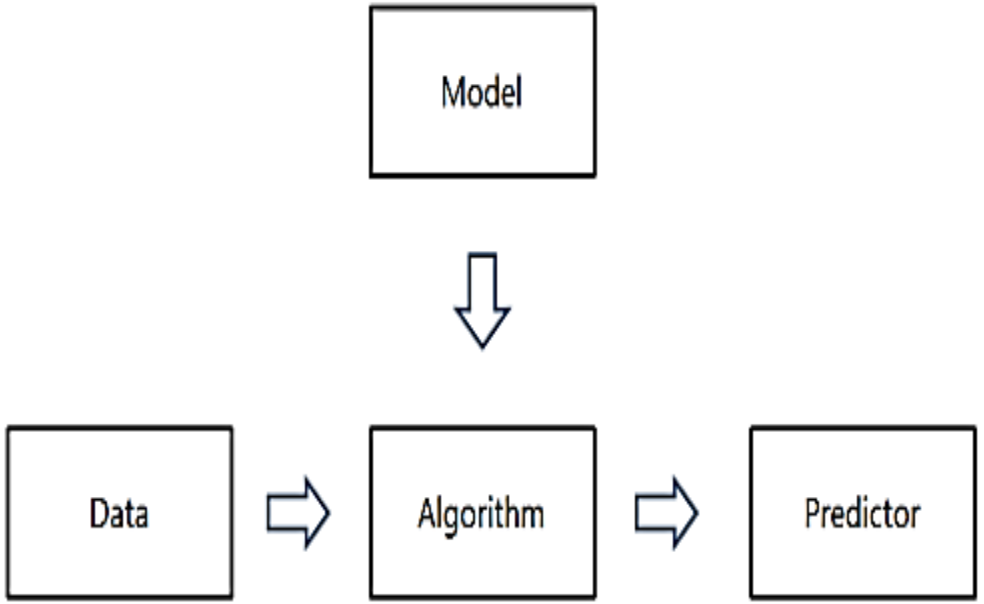
Machine Learning Framework



Model selection

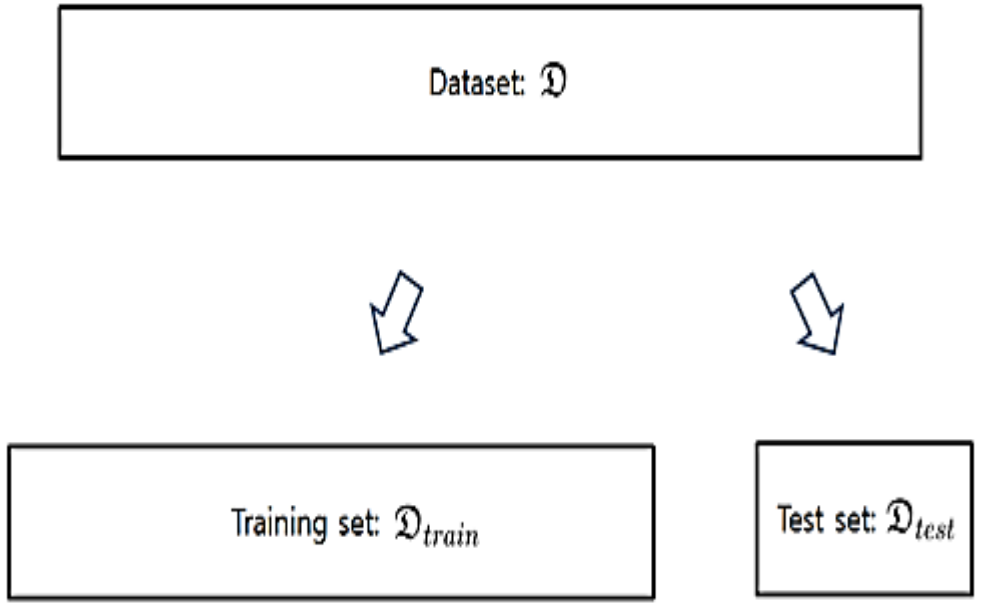
- We do not want our model to say“That’sBob! I remember him! He has dementia!” The reason why is simple.
- When we deploy the model in the future, we will encounter patterns that the model has never seen before.
- Our predictions will only be useful if our model has truly discovered a general pattern.
- We access just a small sample of data. The largest public image data sets contain roughly one million images.

Model selection…

- More often, we must learn from only thousands or tens of thousands of data examples.
- In a large hospital system, we might access hundreds of thousands of medical records.
- When working with finite samples, we run the risk that we might discover apparent associations that turn out not to hold up when we collect more data.

Testing and Training the Dataset

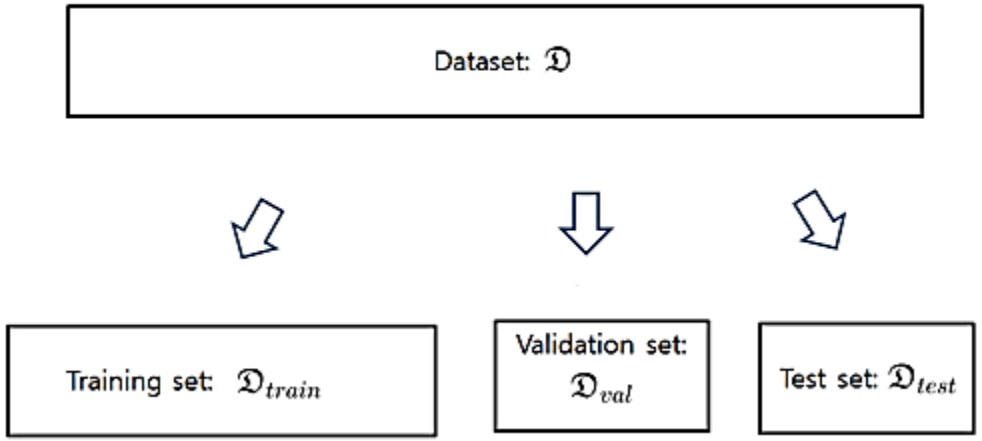
A good predictor must satisfy dual criteria. It has to perform well for the given dataset D, but it also has to work well for similar future data.



Validation and model selection

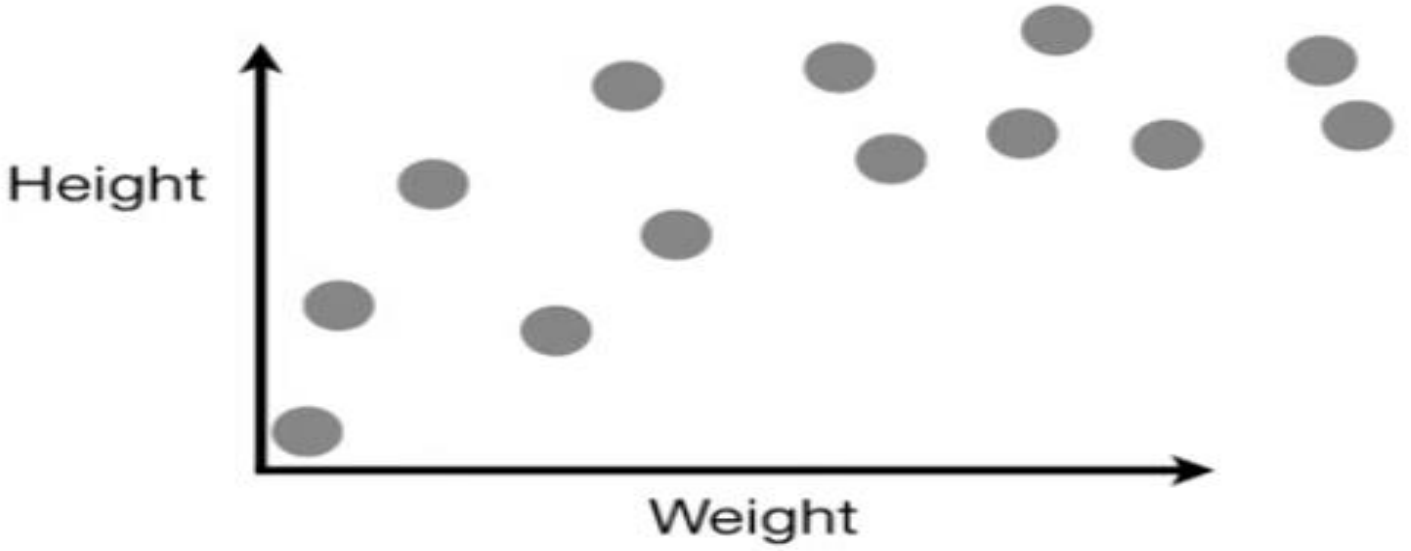
The above simple scheme of training and testing is good for estimating the performance of a predictor.

