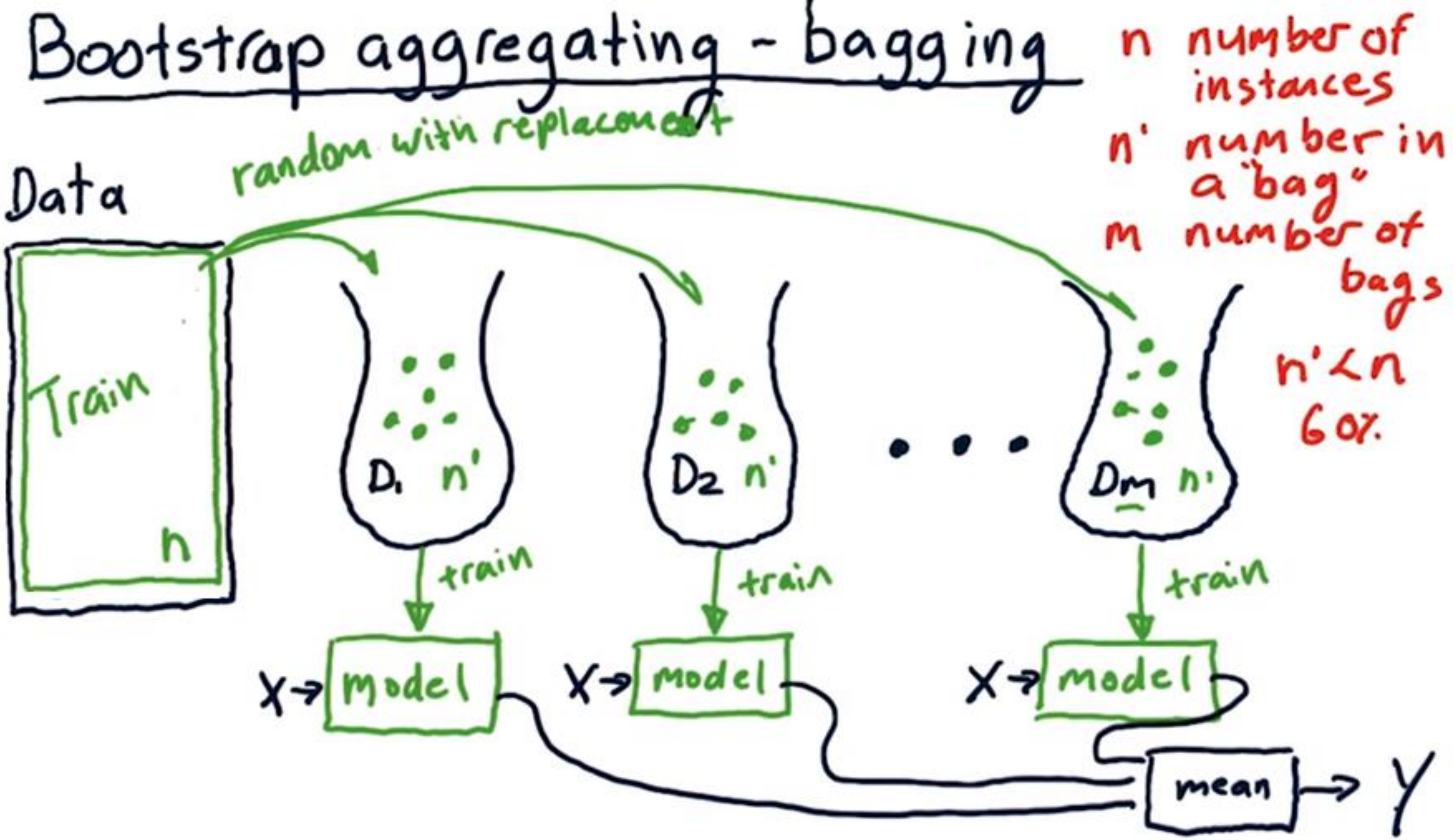
Ensemble Learning

Ensemble learners

Why ensembles?
- Lower error
- Less overfitting

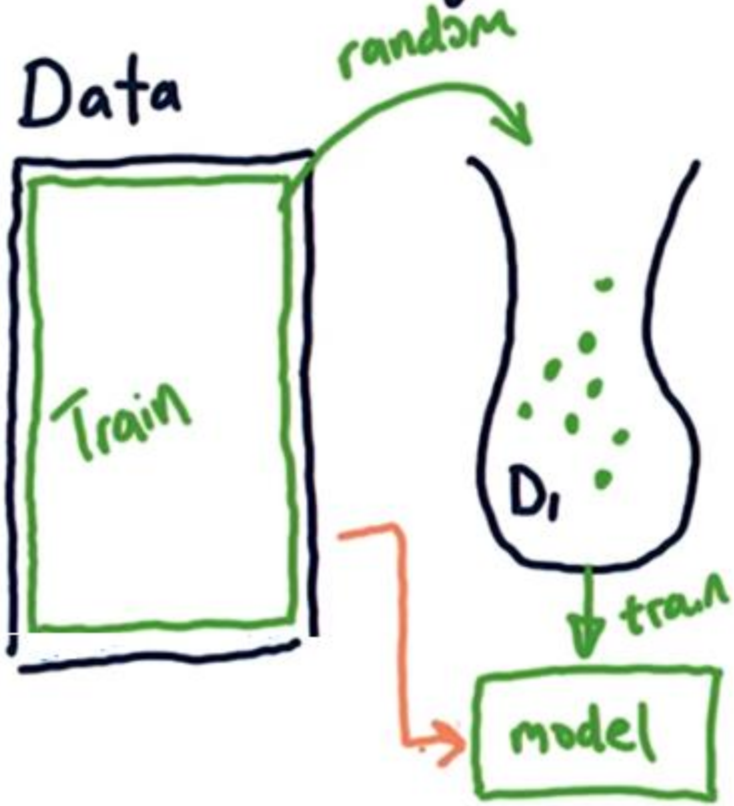


