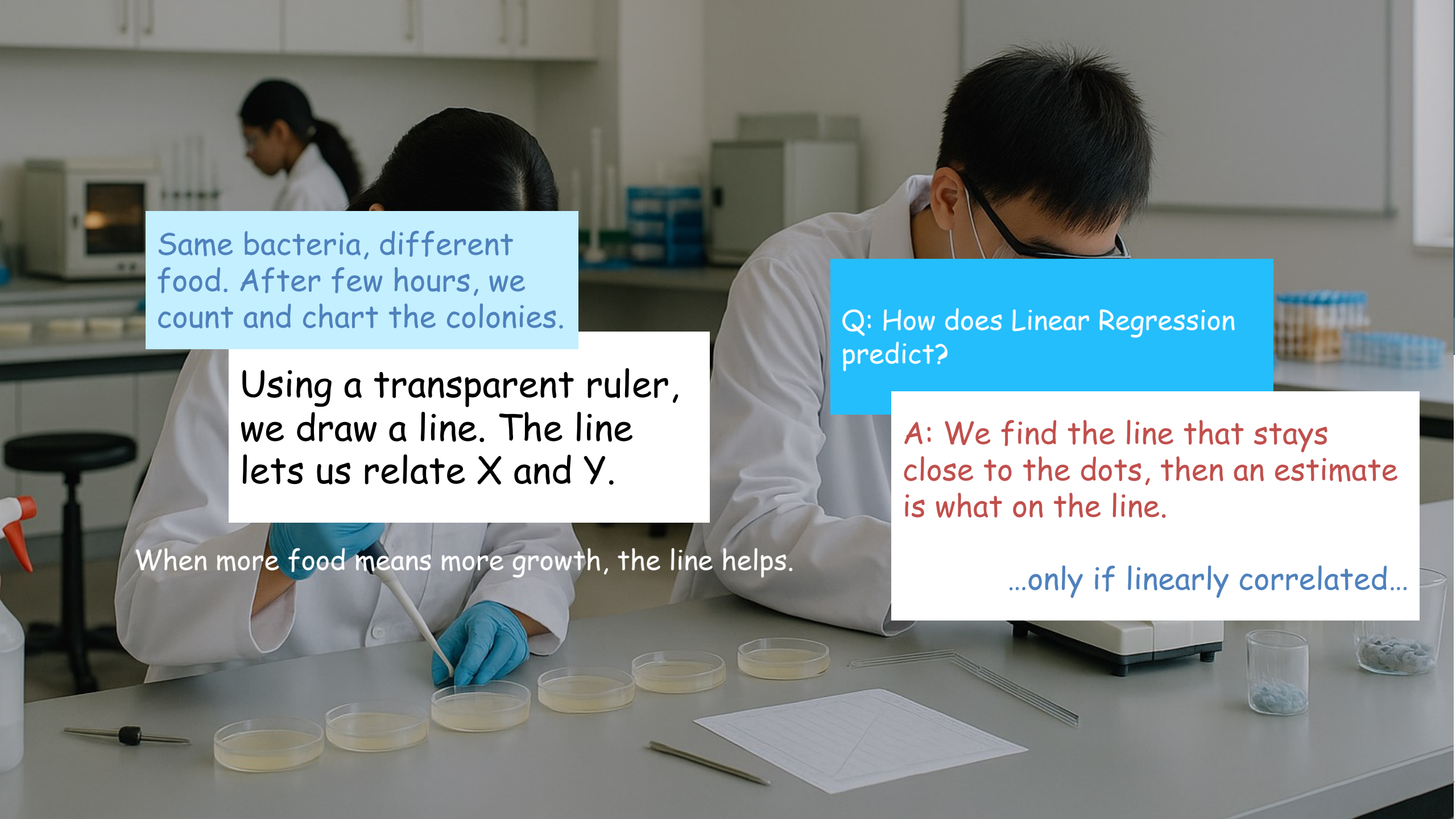
Machine Learning

Linear Regression

Tarapong Sreenuch

8 February 2024

克明峻德，格物致知



Same bacteria, different food. After few hours, we count and chart the colonies.

Using a transparent ruler, we draw a line. The line lets us relate X and Y.

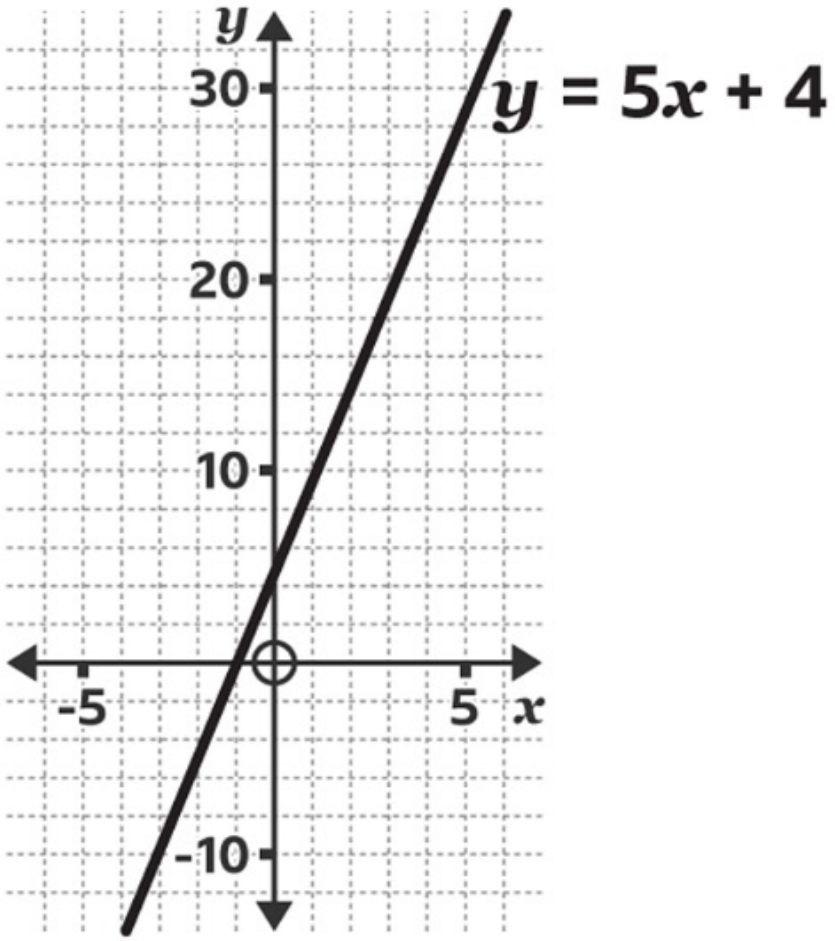
When more food means more growth, the line helps.

Q: How does Linear Regression predict?

A: We find the line that stays close to the dots, then an estimate is what on the line.

...only if linearly correlated...

Linear Equation Graph



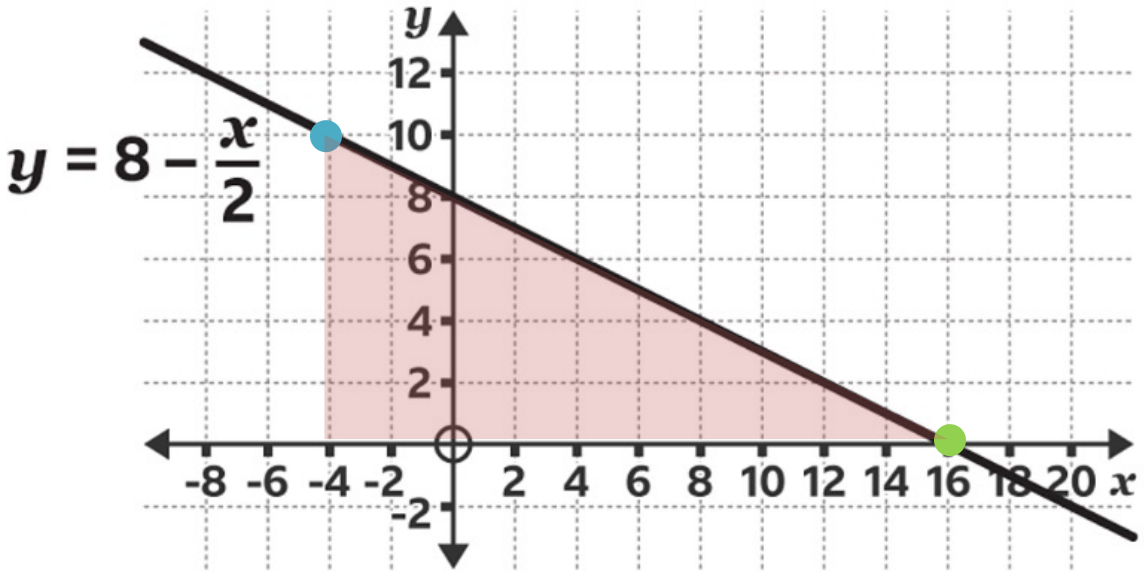
Q: What is the value of y when x = 4?

A: y = 24

Q: What is the value of x when y = 9?

A: x = 1

Linear Equation Graph



Intercept

$y = mx + c$

Gradient (or Slope)

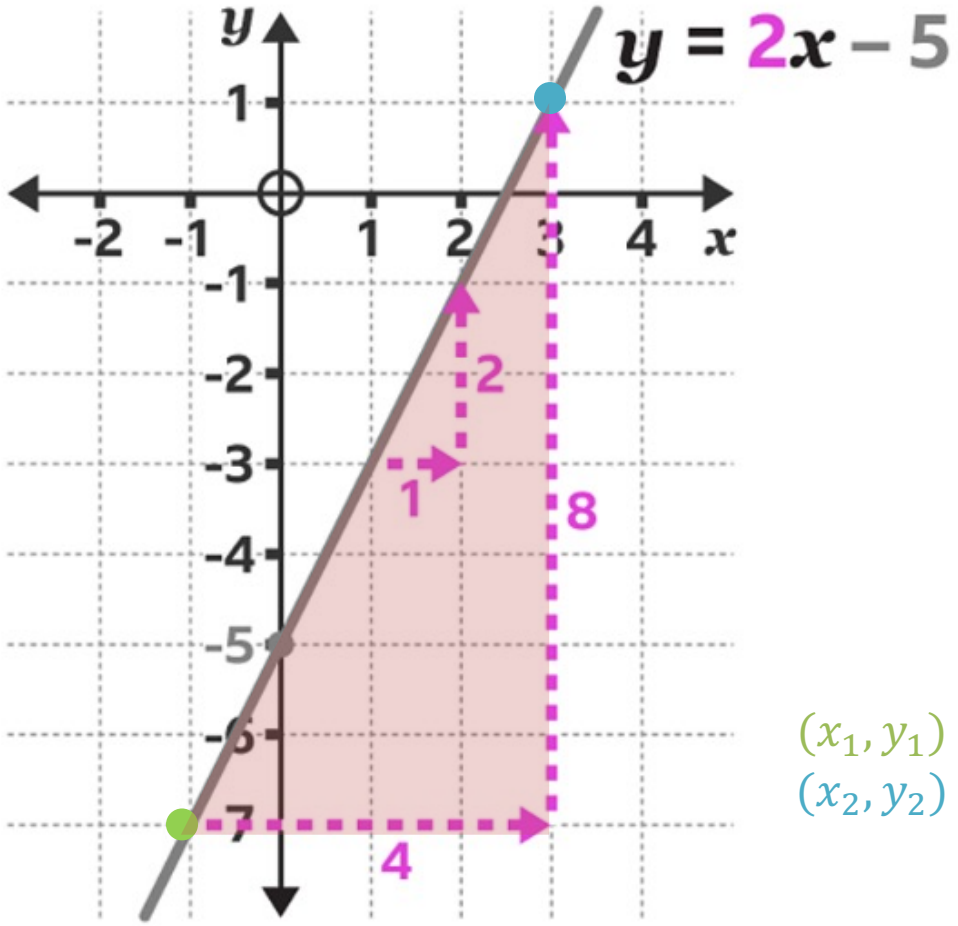
$$\text{Gradient} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{10 - 0}{-4 - 16} = -\frac{1}{2}$$

$(x_1, y_1) = (16,0)$

$(x_2, y_2) = (-4,10)$

$\text{Intercept} = 8$

Linear Equation Graph



$(x_1, y_1) = (-1, -7)$

$(x_2, y_2) = (3,1)$

Intercept

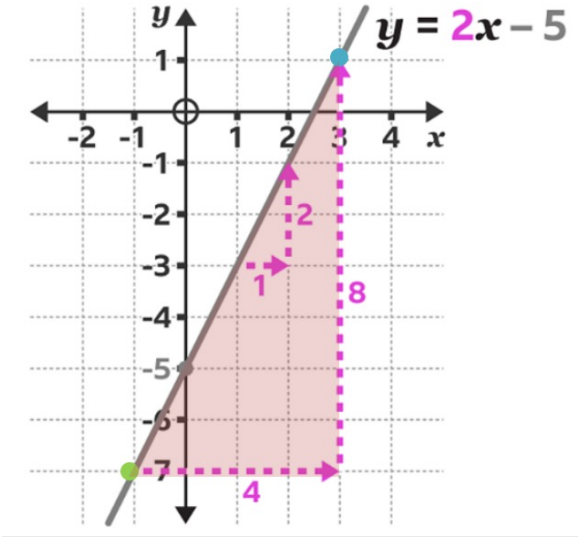
$y = mx + c$

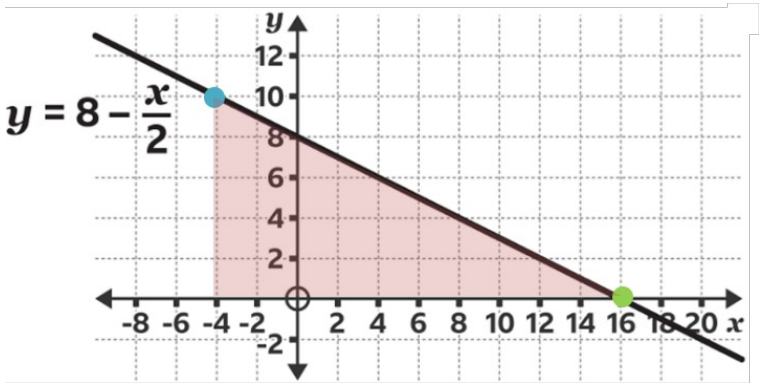
Gradient (or Slope)

$$\text{Gradient} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{1 - (-7)}{3 - (-1)} = 2$$

$$\text{Intercept} = -5$$

Recap: Gradients

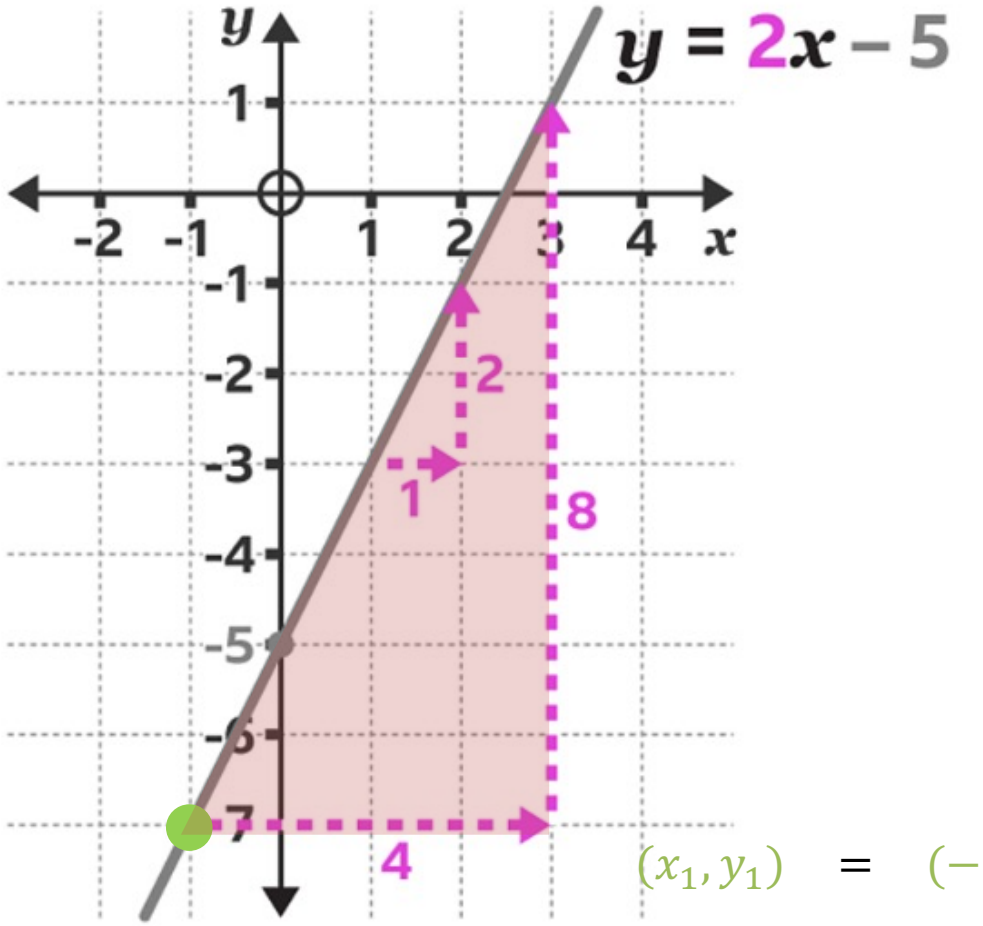




x ↑ y ↑ → ⊕ gradient

x ↑ y ↓ → ⊖ gradient

Linear Equation Graph



$$-7 = 2\times(-1) - 5$$

$$-7 = (-1)\times 2 + 1\times(-5)$$

$$-7 = 1\times(-5) + (-1)\times 2$$

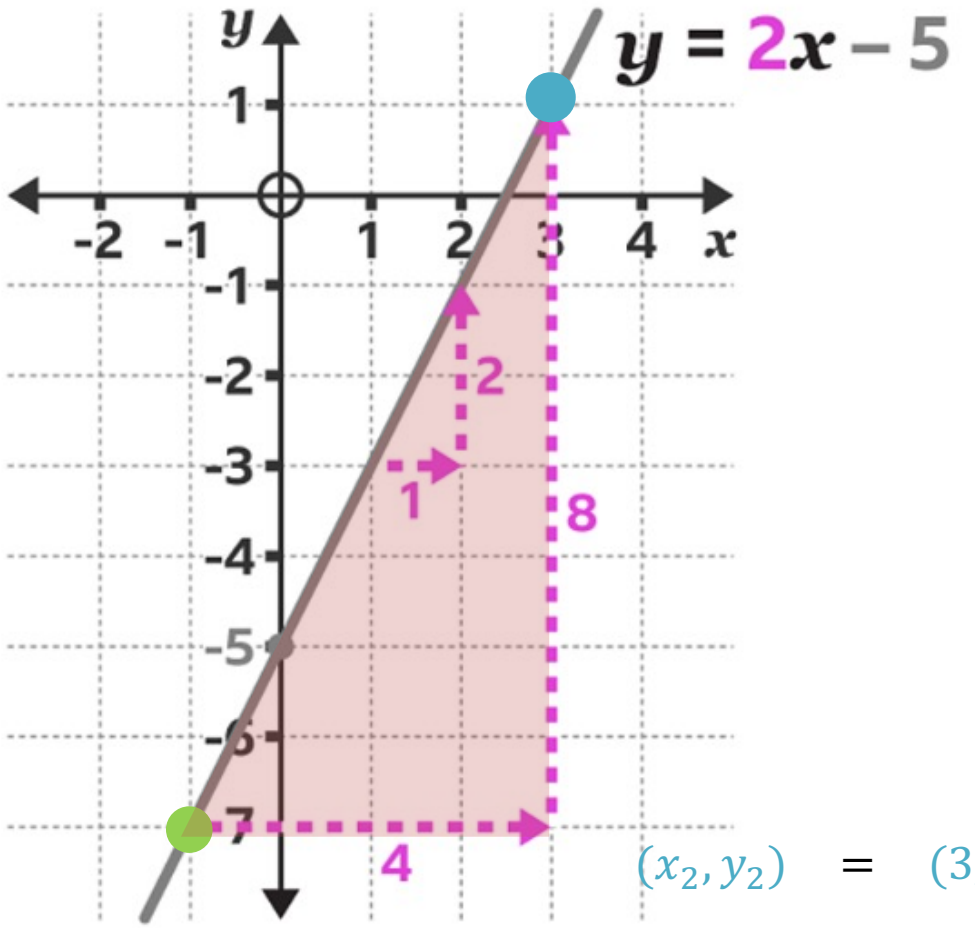
$$-7 = \begin{bmatrix}1 & -1\end{bmatrix}\times\begin{bmatrix}-5\\2\end{bmatrix}$$

$$\hat{y} = \vec{x}^T\times\vec{w}$$

$$\vec{x} = \begin{bmatrix}1\\-1\end{bmatrix} \qquad \vec{W} = \begin{bmatrix}-5\\2\end{bmatrix}$$