when there are several models to choose from, we need a different scheme.

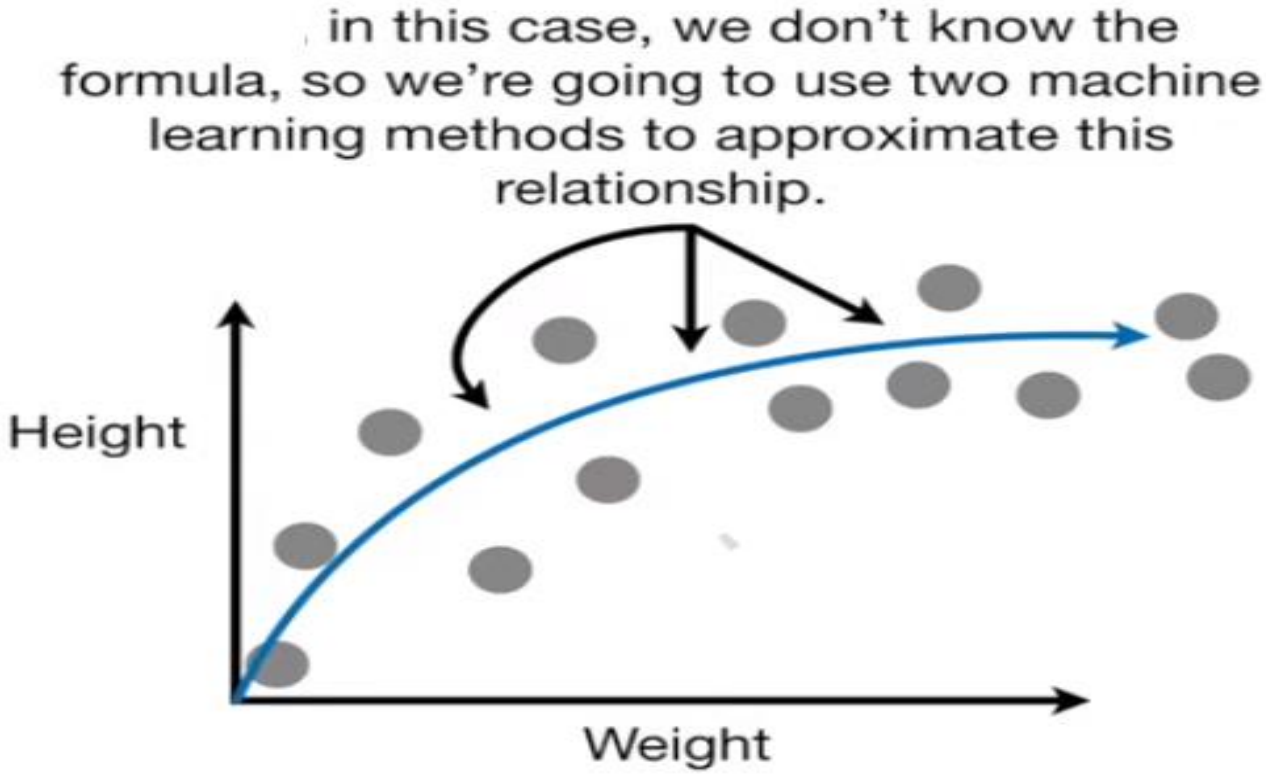


How to select the Model (Bias and Variance)

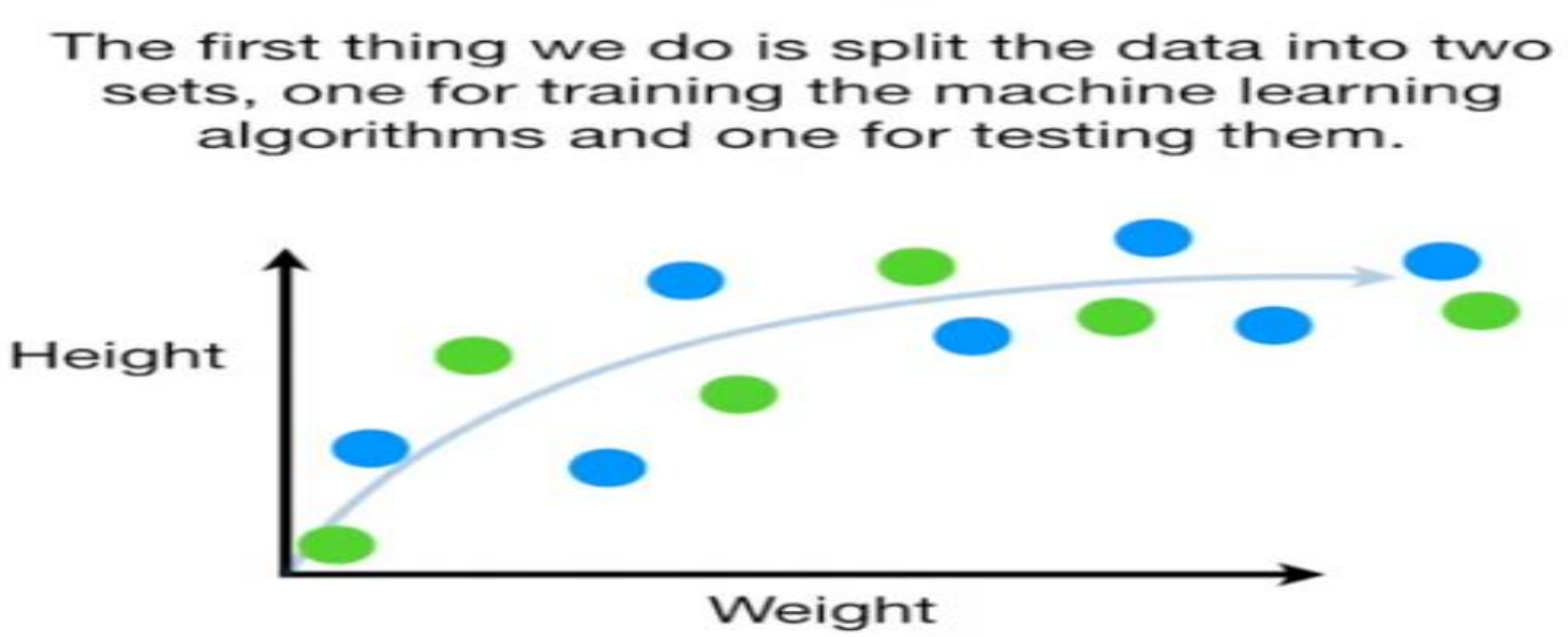
Imagine we measured the weight and height of a bunch of mice and plotted the data on a graph...



