Ensemble Learning

Dr. Gaurav Kumar Ameta
Associate Professor
Department of CSE
Parul Institute of Technology

Introduction to Ensemble Learning

- Suppose if you interested to go for Goa trip with your 4 friends.
- In that case you will call your friends and ask about their availability.
- You received the following response by your friends as follows:

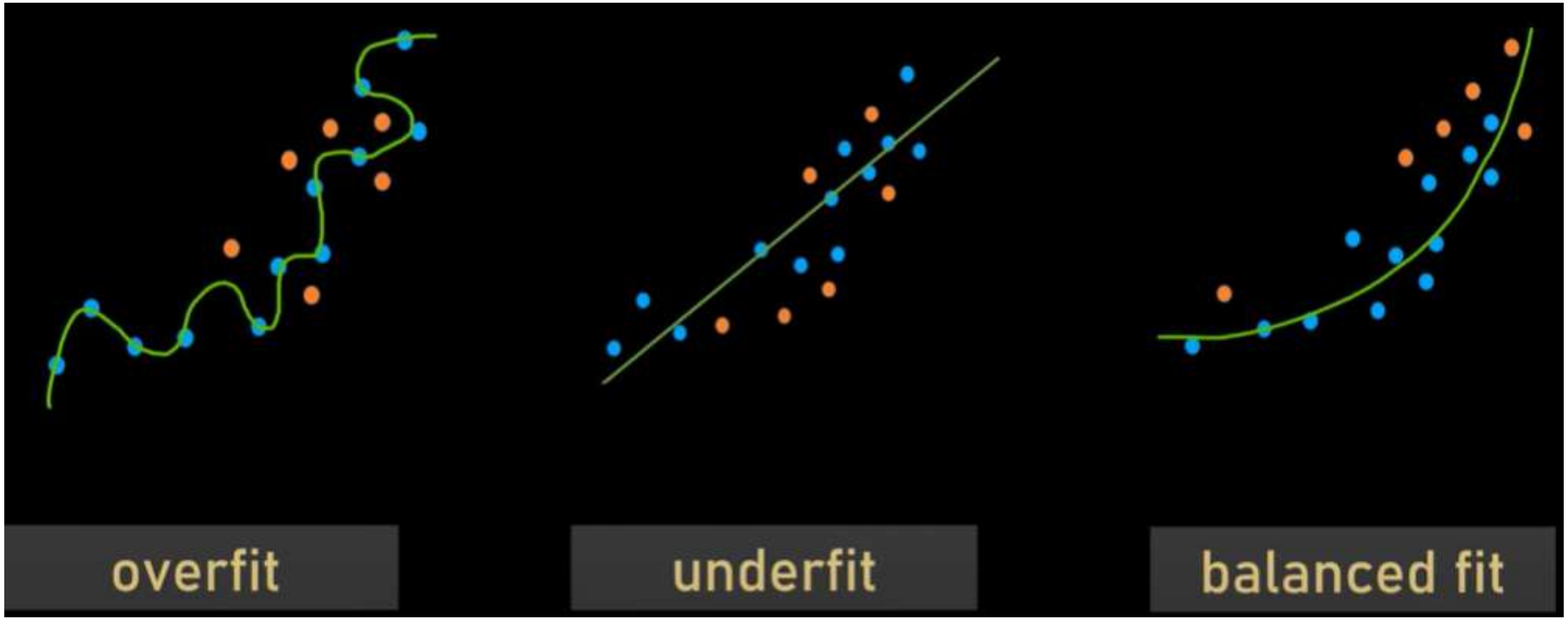
1st Friend→Yes, 2nd Friend→No, 3rd Friend→No, 4th Friend→No

- On the basis of their response 3 No's and 1 Yes received.

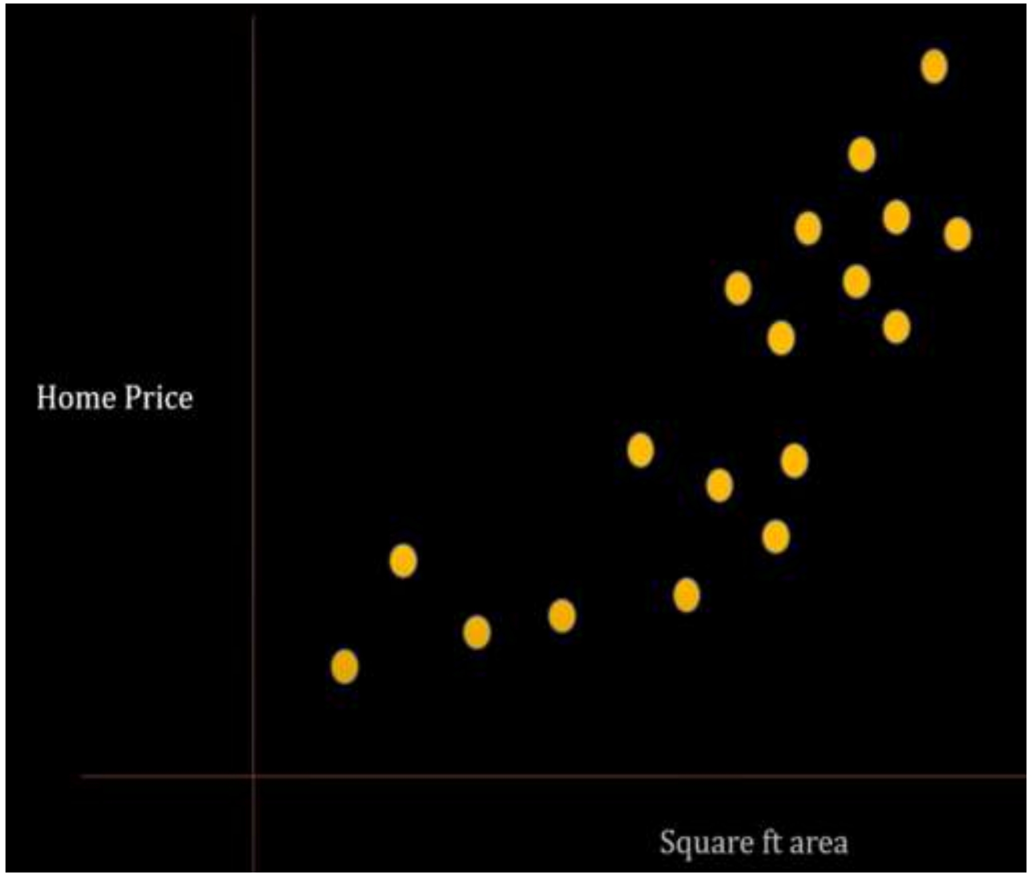
Introduction to Ensemble Learning

- So on the basis of majority you will decide no to go for Goa Trip.
- In the same way, in the Machine Learning, if you have one model and if you train that model using the complete dataset, the model may get overfit or may suffer high variance issues.
- To resolve the high variance issues, we can go for ensemble learning.

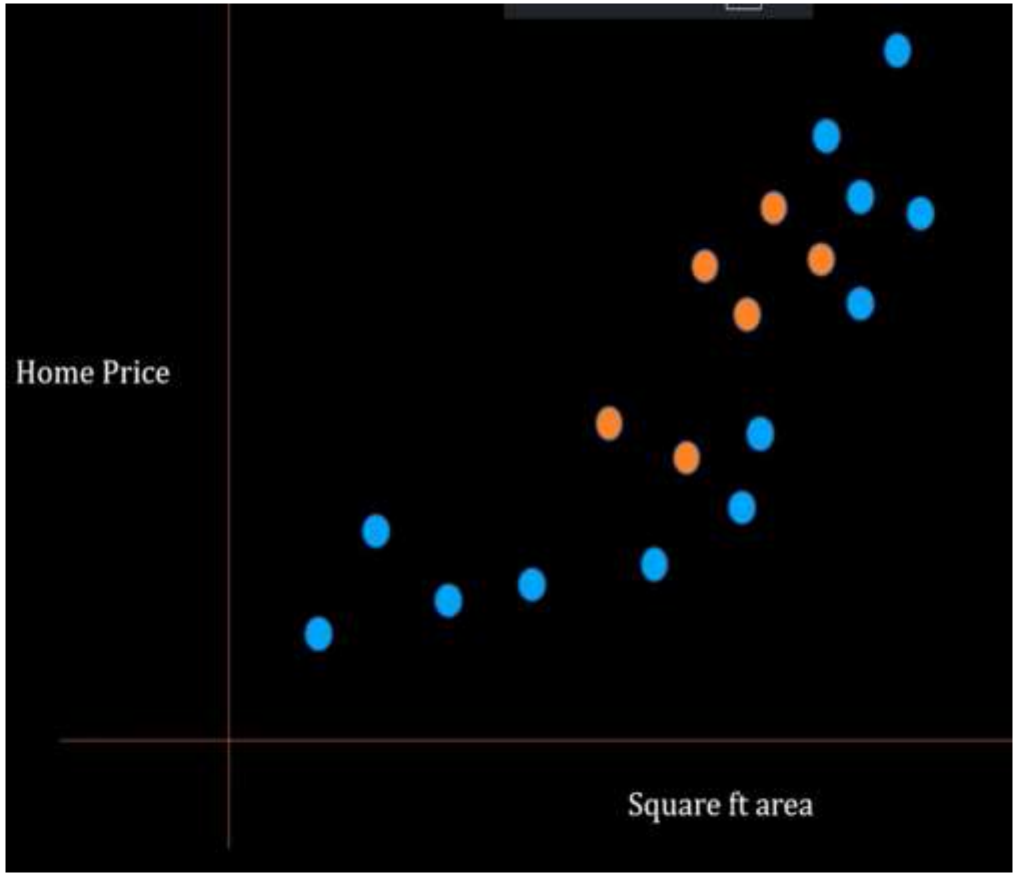
Model Overfit, Underfit and Balanced Fit