$$\hat{y} = -7$$

Linear Equation Graph



$(x_2, y_2) = (3,1)$

$$1 = 2\times 3 - 5$$

$$1 = 3\times 2 + 1\times(-5)$$

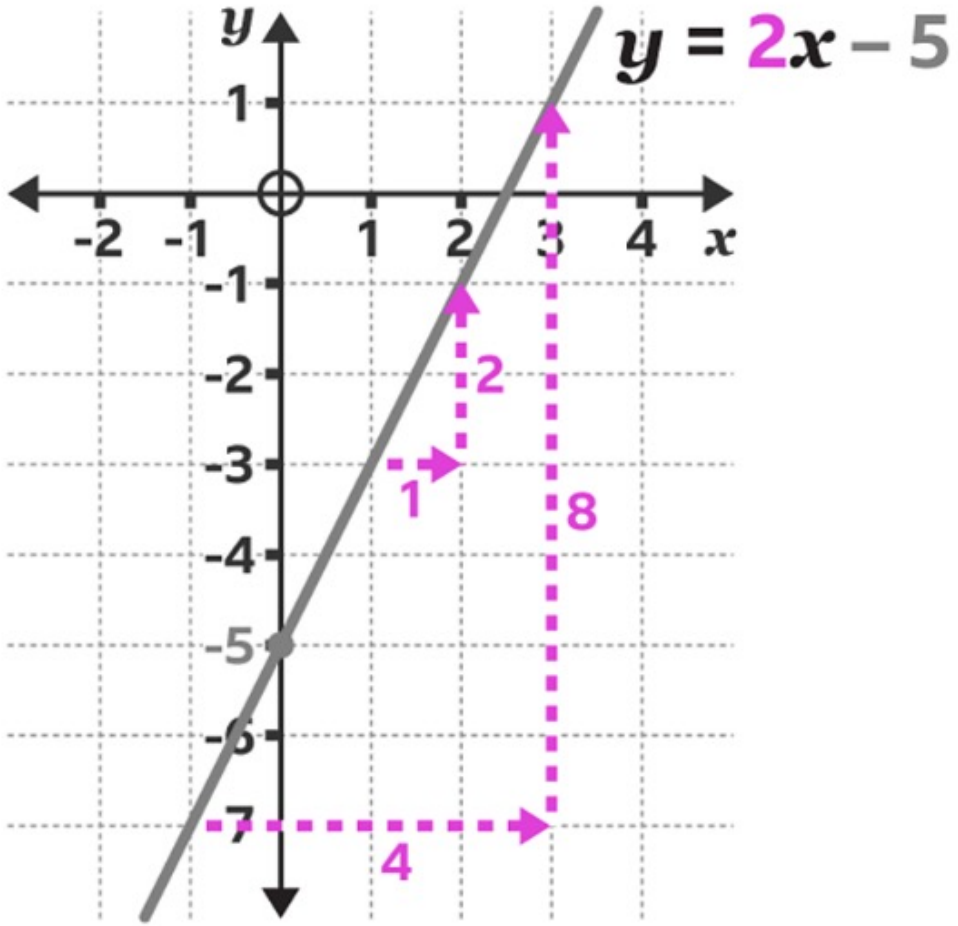
$$1 = [1 \quad 3]\times\begin{bmatrix}-5\\2\end{bmatrix}$$

$$\hat{y} = \vec{x}^T\times\vec{w}$$

$$\vec{x} = \begin{bmatrix}1\\3\end{bmatrix} \qquad \vec{W} = \begin{bmatrix}-5\\2\end{bmatrix}$$

$$\hat{y} = 1$$

Matrix Notations



$$
\begin{aligned}
y &= \vec{x}^T \times \vec{w} \\
&= \begin{bmatrix} 1 & x^{(1)} \end{bmatrix} \times \begin{bmatrix} -5 \\ 2 \end{bmatrix} \\
&= 1 \times (-5) + x \times 2 \\
&= 2x - 5 \\
&= -5 + 2x
\end{aligned}
$$

always equals to 1

Intercept

$$
\vec{x} = \begin{bmatrix} 1 \\ x^{(1)} \\ \vdots \\ x^{(M)} \end{bmatrix} \qquad \vec{W} = \begin{bmatrix} w^{(0)} \\ w^{(1)} \\ \vdots \\ w^{(M)} \end{bmatrix}
$$

Feature Vector

Weight

Slopes

Linear Regression

Q: What is the best estimate of the price of the house?

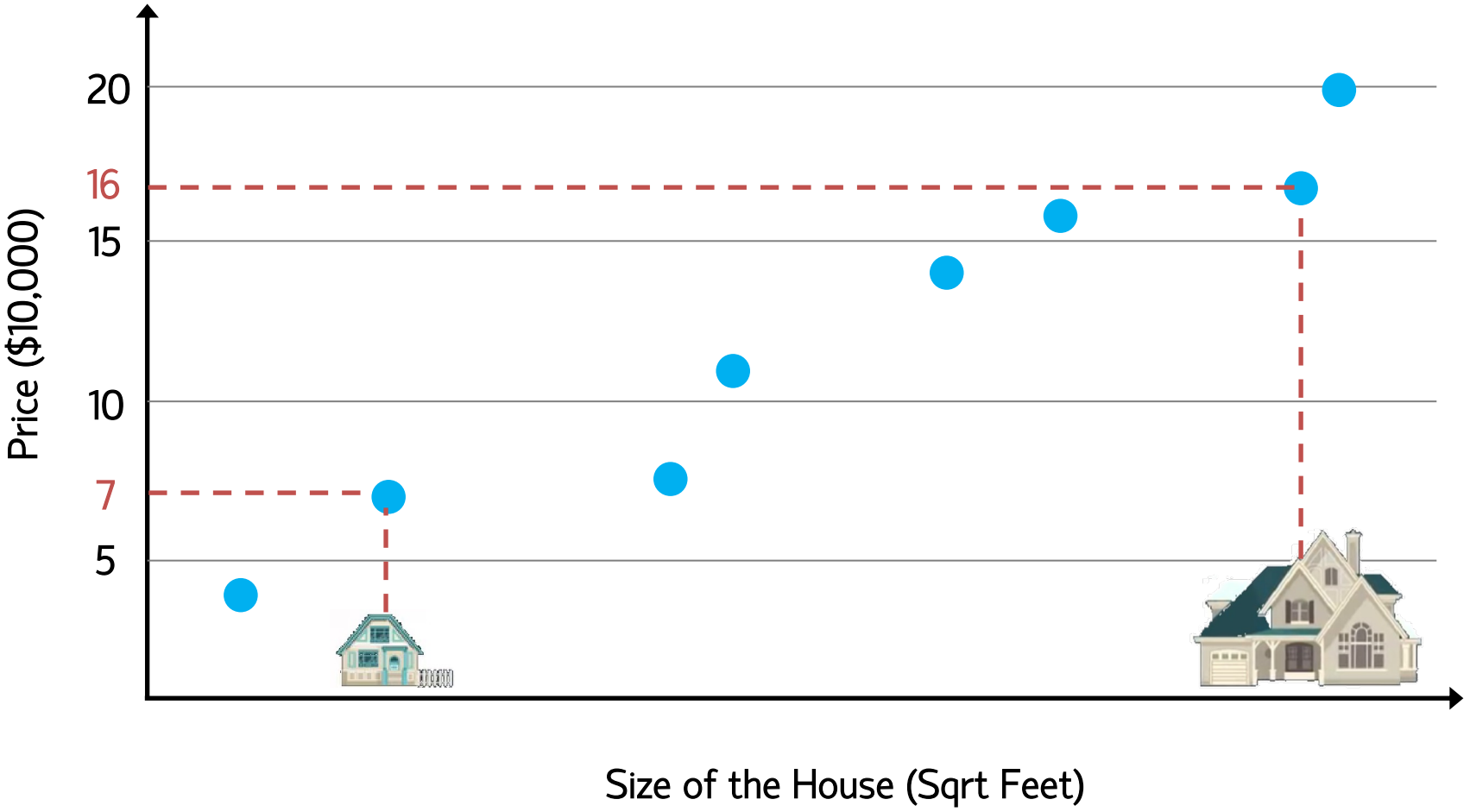
- ☐ $80,000
- ☐ $120,000
- ☐ $190,000

$70,000

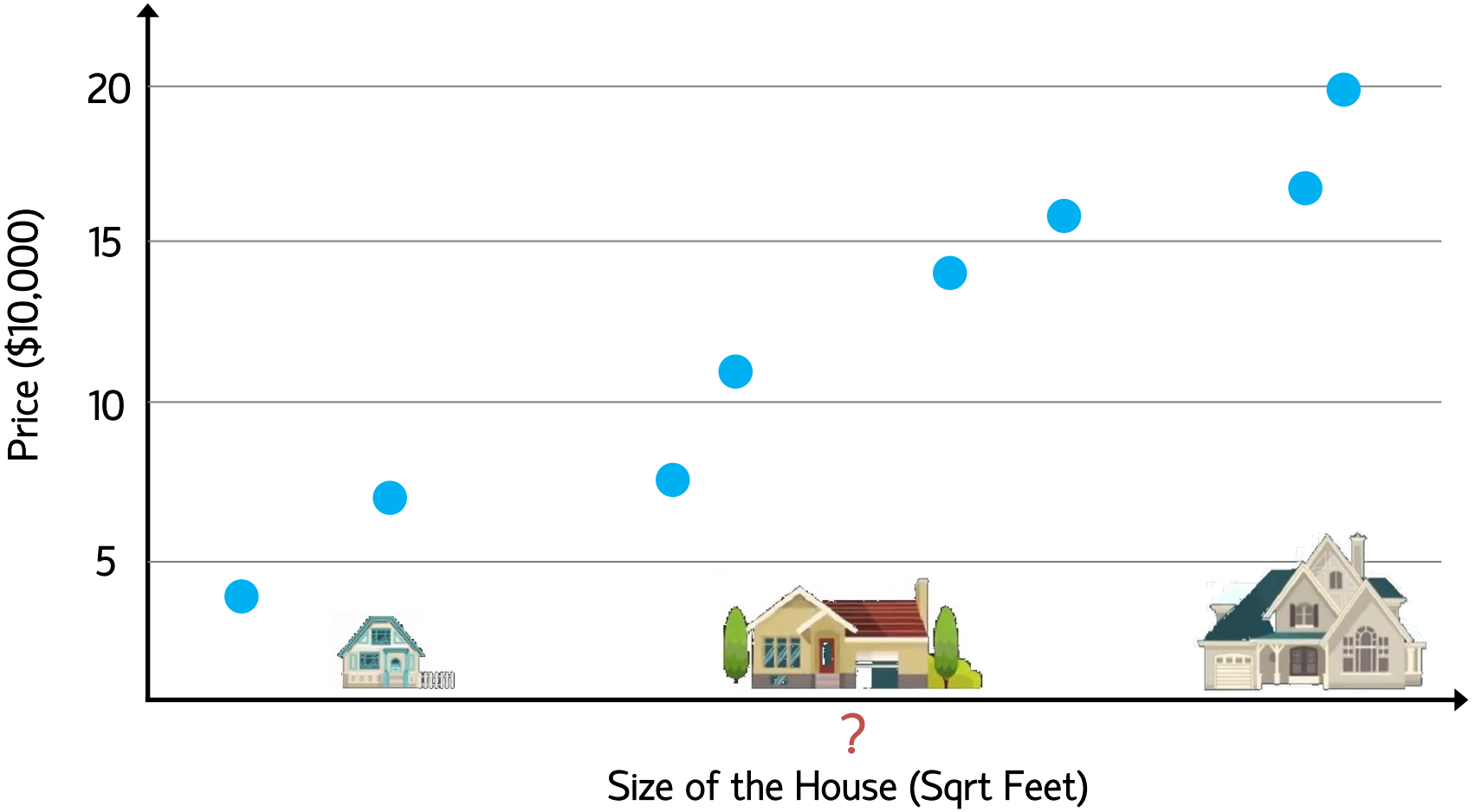
$160,000

Linear Regression

Linear Regression is often used to predict continuous values (aka target variable) from one or more input features.

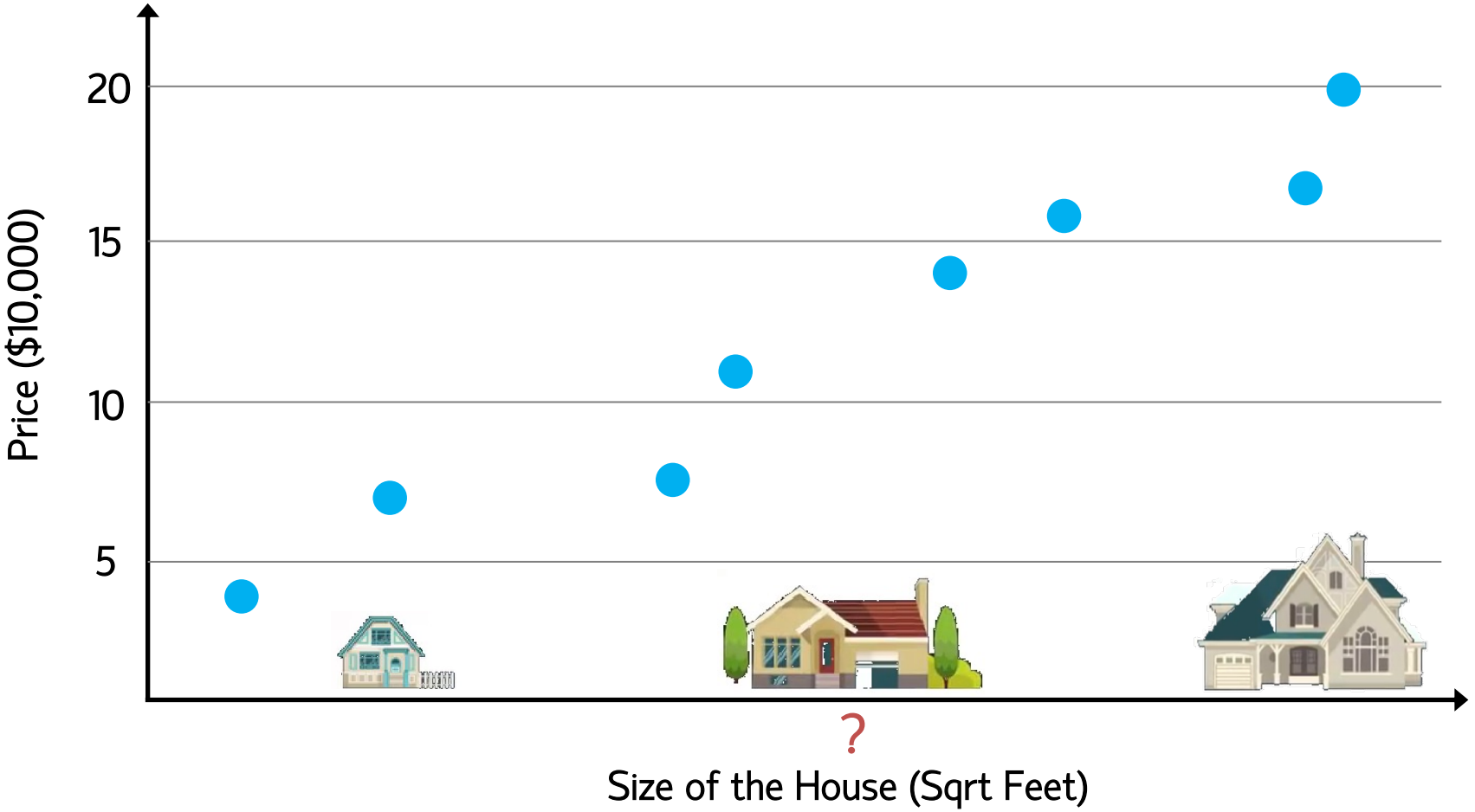


Q: What is the best estimate of the price of the house?



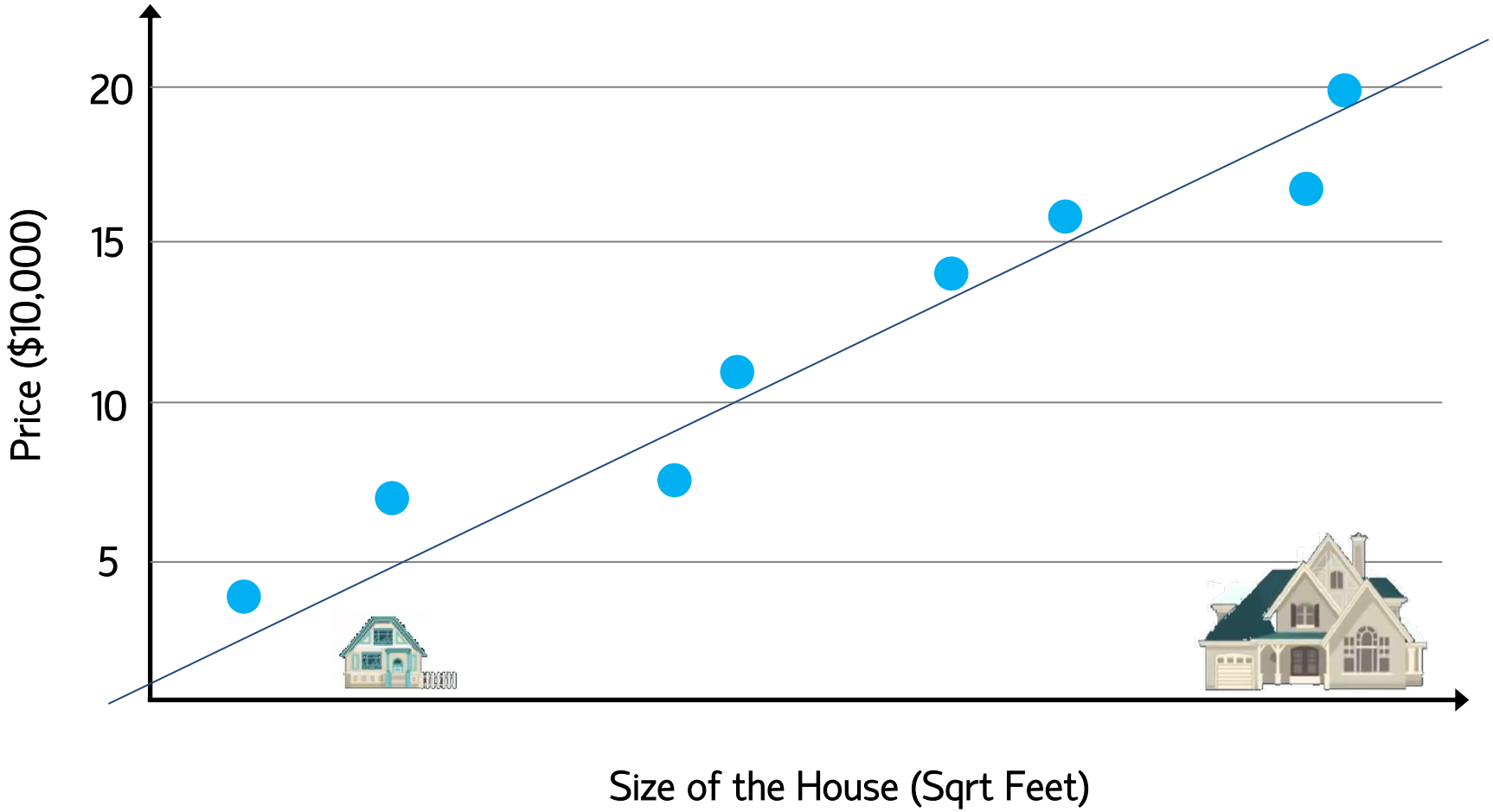
Recall: Prediction

Prediction *is the process of filling in missing information.* Prediction takes information we have, often called '*data*', and uses it to generate information we don't have.