Bias and Variance..

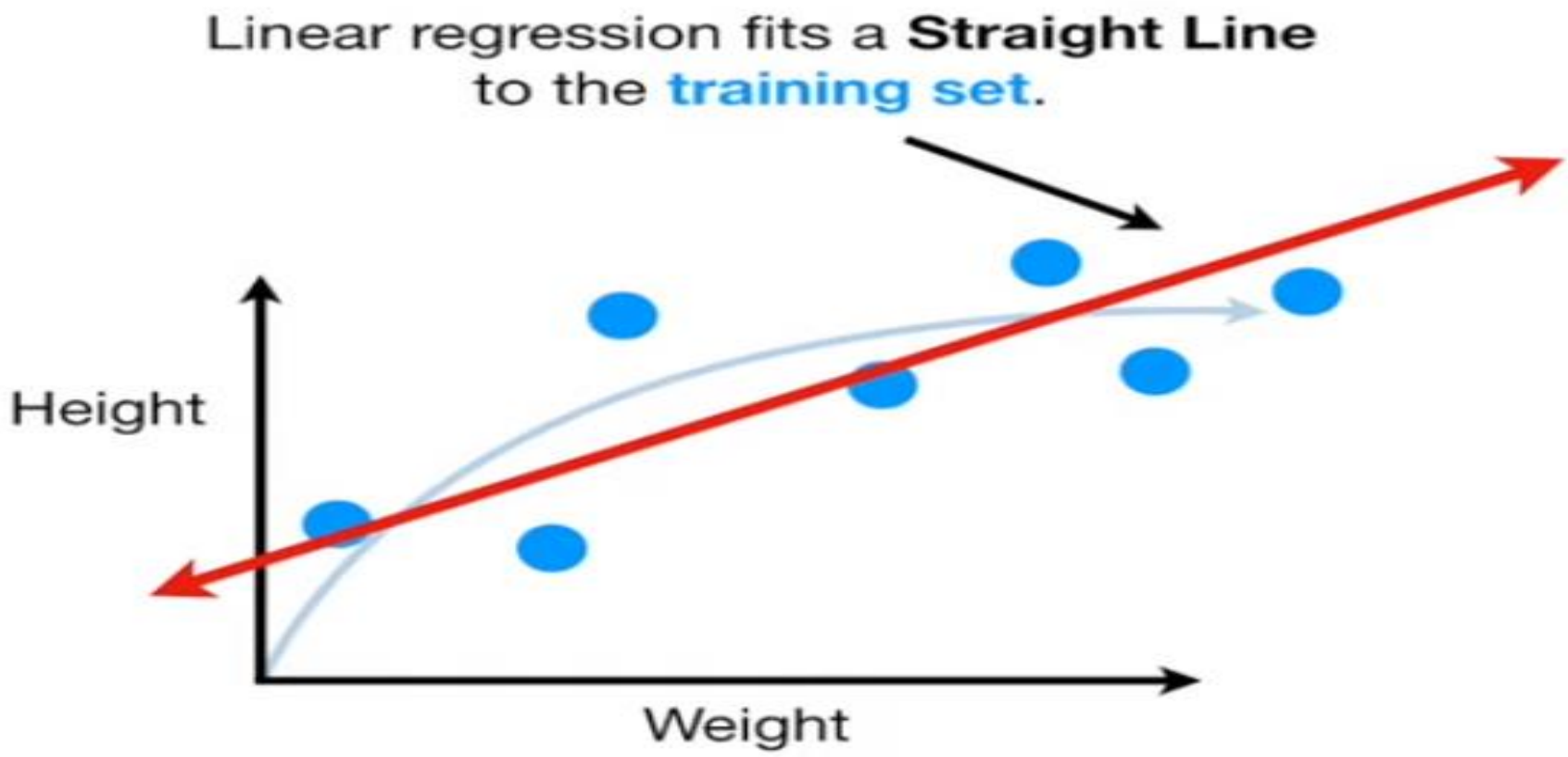


Bias and Variance..