https://www.youtube.com/watch?v=Un9zObFjBH0



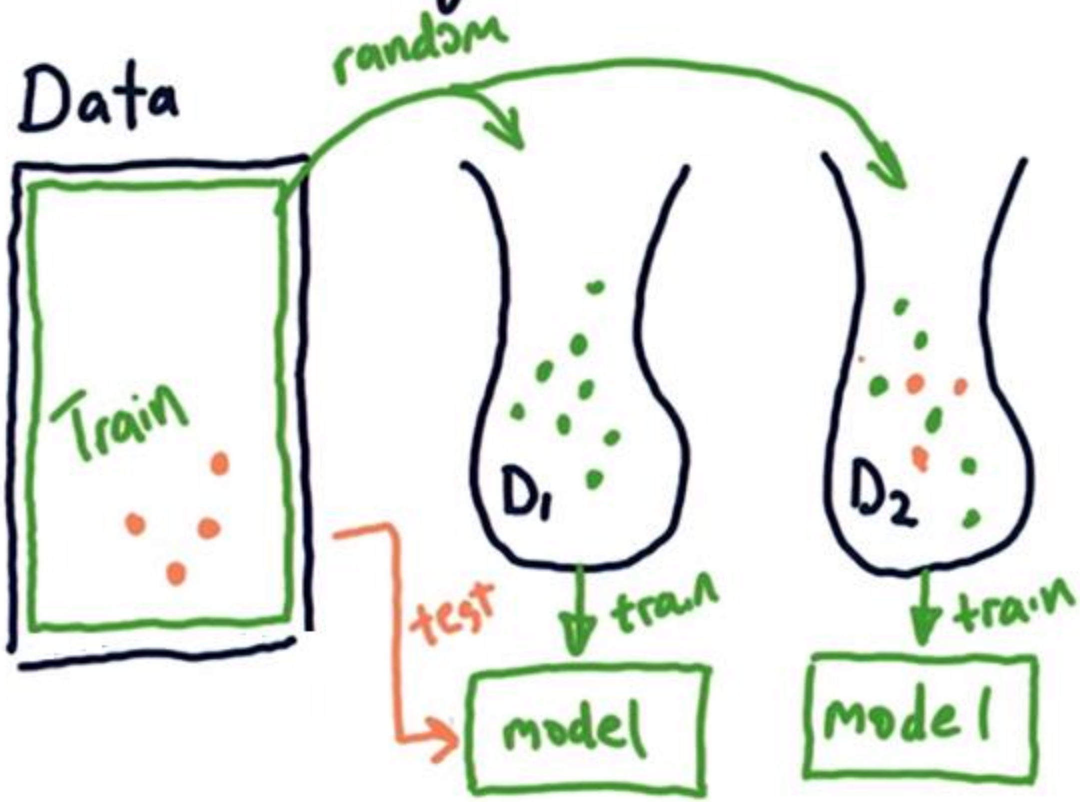
https://www.youtube.com/watch?v=2Mg8QD0F1dQ