Take an example of house price prediction

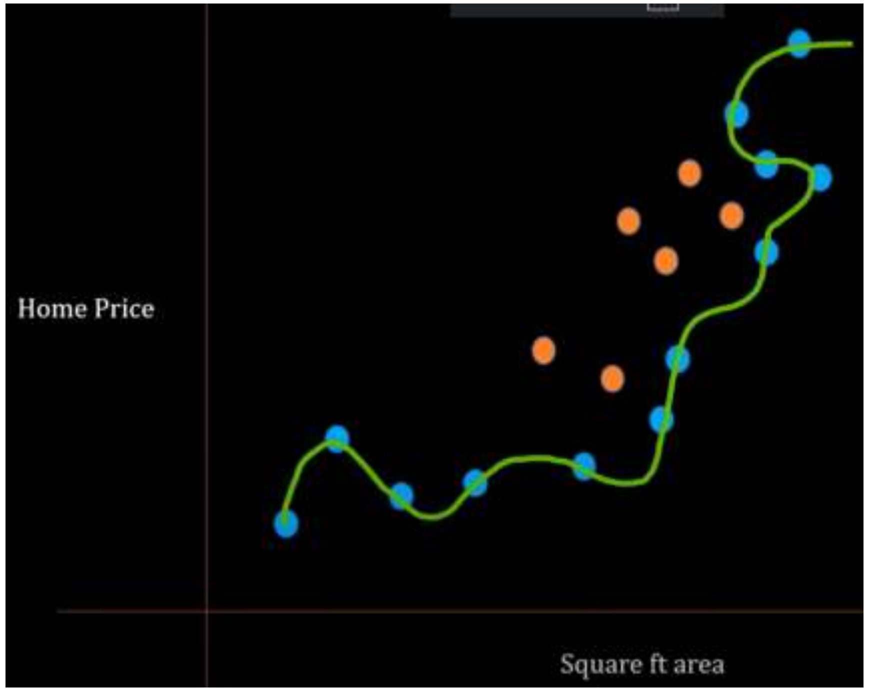


Overall Data

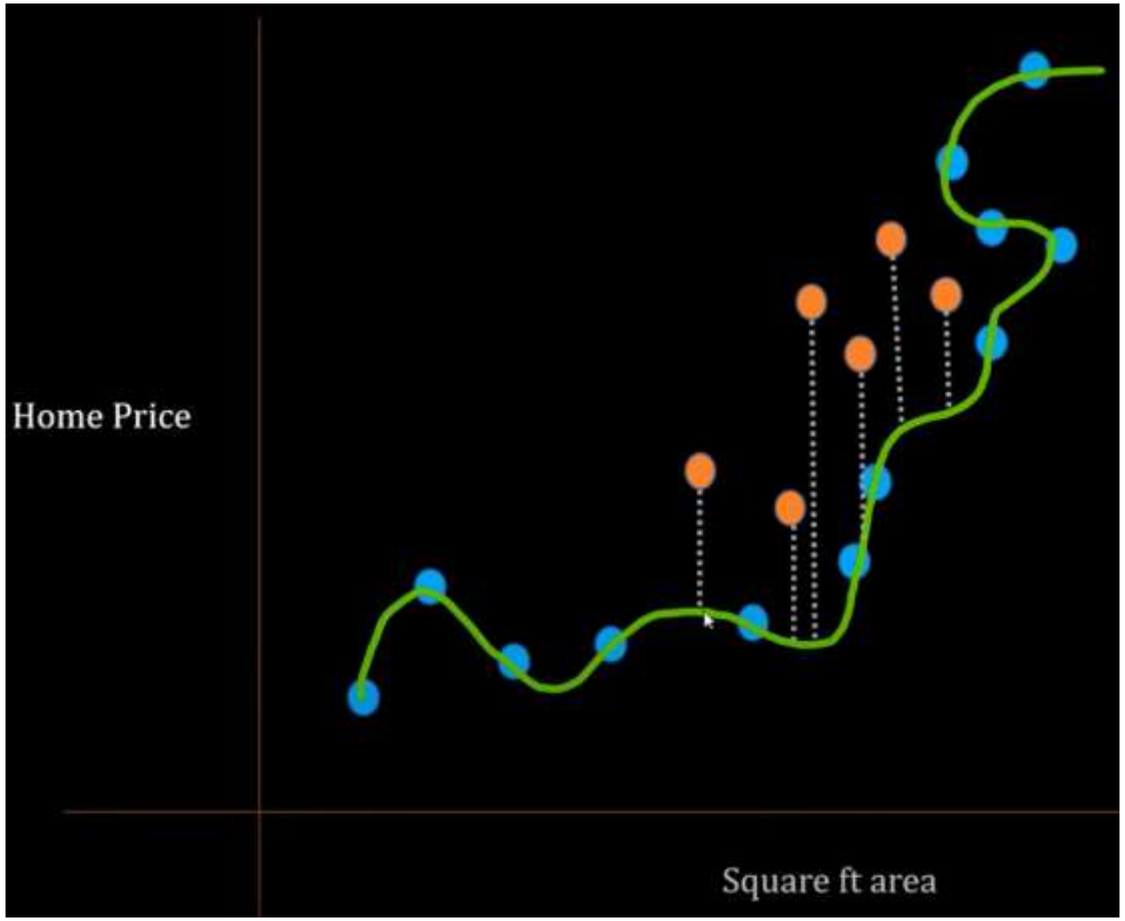


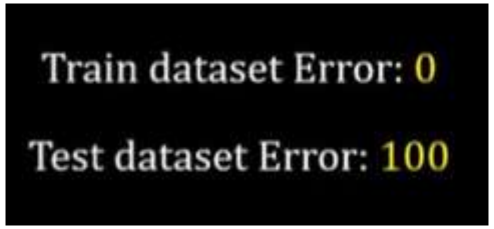
Data divided into Train (Blue) and Test samples (Red)

- We need to train the model which fits to the blue dots.
- Our training methodology is like that our model becomes overfit like below. Model tries to fit exactly to the training samples. **Here the training dataset error is close to 0.**



- **But the test dataset is having an error (See the next diagram).**



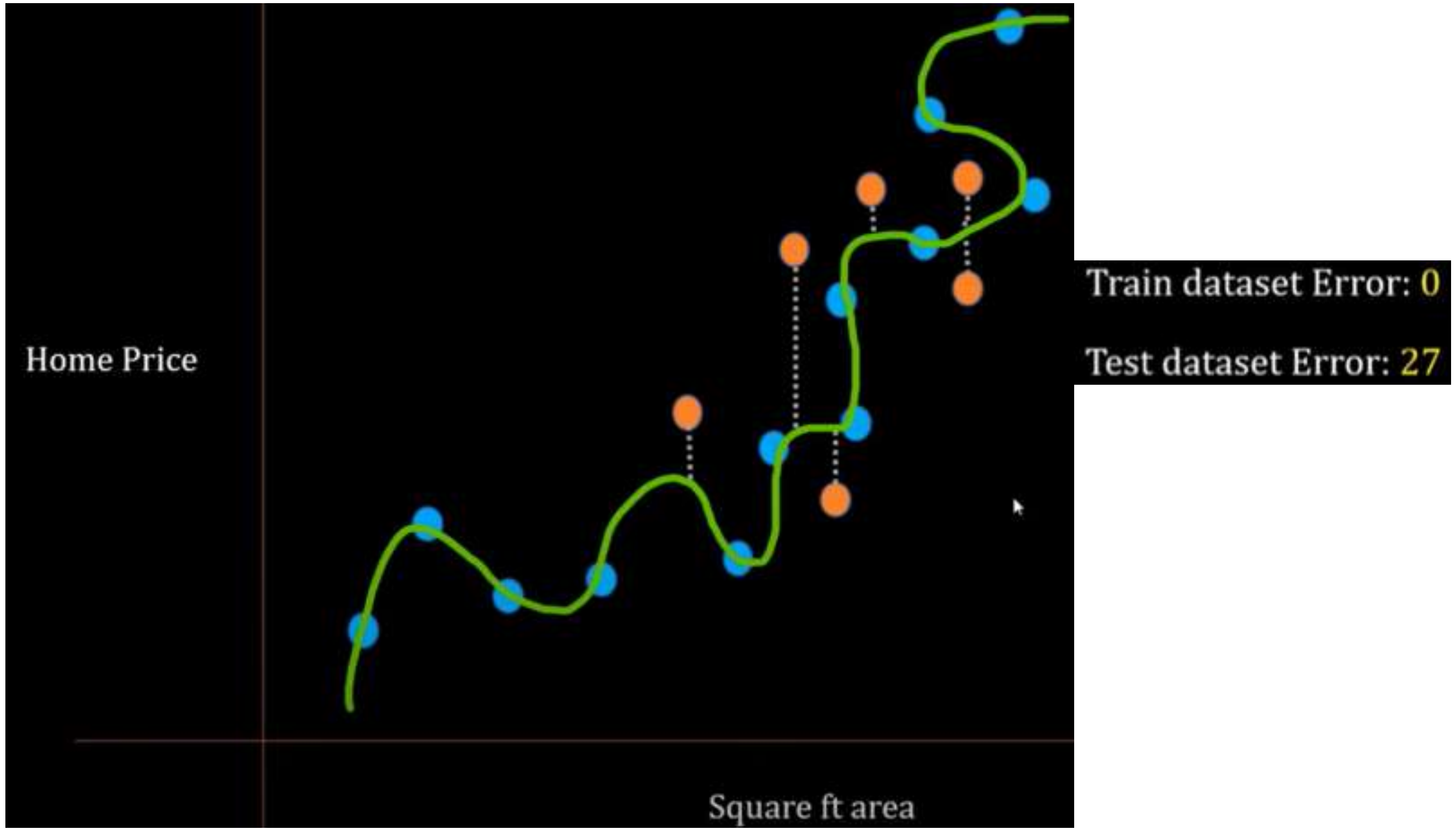

Train dataset Error: 0

Test dataset Error: 100

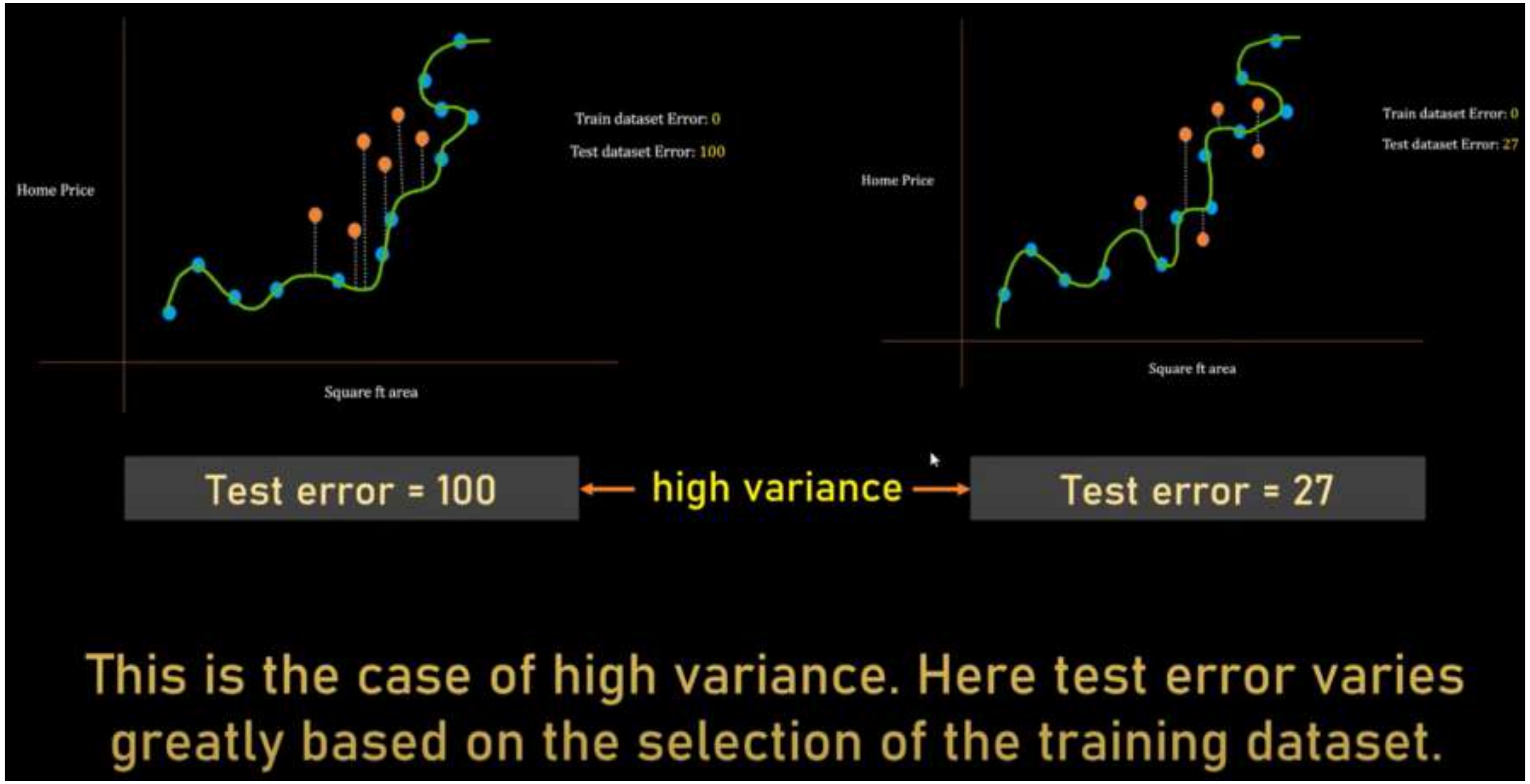
100 is the just an assumed value as an overall error.

These vertical dotted white lines shows the test errors for the model trained.

- To avoid these errors in the testing, it is advised to pick random samples for the training. So, at the time of testing, error could be low and model could be little bit generalized. (like below Figure)



Compare above two scenarios



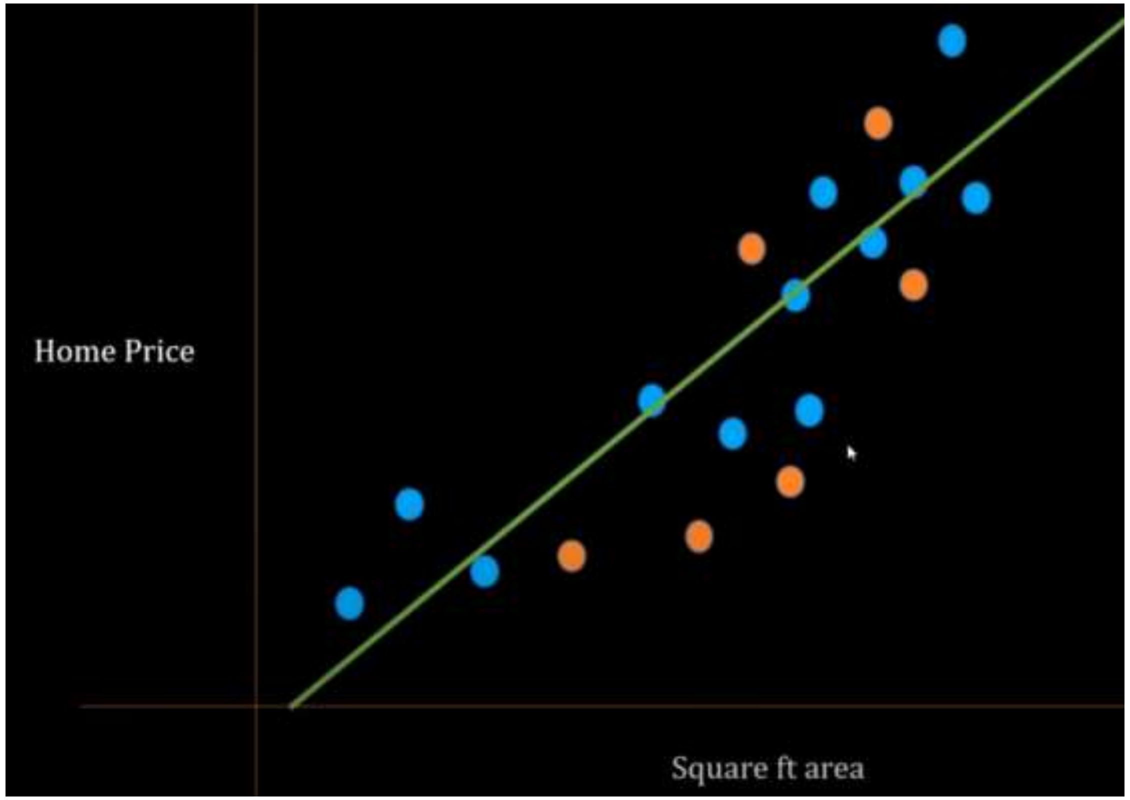
Test error = 100 ← high variance → Test error = 27

This is the case of high variance. Here test error varies greatly based on the selection of the training dataset.