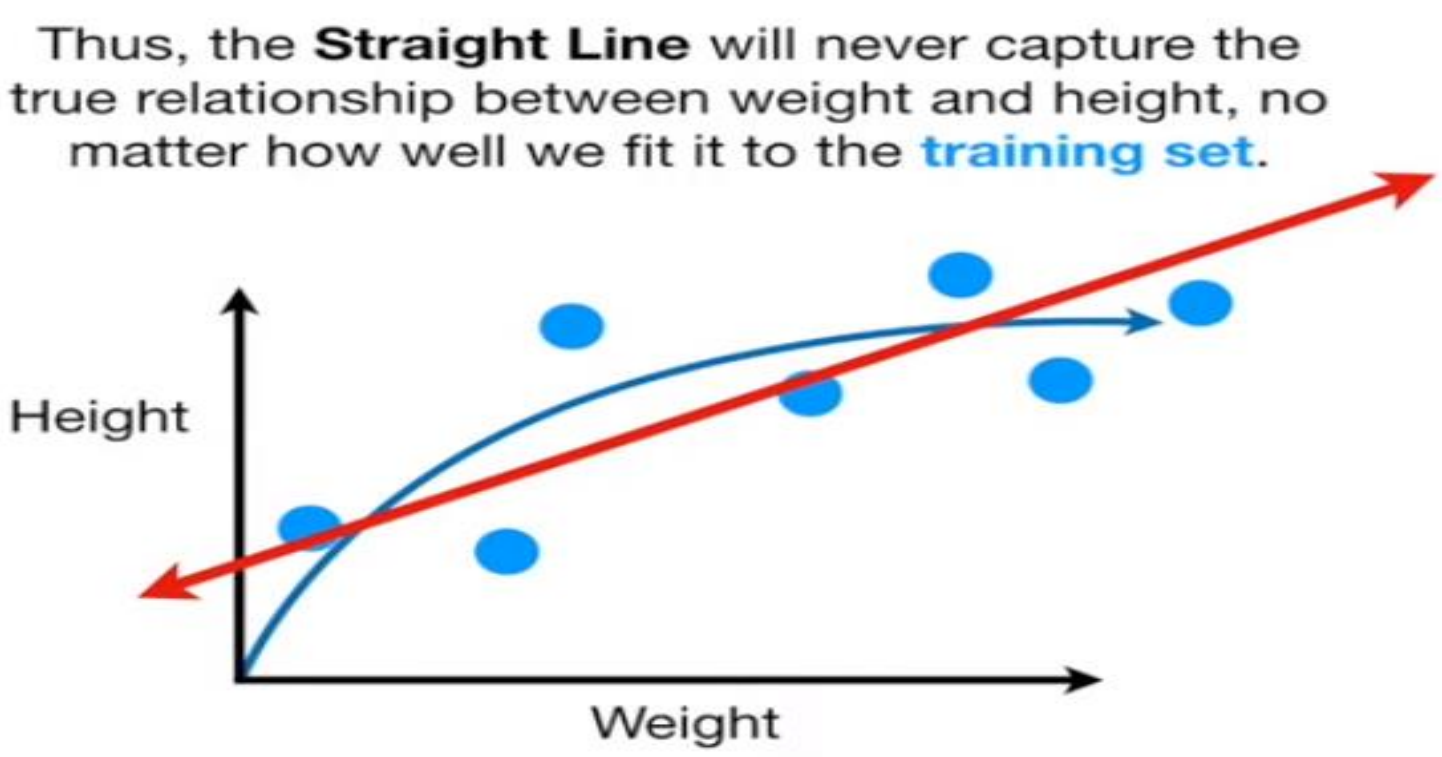


Blue for training and Green for tesing

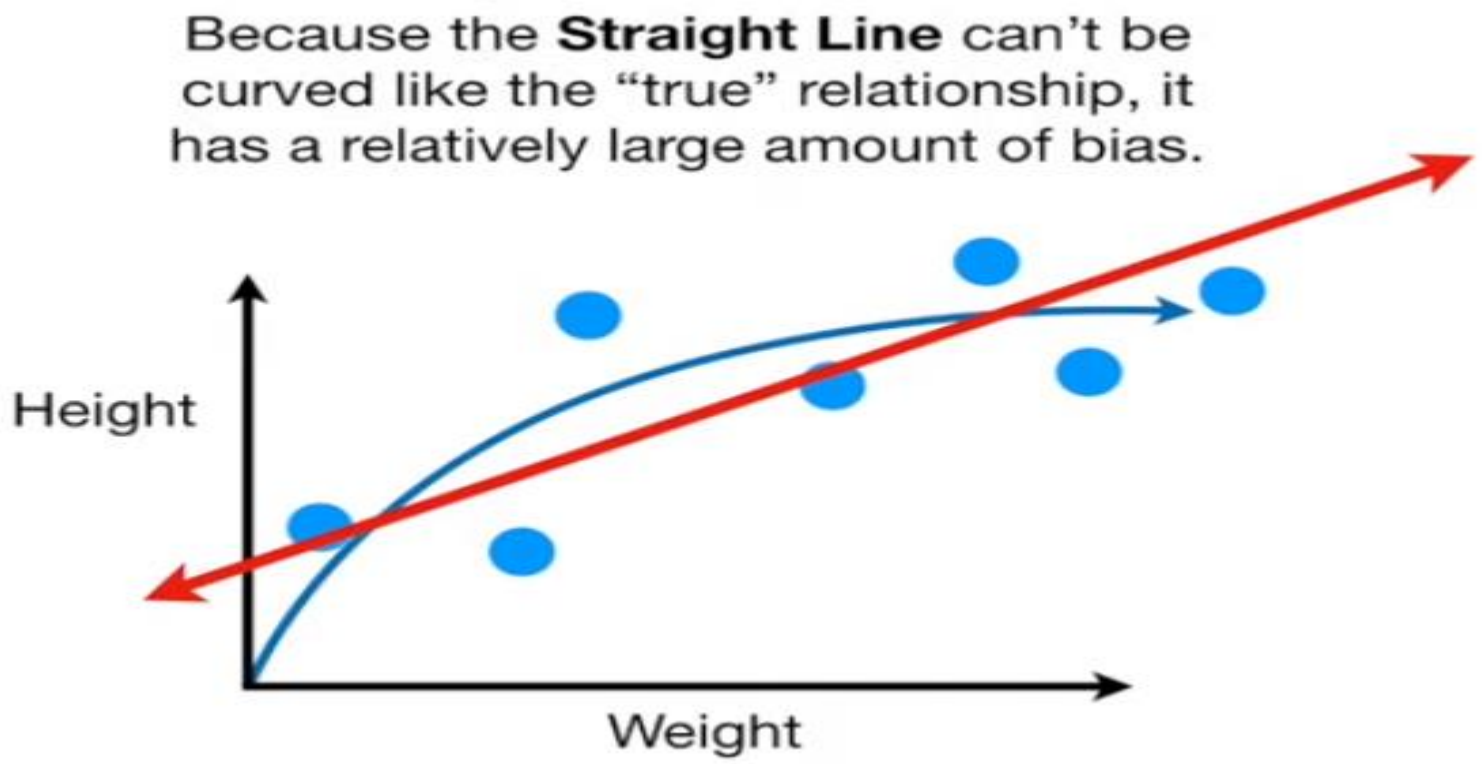
Bias and Variance..



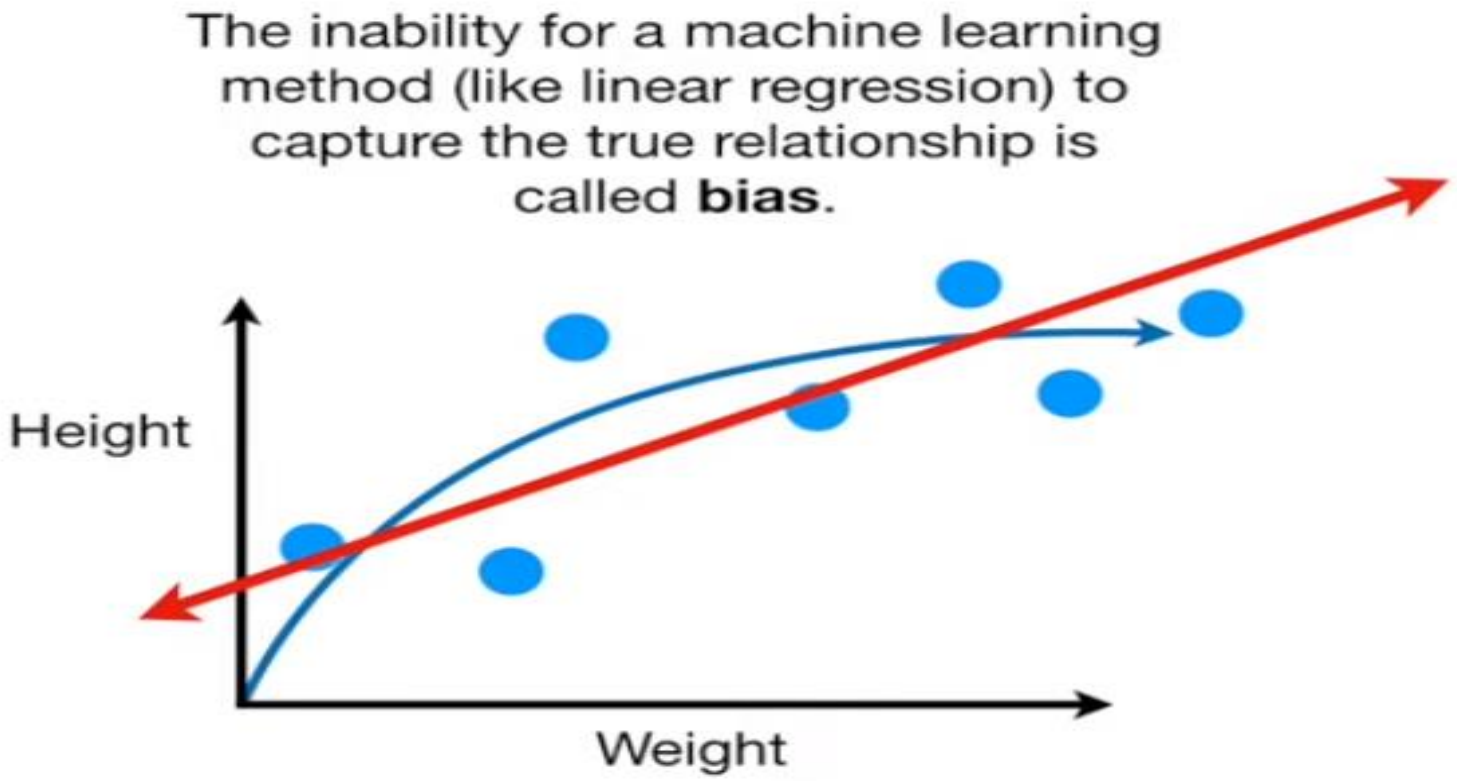
Bias and Variance..