Boosting: Ada Boost



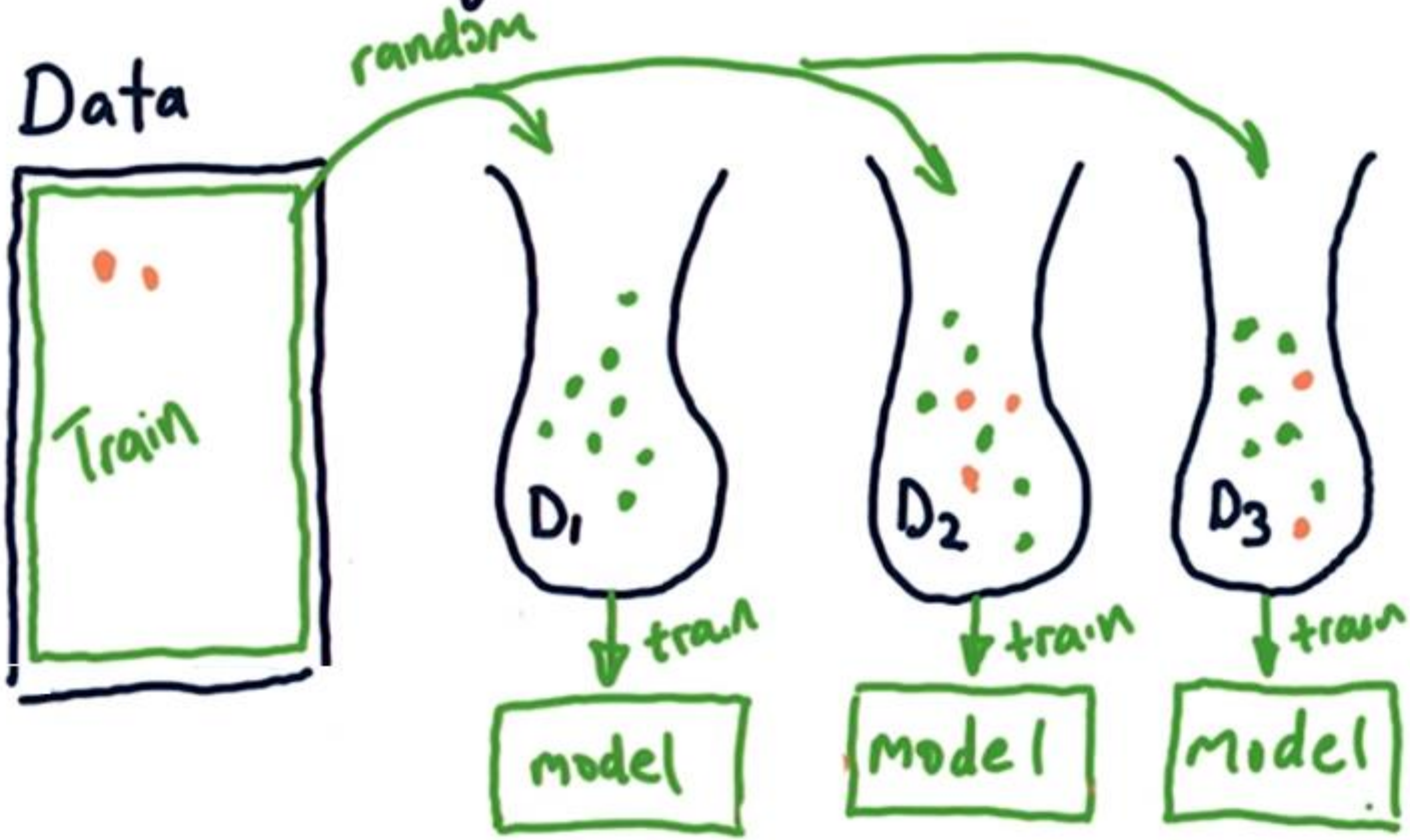
https://www.youtube.com/watch?v=GM3CDQfQ4sw&t=7s

Boosting: Ada Boost



https://www.youtube.com/watch?v=GM3CDQfQ4sw&t=7s

Boosting: Ada Boost



https://www.youtube.com/watch?v=GM3CDQfQ4sw&t=7s

See also:

https://github.com/knathanieltucker/data-science-foundations

XGBoost

http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting

Example

Task: Predict Age

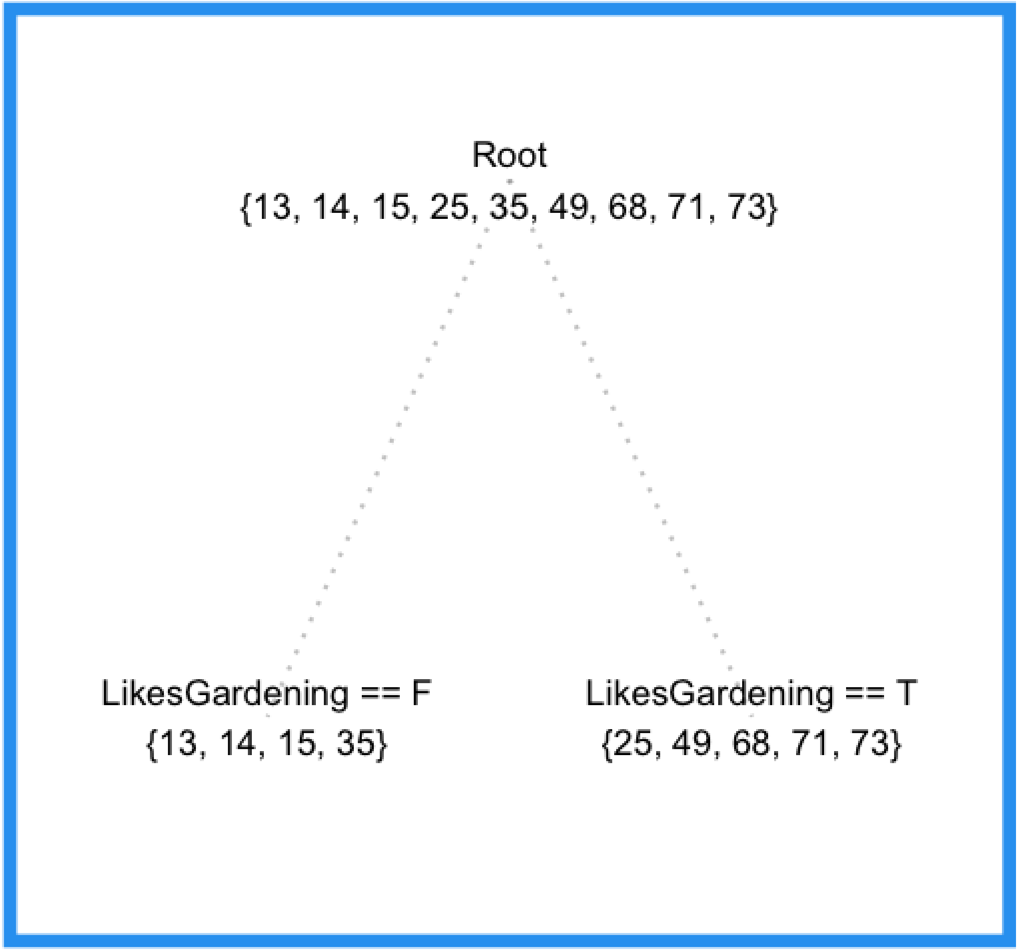
PersonID	LikesGardening	PlaysVideoGames	LikesHats	Age
1	FALSE	TRUE	TRUE	13
2	FALSE	TRUE	FALSE	14
3	FALSE	TRUE	FALSE	15
4	TRUE	TRUE	TRUE	25
5	FALSE	TRUE	TRUE	35
6	TRUE	FALSE	FALSE	49
7	TRUE	TRUE	TRUE	68
8	TRUE	FALSE	FALSE	71
9	TRUE	FALSE	TRUE	73