In Simple Words, **When test results vary largely /highly dependent on the basis of the selection of training and test samples**. It is called as **high variance.**

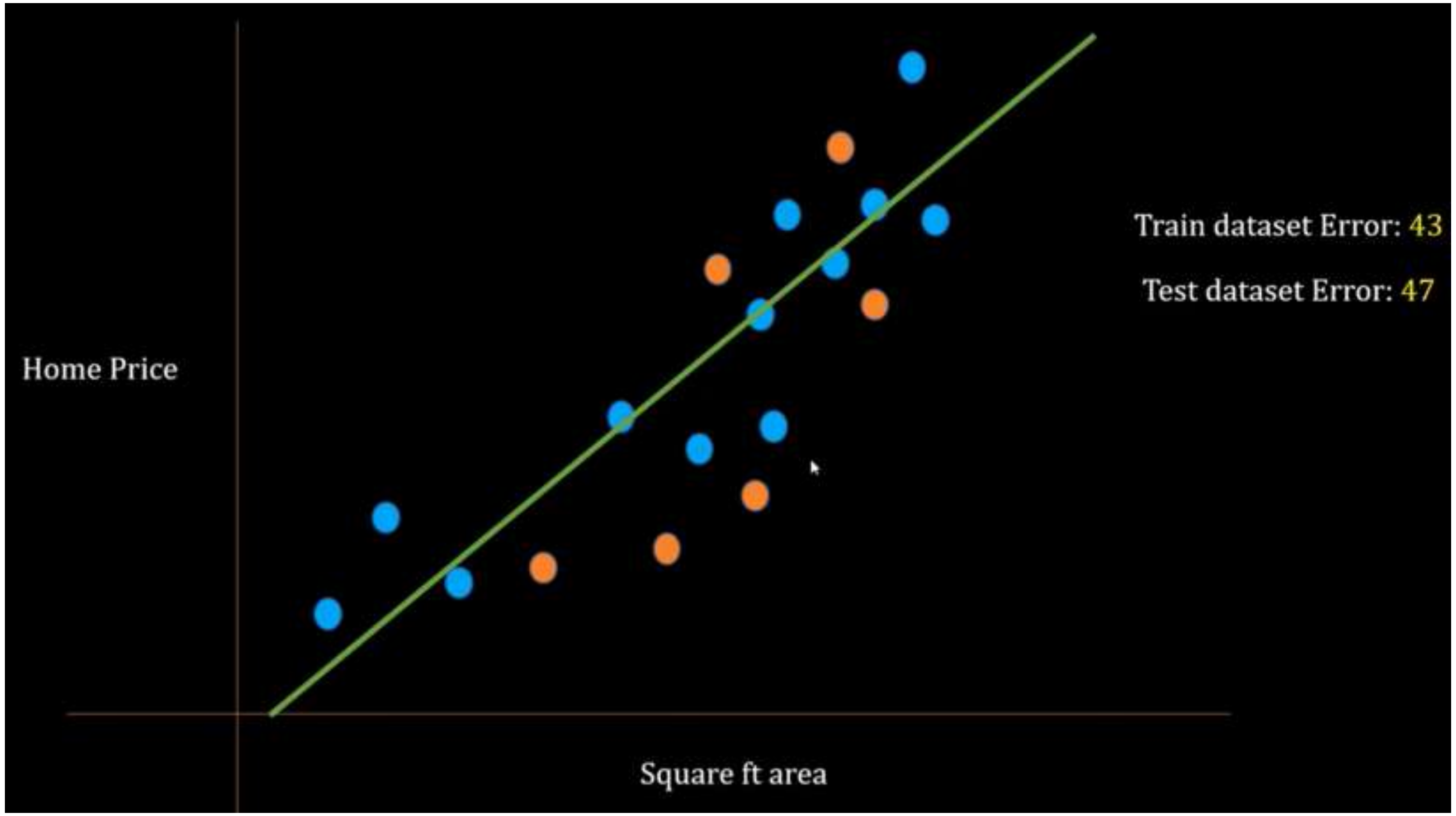
Overfitting= Low Bias and High Variance

Now we come up with a very simple approach of linear regression (Straight Line)

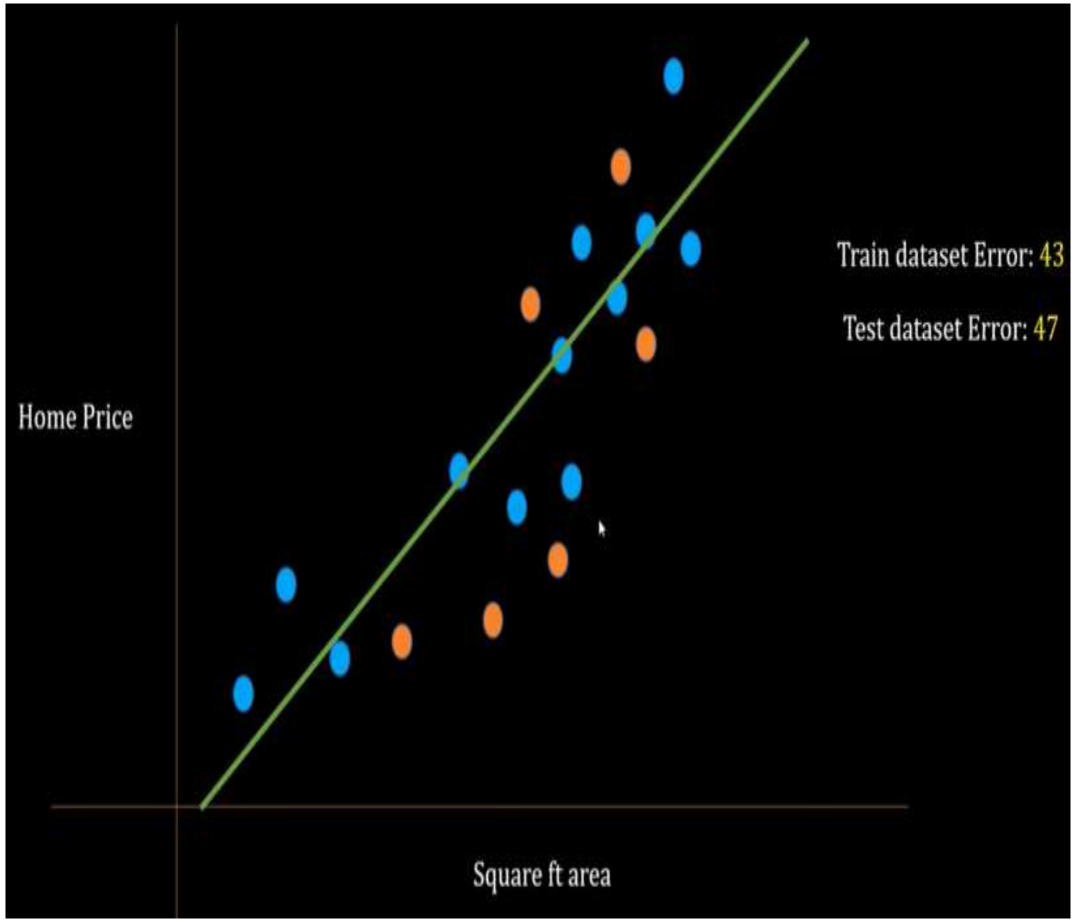


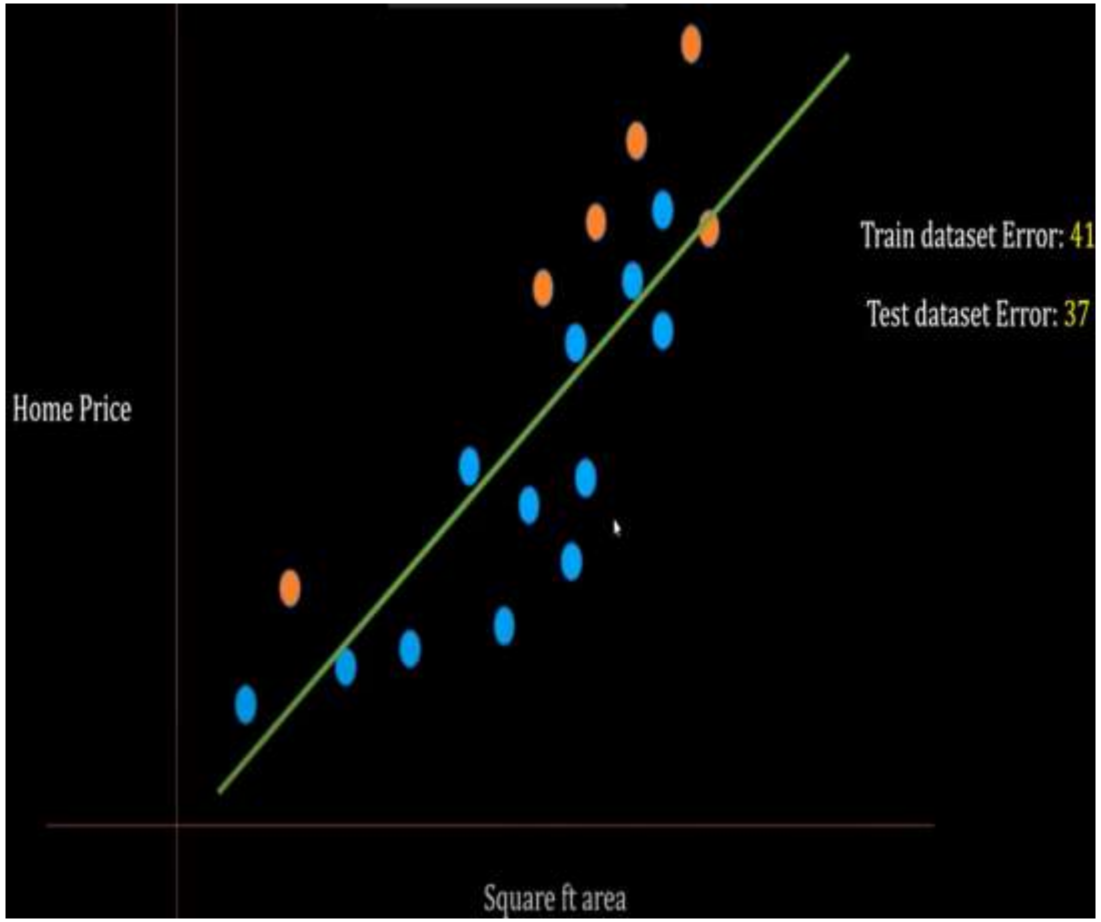
Clearly, it is an example of underfitting equation. It cannot truly capture the pattern of training samples.

So, in the training and testing both we are having big error.