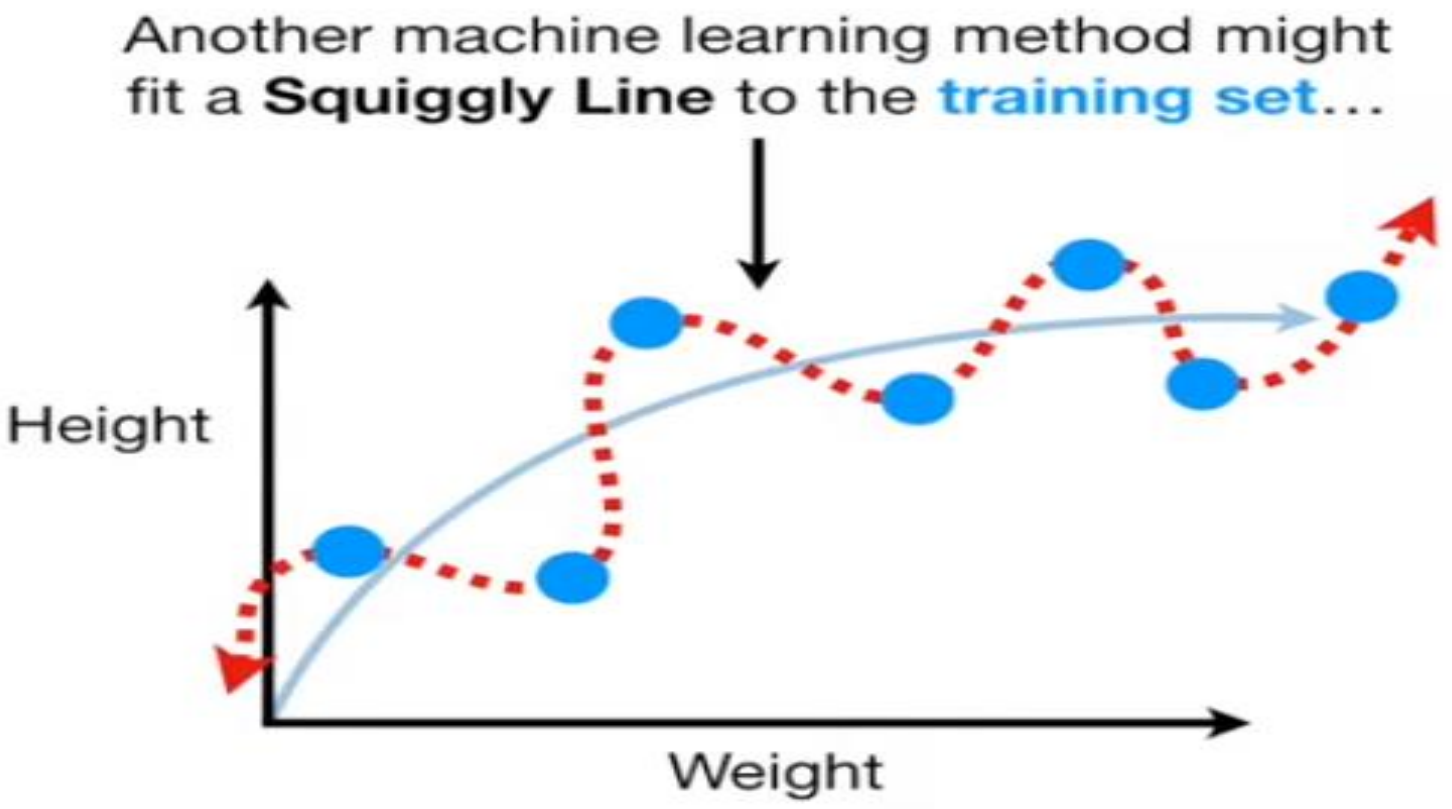
Bias and Variance..



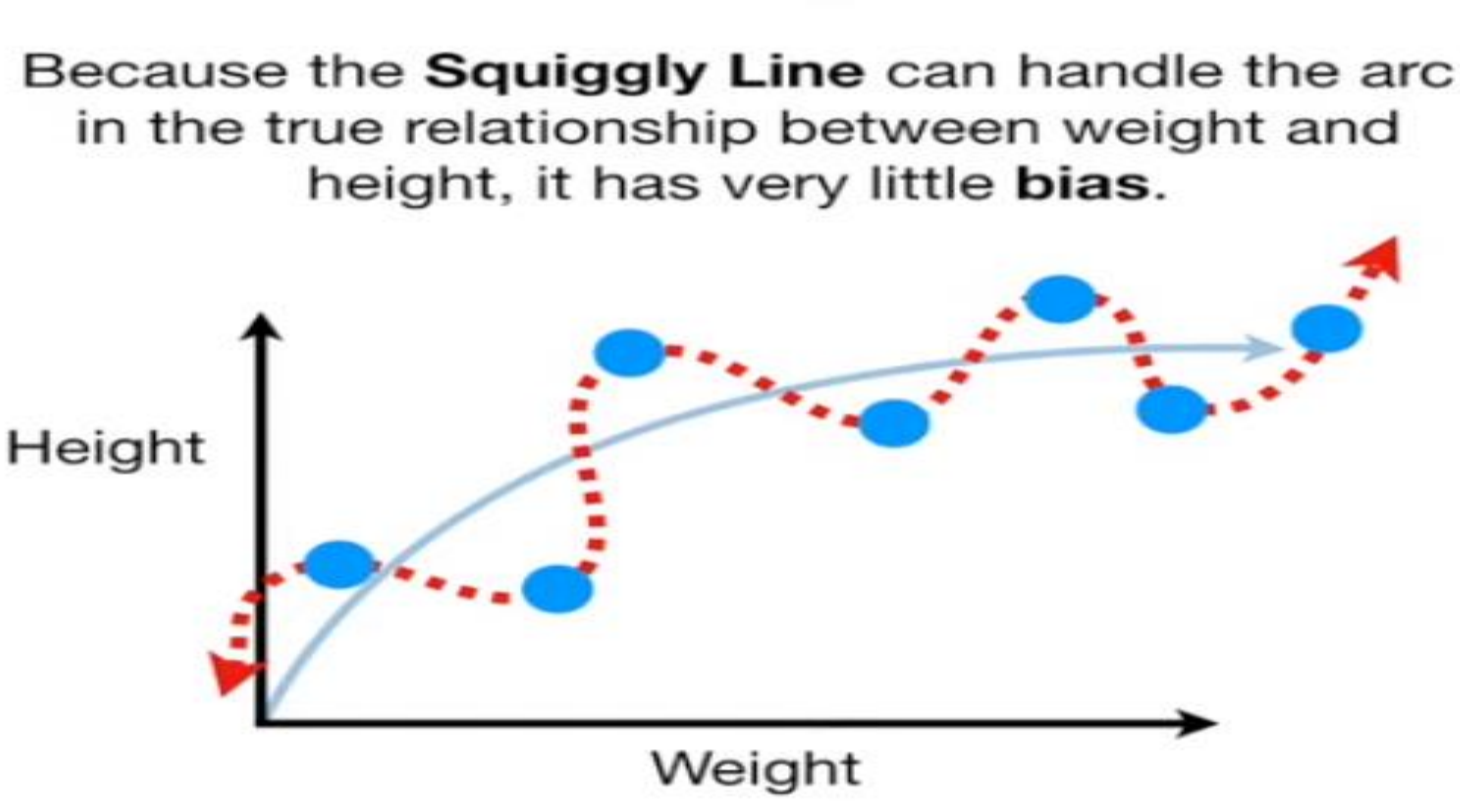
Bias and Variance..