Intuitively, we might expect
– The people who like gardening are probably older
– The people who like video games are probably younger
– *LikesHats* is probably just random noise

Let's check these assumptions:

Feature	**FALSE**	**TRUE**
LikesGardening	{13, 14, 15, 35}	{25, 49, 68, 71, 73}
PlaysVideoGames	{49, 71, 73}	{13, 14, 15, 25, 35, 68}
LikesHats	{14, 15, 49, 71}	{13, 25, 35, 68, 73}

Tree 1

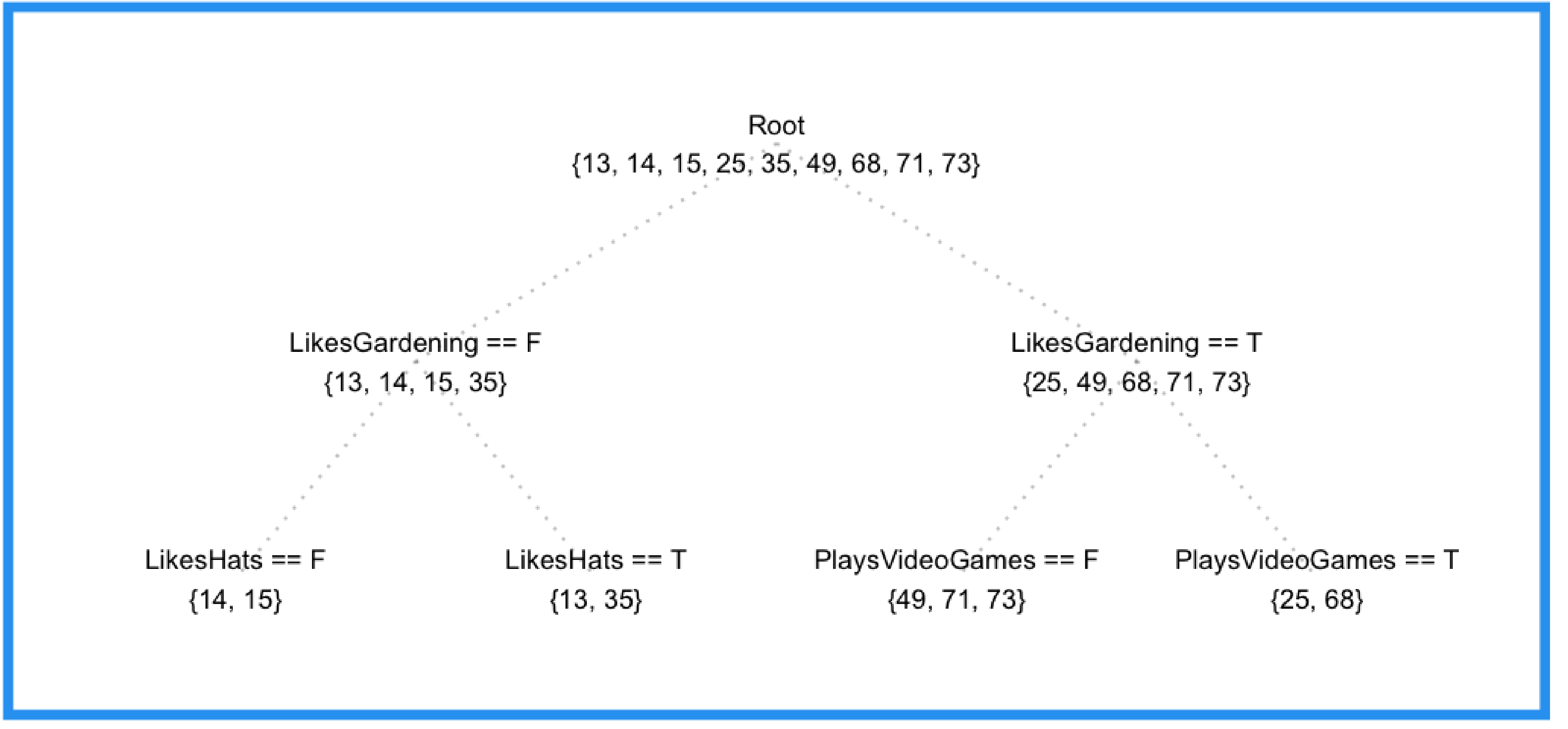


Let’s model the data with a regression tree.