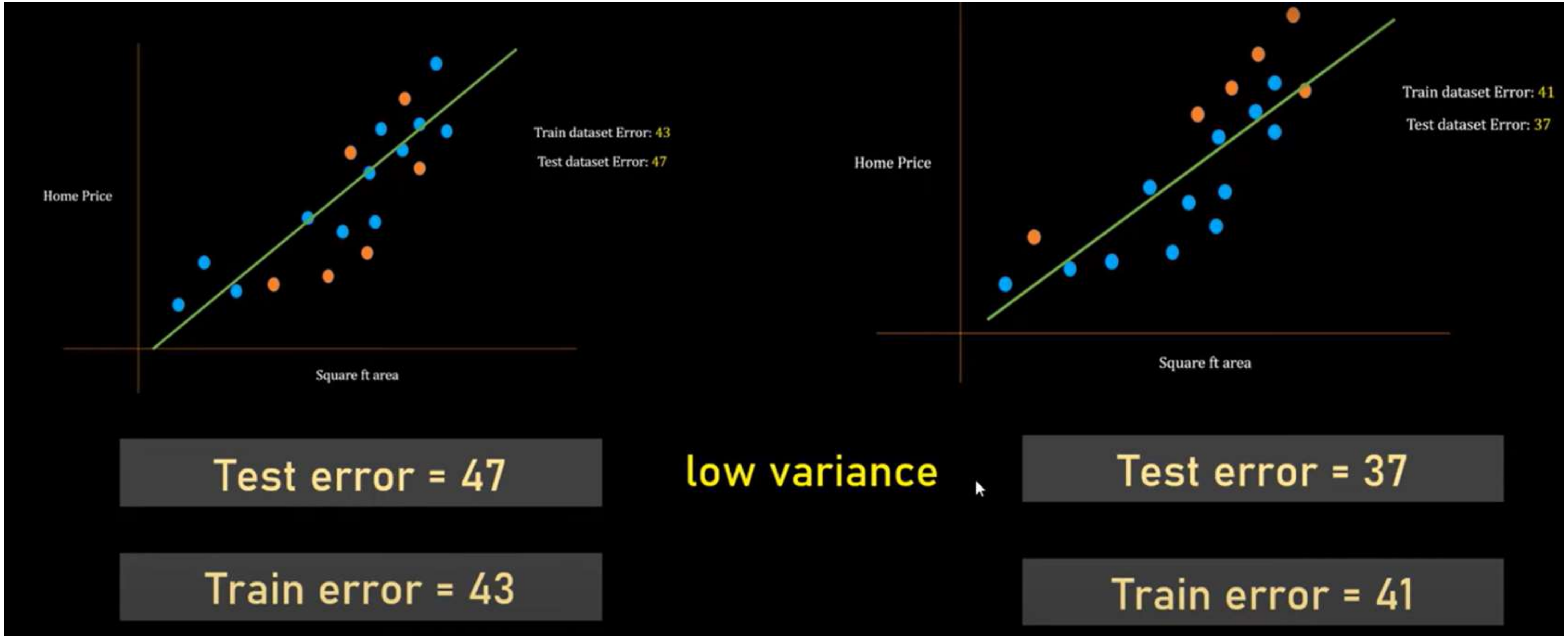


If you change the train and test samples around the line





Then, also there is not a big change or improvement in the training and testing error.

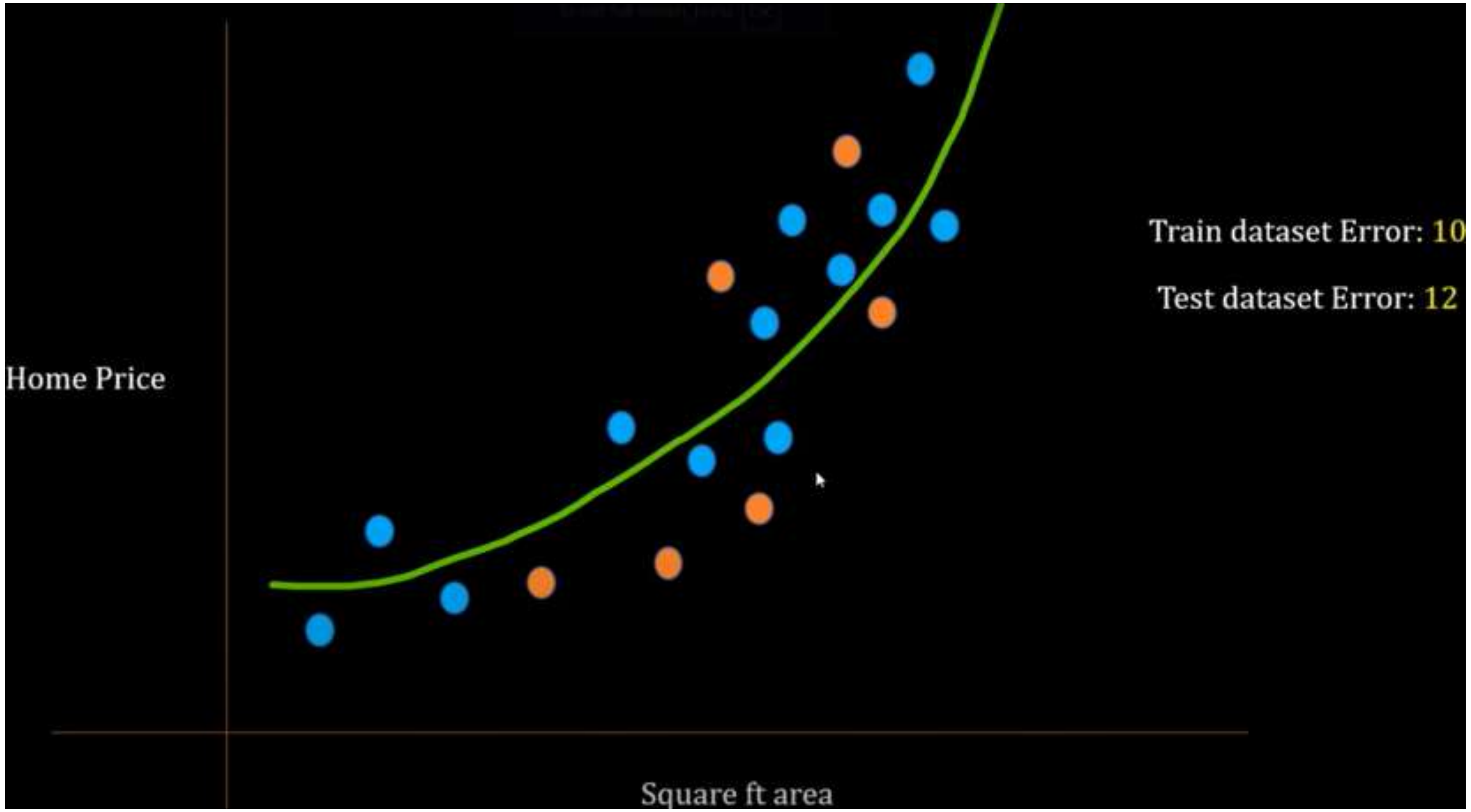


Even if we are changing the data samples for the **training** but **test error does not vary too much. It is called low variance.**

Bias is a measurement of how accurately a model can capture a pattern in the training dataset. **In above case training error is big so, it is said to have a high bias. [Underfitting→ High Bias and Low Variance]**

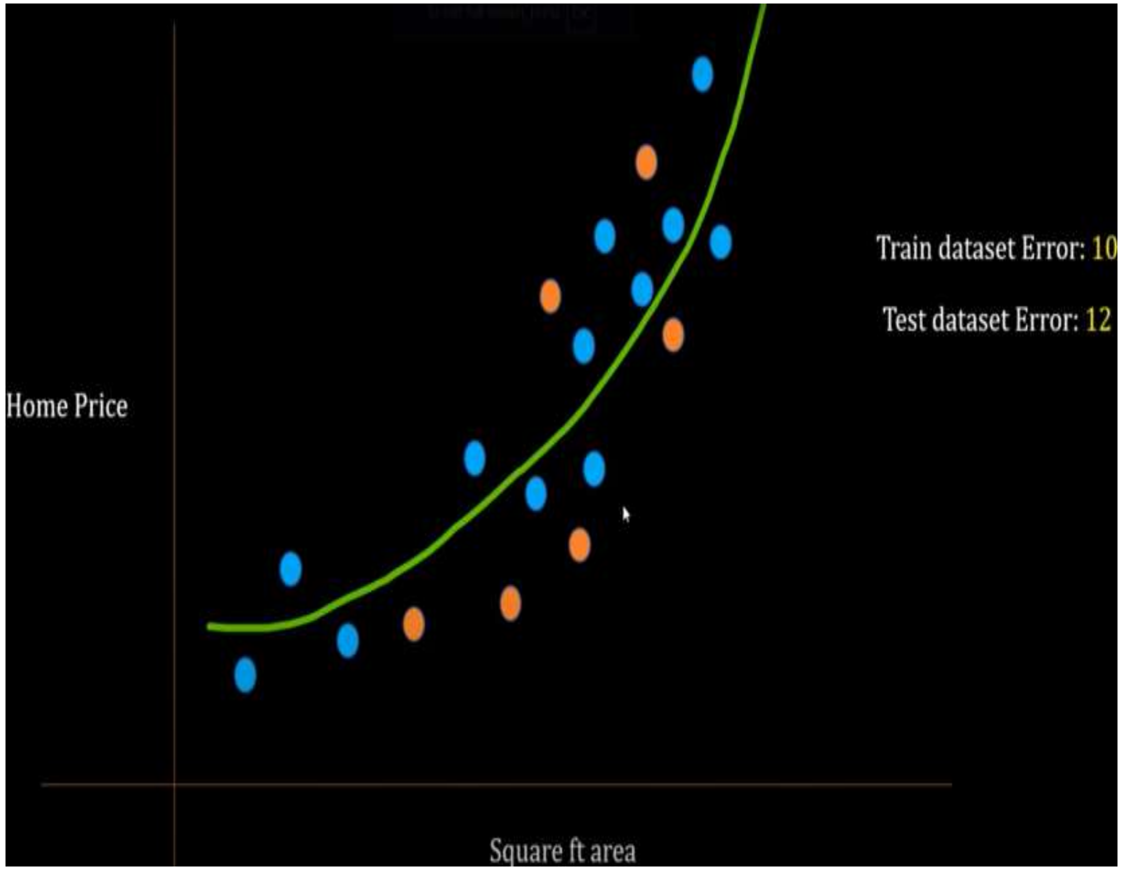
- So, when we talk about Bias, it will represents the training error.
- When we discuss about Variance, it will represents to the test error.

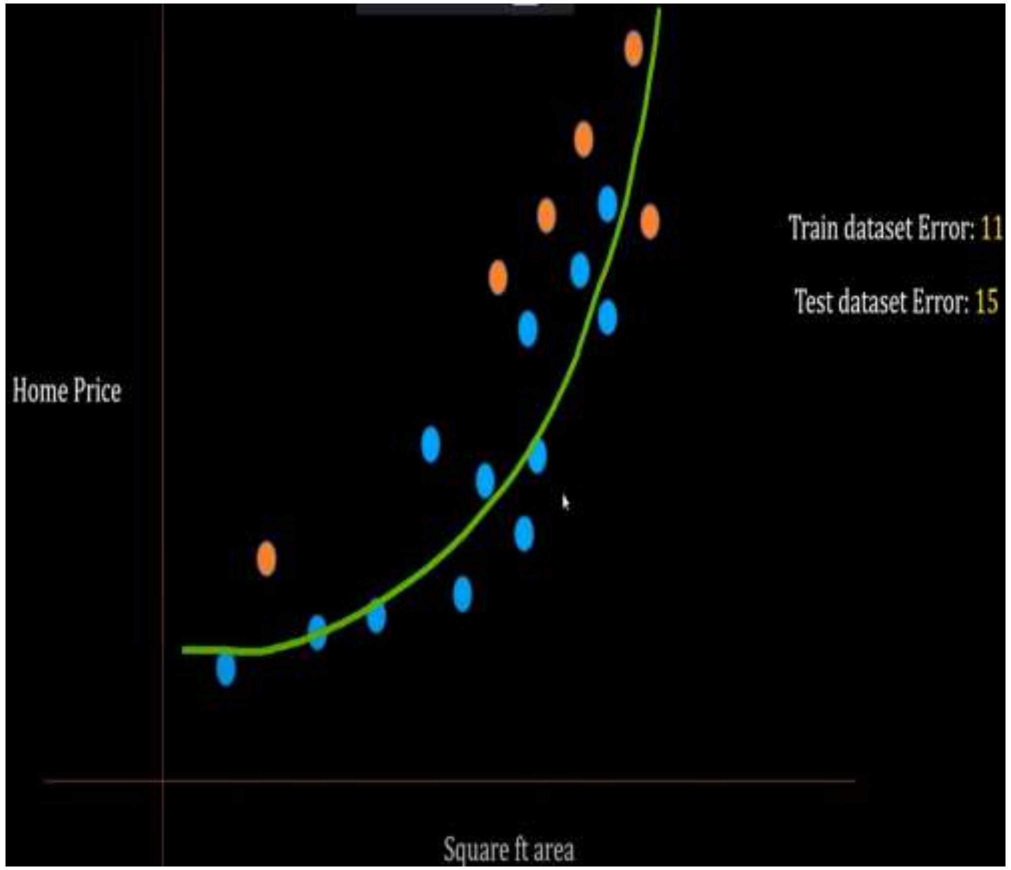
Now we will try to focus on the model which most accurately describe the point of the dataset (Best Fit)



Here our training error and testing error both are low.