Bias and Variance..

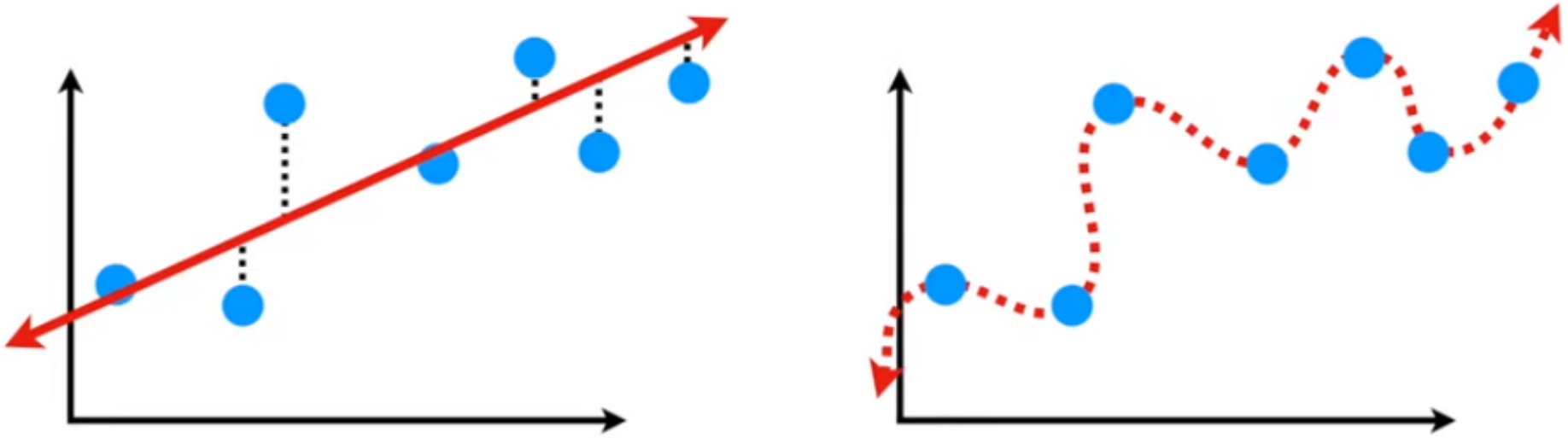


Bias and Variance..

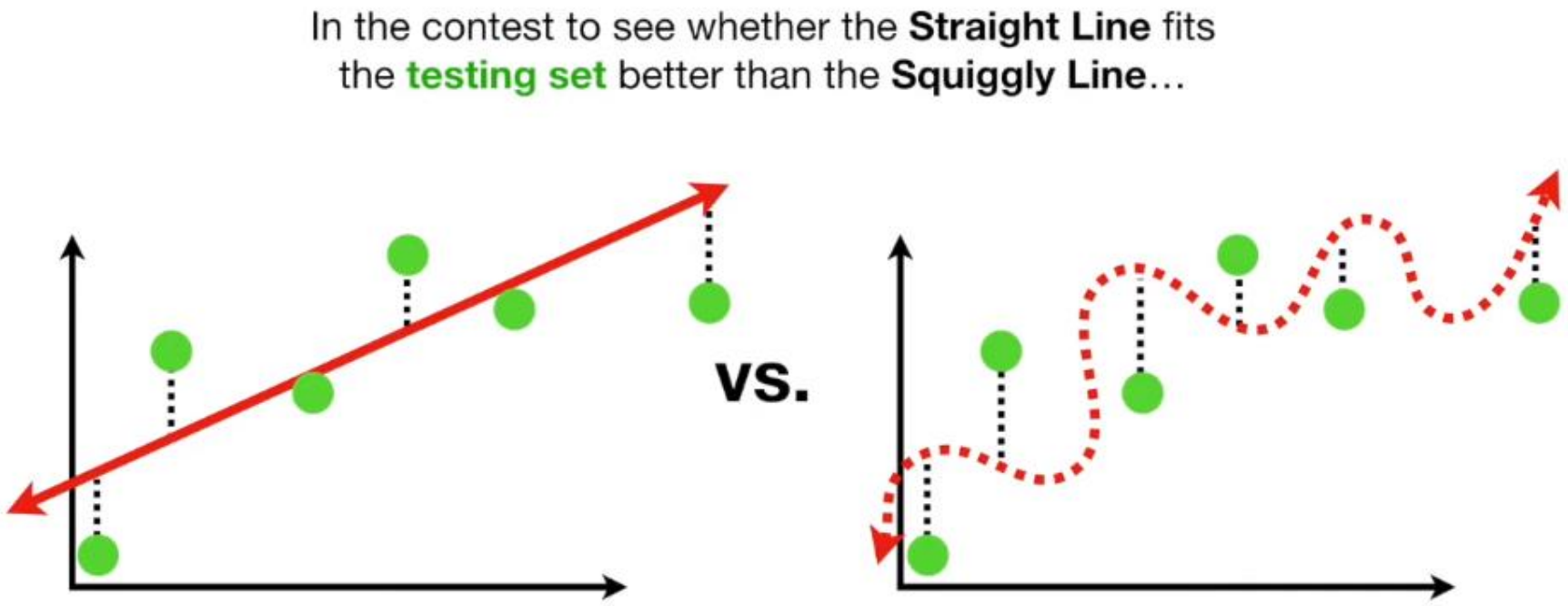


Bias and Variance..

We can compare how well the **Straight Line** and the **Squiggly Line** fit the **training set** by calculating their sums of squares.

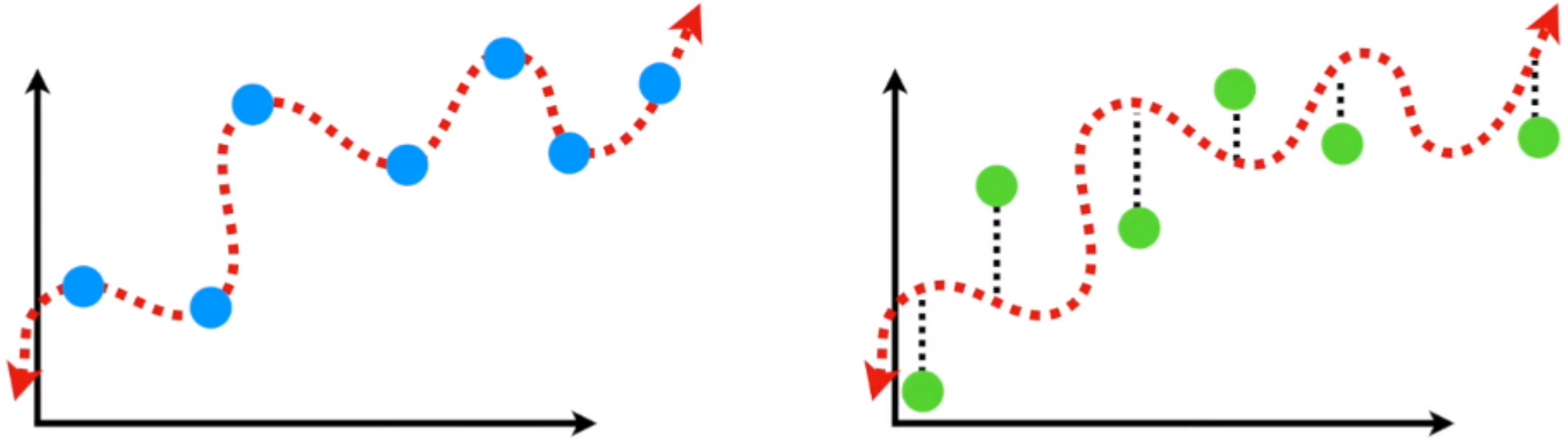


Bias and Variance..