To start, we’ll require that terminal nodes have at least three instances.

With this condition, the regression tree will make its first and last split on **LikesGardening**.

Overfit Tree



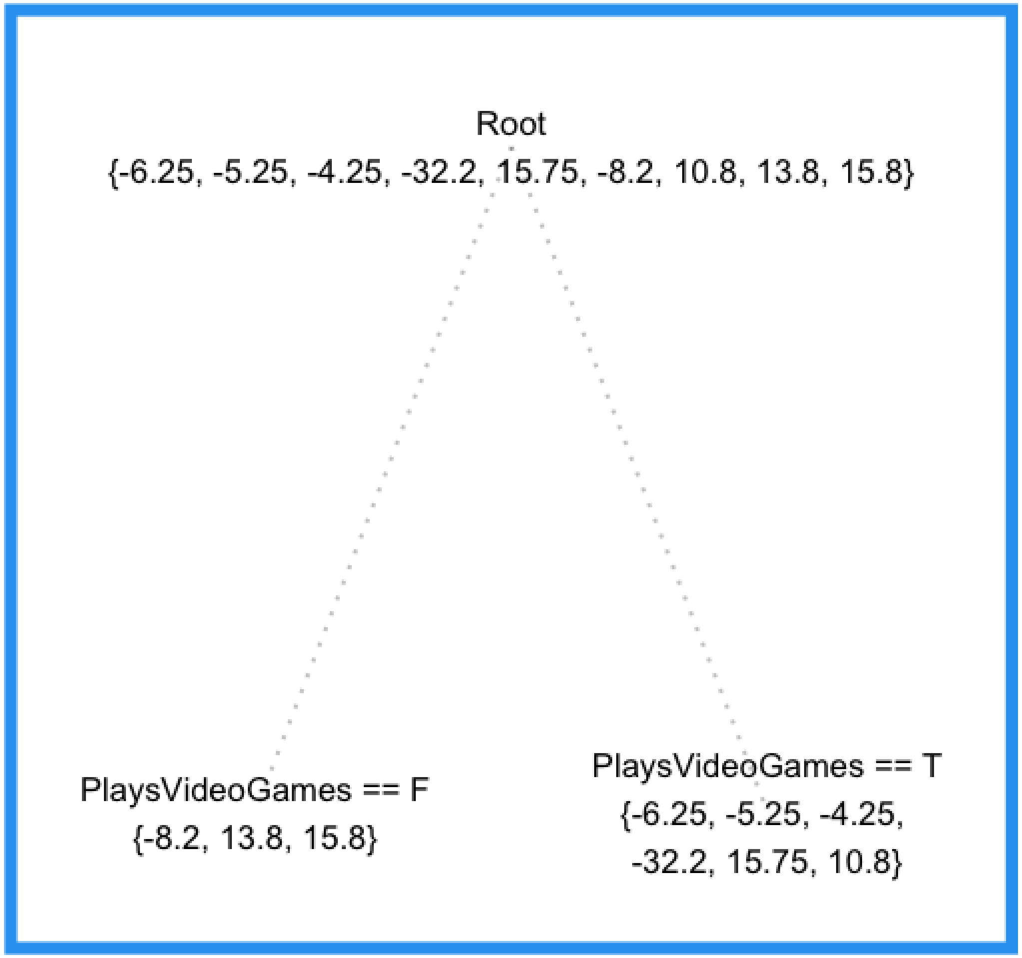
Let's try letting terminal nodes have 2 instances.

Here we pick up some information from *PlaysVideoGames* but we also pick up information from *LikesHats* – a good indication that we're overfitting and our tree is splitting random noise.

A better approach – model residuals

PersonID	Age	Tree1 Prediction	Tree1 Residual
1	13	19.25	-6.25
2	14	19.25	-5.25
3	15	19.25	-4.25
4	25	57.2	-32.2
5	35	19.25	15.75
6	49	57.2	-8.2
7	68	57.2	10.8
8	71	57.2	13.8
9	73	57.2	15.8

Tree2



Now we can fit a second regression tree to the residuals of the first tree.

In other words, for the second regression tree, we will not use **age** as y, but **Tree1Residual**.

Doing that, selects **PlayVideoGames** as the attribute at the root.

Now we can improve the predictions from our first tree by adding the “error-correcting” predictions from this tree.