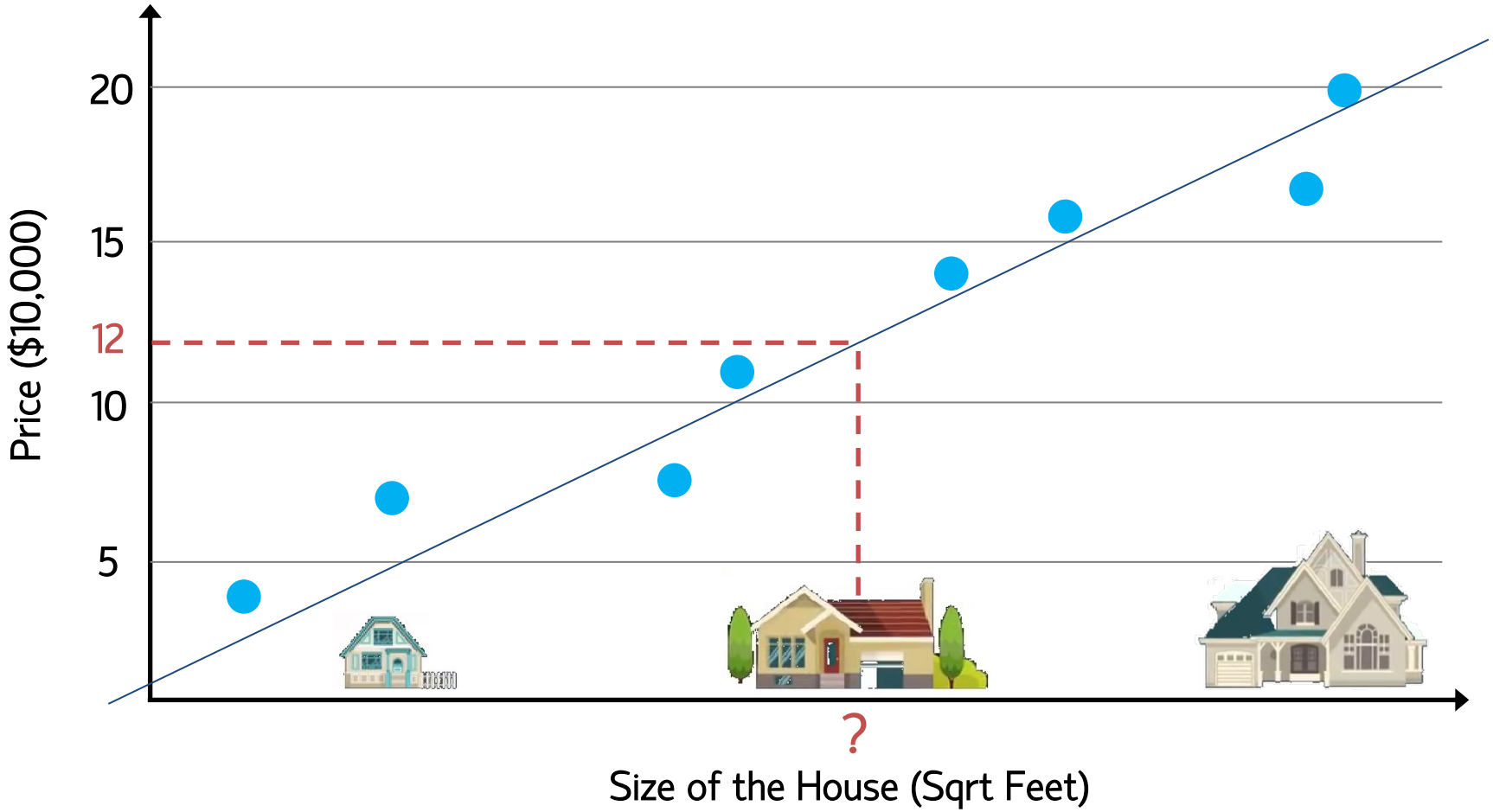
Linear Regression

The Linear Regression algorithm predicts the target value for a test dataset by modeling the linear relationship between the input features and the target variable. The model assumes that the target value can be represented as a weighted sum of the input features plus an intercept.

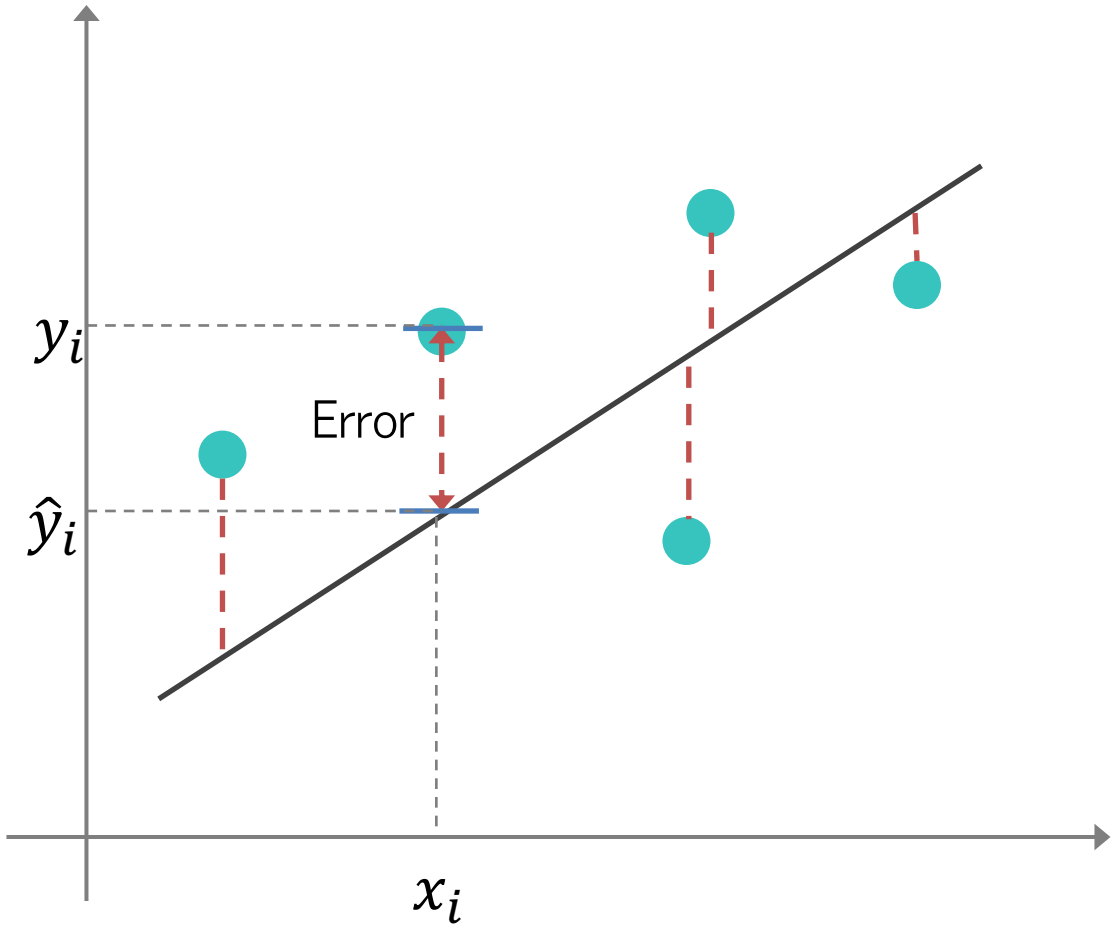


Linear Regression

The Linear Regression algorithm predicts the target value for a test dataset by modeling the linear relationship between the input features and the target variable. The model assumes that the target value can be represented as a weighted sum of the input features plus an intercept.



Best Fit Line



$$\text{Error (Residue)} \quad = \quad y - \hat{y}$$

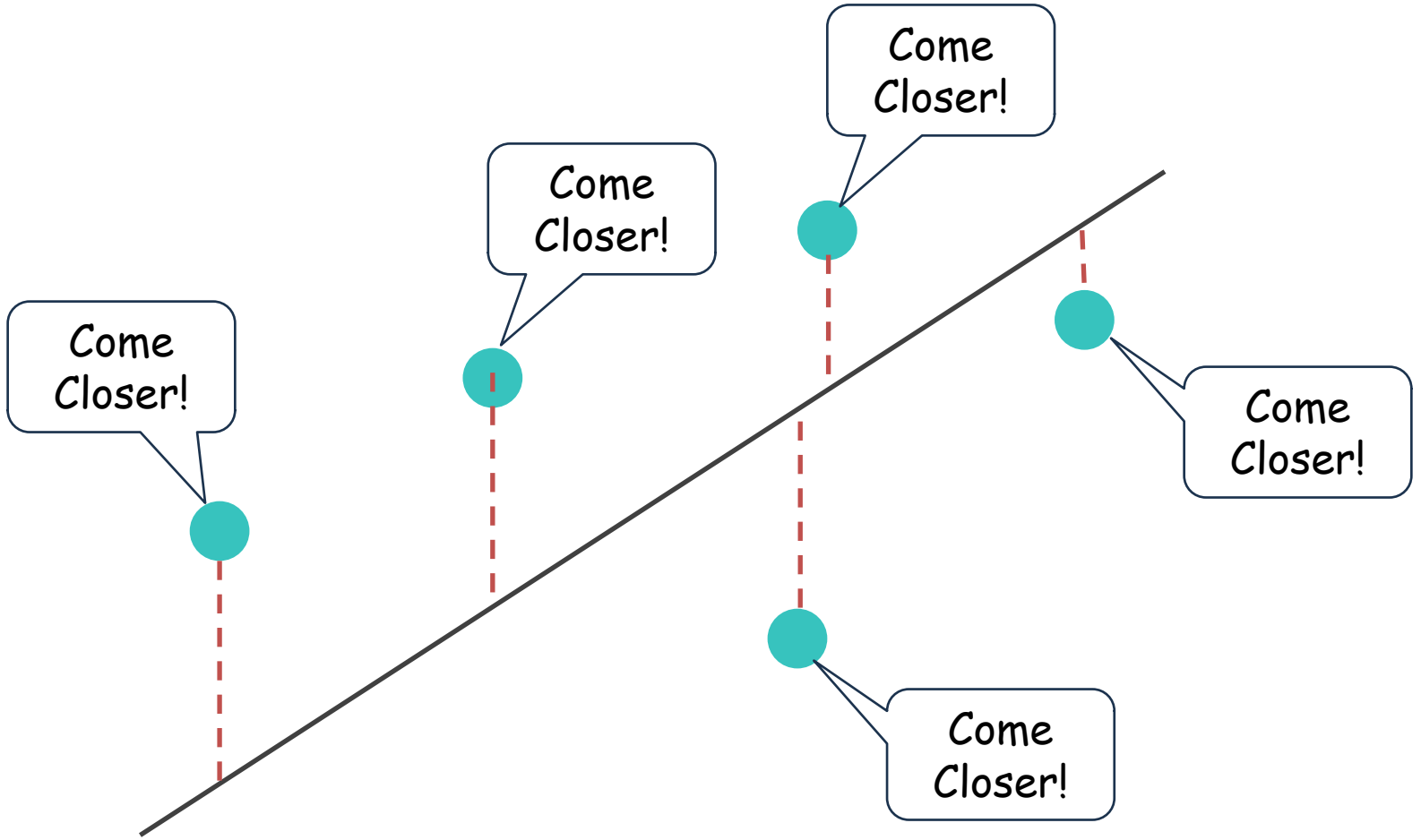
where y and $\hat{y}$ are the observed and predicted values, respectively.

Mean Squared Error (MSE):

$$\text{MSE} \quad = \quad \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

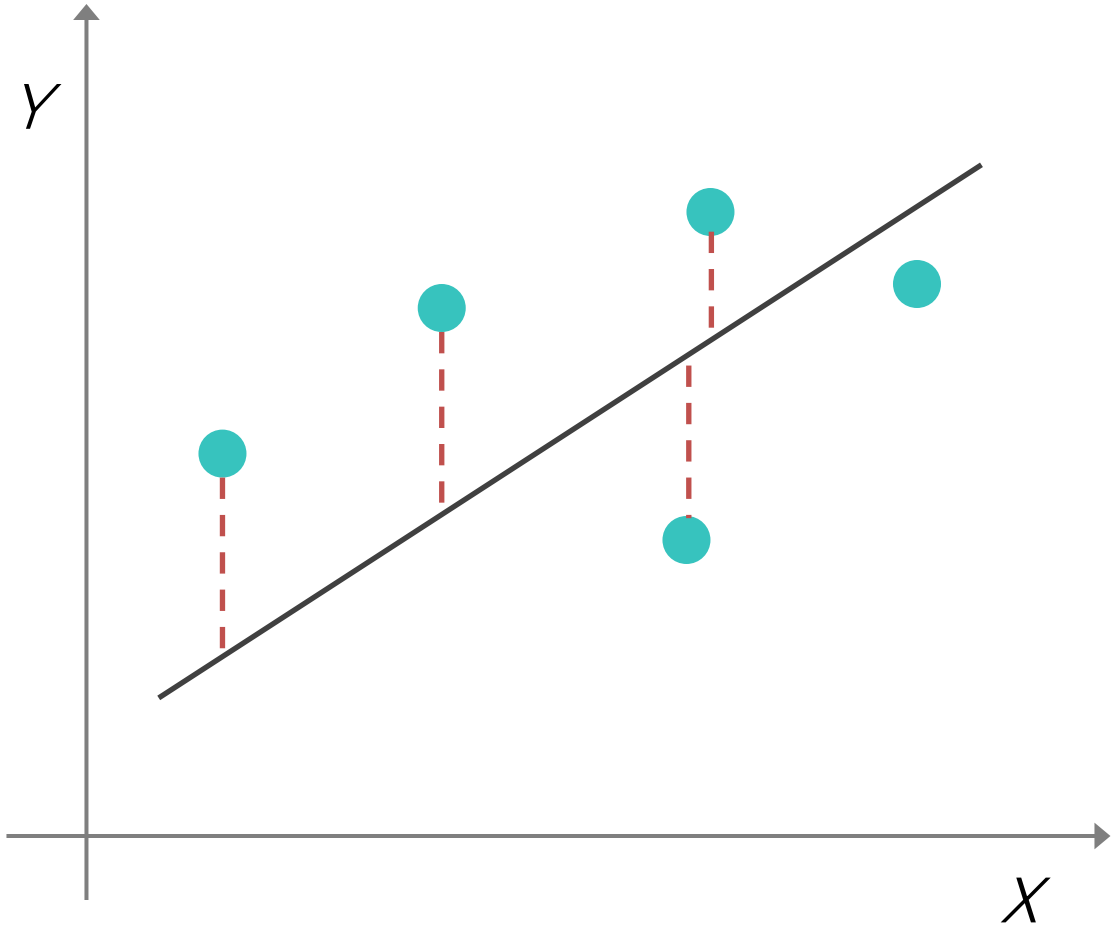
where N is the total number of data points.

Finding A Line



Best Fit Line

The weights, or coefficients, are estimated by minimizing the difference between the predicted and actual values, typically using a method like Ordinary Least Squares.



Minimise function over all possible $\vec{W}$

$$\min_{\vec{W}} \sum_{i=1}^{N} (y_i - \vec{x}_i^{\,T} \times \vec{w})^2$$

Sum Squared Error (SSE):

$$\text{SSE} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

where

$$\hat{y}_i = \vec{x}_i^{\,T} \times \vec{w}$$

SSE: Matrix Notations

Sum Squared Error (SSE):

$$\text{SSE} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Predicted Value

$$= \sum_{i=1}^{N} \left(y_i - \vec{x}_i^{\,T} \times \vec{w}\right)^2$$

$$= (Y - X \times \vec{w})^T (Y - X \times \vec{w})$$

Errors

Feature Vector (Data Point)

Inputs

$$X = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_1^{(M)} \\ 1 & x_2^{(1)} & \dots & x_2^{(M)} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_N^{(1)} & \dots & x_N^{(M)} \end{bmatrix}$$

Intercept

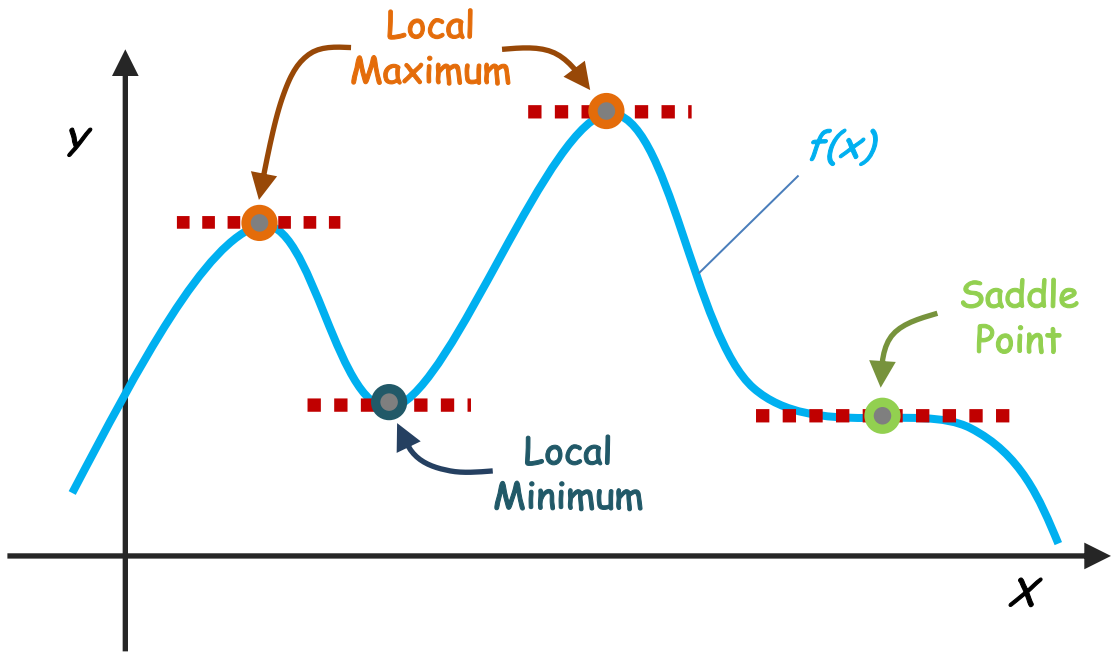
$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \qquad \vec{w} = \begin{bmatrix} w^{(0)} \\ w^{(1)} \\ \vdots \\ w^{(M)} \end{bmatrix}$$

Outputs

Weights

Slopes

Calculus



Q: Which value of x will f(x) be either minimum or maximum?

Hint: What will happen to Slope (or Gradient) at those x(s)?

A: … Slope (or Gradient) = 0 …

Ordinary Least Square

Sum Squared Error (SSE):

$$\text{SSE} = (Y - X \times \vec{w})^T (Y - X \times \vec{w})$$

SSE Derivative:

$$\nabla\text{SSE} = 2X^T (Y - X \times \vec{w})$$

To minimize SSE, we set $\nabla\text{SSE} = 0$ and then solve for $\vec{w}$.

$$0 = 2X^T (Y - X \times \vec{w})$$

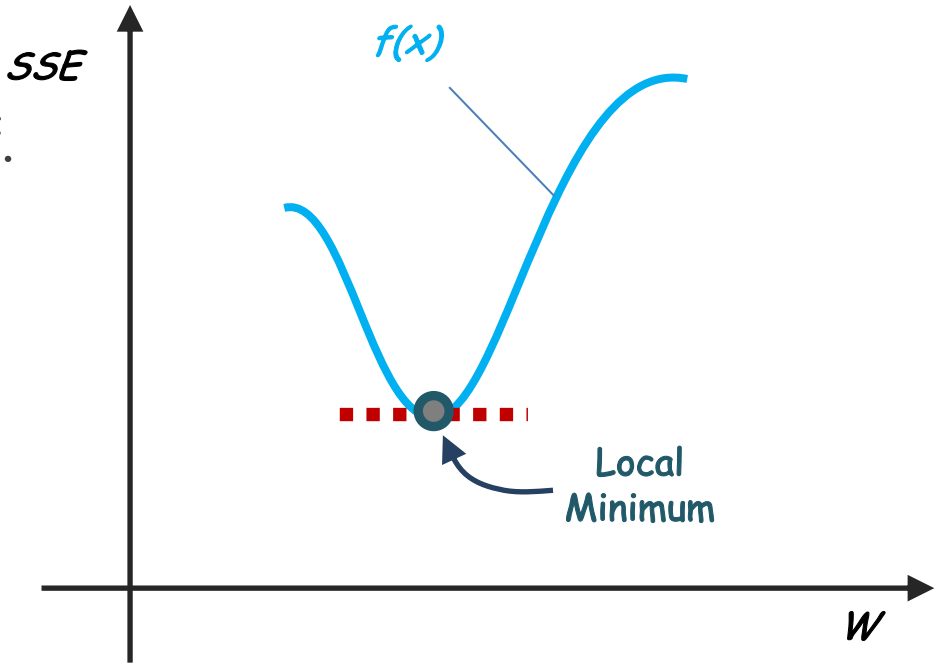
$$0 = X^T Y - X^T X \vec{w}$$

$$(X^T X)^{-1}(X^T X)\vec{w} = (X^T X)^{-1} X^T Y \quad (= 1)$$

$$\vec{w} = (X^T X)^{-1} X^T Y$$

Q: Which value of x will f(x) be minimum?

A: ... Slope (or Gradient) = 0 ...