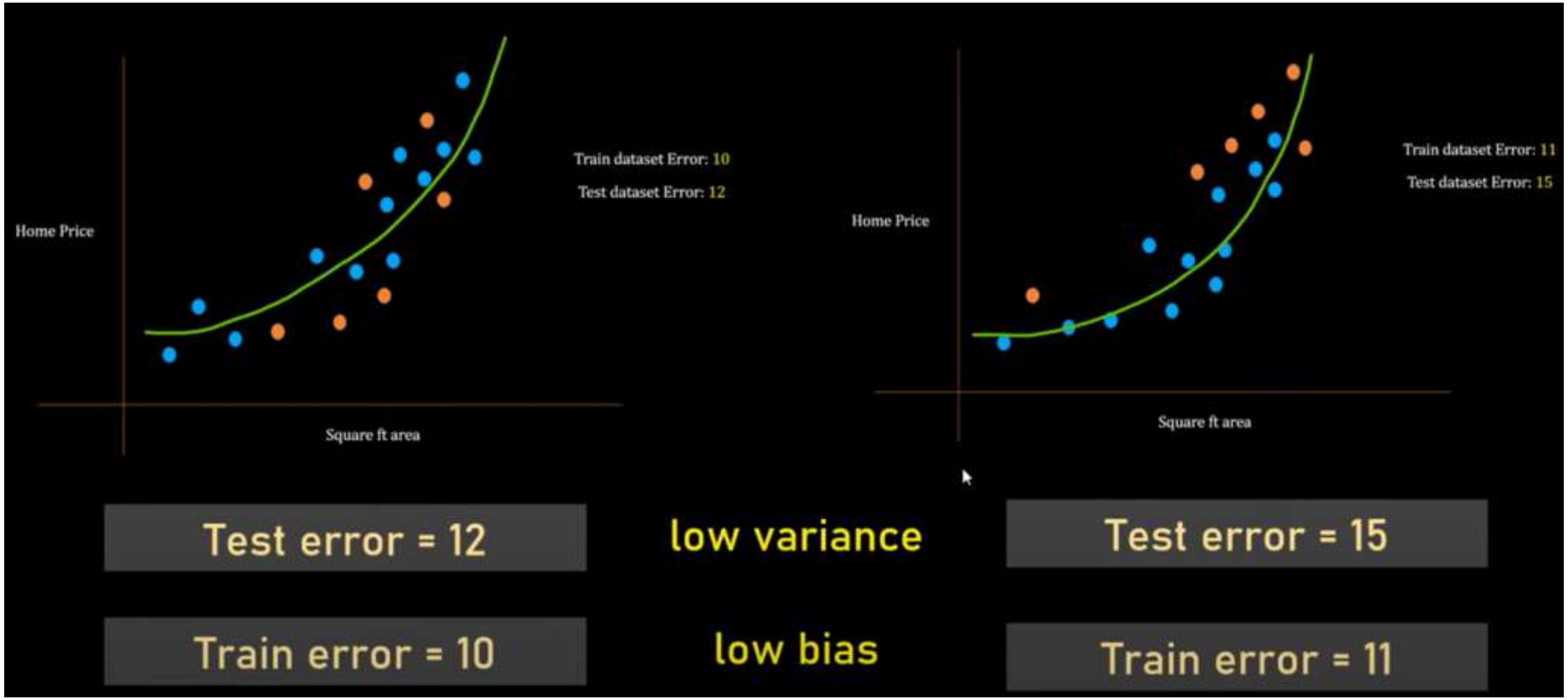
Even if you change the train and test data still the train and test error is low.



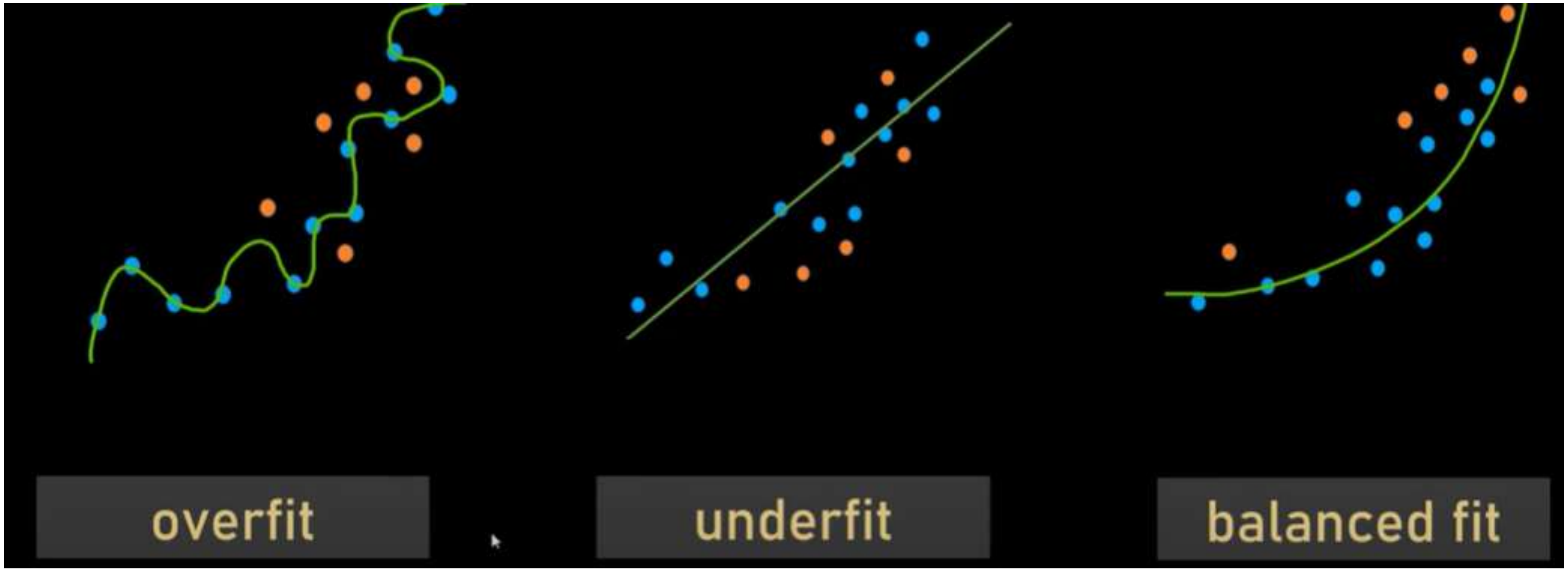


You model selection is such that the train and test error is low, it will be less affected (dependent) on the fact that which samples you are selecting for training and testing purposes.

The best fit case will be known as Low Variance, Low Bias model



- **Low Variance:** Because the **test error is not too much, varying** even if we are changing the train test samples.
- **Low Bias:** Because the **train error is not too much,** even if we are changing the train test samples.
- So, Low Variance and Low Bias situation is there.



Low Bias
High Variance

High Bias
Low Variance

Low Bias
Low Variance

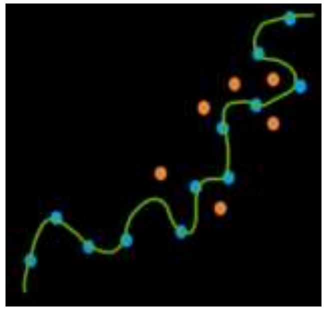
Balance fit model is always a good choice, to get a balance fit model we use techniques like Cross Validation, Regularization, Dimensionality Reduction, Ensemble Techniques (Bagging, Boosting) etc.

Introduction to Ensemble Learning

- In the case of decision of Goa trip we didn’t called a single friend. By calling a single friend a decision could be biased. I wanted to make a good decision. So, I talked to all my 4 closed friends and took a majority vote.
- In Ensemble learning, we train multiple models on the same dataset and then when we do prediction, we consider the results (outputs) of all those models, we combine those results somehow to get our final result (output).
- **Bagging and Boosting are the two main techniques used in ensemble learning.**

Bagging

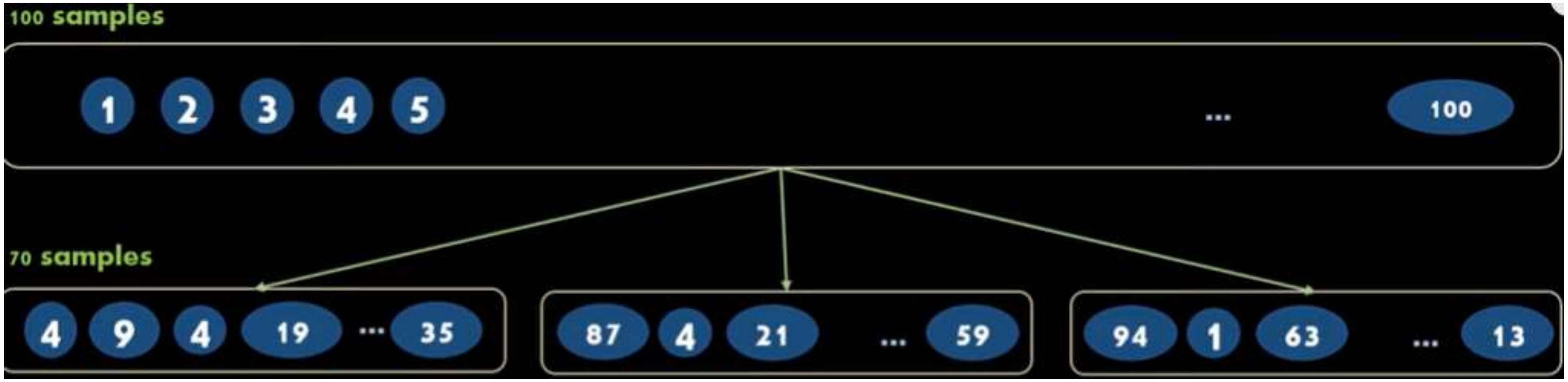
- Suppose, I am having insurance dataset and I want to take decision whether I should buy insurance or not?
- I am having dataset of 100 samples.
- When I train my machine learning model, I may face the problem of overfitting.



- Overfitting happens due to the nature of the dataset. Generally, overfitting leads to high variance problem.

Bagging

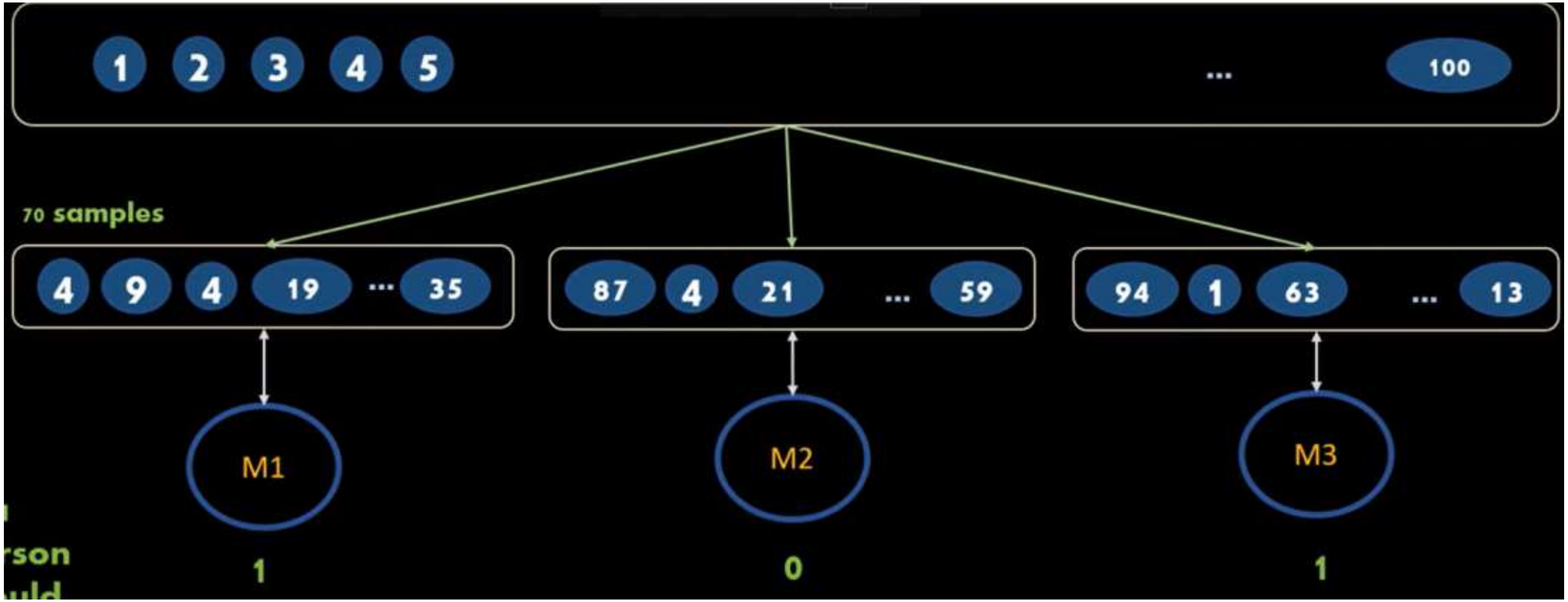
- To deal with this overfitting issue, you can create a multiple small dataset of 70 samples out of 100.
- You need to use '**resampling by replacement**' method to generate this dataset. Means, we will generate many data of 70 samples each like below