PersonID	Age	Tree1 Prediction	Tree1 Residual	Tree2 Prediction	Combined Prediction	Final Residual
1	13	19.25	-6.25	-3.567	15.68	2.683
2	14	19.25	-5.25	-3.567	15.68	1.683
3	15	19.25	-4.25	-3.567	15.68	0.6833
4	25	57.2	-32.2	-3.567	53.63	28.63
5	35	19.25	15.75	-3.567	15.68	-19.32
6	49	57.2	-8.2	7.133	64.33	15.33
7	68	57.2	10.8	-3.567	53.63	-14.37
8	71	57.2	13.8	7.133	64.33	-6.667
9	73	57.2	15.8	7.133	64.33	-8.667

Tree1 SSE	Combined SSE
1994	1765

Gradient Boosting

1. Fit a model to the data, $F_1(x) \sim y$
2. Fit a model to the residuals, $h_1(x) \sim y - F_1(x)$
3. Create a new model, $F_2(x) = F_1(x) + h_1(x)$

We can repeat this process

4. Fit a model to the residuals, $h_2(x) \sim y - F_2(x)$
5. Create a new model, $F_3(x) = F_2(x) + h_2(x)$

…

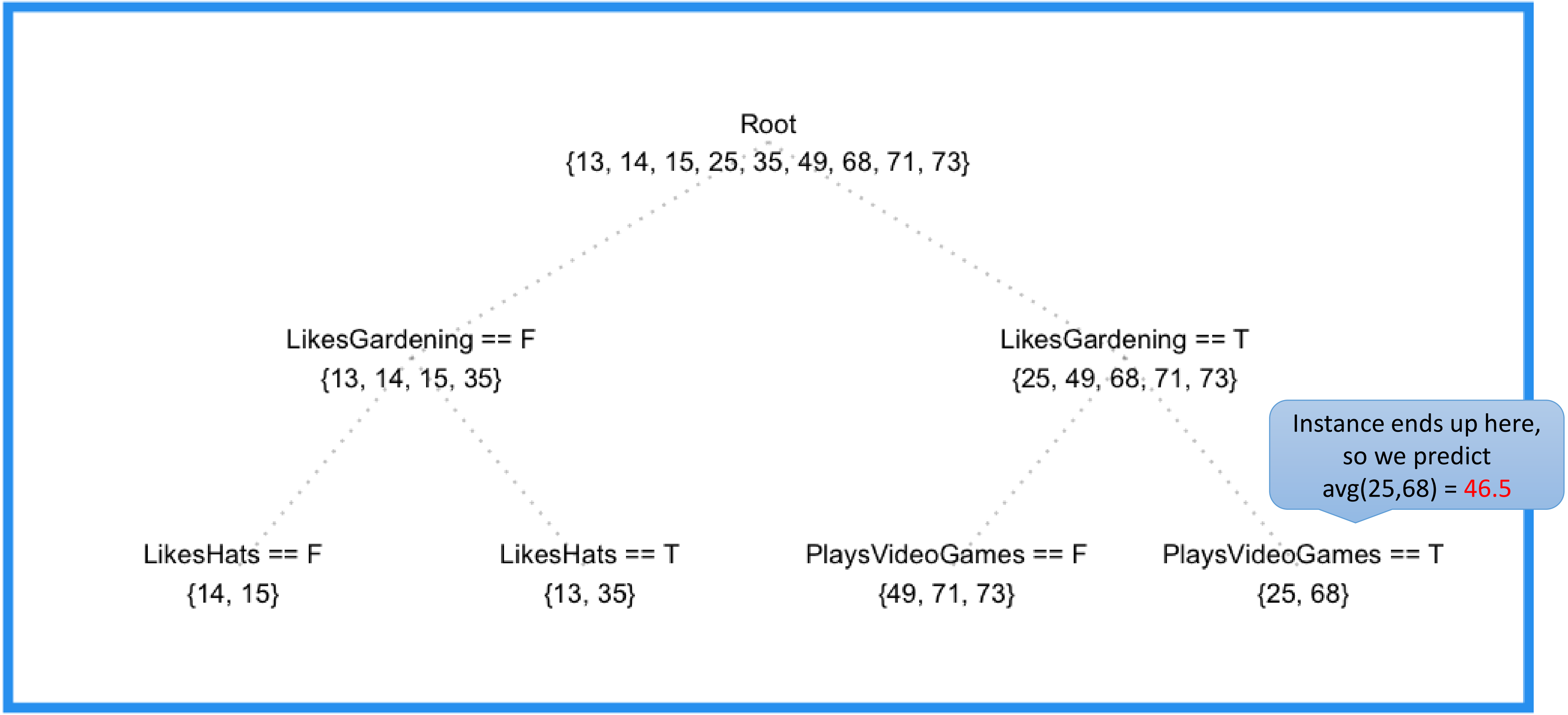
What we get, XGBoost, is one of most used algorithms in Kaggle competitions.

How do we predict?

Let's see first, what prediction we get with the overfit tree

PersonID	LikesGardening	PlaysVideoGames	LikesHats	Age
100	TRUE	TRUE	TRUE	?