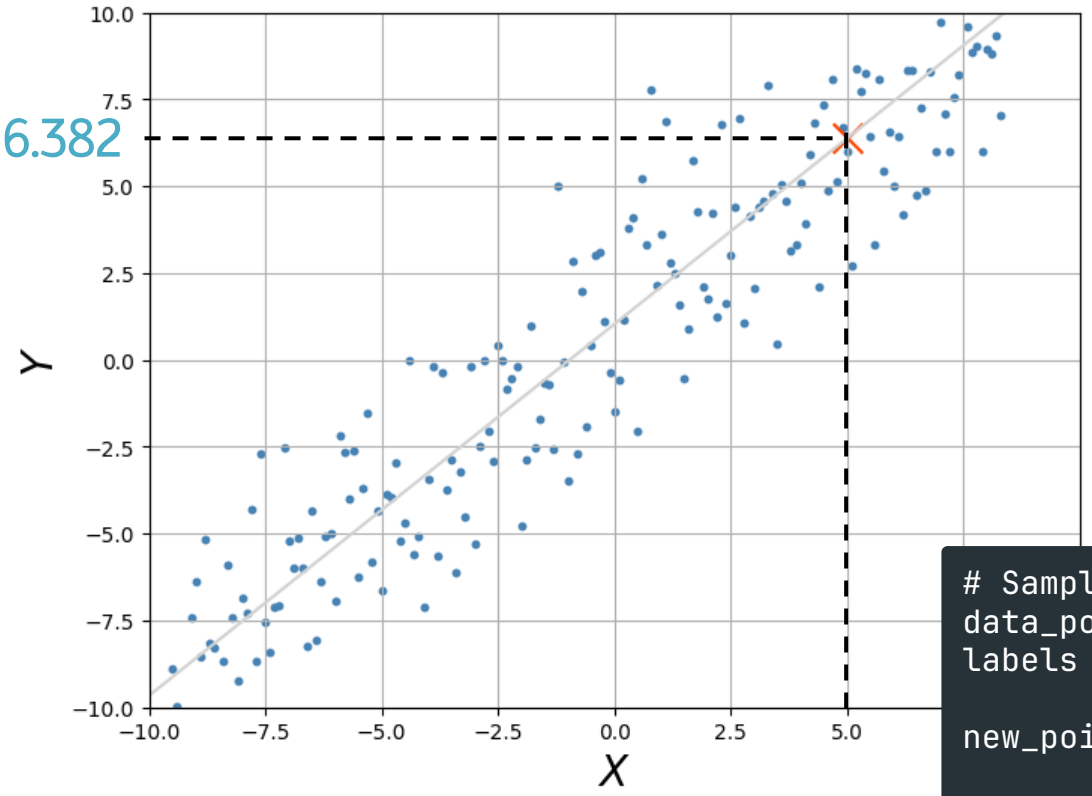
Pseudocode for Linear Regression

```
Function Linear_Regress(new_point, data_points, labels)
    Begin
        // Step 1: Add bias term (column of ones) to data_points
        Add a column of ones to data_points: X

        // Step 2: Find best fit line by calculate the closed-form solution for the weights (theta)
        weights = inverse(transpose(X) * X) * transpose(X) * labels

        // Step 3: Add bias term to new_point
        Add a 1 (bias term) to new_point: x

        // Step 4: Predict the value for new_point
        prediction = transpose(x) * weights

        Return prediction
    End
```

Python Code Snippet

```
def Linear_Regress(new_point, data_points, labels):
    # Step 1: Add bias term (column of ones) to data_points
    X = np.hstack([np.ones((data_points.shape[0], 1)), data_points])

    # Step 2: Calculate the closed-form solution for the weights (theta)
    weights = np.linalg.inv(X.T @ X) @ (X.T @ labels)

    # Step 3: Add bias term to new_point and rename it as x
    x = np.hstack([1, new_point])

    # Step 4: Predict the value for new_point
    predicted_value = x @ weights

    # Return the predicted value
    return predicted_value
```

Usage Example




```
# Sample data
data_points = np.arange(-9.5, 8.5, 0.1)
labels = data_points + 1 + np.random.normal(0, 2, len(data_points))

new_point = np.array([5])

# Make prediction for the new point
predicted_value = Linear_Regress(new_point, data_points, labels)

print(f"Predicted Value: {predicted_value}")
```

```
Predicted Value: 6.382
```

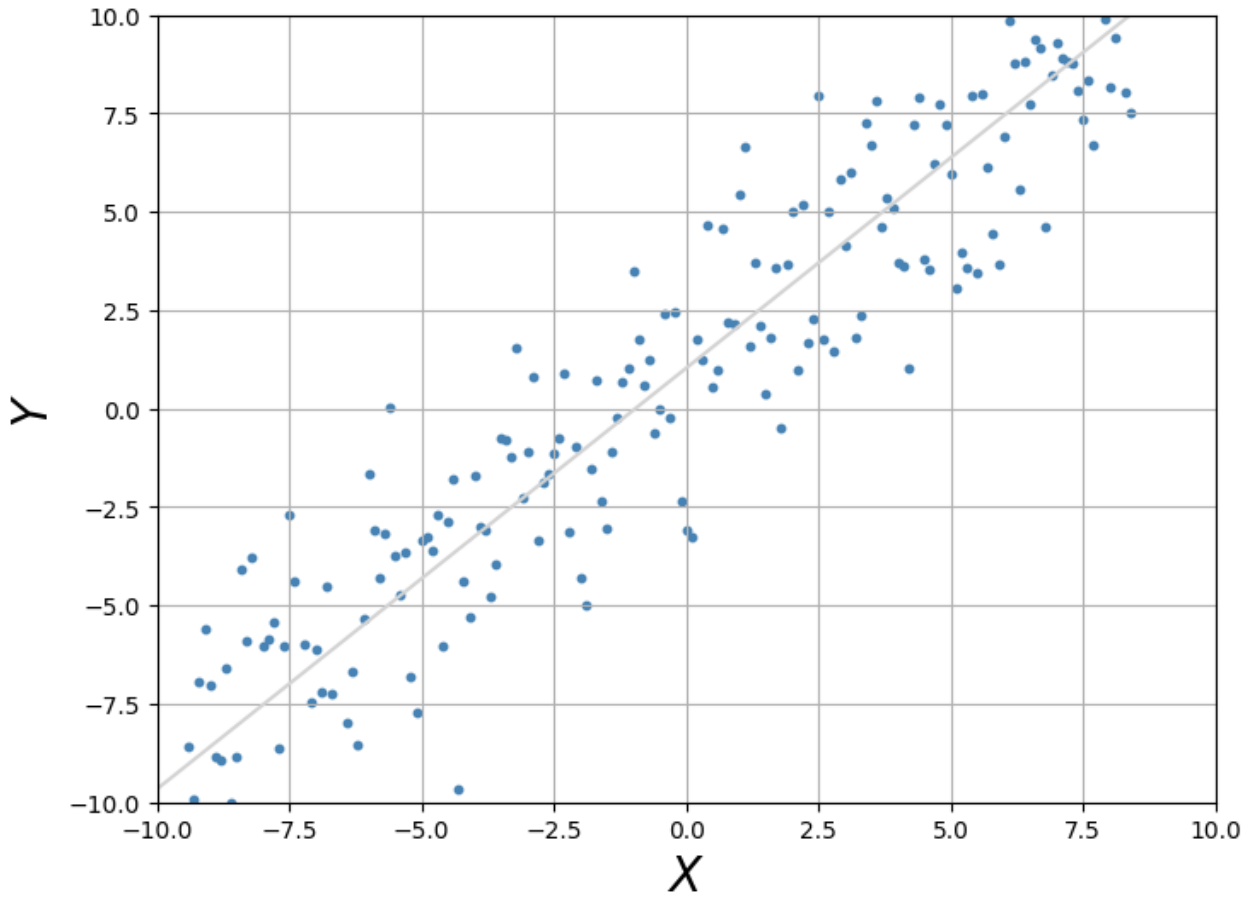
Step 1: Adding Column of Ones

```
# Step 1: Add bias term (column of ones) to data_points
X = np.hstack([np.ones((data_points.shape[0], 1)), data_points])
```

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$

Step 2: Finding Best Fit Line

```
# Step 2: Calculate the closed-form solution for the weights
weights = np.linalg.inv(X.T @ X) @ (X.T @ labels)
```



Weights = [1.069, 1.035]

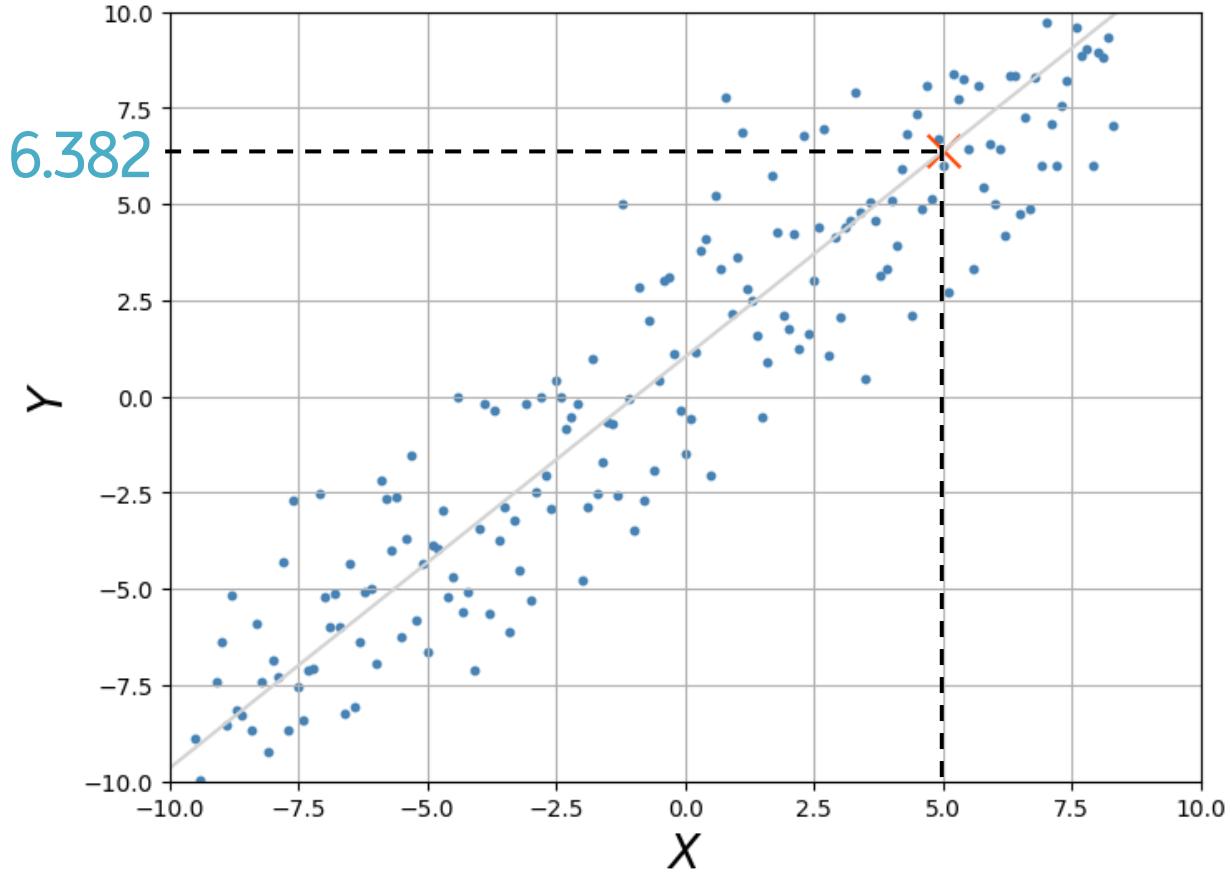
Step 3: Adding Bias Term

```
# Step 3: Add bias term to new_point and rename it as x
x = np.hstack([1, new_point])
```

 # New Point = [5]

$$[5] \quad \Rightarrow \quad [1 \quad 5]$$

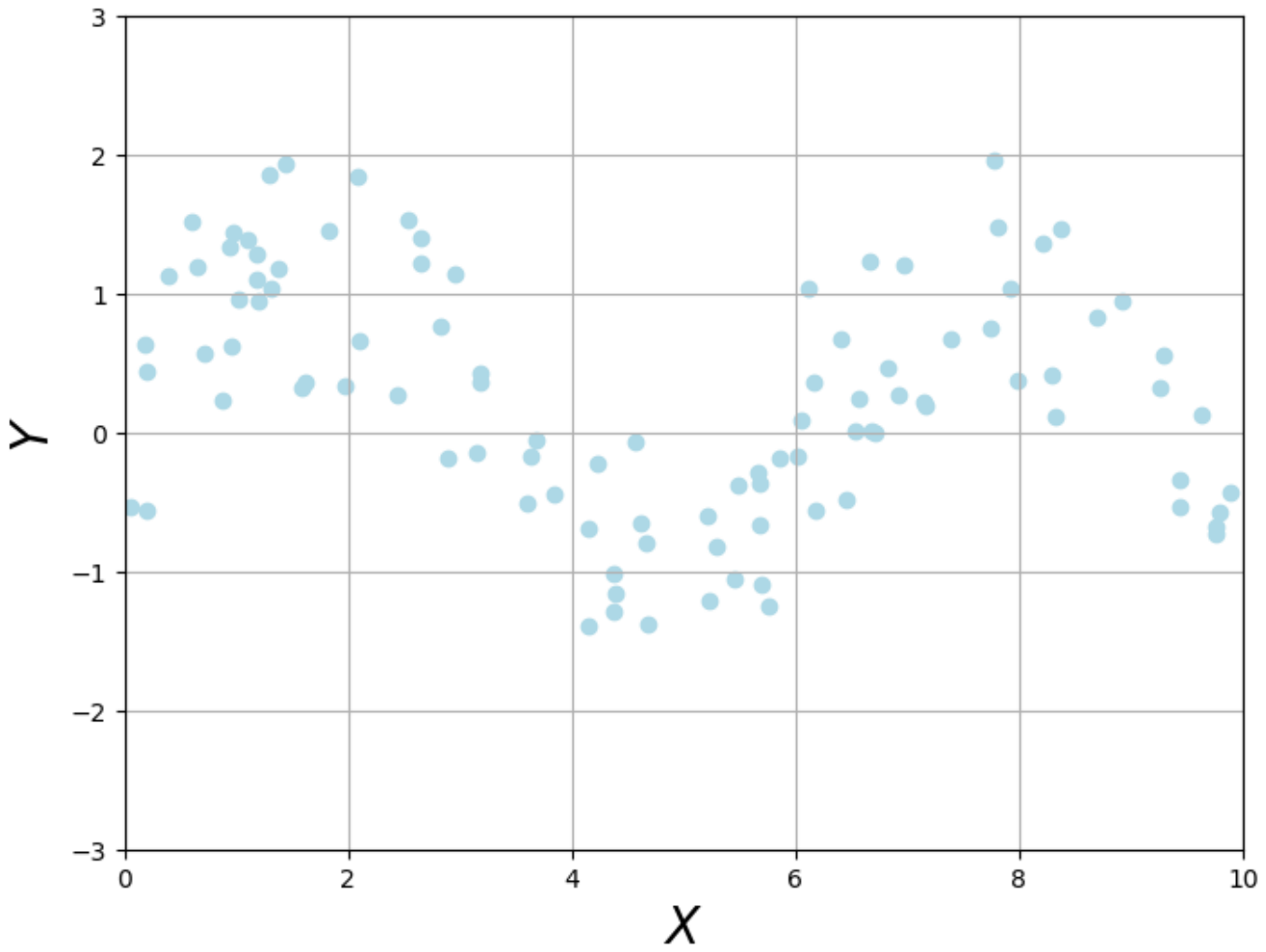
Step 4: Prediction based on Regression




```
# Step 4: Predict the value for new_point
predicted_value = x @ weights
```