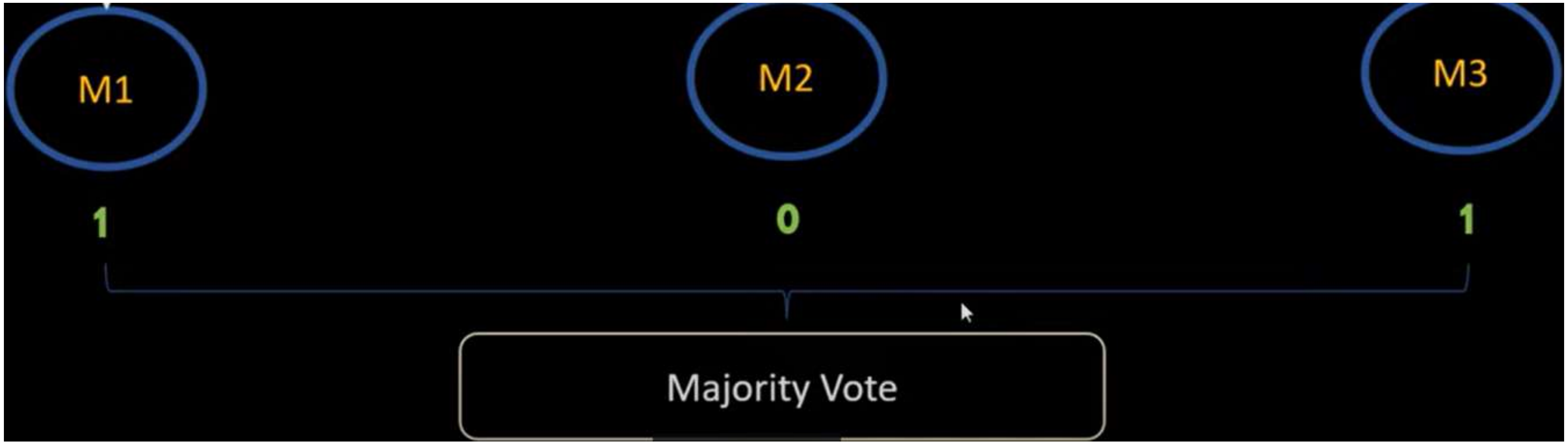
Bagging

- Then on individual dataset of 70 samples, I will train my machine learning model.
- Suppose, I need to check whether a person will buy insurance or not using Logistic Regression.
- In next diagram M1, M2 and M3 are the logistic regression models, we are using to take the decision.
- Each model M1, M2 and M3 is trained on the different dataset.
- Each model may be same (like all logistic or may be different also NB, Decision Tree etc.)
- At the end, you need to perform prediction in parallel on all the three models and finally, you will get the results (Refer Next Slide).

Bagging


1
2
3
4
5
...
100
70 samples
4
9
4
19
...
35
87
4
21
...
59
94
1
63
...
13
M1
M2
M3
1
0
1
rson

Bagging



Here M1 and M3 are in favour of buying the insurance and M2 is not in the favour of buying. So, decision will be taken on the basis of majority. Finally, I will decide to buy the insurance. **The above Classification problem we solved.**

Bagging: Advantage of this technique

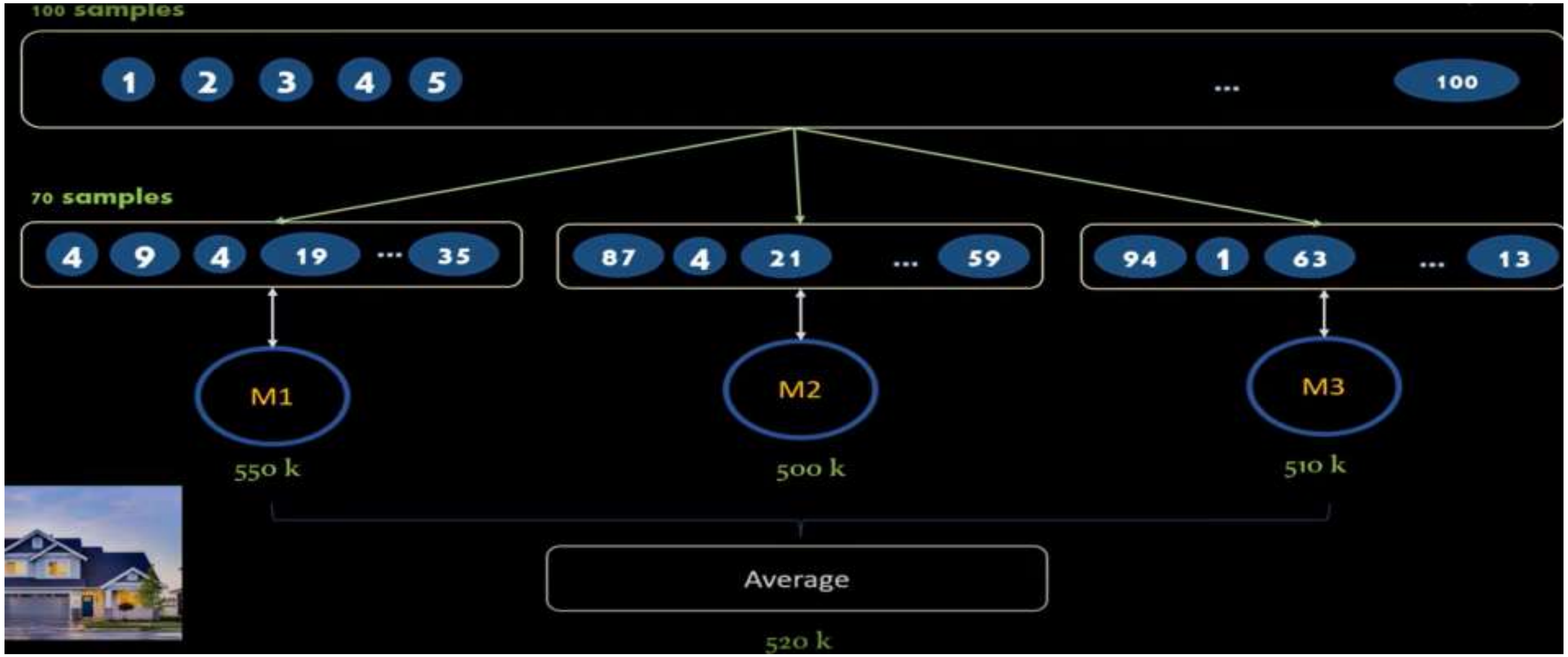
- These individual models (M1, M2 and M3) are considered as the ‘**Weak learners**’, because they are trained on partial dataset (**different subsets of the main dataset)**.
- These subsets might miss some key patterns, making the individual models less accurate on their own.
- However, **by combining multiple weak learners, bagging reduces overfitting.**
- The model’s errors, which vary due to different training data, tend to cancel each other out when averaged or voted upon.
- **This aggregation process lowers the variance of the final model, leading to improved generalization and reduced risk of overfitting compared to any single model.**
- So, they will not stuck in overfit condition.

Bagging: Advantage of this technique

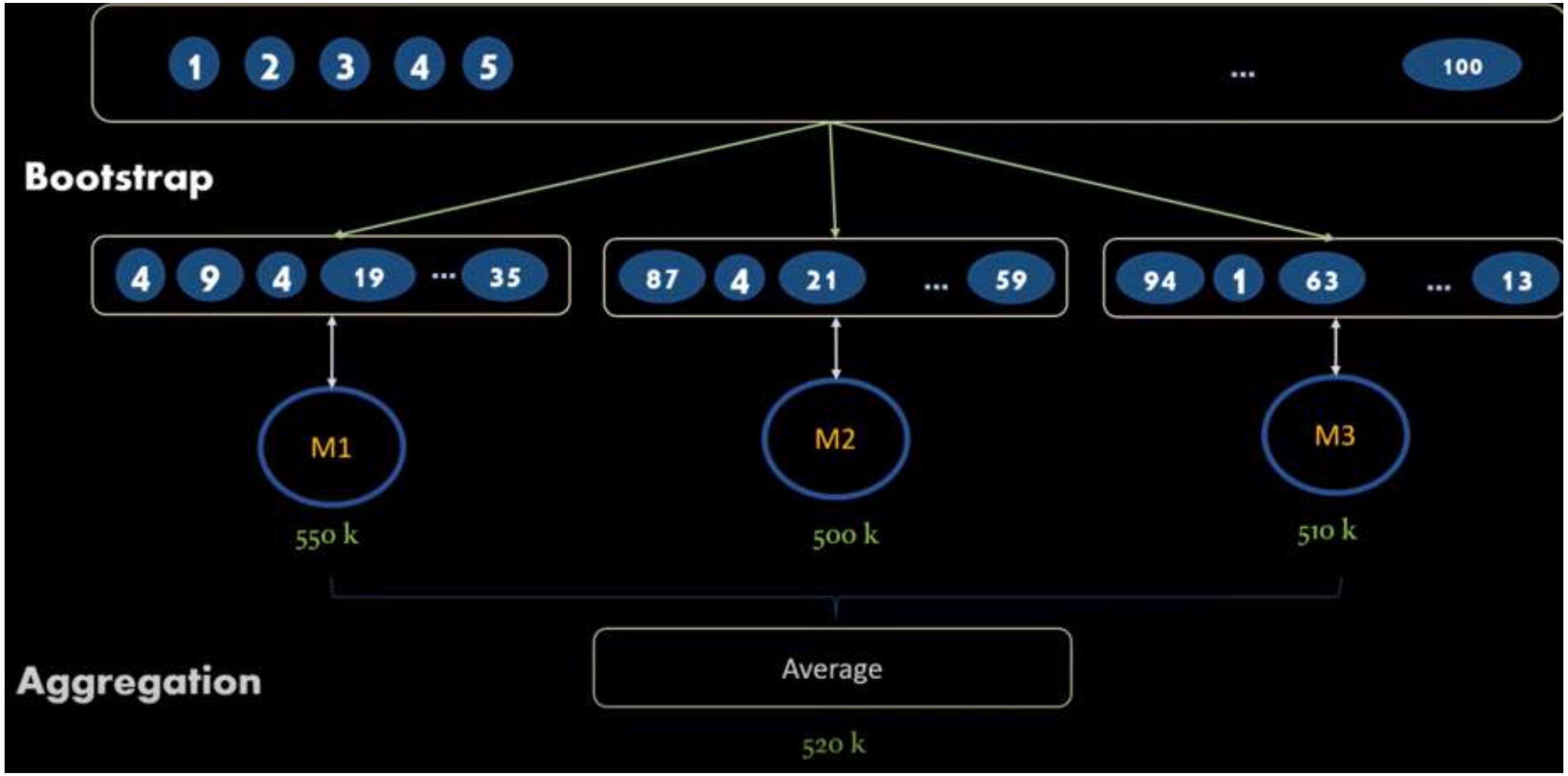
- This technique will help model to being generalize instead of being overfit.
- By combining all the weak learners at the end, we will get a good quality result.

In case of Regression

- If you are asked to predict the price of a house, then we can take an average of price predicted by each individual model, which we considered, like below.