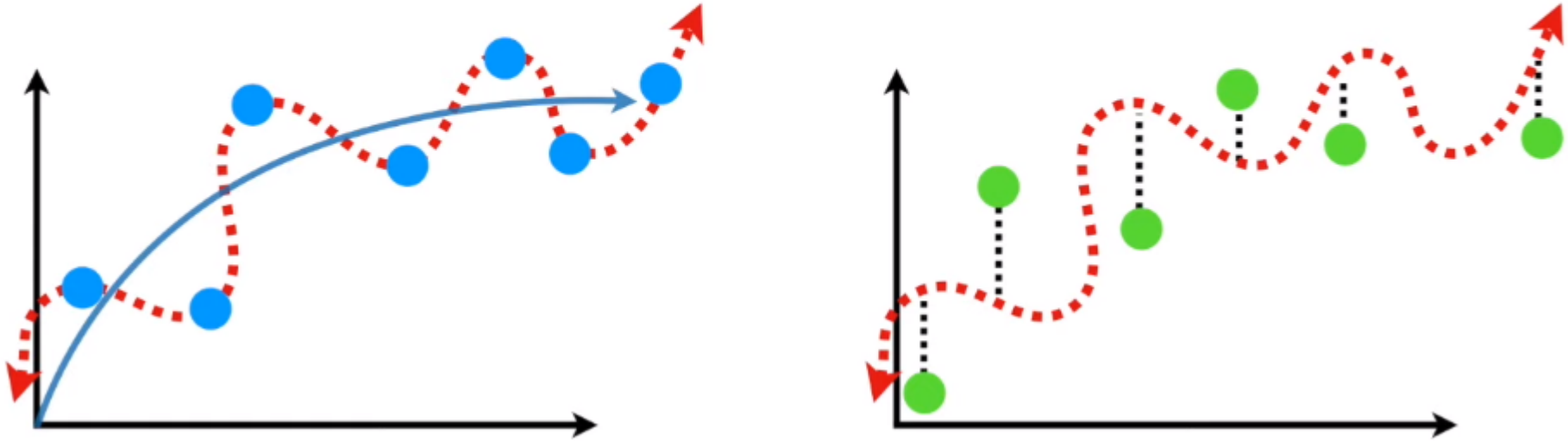
Bias and Variance..

In Machine Learning
difference in fits between data sets is
called **Variance**.

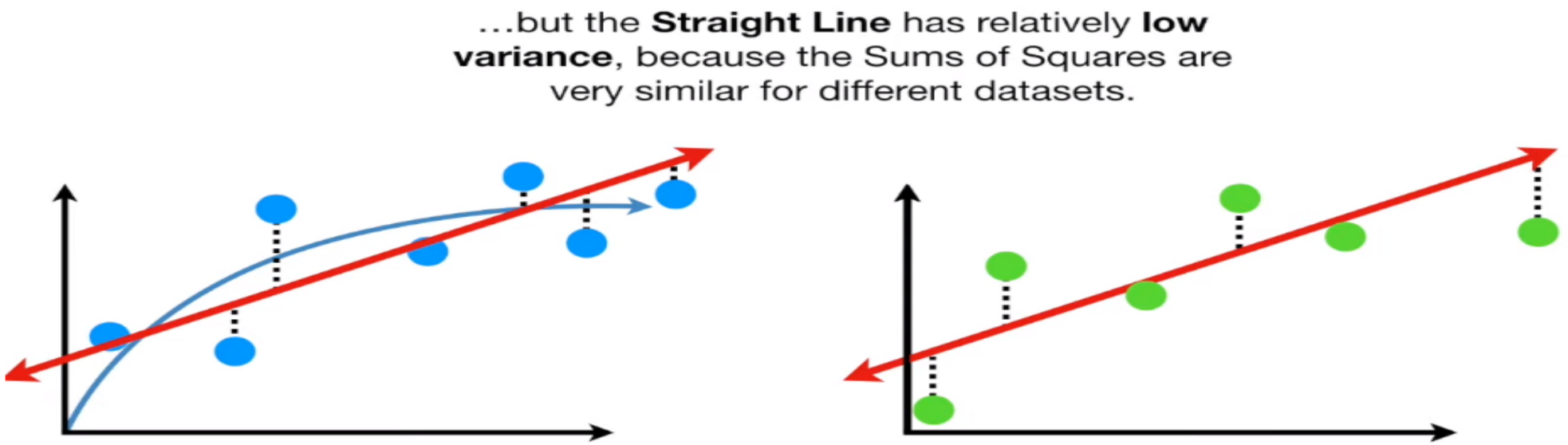


Bias and Variance..

it's hard to predict how well the **Squiggly Line** will perform with future data sets. It might do well sometimes, and other times it might do terribly.

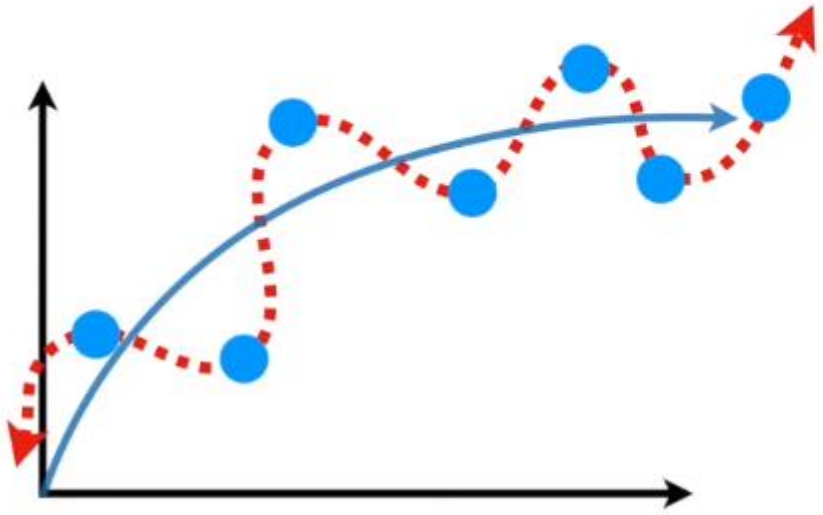


Bias and Variance..

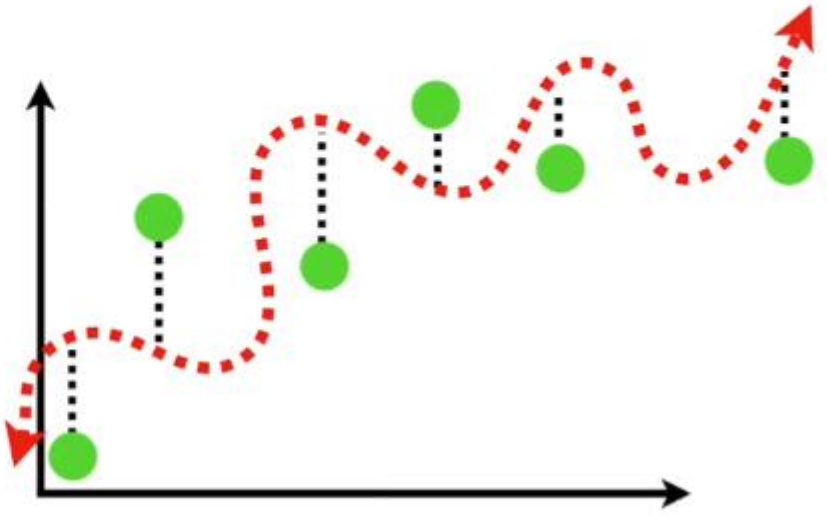


So Straight line or linear Regression is the best fit for the mice data setfor prediction the height for the given weight

Bias and Variance..