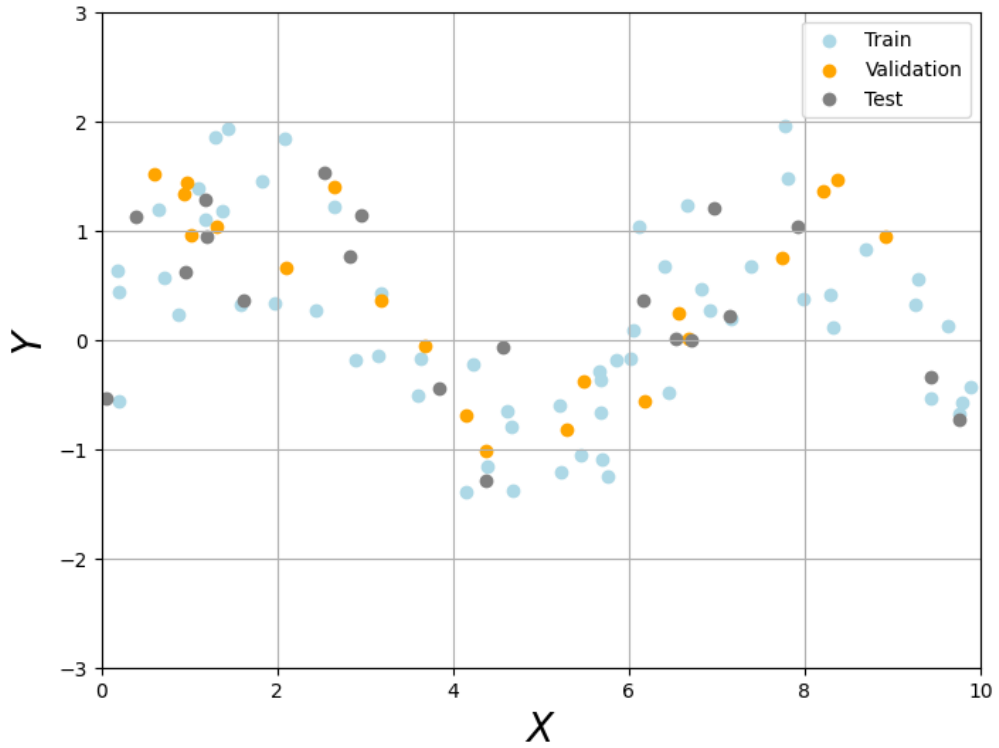
Prediction: 6.382

Example Dataset



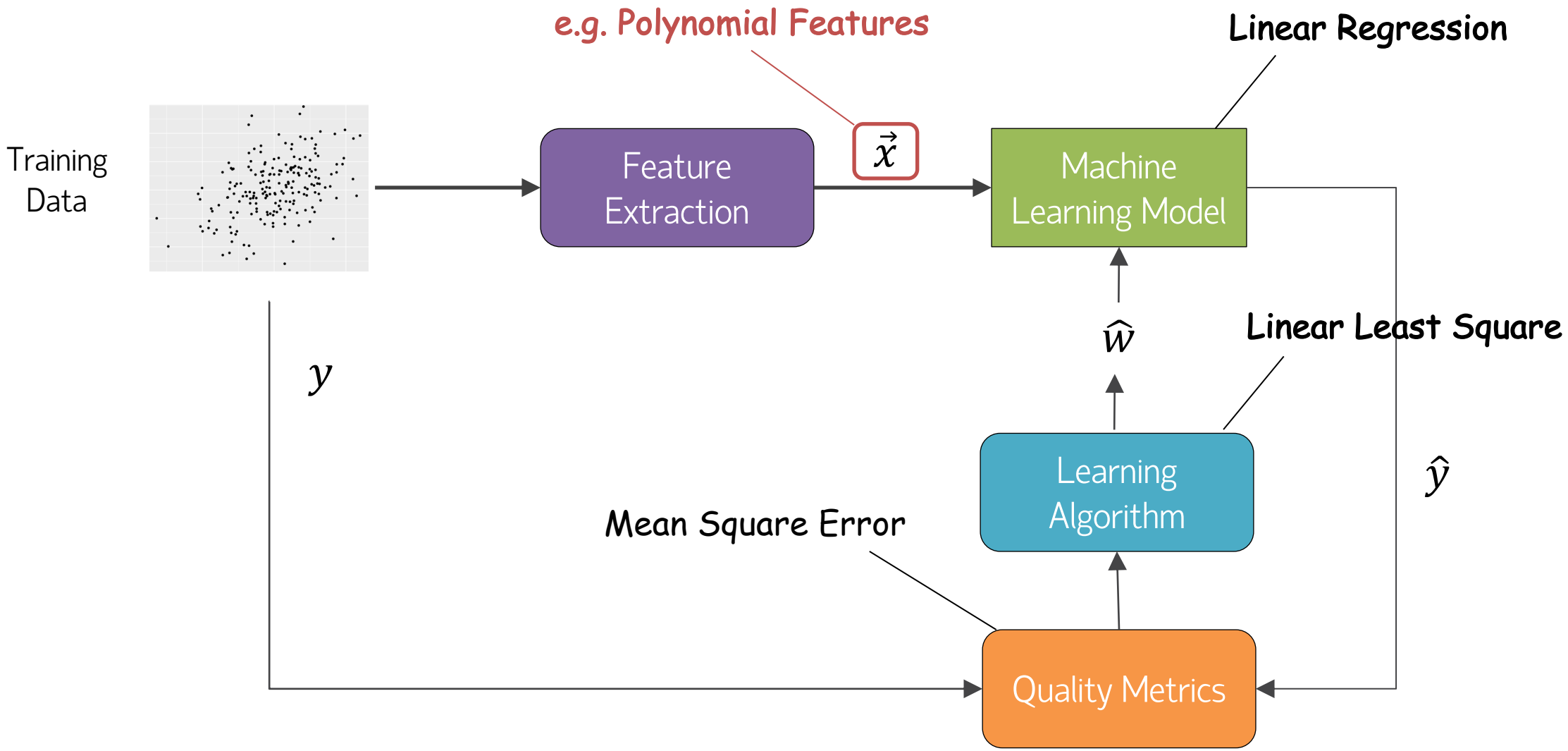
Train, Validation and Test Datasets

```
from sklearn.model_selection import train_test_split

# First, split the data into train (60%) and temp (40%)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.4, random_state=42)

# Then split the temp data into validation (50% of 40% ⇒ 20% of total) and test (remaining 50% of 40% ⇒ 20% of total)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```



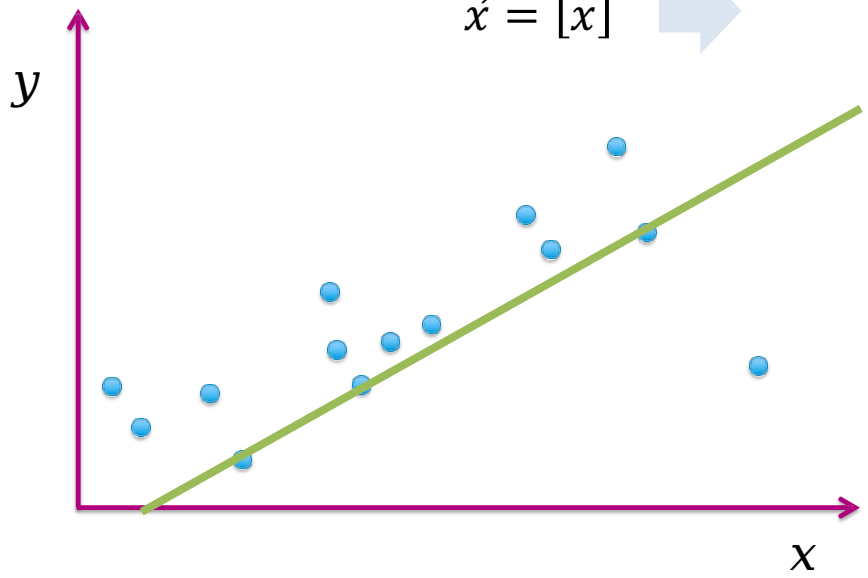
- (Train+Validation):Test is typicaly either 0.8:0.2 or 0.7:0.3.
- **Test** dataset is a proxy of unseen data, and it will only be used in the final evaluation.
- We train our ML model on the **Train** dataset.
- **Validation** datest is used to fine-tune or optimise the ML model. Here, we find a set of hyper-parameters, e.g. a set of features in the Linear context, that will result in the best performance on the Validation dataset.

Workflow: Linear Regression



Feature Extraction: Polynomial Features

$$\vec{x} = [x] \Rightarrow \begin{aligned} y &= \vec{x}^T \times \vec{w} \\ &= w^{(0)} + w^{(1)}x \end{aligned}$$



1st Degree: $\vec{x} = [x]$

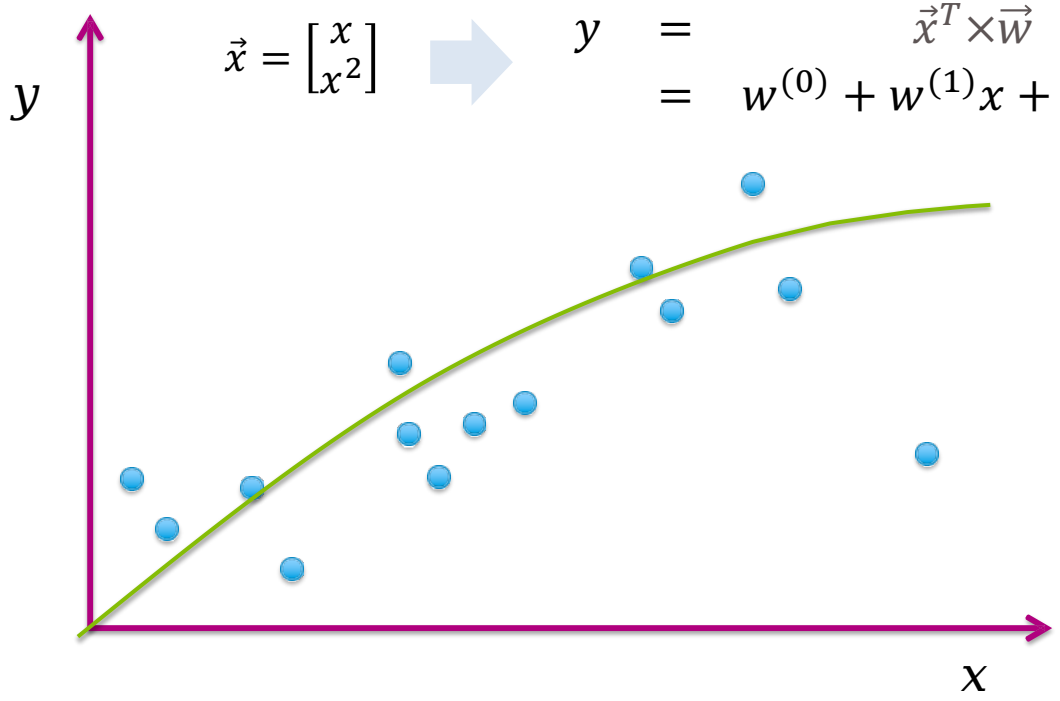
2nd Degree: $\vec{x} = [x \quad x^2]^T$

3rd Degree: $\vec{x} = [x \quad x^2 \quad x^3]^T$

$\vdots$

N^{th} Degree: $\vec{x} = [x \quad \cdots \quad x^N]^T$

$$\vec{x} = \begin{bmatrix} x \\ x^2 \end{bmatrix} \Rightarrow \begin{aligned} y &= \vec{x}^T \times \vec{w} \\ &= w^{(0)} + w^{(1)}x + w^{(2)}x^2 \end{aligned}$$

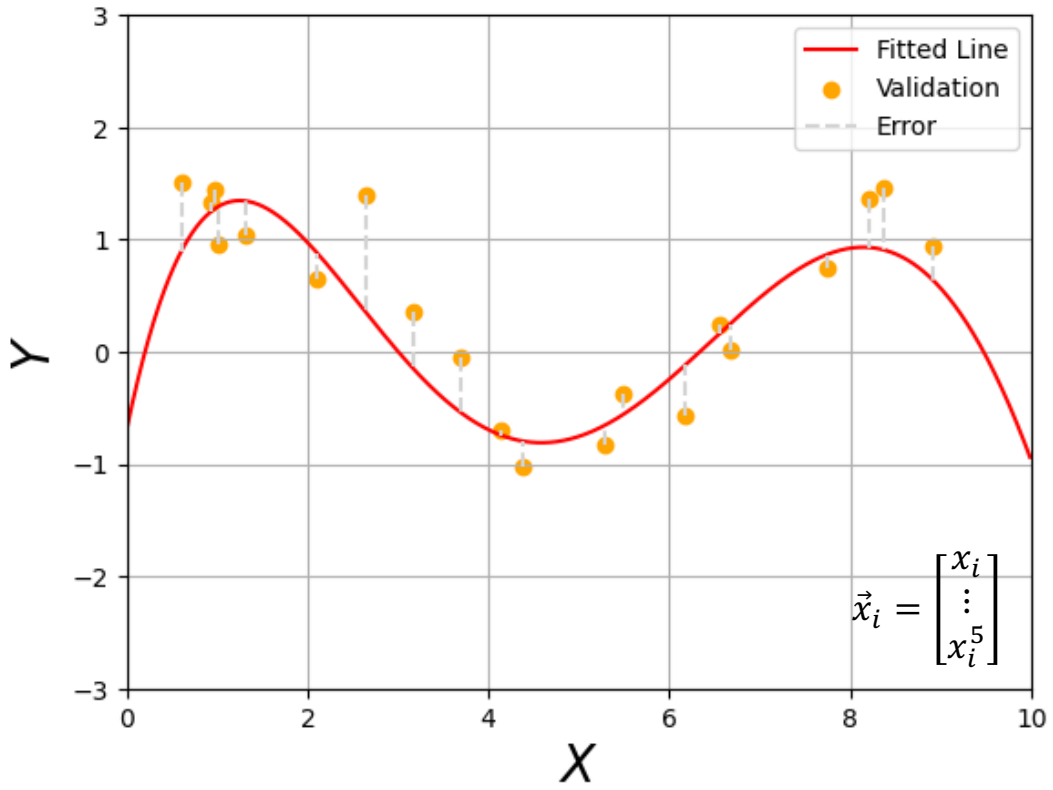


scikit-learn: Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error

# Define polynomial order
order = 5
poly = PolynomialFeatures(degree=order)

# Polynomial featurse, i.e. engineered features
Xp_train = poly.fit_transform(X_train)
Xp_test = poly.fit_transform(X_val)

# Create an instance of LinearRegression
reg = LinearRegression()

# Fit the model on the training data
reg.fit(Xp_train, y_train)

# Predict the target values on the validation set
predictions = reg.predict(Xp_val)

# Calculate the mean squared error of the predictions
mse = mean_squared_error(y_val, predictions)
print("Mean Squared Error of the linear regressor on the validation set: {:.2f}".format(mse))
```


```
Mean Squared Error of the k-NN regressor on the validation set: 0.16
```

Regression Line



$$\text{Error (Residue)} \quad = \quad y - \hat{y}$$

where y and $\hat{y}$ are the observed and predicted values, respectively.

Mean Squared Error (MSE):

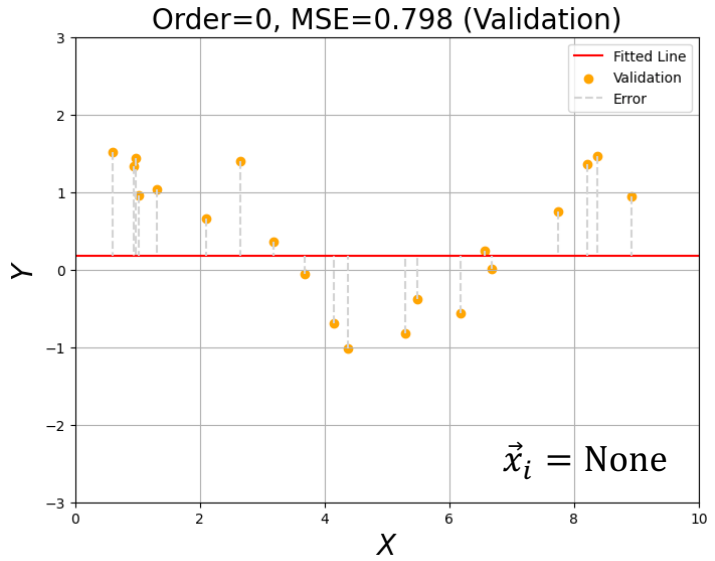
$$\text{MSE} \quad = \quad \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

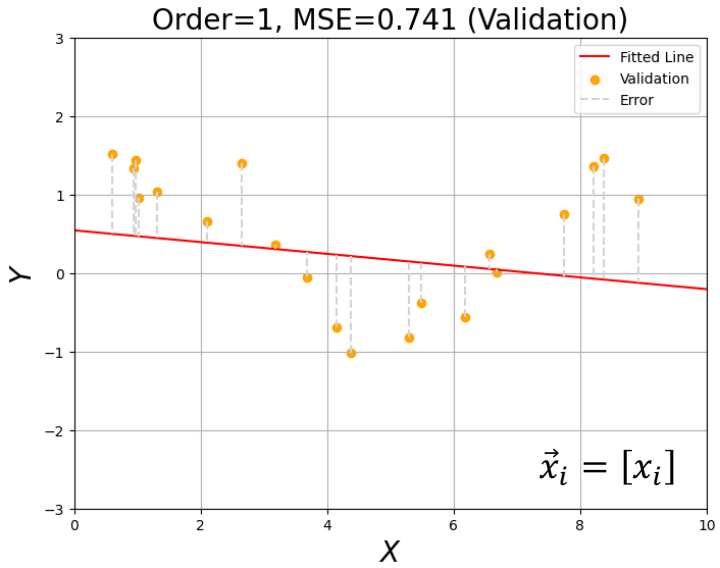
where N is the total number of data points.

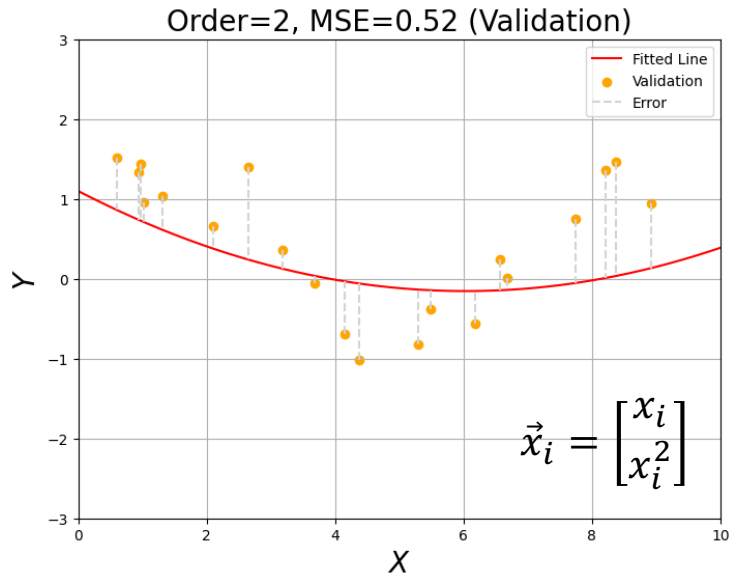
MyLinearRegressor (Revised)

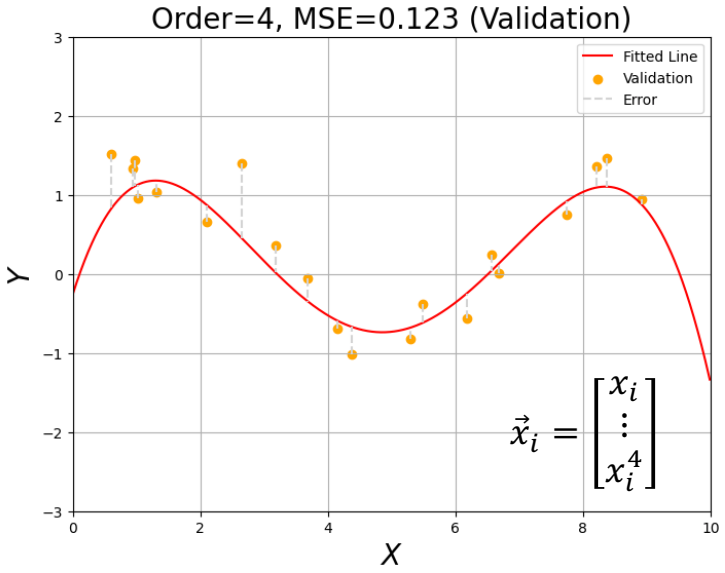
```
from sklearn.base import BaseEstimator, RegressorMixin
import numpy as np

class MyLinearRegressor(BaseEstimator, RegressorMixin):
    def __init__(self):
        pass

    def fit(self, X, y):
        # Add bias (column of ones) to the input data
        X_bias = np.c_[np.ones(X.shape[0]), X]

        # Closed-form solution for linear regression: weights = (X.T @ X)^(-1) @ X.T @ y
        self.weights_ = np.linalg.inv(X_bias.T @ X_bias) @ X_bias.T @ y
        return self

    def predict(self, X):
        # Add bias to the input data (X)
        X_bias = np.c_[np.ones(X.shape[0]), X]

        # Predict using the learned coefficients (weights)
        return X_bias @ self.weights_
```

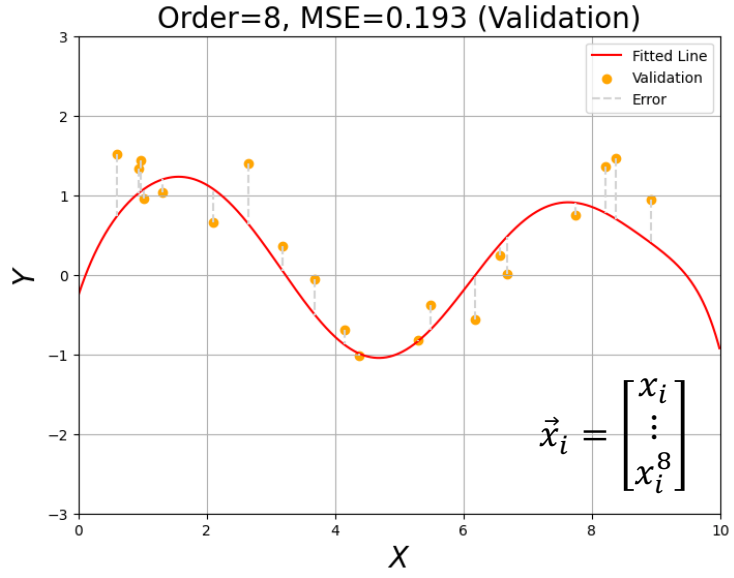
Model Complexity

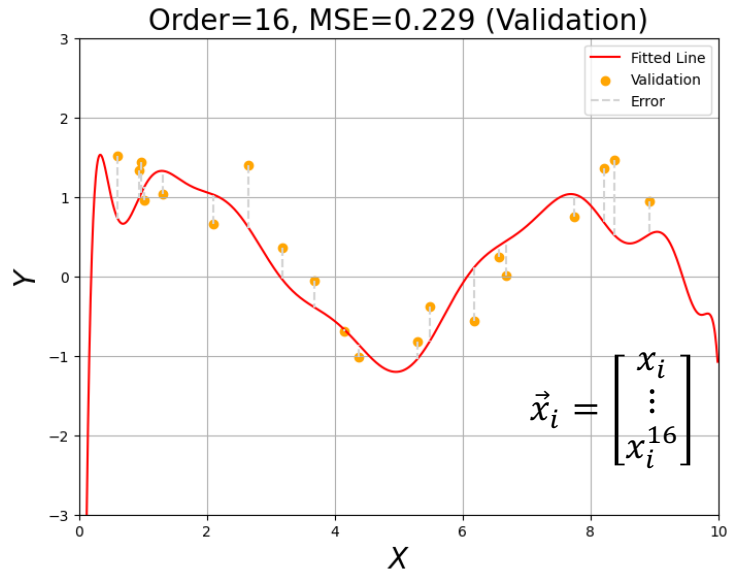




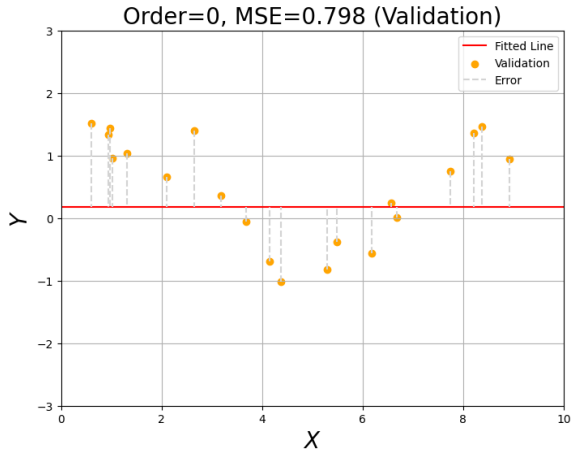




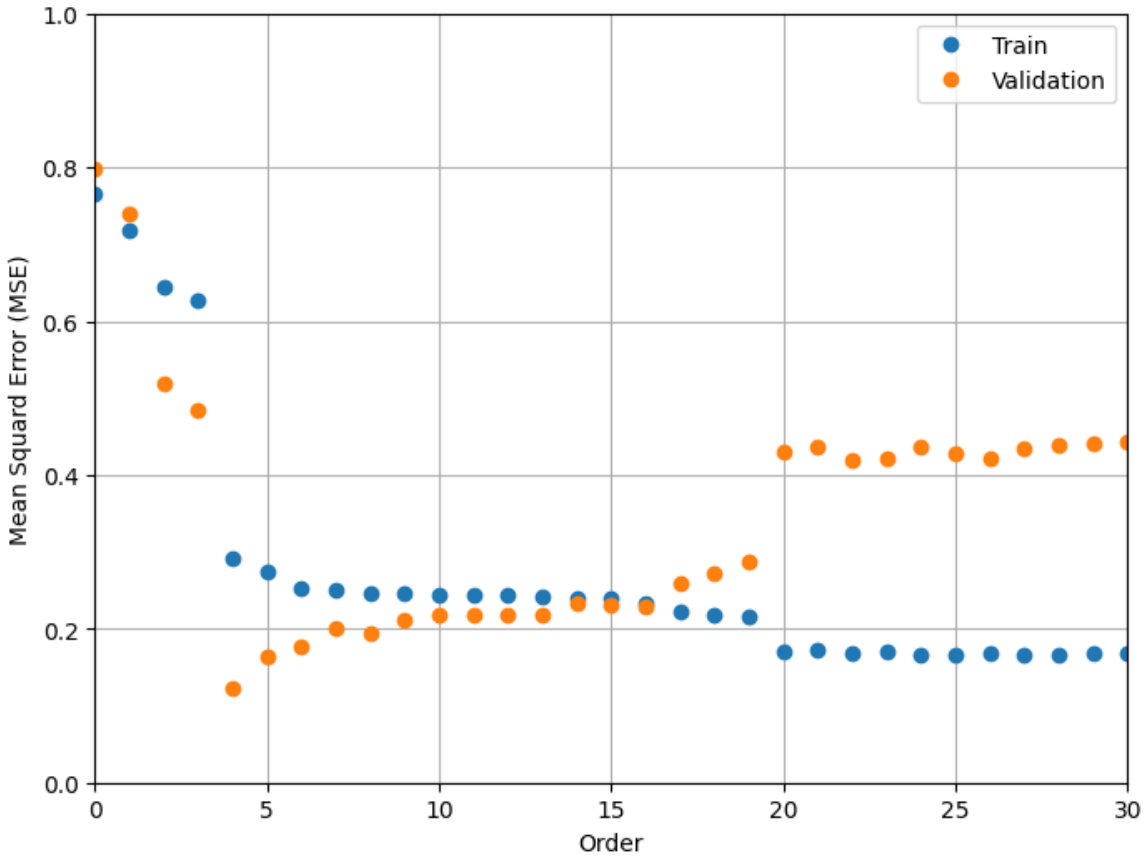


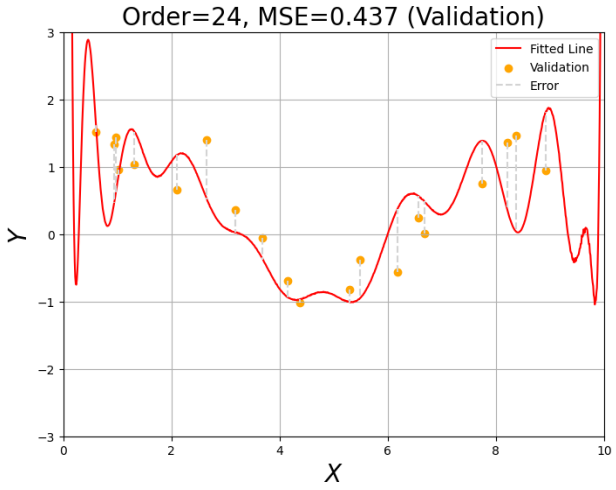


Bias-Variance Trade-Offs



- High Bias, Low Variance
- Simple Fitted Line
- High Train MSE
- High Validation MSE