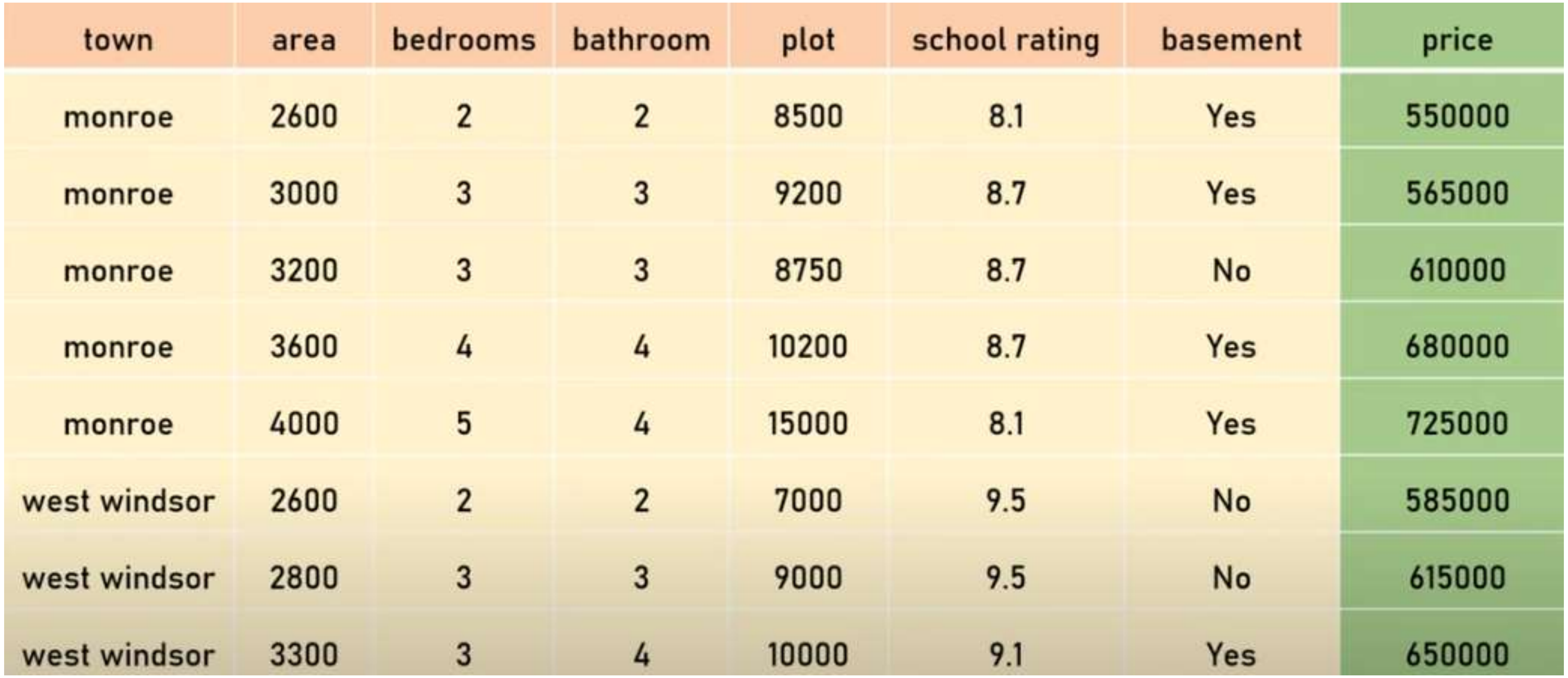
This technique is also called '**Bootstrap Aggregation**'



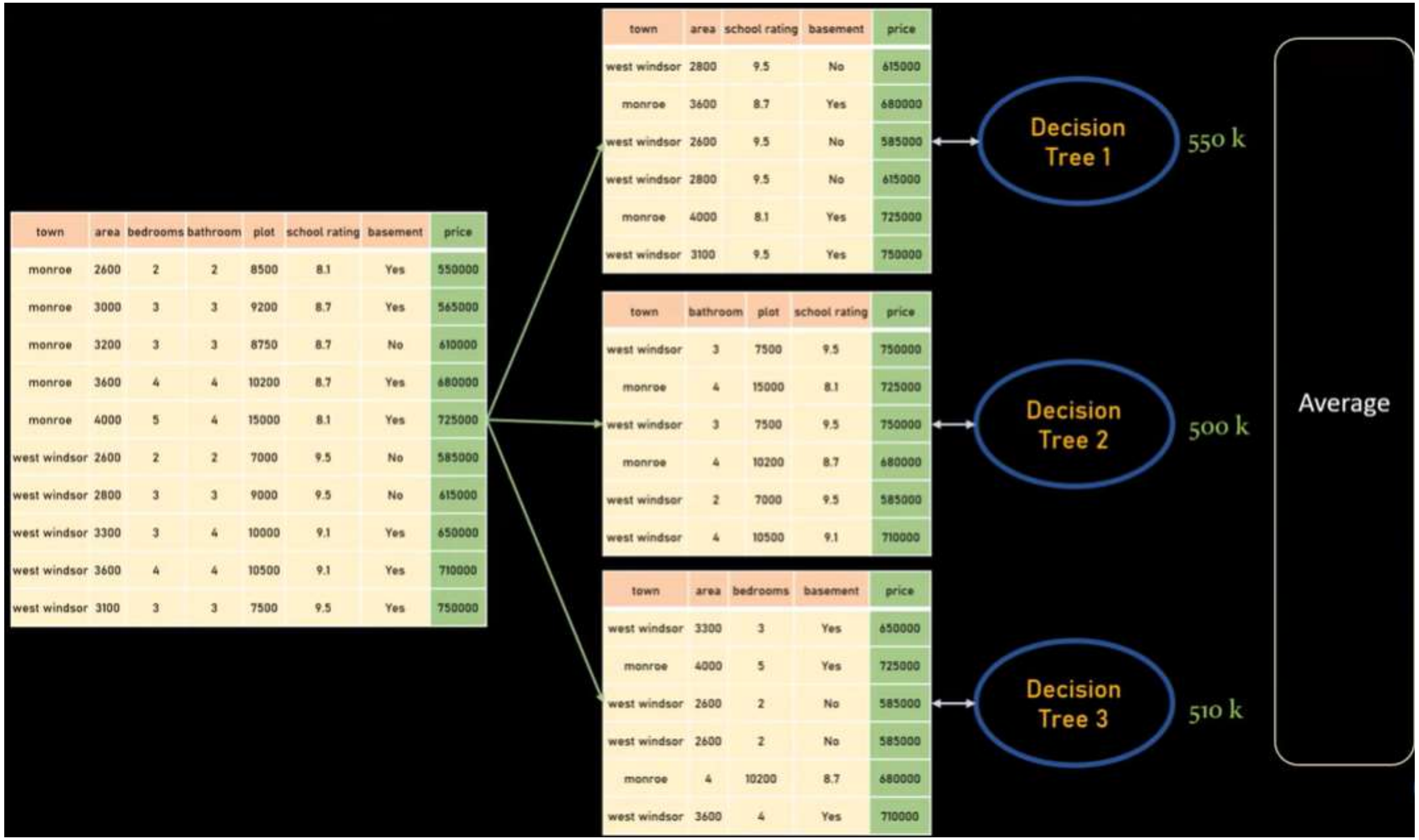
- ‘**Resampling with replacement**’ procedure is called ‘**Bootstrap**’
- When we combine the result either by ‘Majority Voting’ or ‘by taking average’ is called ‘Aggregation’.
- So the bagging is also called ‘BOOTSTRAP AGGREGATION’.

Random Forest

- Random Forest is the example of bagging technique with one difference. In bagging technique, we shuffle (sampled) only the rows in order to generate subsets in the Bootstrapping process.
- But, In Random Forest **Rows and Columns (Features) both are shuffled (sampled),** in order to make the model more robust.
- Let’s take an example of House Price Prediction to understand it more clearly.



town	area	bedrooms	bathroom	plot	school rating	basement	price
monroe	2600	2	2	8500	8.1	Yes	550000
monroe	3000	3	3	9200	8.7	Yes	565000
monroe	3200	3	3	8750	8.7	No	610000
monroe	3600	4	4	10200	8.7	Yes	680000
monroe	4000	5	4	15000	8.1	Yes	725000
west windsor	2600	2	2	7000	9.5	No	585000
west windsor	2800	3	3	9000	9.5	No	615000
west windsor	3300	3	4	10000	9.1	Yes	650000



town	area	bedrooms	bathroom	plot	school rating	basement	price
monroe	2600	2	2	8500	8.1	Yes	550000
monroe	3000	3	3	9200	8.7	Yes	565000
monroe	3200	3	3	8750	8.7	No	610000
monroe	3600	4	4	10200	8.7	Yes	680000
monroe	4000	5	4	15000	8.1	Yes	725000
west windsor	2600	2	2	7000	9.5	No	585000
west windsor	2800	3	3	9000	9.5	No	615000
west windsor	3300	3	4	10000	9.1	Yes	650000
west windsor	3600	4	4	10500	9.1	Yes	710000
west windsor	3100	3	3	7500	9.5	Yes	750000

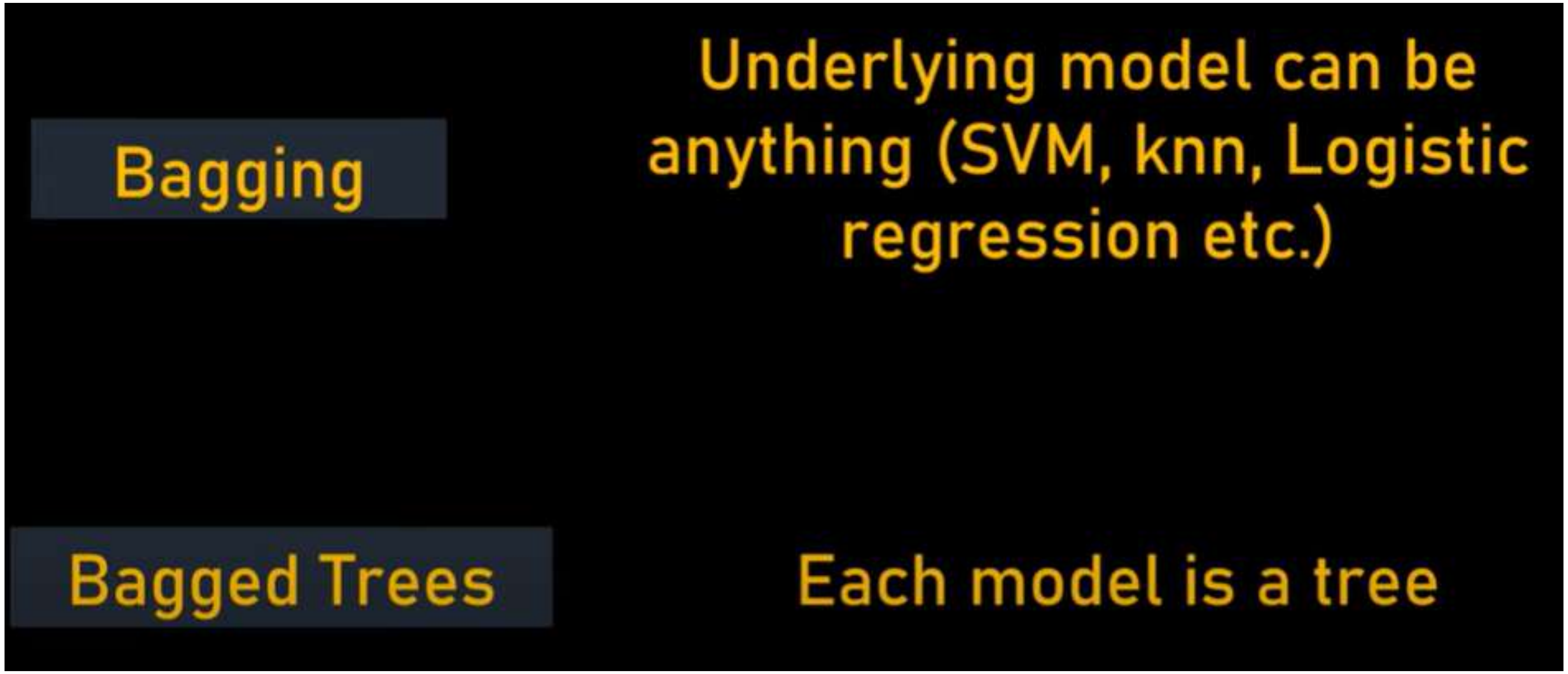
town	area	school rating	basement	price
west windsor	2800	9.5	No	615000
monroe	3600	8.7	Yes	680000
west windsor	2600	9.5	No	585000
west windsor	2800	9.5	No	615000
monroe	4000	8.1	Yes	725000
west windsor	3100	9.5	Yes	750000

town	bathroom	plot	school rating	price
west windsor	3	7500	9.5	750000
monroe	4	15000	8.1	725000
west windsor	3	7500	9.5	750000
monroe	4	10200	8.7	680000
west windsor	2	7000	9.5	585000
west windsor	4	10500	9.1	710000

town	area	bedrooms	basement	price
west windsor	3300	3	Yes	650000
monroe	4000	5	Yes	725000
west windsor	2600	2	No	585000
west windsor	2600	2	No	585000
monroe	4	10200	8.7	680000
west windsor	3600	4	Yes	710000

Average= 520k
Answer

Random Forest



Bagging	Underlying model can be anything (SVM, knn, Logistic regression etc.)
Bagged Trees	Each model is a tree

Random Forest is an Example of Bagged Tree

Implementation

- https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database
- Download this dataset.
- The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.