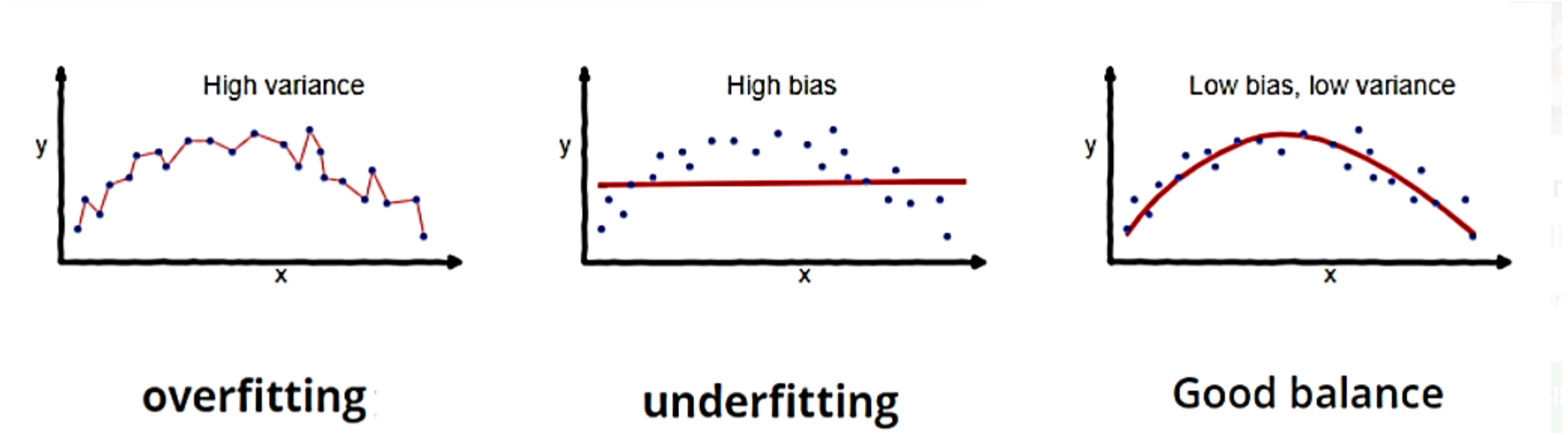


Because the **Squiggly Line** fits the **training set** really well, but not the **testing set**, we say that the **Squiggly Line** is **overfit**.



Bias/Variance Trade -off

There are two sources of error, namely Bias and Variance, which acts as a hindrance for any algorithm to generalise.



Loss Function

Loss functions quantify how close a given neural network is to the ideal toward which it is training.

Steps to calculate the Loss:

- Calculate a metric based on the error we observe in the network's predictions.
- Aggregate these errors over the entire dataset and average them and compare the value with the ideal case.
- Adjust the parameters(Weights) to reduce the Loss due to error.

Loss Function Notation….

The loss function notation indicates that its value depends only on

W and b, the weights and the biases of the neural network.

Types of Loss Functions:

- Squared loss
- Logistic loss
- Hinge loss
- Negative log likelihood

Loss Function Notation

- Consider the dataset gathered to train a neural net. Let "*N*" denote the number of samples (set of inputs with corresponding outcomes) that have been gathered.

- Consider the nature of the input and output collected. Each data point records some set of unique input features and output features. Let "*P*" denote the number of input features gathered and "*M*" denote the number of output features that have been observed.

Loss Function Notation….

- We will use (X,Y) to denote the input and output data we collected. Note that there will be N such pairs where the input is a collection of P values and the output Y is a collection of M values. We will denote the *ith* pair in the dataset as X_i and Y_i.

- We will use $\hat{Y}$ to denote the output of the neural net. Of course, $\hat{Y}$ is the network's guess at Y and therefore it will also have M features.

Loss Function Notation….

- We will use (X,Y) to denote the input and output data we collected. Note that there will be N such pairs where the input is a collection of P values and the output Y is a collection of M values. We will denote the *ith* pair in the dataset as X_i and Y_i.

- We will use $\hat{Y}$ to denote the output of the neural net. Of course, $\hat{Y}$ is the network's guess at Y and therefore it will also have M features.

Loss Functions for Regression:

Loss function in regression is the least squares

$$L(W, b) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{Y}_i - Y_i\right)^2$$

Where Y_i is the desired output

$\hat{Y}_i$ is the predicted output

N is the size of the dataset

In above case we have considered we have only one output feature(M=1).

Loss Function…

When We have more than one output features the loss function is:

$$L(W, b) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{j=1}^{M}(\hat{y}_{ij} - y_{ij})^2$$

When M=N

$$L(W, b) = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{M}(\hat{y}_{ij} - y_{ij})^2$$

Loss Function…

Loss Functions for Classification :

We can build neural networks to bin data points into different categories; for example, fraud or not a fraud.0= no fraud and 1 =fraud, which by convention is called a **0-1 classifier. (Binary classifier)**

- Hinge loss is the most commonly used loss function when the network must be optimized for a hard classification.
- **Hinge loss is also seen in a class of models called maximum-margin classification models (e.g., support vector machines)**

Loss Function….

(Soft classifiers: Estimate the class conditional probabilities and then perform classification based on estimated probabilities.

Hard classifiers: **Directly target on the classification decision boundary without producing the probability estimation.).**

Hing Loss equation:

$$L(W, b) = \frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_{ij} \times \hat{y}_{ij})$$

Logistic Loss:

- Logistic loss functions are used when probabilities are of greater interest than hard classifications.
- Let us consider the case in which our network predicts a probability for two classes, like the fraud and not fraud 0–1 classifier
- For the given **set of bias b and weight w .The probability of 1 and 0**

$$P(y_i = 1 \mid X_i; \mathbf{W}, \mathbf{b}) = h_{\mathbf{W},\mathbf{b}}(X_i)$$

$$P(y_i = 0 \mid X_i; \mathbf{W}, \mathbf{b}) = 1 - h_{\mathbf{W},\mathbf{b}}(X_i)$$

We can combine these equations and express them as follows:

$$P(y_i \mid X_i; \mathbf{W}, \mathbf{b}) = (h_{\mathbf{W},\mathbf{b}}(X_i))^{y_i} \times (1 - h_{\mathbf{W},\mathbf{b}}(X_i))^{1-y_i}$$

Negative log likelihood

For the sake of mathematical convenience, when dealing with the **product of probabilities, convert the probabilities into a log of probabilities.**

The product of the probabilities transforms into the sum of the log of the probabilities.

$$L(W, b) = -\sum_{i=1}^{N} y_i \times \log \hat{y}_i + (1 - y_i) \times \log (1 - \hat{y}_i)$$

If there are M classes then the equation for negative log-likelihood will change to:-

$$L(W, b) = -\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j} \times \log \hat{y}_{i,j}$$

Building Blocks of Deep Networks

Building deep networks goes beyond basic feed-forward multilayer neural networks.

Three specific building blocks:-

- Feed-forward multilayer neural networks
- RBMs
- Autoencoders

Feed-Forward Networks

- Feed-Forward Networks are the simplest Artificial Neural Networks.

- They are composed of an input layer, one or many hidden layers, and an output layer.