Overfit Tree

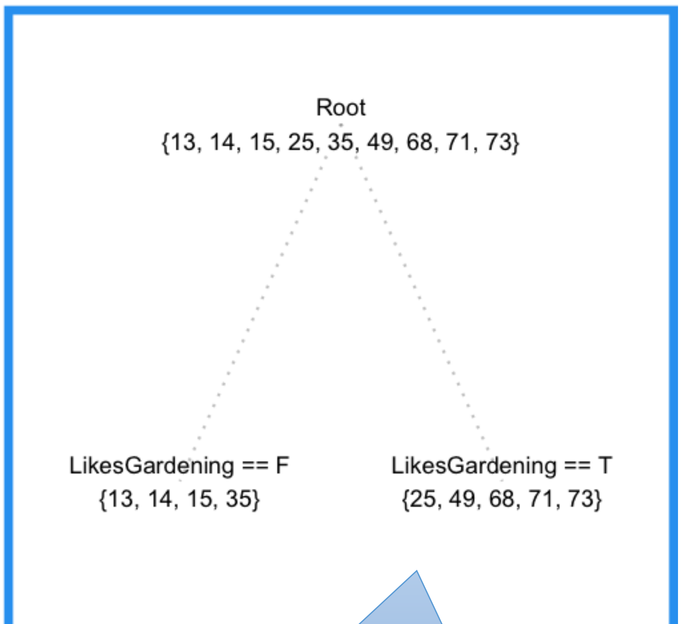


Now let's see prediction with XGBoost.
We run the instance on both Tree1 and Tree 2 and add the results.

How do we predict?

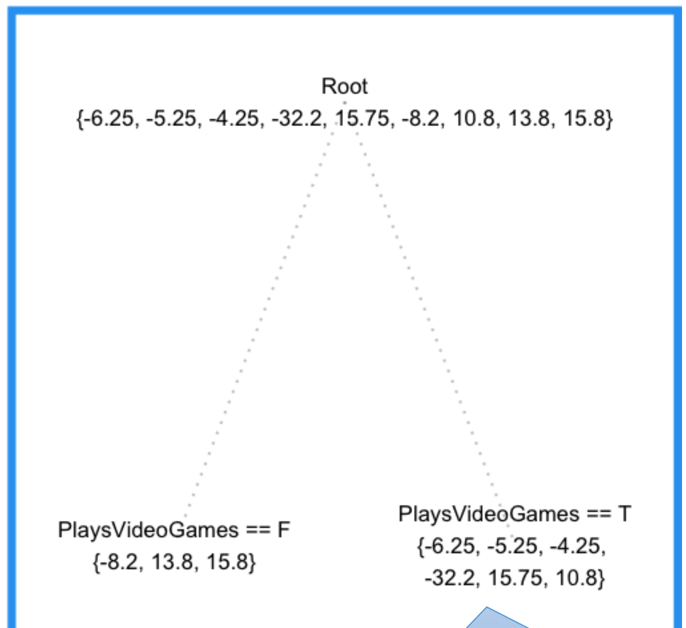
PersonID	LikesGardening	PlaysVideoGames	LikesHats	Age
100	TRUE	TRUE	TRUE	?

Tree 1



Instance ends up here, so we predict
avg(25, 49, 68 ,71, 73) = 57.2

Tree2



Instance ends up here, so we predict
avg(-6.25, -5.25, -4.25, -32.2, 15.75, 10.8) = -3.6