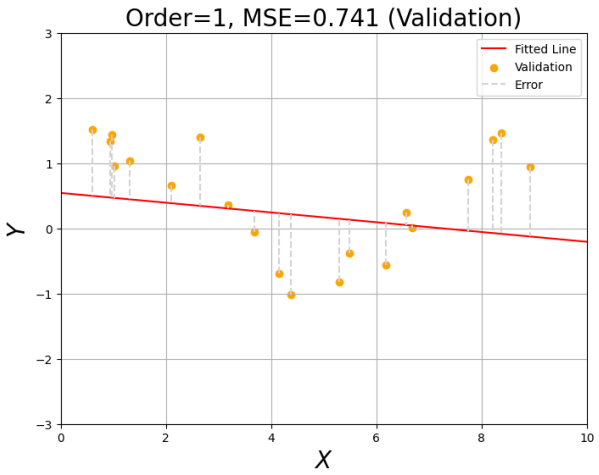




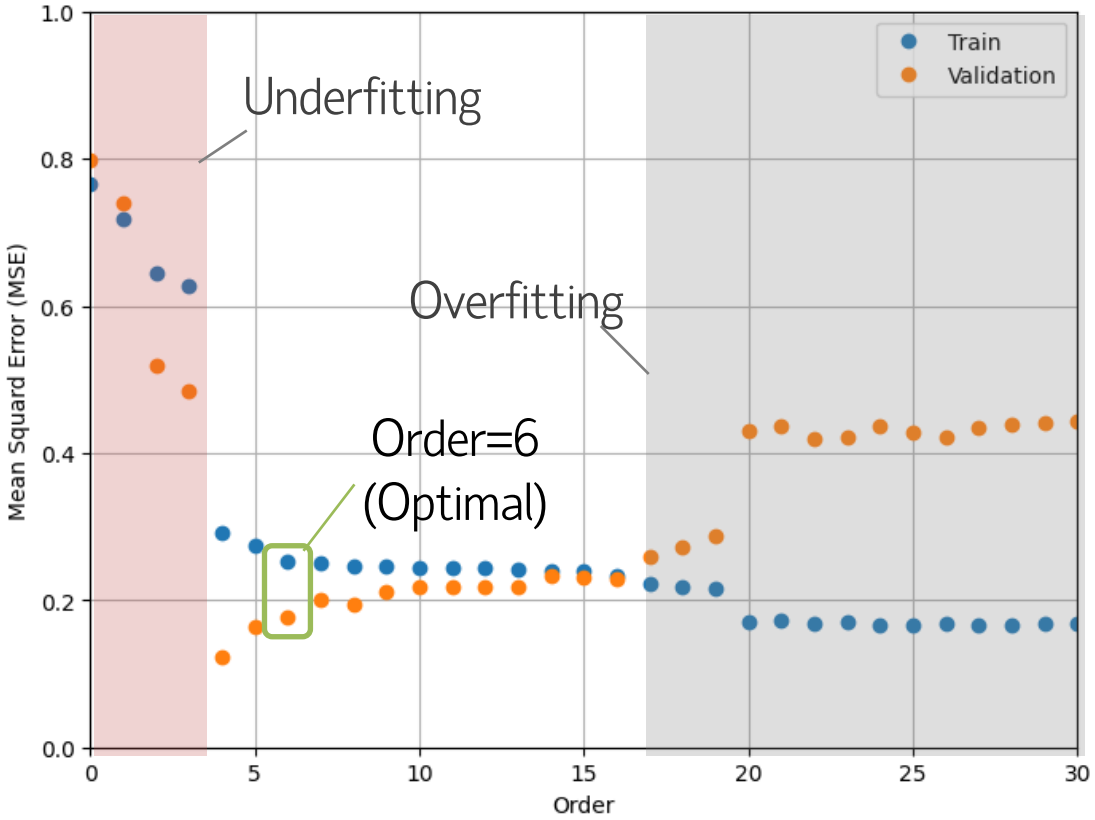
- High Variance, Low Bias
- Complex Fitted Line
- Low Train MSE
- High Validation MSE

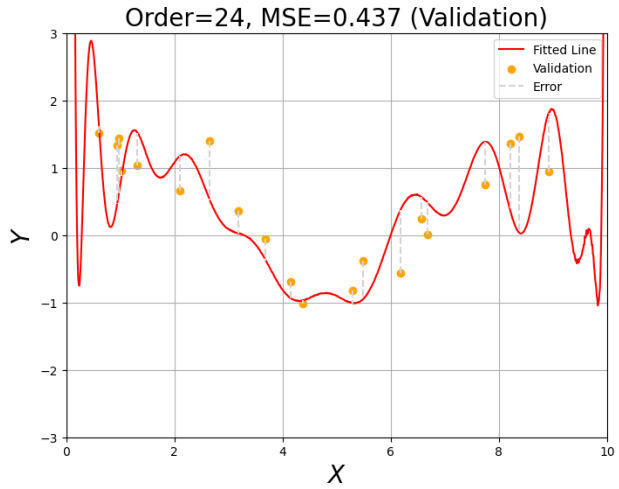
Overfitting vs Underfitting

- **Overfitting** occurs with a small k, where the model captures noise in the training data as if it were significant signal. This results in a model that performs well on the training data but poorly on validation data.
- **Underfitting** arises with a large k, leading to a model that is too generalized. It overlooks important patterns in the data, causing subpar performance on both the training and validation datasets as it fails to capture the underlying data structure.



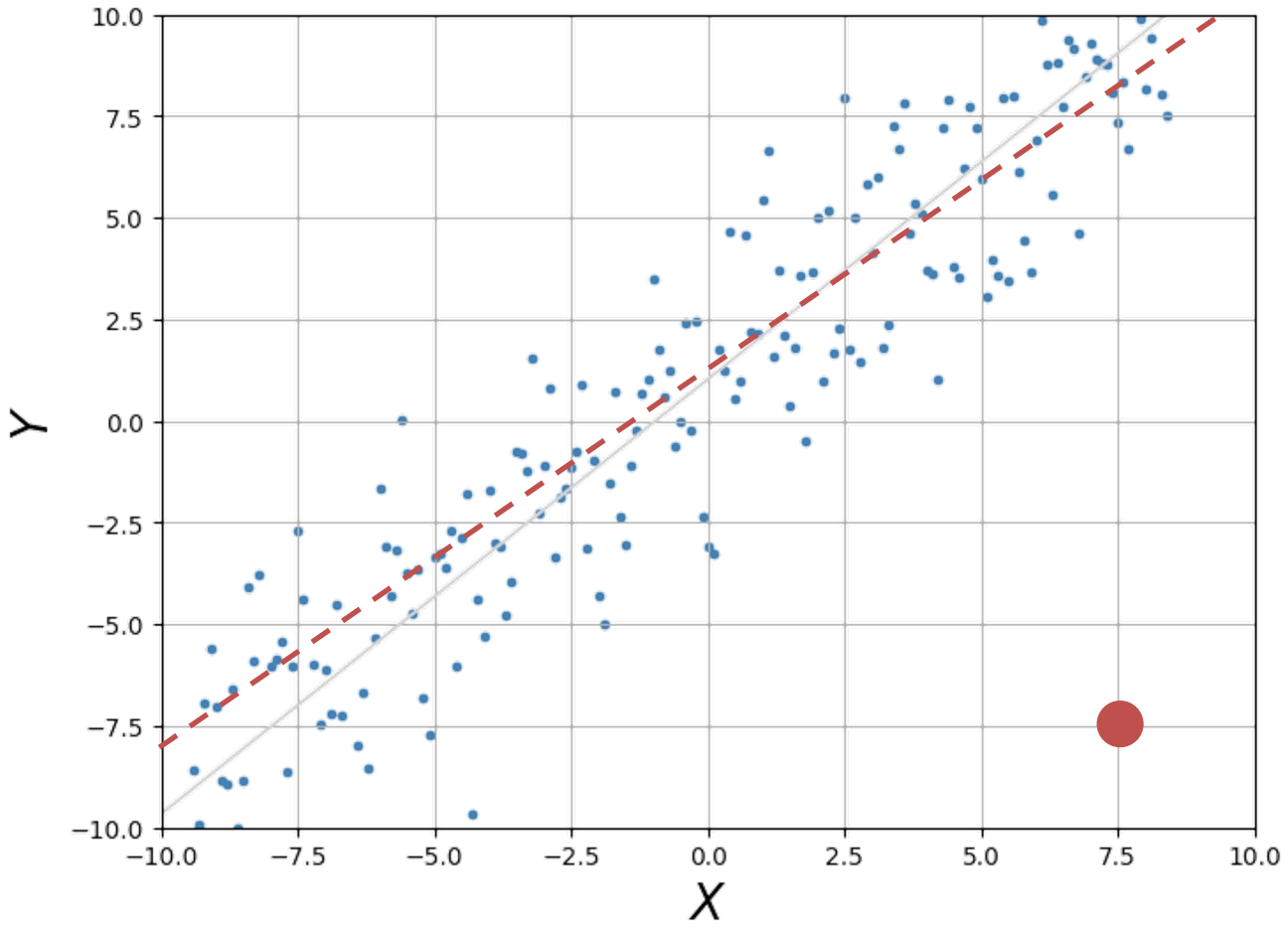
- High Bias, Low Variance
- Simple Fitted Line
- High Train MSE
- High Validation MSE





- High Variance, Low Bias
- Complex Fitted Line
- Low Train MSE
- High Validation MSE

Outliers



- Outliers can significantly distort linear regression by pulling the regression line towards themselves, leading to biased coefficient estimates and reduced model accuracy, as the model tries to fit these extreme points rather than the overall data trend.

Q: What to expect from our ML model?

A: The model must perform well in most of the cases. Hence, outliers must be excluded from the training data.

Measure of Centre

Example

9 students' exam scores: 75, 69, 88, 93, 95, 54, 87, 88, 27

Mean: $\frac{75 + 69 + 88 + 93 + 95 + 54 + 87 + 88 + 27}{9} \approx 75.11$

Mode: 88

Median: 27, 54, 69, 75, 87, 88, 88, 93, 95

Mean

Arithmetic Average
$\bar{x}$ (or μ) – Sample Mean

Median

Midpoint of the Distribution
(50th percentile)

Mode

Most Frequent Observation

Median Example

10 students' exam scores:

27, 54, 69, 75, 87, 88, 88, 93, 95, 100

$$\frac{87 + 88}{2} = 87.5$$

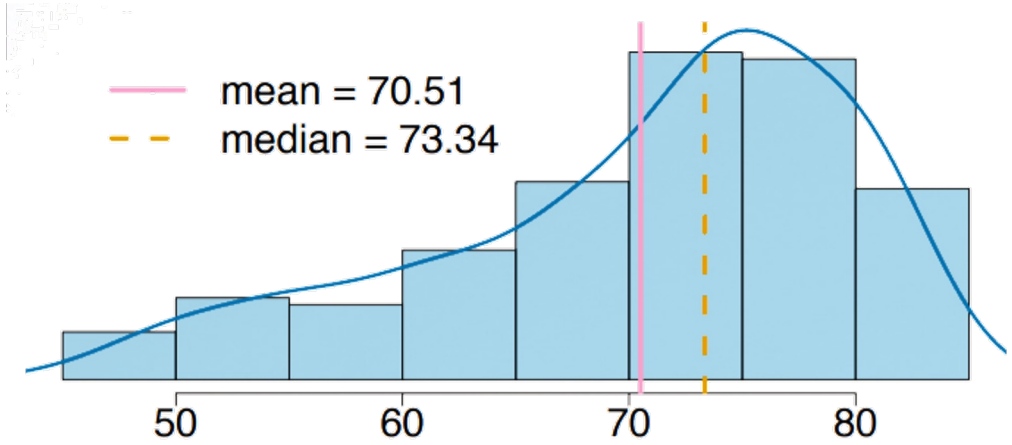
Median

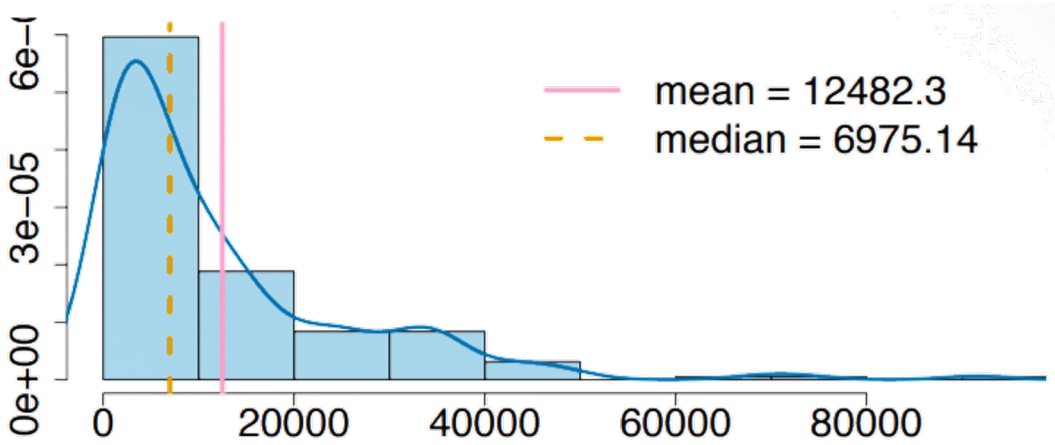
Midpoint of the Distribution (50^{th} percentile)

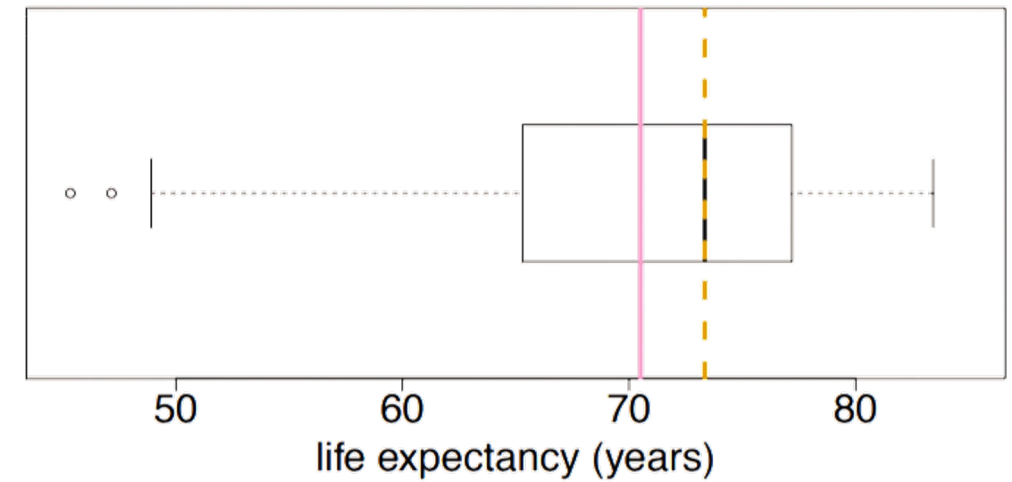
Gapminder Dataset

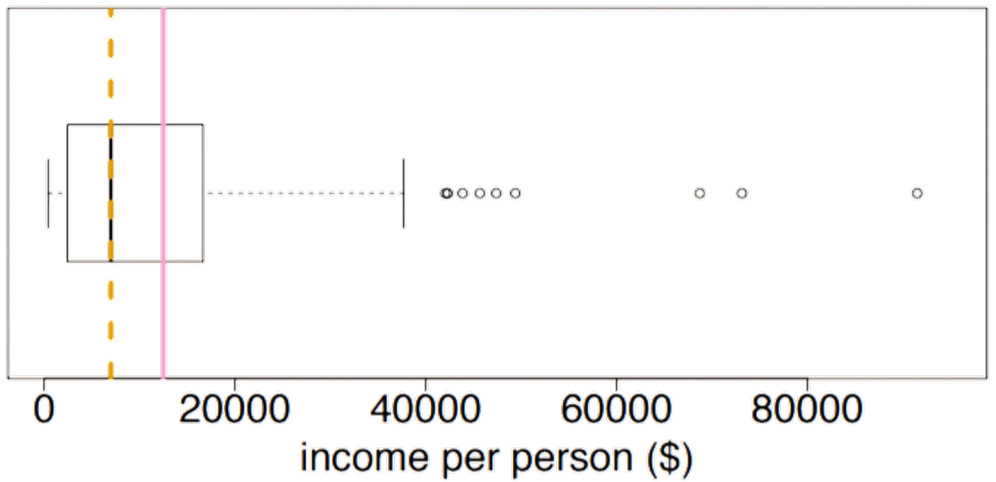
data	income per person ($, 2012)	life expectancy (years, 2012)
Afghanistan	1359.7	60.254
Albania	6969.3	77.185
Algeria	6419.1	70.874
…	…	…
Zimbabwe	545.3	58.142