Boosting Algorithms

- In **bagging (Random Forest),** models are parallelly connected, but in **boosting models are connected in a sequence**.
- In the Random Forest. all models are having equal share in the decision making.
- In the boosting, each model is treated as weak learner because each model does some wrong predictions. **Each model forwards its wrongly predicted data to the next model in its training phase**.
- Each next model will be dependent on its previous model for its input.
- But combining all the models in a sequence provides them a strength, so overall model becomes a strong learner.
- Example : AdaBoost, GradientBoost etc.

AdaBoost

- AdaBoost is known as ‘**Adaptive Boosting**’
- This is also a sequential type of boosting technique.
- In AdaBoost all the models participating in the chain are **not having equal sharing/importance**, as we were having in the bagging. Some weak learners are having higher and some have the lower priority.
- In **Random Forest each model is treated as fully grown decision tree**.
- In AdaBoost trees are not fully grown, they are having just one root node and two child nodes. These trees are called ‘**Stumps**’.

How AdaBoost works? Example

Age	BMI	Gender
25	24	F
41	31	F
56	28	M
78	26	F
62	30	M

Dataset is given as above. Here Gender is the target variable. On the basis of Age and BMI we need to predict Gender. So at first we need to assign initial weights (IW) to these rows. We are assigning equal weight to all tuples.

Age	BMI	Gender	IW
25	24	F	1/5
41	31	F	1/5
56	28	M	1/5
78	26	F	1/5
62	30	M	1/5

- After initializing the weights. The first stump is created from the whole dataset. Stump is generated on the basis of Gini Index/Entropy etc.
- Once the stump is created the data is tested for the accuracy.
- When on this training data, testing is performed on this data and predictions are made

Age	BMI	Gender	IW	Predictions	
25	24	F	1/5	Correct	
41	31	F	1/5	Correct	
56	28	M	1/5	**Wrong**	Misclassified in first iteration
78	26	F	1/5	Correct	
62	30	M	1/5	Correct	

- In the first iteration the 4 predictions made are correct and one is incorrect.

- **Now, for the next model in chain, the weight for the uncorrected predicted tuple is increased. For correctly predicted tuples weight is reduced.**
- In the next model/iteration more importance (weightage) is given to previously incorrectly classified record, so that next model will try to classify it correctly. Model gives more focus to correctly classify the previously misclassified tuple.

Age	BMI	Gender	IW	Predictions	Next Model Weight
25	24	F	1/5	Correct	Decrease
41	31	F	1/5	Correct	Decrease
56	28	M	1/5	**Wrong**	**Increase**
78	26	F	1/5	Correct	Decrease
62	30	M	1/5	Correct	Decrease

The final model will gives us the output, which reflects all the goodness from the chain and deliver at the end of the sequence.

- This approach works well on larger scale of number of tuples are in thousands/lacs.

Another example

- Predicting if a Fruit is an Apple or Orange
- **1. Dataset**
- Imagine we have a dataset with three features:
- **Color** (Red, Orange)
- **Size** (Small, Large)
- **Weight** (Light, Heavy)

And we want to predict if a fruit is an Apple (A) or an Orange (O).

Dataset

Color	Size	Weight	Label
Red	Small	Light	A
Orange	Large	Heavy	O
Red	Large	Heavy	A
Orange	Small	Light	O
Red	Small	Heavy	A
Orange	Large	Light	O

- **2. Initial Setup**

- **Initial Weights**: Each data point starts with an equal weight. For example, if we have 6 samples, each sample could start with a weight of 1/6≈0.1673**. Training Weak Classifiers**
- Let's assume we use a simple decision stump (a tree with one split) as our weak classifier. AdaBoost will iteratively train several of these classifiers, each focusing on different aspects of the data.

Iteration 1:

- Classifier 1: The first decision stump might focus on the **Color feature**.
- If **Color = Red**, predict Apple (A).
- If **Color = Orange**, predict Orange (O).
- **Accuracy:** Let's say this classifier gets 4 out of 6 correct (all Red are classified correctly, but 2 Orange samples are misclassified).
- **Update Weights:** Increase the weights of the misclassified samples, so that the next classifier pays more attention to them.

Iteration 2:

Classifier 2:

- The next decision stump might focus on the Size feature.
- If **Size = Small, predict Apple (A).**
- If **Size = Large, predict Orange (O).**
- **Accuracy: This classifier might get 5 out of 6 correct**.
- **Update Weights: Again, increase the weights of the misclassified samples.**

Iteration 3:

- Classifier 3: The final decision stump might focus on the Weight feature.
- If **Weight = Light**, predict Orange (O).
- If **Weight = Heavy**, predict Apple (A).
- **Accuracy:** This classifier might get 6 out of 6 correct.
- **Update Weights:** At this point, the model might be perfect, or very close to it.

4. Combining Classifiers

- AdaBoost combines these weak classifiers into a strong classifier by giving more weight to the classifiers that performed better. The final prediction is based on a weighted vote of all the classifiers.

5. Testing the Model

- Once the model is trained, we test it on a separate test dataset (which wasn't used during training). This step evaluates the model's ability to generalize to new, unseen data.

6. Outcome Prediction: For a new fruit, the AdaBoost model will consider the predictions of all three weak classifiers and combine them to make the final decision

—whether the fruit is an Apple or an Orange.

Summary:

In this example, AdaBoost iteratively trains weak classifiers, focusing on misclassified samples each time.

It then combines these classifiers into a strong model.

The training data is used to build the model, and separate test data is used to evaluate it. This ensures that the model is both accurate and generalizable.

Implementation of AdaBoost Algorithm

Gradient Boost

Difference between AdaBoost and Gradient Boost

In AdaBoost weights of records are updated. Misclassified samples are given more weightage to the next model in the sequence.

In Gradient Boost also multiple models are combined but learning happens by optimizing the loss function.

In AdaBoost, trees are creates in form of **‘Stumps’**. Trees are not fully grown.

In Gradient Boost, leaf node ranges from 8 to 32. Trees are bigger in size in Gradient Boost.

In AdaBoost learning (boosting) happens by adjusting the weights.

In Gradient Boost learning (boosting) happens by optimizing the loss.

- **Gradient Boost works in same way to optimize its loss like the concept of Gradient Descent to minimize the error in the Linear Regression.**

XGBoost Algorithm

- e**X**treme Gradient Boost
- It is a framework that can support multiple languages like C++, R, Python, Julia etc.
- It is platform free (independent). Mostly it could be seamlessly imported/deployed to many platforms on Windows/Linux/Mac OS etc.
- It is integrable with cloud computing platform.
- XGBoost is popular due to two main things, **its processing speed is very fast and the results generated through this algorithm are of good quality.**

XGBoost Algorithm

- It is implemented on the top of Gradient Boost.
- Unlikely to AdaBoost, it directly attacks on residuals (errors) an tries to minimize them after many iterations.

XGBoost Algorithm

- XGBoost (**eXtreme Gradient Boosting**) is an advanced machine learning algorithm based on **boosting**. It is widely used for classification and regression tasks because it is **fast, efficient, and accurate**.
- **Why XGBoost?**
- Faster and more optimized than traditional boosting methods.
- Handles missing values and large datasets well.
- Prevents overfitting using **regularization**.
- Supports parallel processing.

XGBoost Algorithm

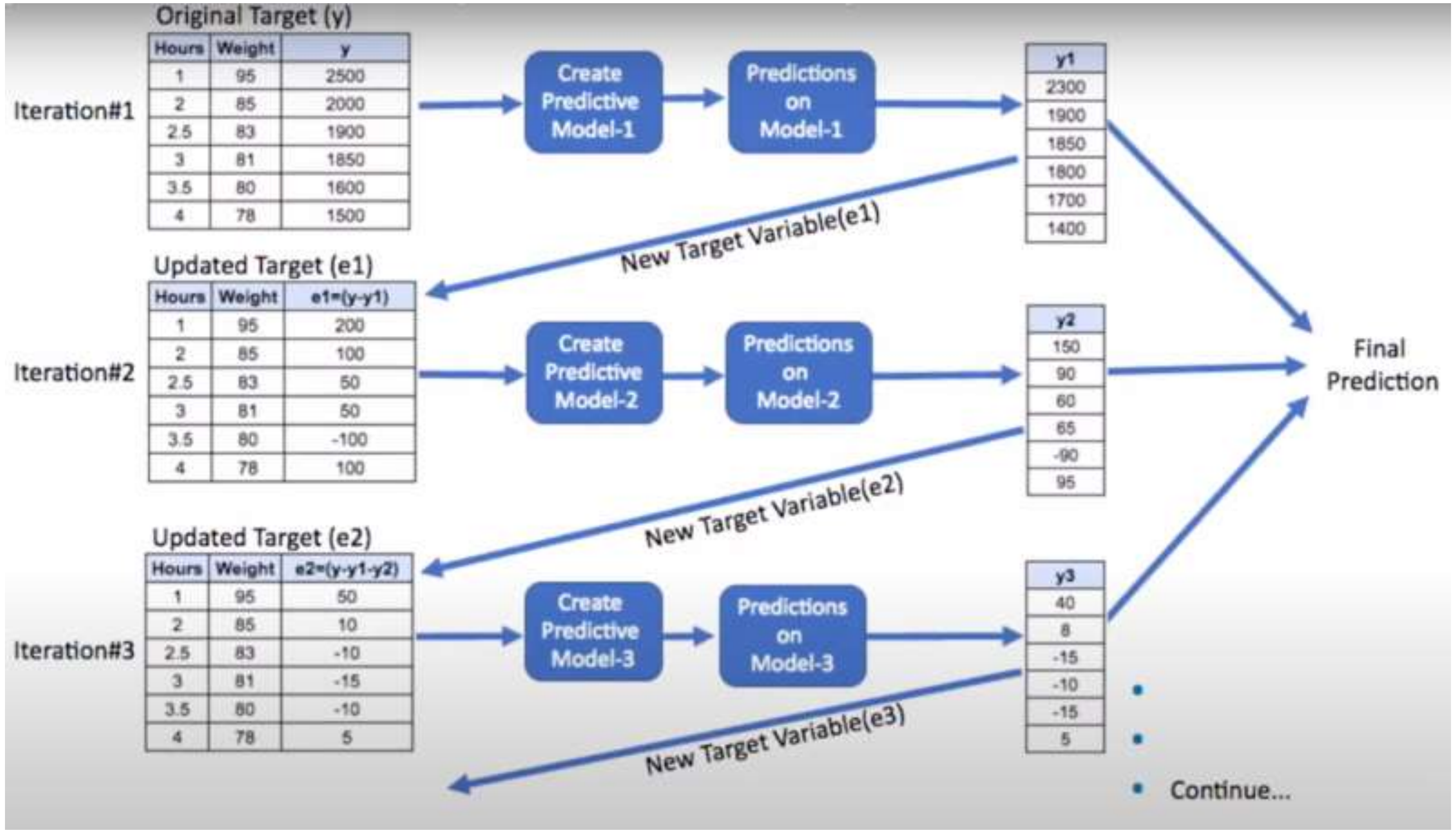
1. Start with a weak model (like a simple decision tree).
2. Calculate errors in predictions.
3. Train a new model to fix those errors (boosting technique).
4. Repeat multiple times, improving step by step.
5. Combine all models to make a strong final prediction.

XGBoost Algorithm

- Imagine you are predicting student grades based on study hours and attendance.

1. **First Tree**: Makes an initial guess (e.g., all students score 60%).
2. **Find Errors**: Some students scored higher or lower.
3. **Second Tree**: Focuses on students with wrong predictions.
4. **Repeat**: Keep adding trees to correct errors.
5. **Final Prediction**: All trees work together for accurate results.

XGBoost Flow



Original Target (y)

Hours	Weight	y
1	95	2500
2	85	2000
2.5	83	1900
3	81	1850
3.5	80	1600
4	78	1500

y1
2300
1900
1850
1800
1700
1400

Updated Target (e1)

Hours	Weight	e1=(y-y1)
1	95	200
2	85	100
2.5	83	50
3	81	50
3.5	80	-100
4	78	100

y2
150
90
60
65
-90
95

Updated Target (e2)

Hours	Weight	e2=(y-y1-y2)
1	95	50
2	85	10
2.5	83	-10
3	81	-15
3.5	80	-10
4	78	5

y3
40
8
-15
-10
-15
5

Final Prediction will be y1+y2+y3. XGBoost is known as additive model.

XGBoost

- **Key Takeaways**

✓ XGBoost is an **optimized boosting algorithm**
✓ It **learns from mistakes** by focusing on errors
✓ Faster and **more efficient than traditional methods**