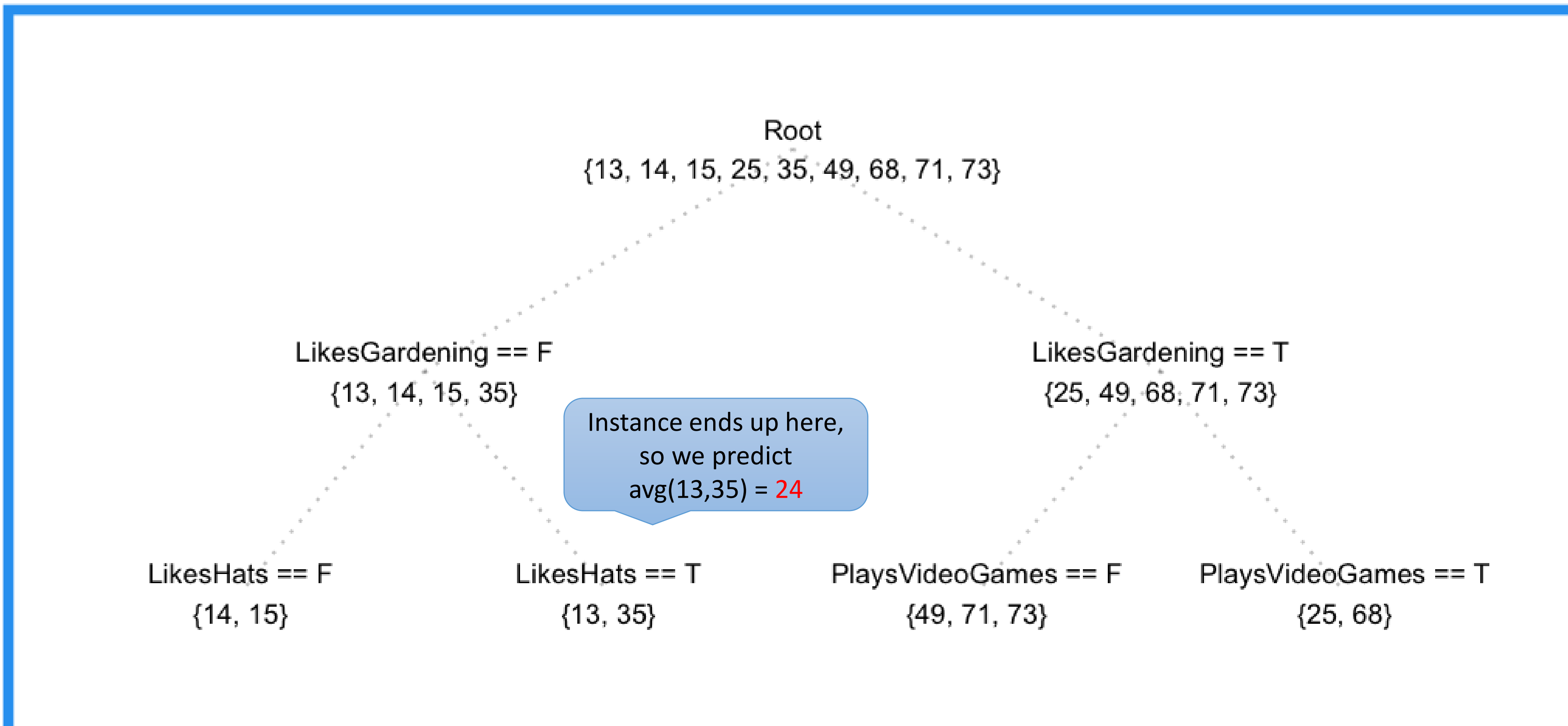
Combined prediction
57.2 - 3.6 = 53.6

Another instance

Let's see first, what prediction we get with the overfit tree

PersonID	LikesGardening	PlaysVideoGames	LikesHats	Age
100	FALSE	FALSE	TRUE	?

Overfit Tree

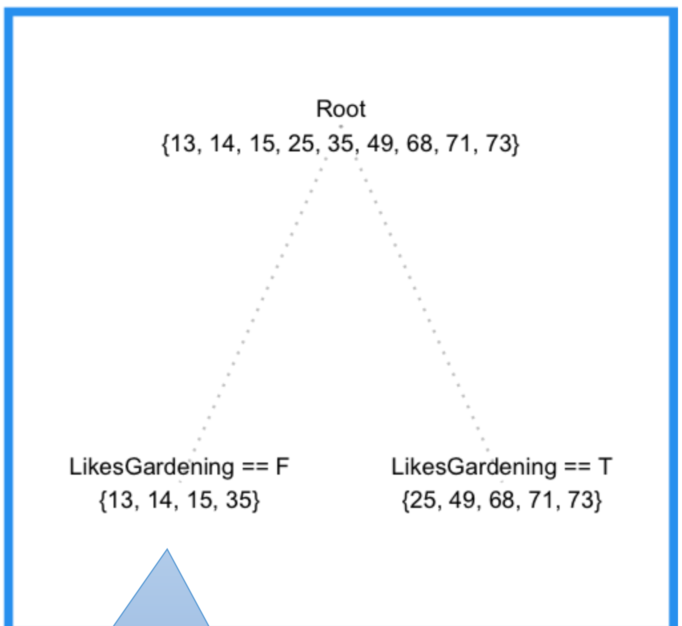


Now let's see prediction with XGBoost.
We run the instance on both Tree1 and Tree 2 and add the results.

Another instance

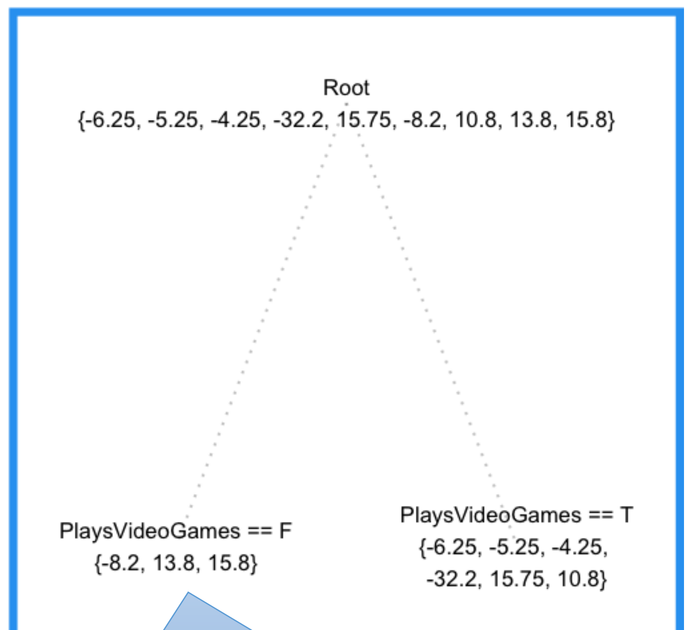
PersonID	LikesGardening	PlaysVideoGames	LikesHats	Age
100	FALSE	FALSE	TRUE	?

Tree 1



Instance ends up here, so we predict avg(13, 14, 15 ,35) = 19.25

Tree2



Instance ends up here, so we predict avg(-8.2, 13.8, 15.8) = 7.1

Combined prediction
19.25 + 7.1 = 26.35