Gapminder: Measure of Centre



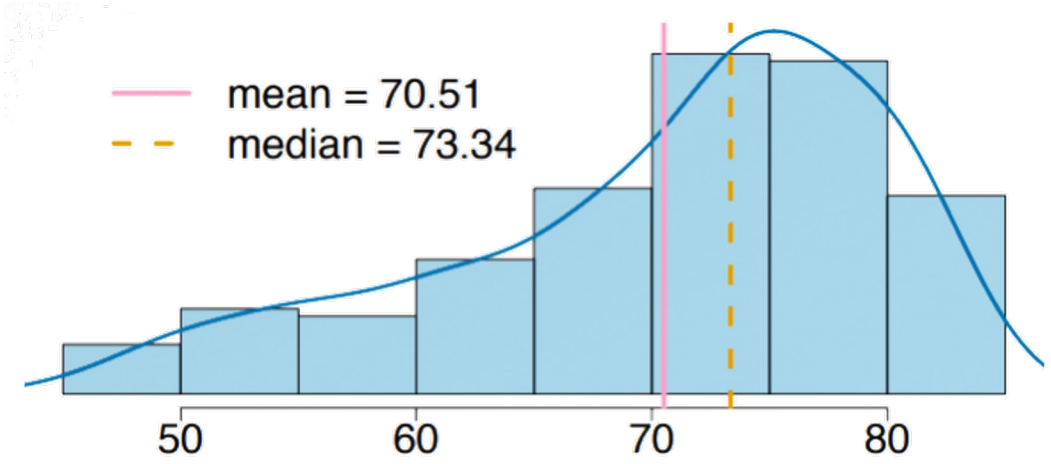




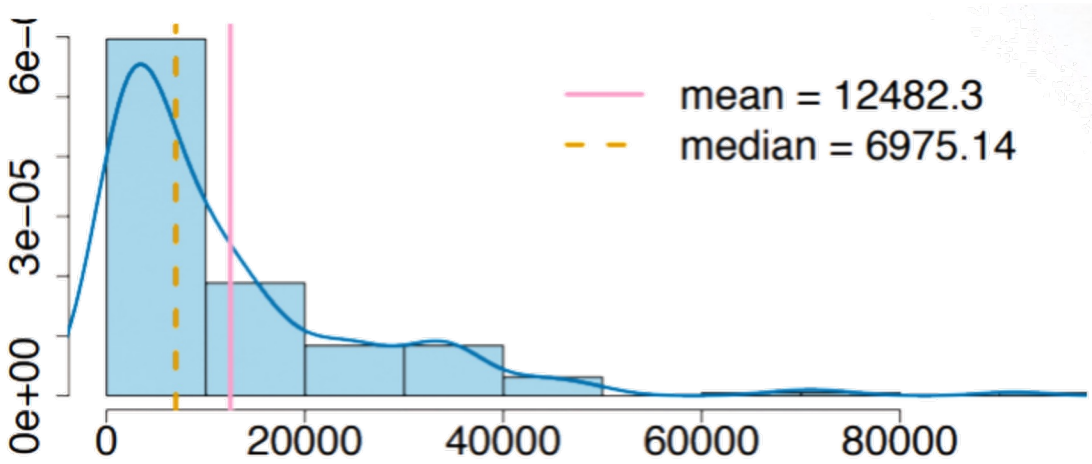


Skewness and Measures of Centre

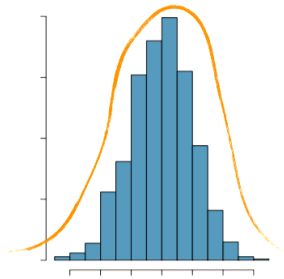
Left Skewed: Mean < Median



Right Skewed: Mean > Median



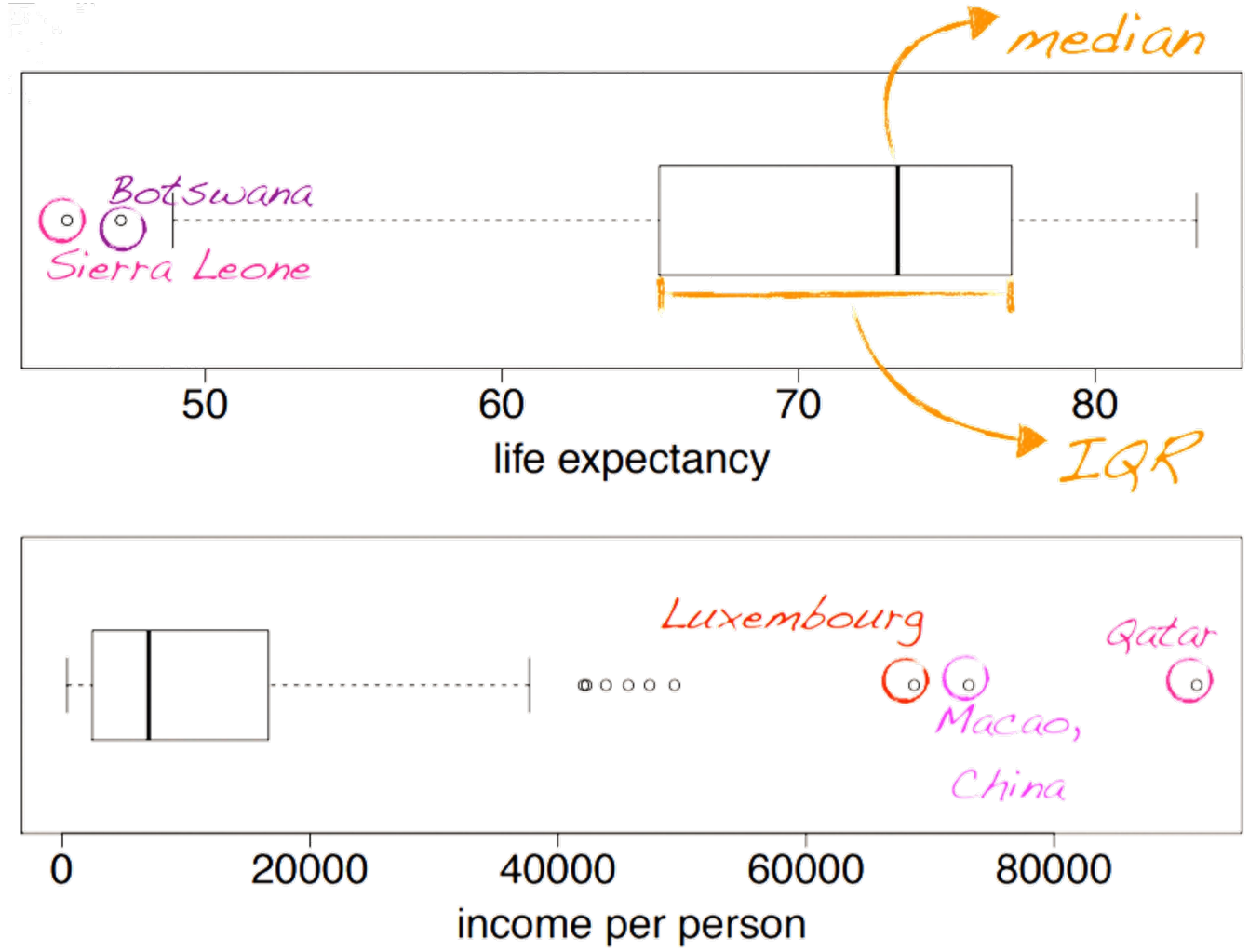
How about Symmetric?

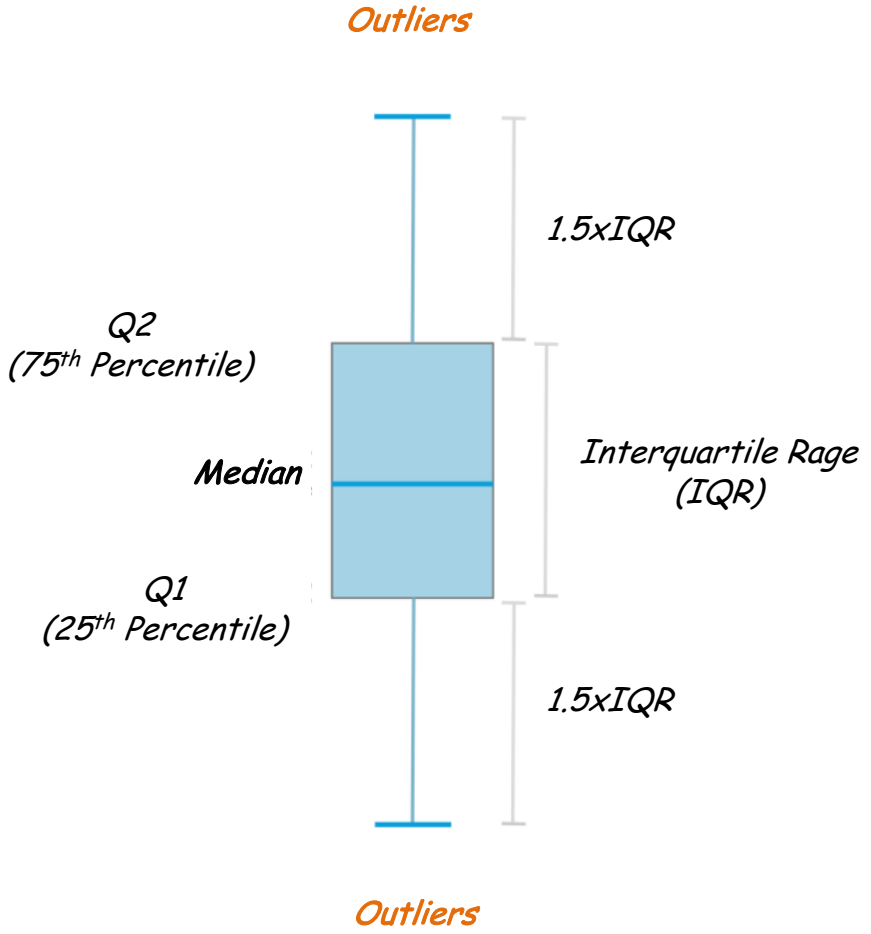


Mean ≈ Median

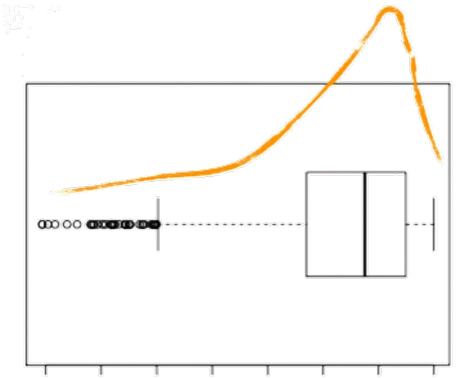
Box Plot

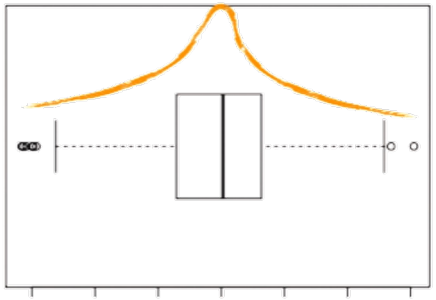
Box plot is useful for highlighting outliers, median and IQR.

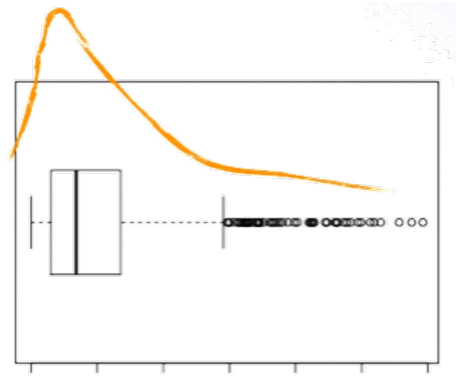


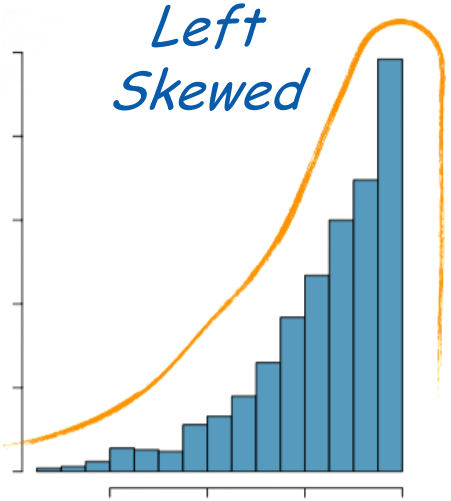


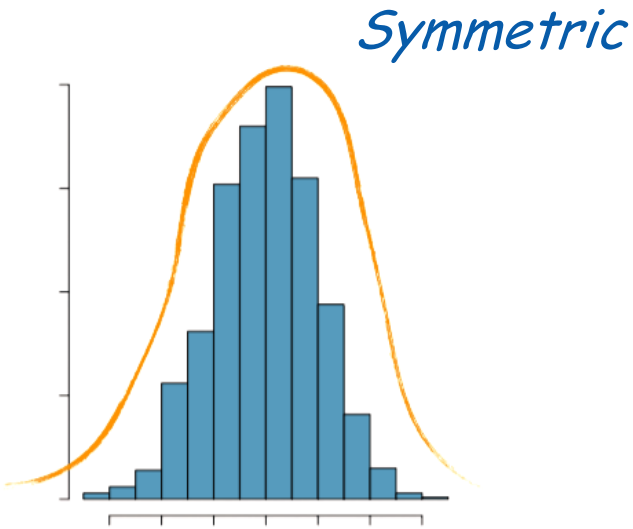
Skewness and Box Plot

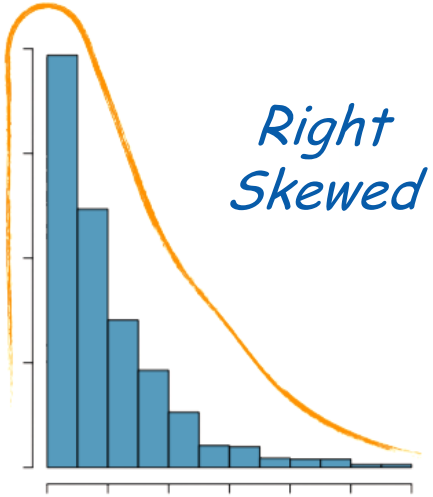






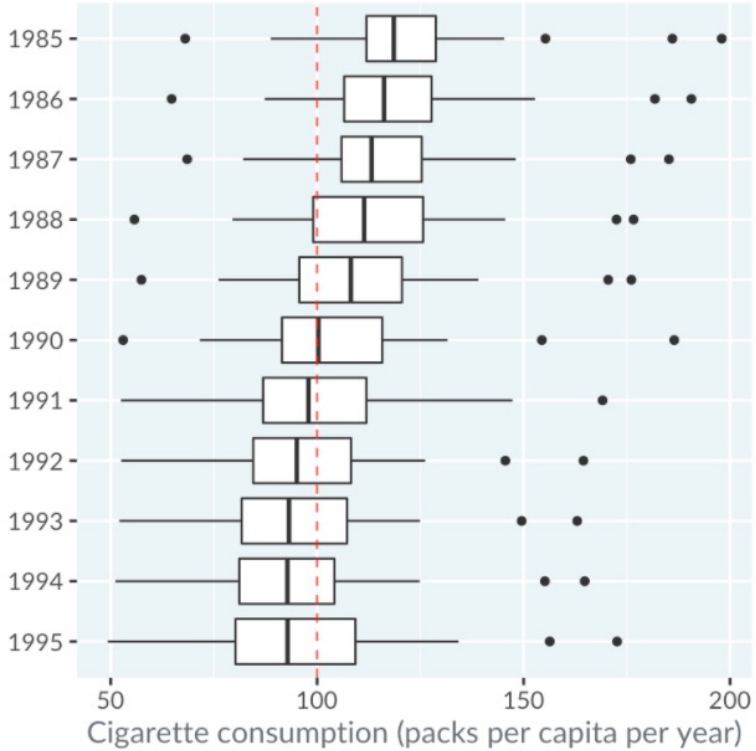






Interpreting Box Plots

Here are box plots of cigarette consumption per person in the USA from 1985 to 1995 (Alaska and Hawaii are not included). Each observation in the dataset is the average number of packets of cigarette smoked per person in one state in one year. Thus each box plot represents the distribution of 48 data points (because there are 48 US states included in the dataset).



Instruction

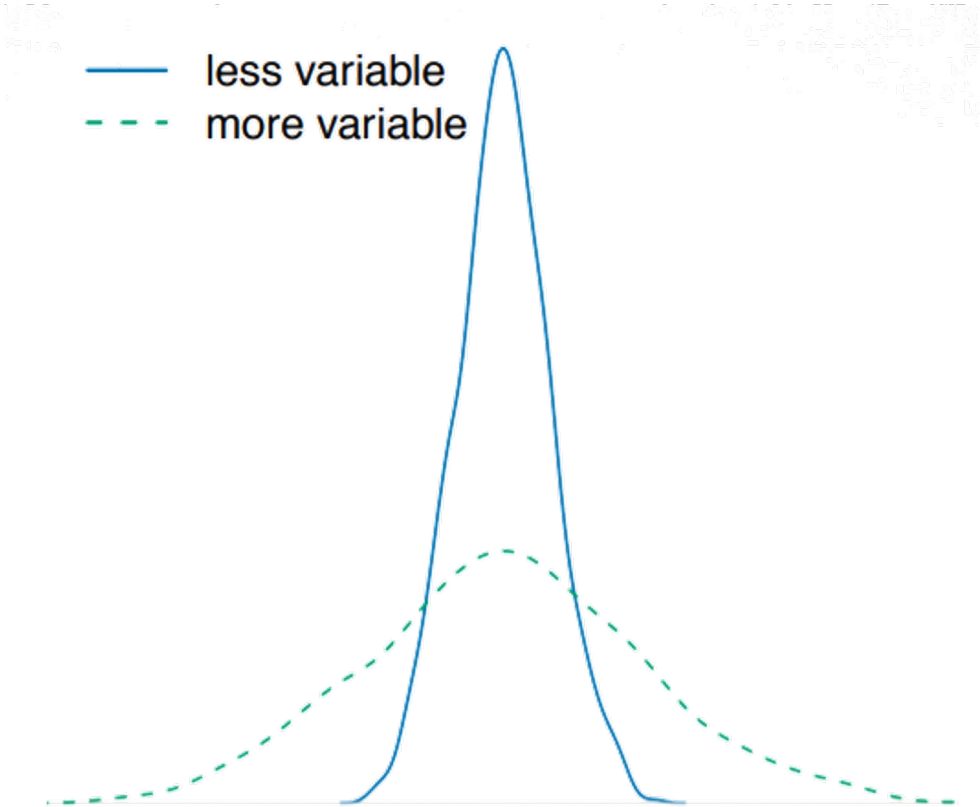
- Categorize these statements about the box plots as true or false.

Drag the Items into the Correct Bucket

True	False
In 1990, three states were considered to have extreme values in the number of packets of cigarettes smoked per capita.	The inter-quartile range of the number of packets of cigarettes smoked per capita decreased every year from 1985 to 1995.
The median number of packets of cigarettes smoked per capita was below 100 from 1991 onwards.	The upper quartile number of packets of cigarettes smoked per capita decreased every year from 1985 to 1995.
The lower quartile number of packets of cigarettes smoked per capita decreased every year from 1985 to 1995.	The inter-quartile range of the number of packets of cigarettes smoked per capita was smallest in 1992.

Measure of Spread

- Range: (*Max* – *Min*)
- Variance
- Standard Deviation
- Inter-Quartile Range



Variance

Variance (σ^2) is roughly the average squared deviation from the mean:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

Example

Given that the average life expectancy is 70.5, and there are 201 countries in the dataset:

$$\sigma^2 = \frac{(60.3 - 70.5)^2 + (77.2 - 70.5)^2 + \ldots + (58.1 - 70.5)^2}{201}$$

$$= 87.5 \text{ years}^2$$

	country	life exp
1	Afghanistan	60.3
2	Albania	77.2
3	Algeria	70.9
	…	…
201	Zimbabwe	58.1

Squared Distance

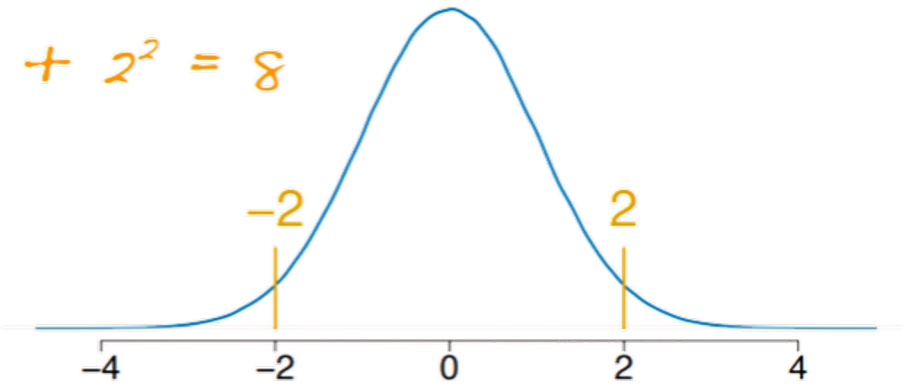
Q: Why do we square the difference?

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

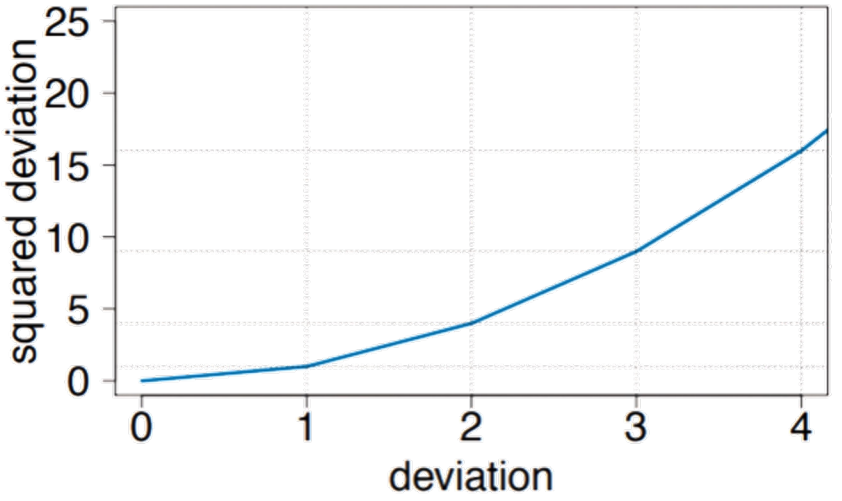
A: It is to get rid of negatives so that negatives and positives don't cancel each other when added together. do we square the difference?

$(-2) + 2 = 0$

$(-2)^2 + 2^2 = 8$

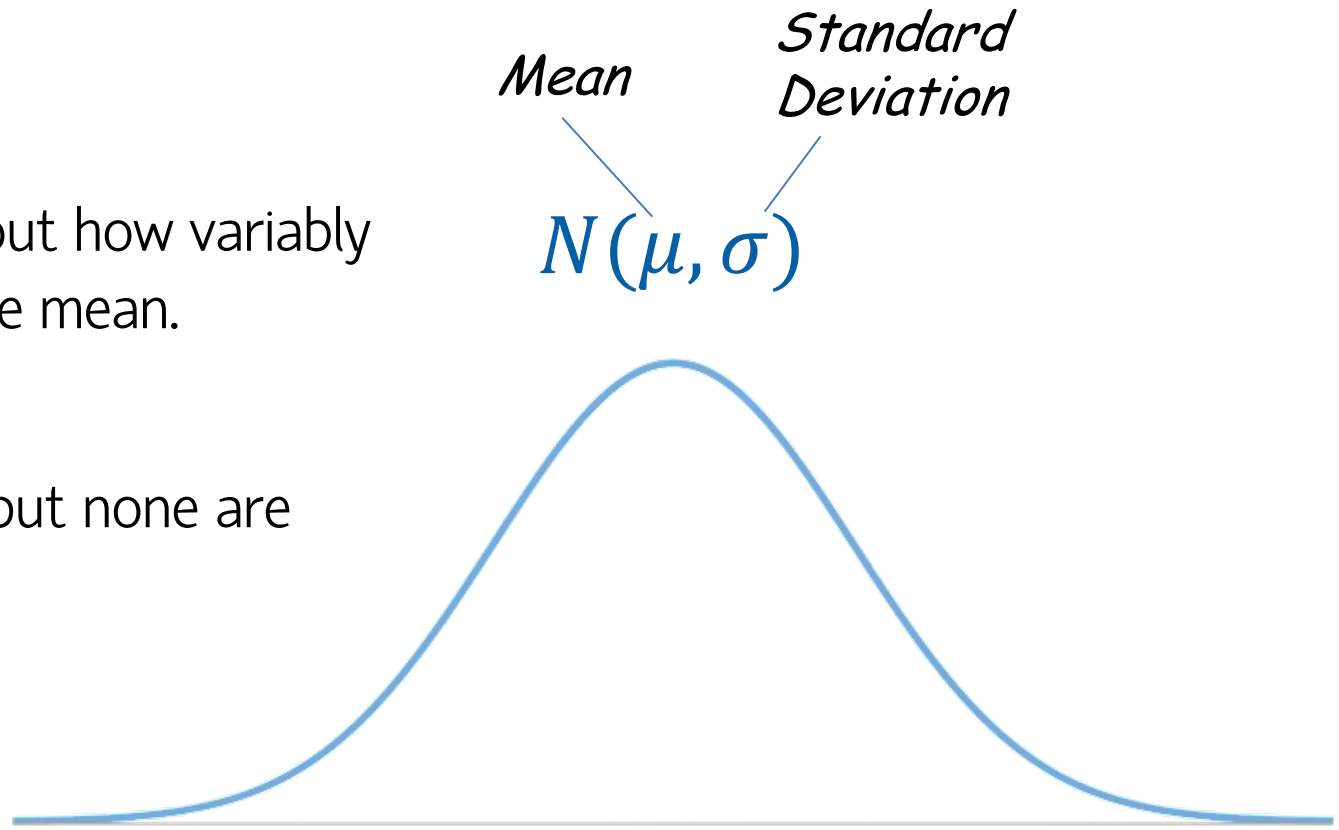


A: The larger deviations are weighed more heavily than the smaller ones.

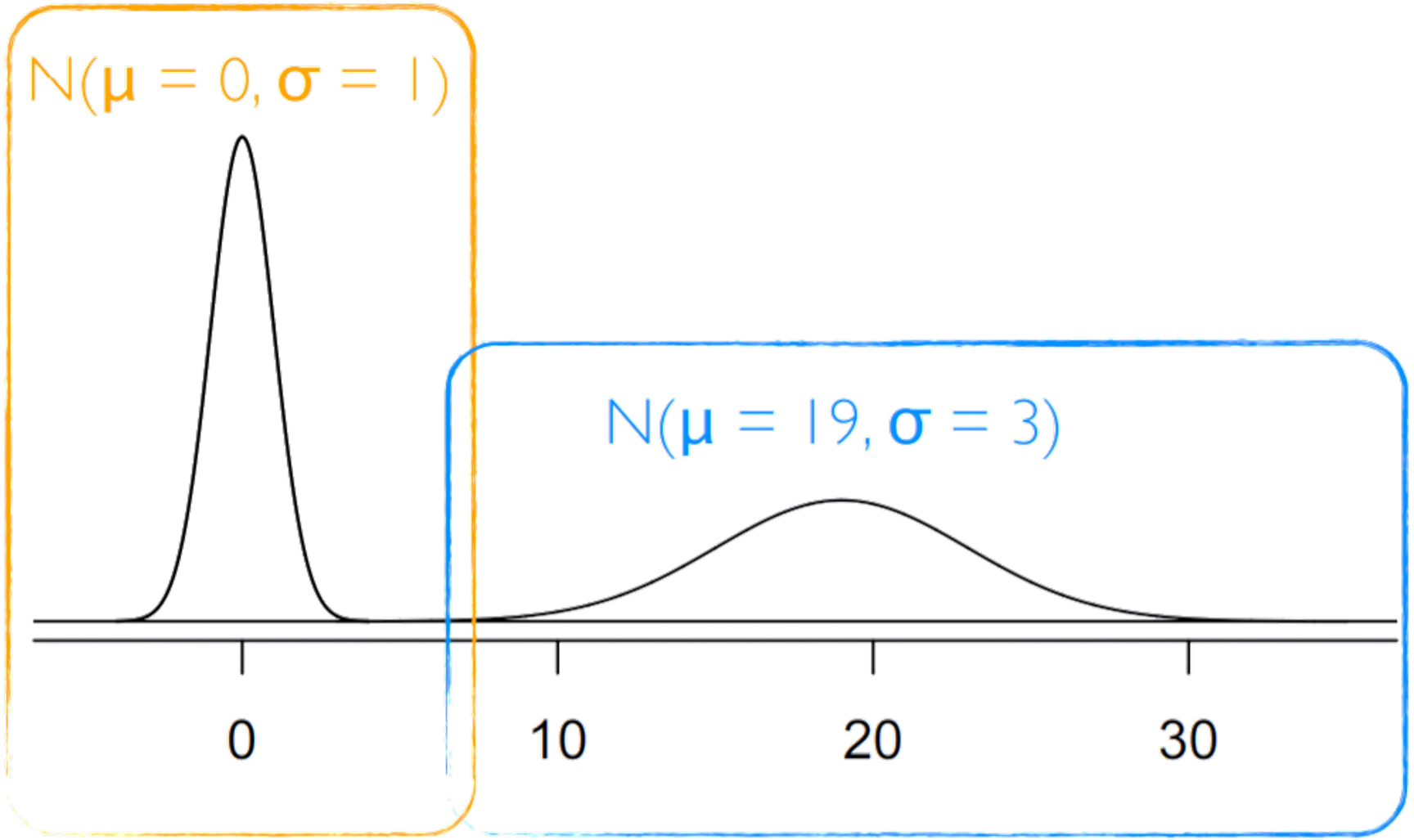


Normal Distribution

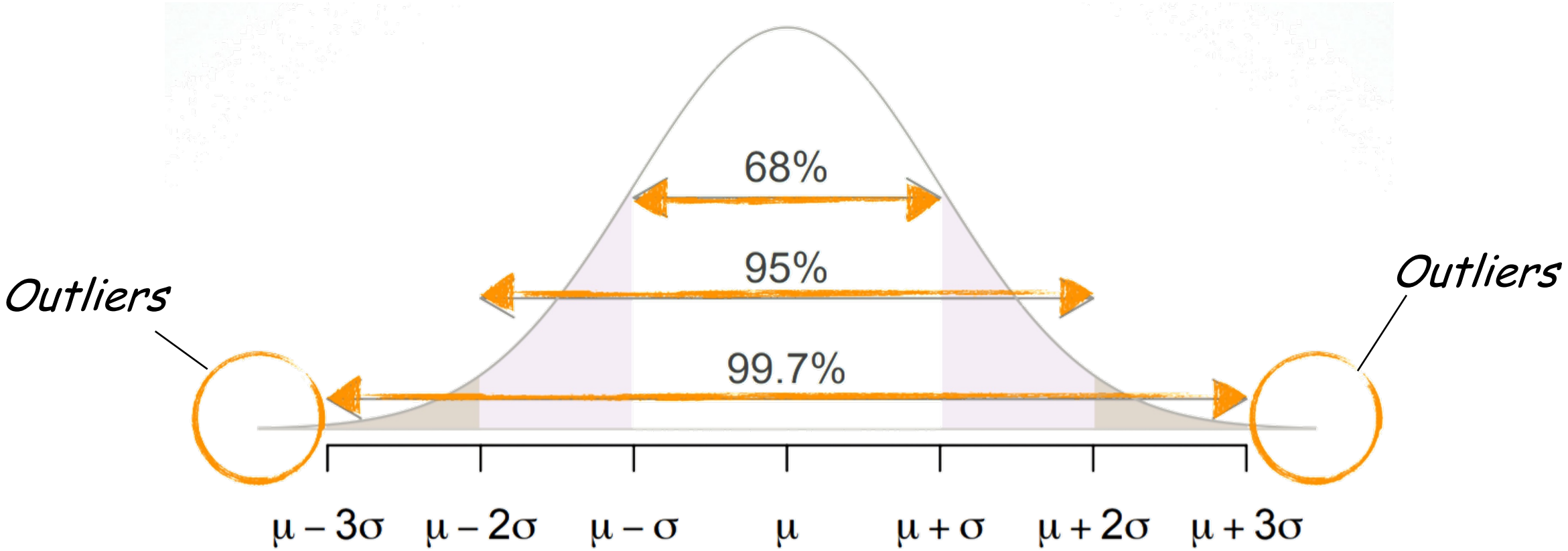
- Unimodal and Symmetric
  - Bell Curve
- It follows very strict guidelines about how variably the data are distributed around the mean.
- Many variables are nearly normal, but none are exactly normal.



Normal Distribution (cont.)



68 – 95 – 99.7% Rule



Example: 99.7% Rule

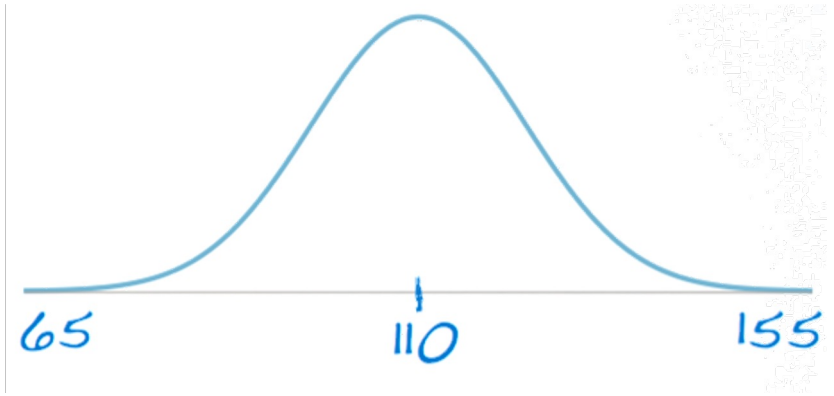
A doctor collects a large set of heart rate measurements that approximately follow a normal distribution. He only reports 3 statistics, the mean = 110 beats per minute, the minimum = 65 beats per minute, and the maximum = 155 beats per minute. Which of the following is most likely to be the standard deviation of the distribution?

A. 5 $110 \pm (3 \times 5) = (95, 125)$

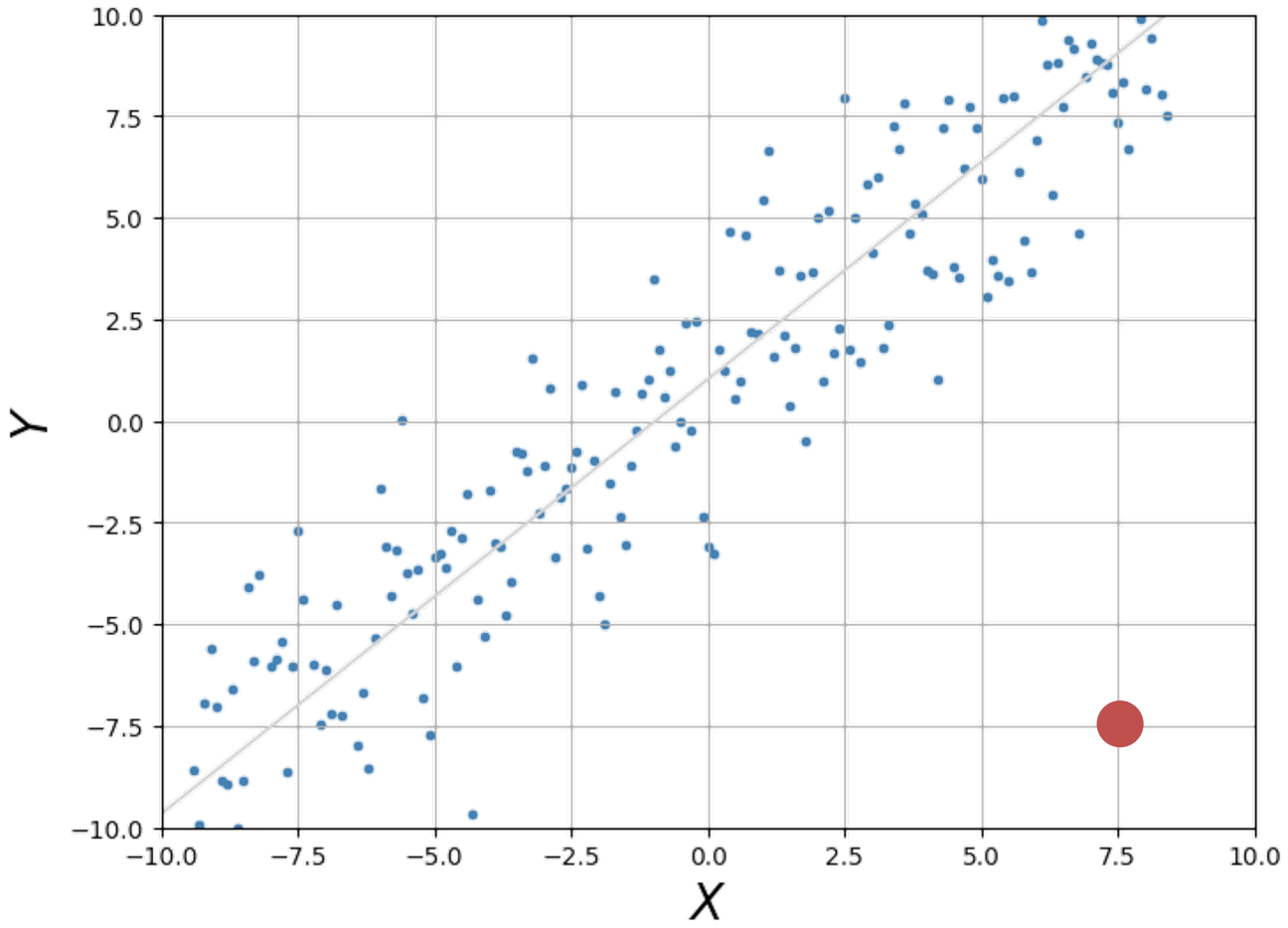
B. 15 $110 \pm (3 \times 15) = (65, 155)$

C. 35 $110 \pm (3 \times 35) = (5, 215)$

D. 90 $110 \pm (3 \times 90) = (160, 380)$



Outliers: 1.5×IQR and 3σ Rules



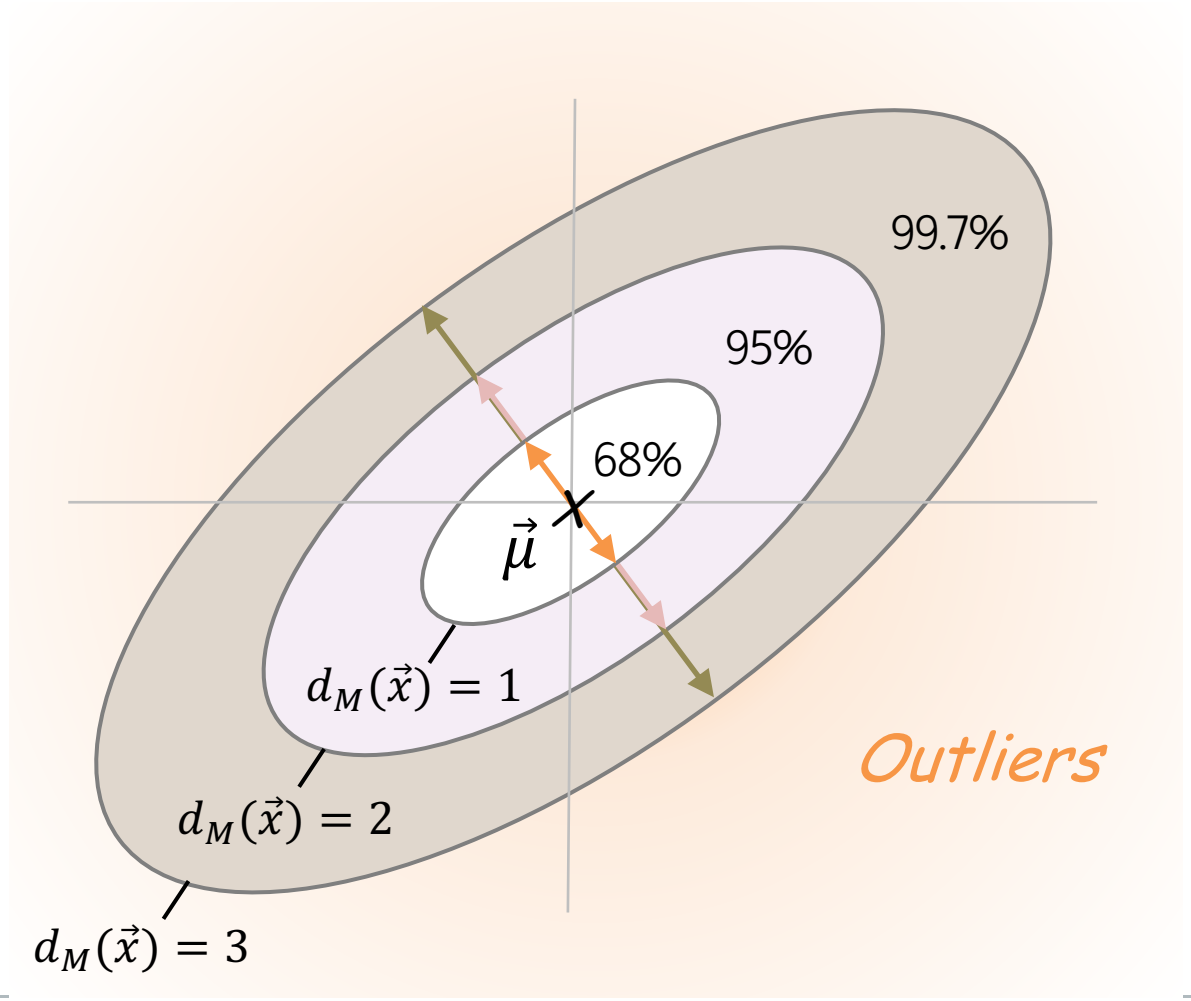
Q: Is ● an outlier?

A: No. ● is within 1.5xIQR rule and also ±3σ range along the X axis. Similarly for Y dimension.

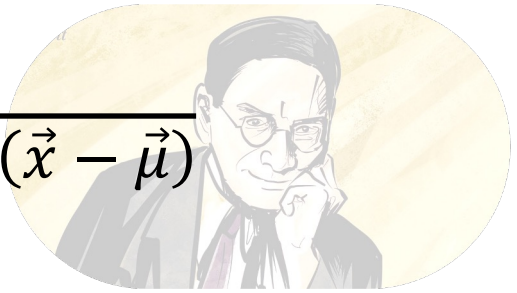
Eh? This's not quite right. 1.5xIQR and 3σ rules are univariate, meaning we are only considering one dimension at a time.

Mahalanobis Distance

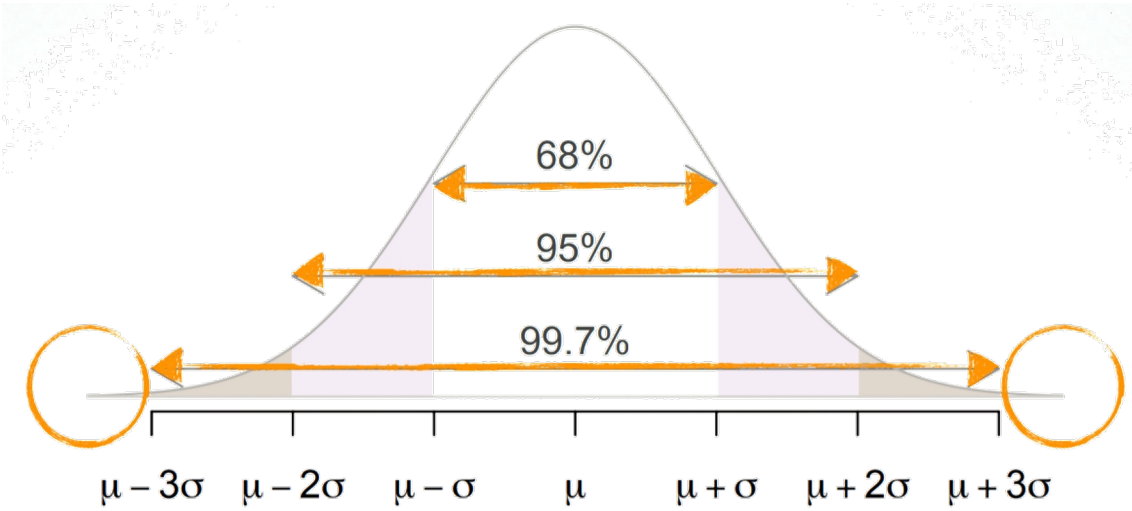
Think of *Mahalanobis* distance as an extension of the 1D concept of *distance from the mean in terms of standard deviation,* but adapted to account for multiple directions in multi-dimensional data.



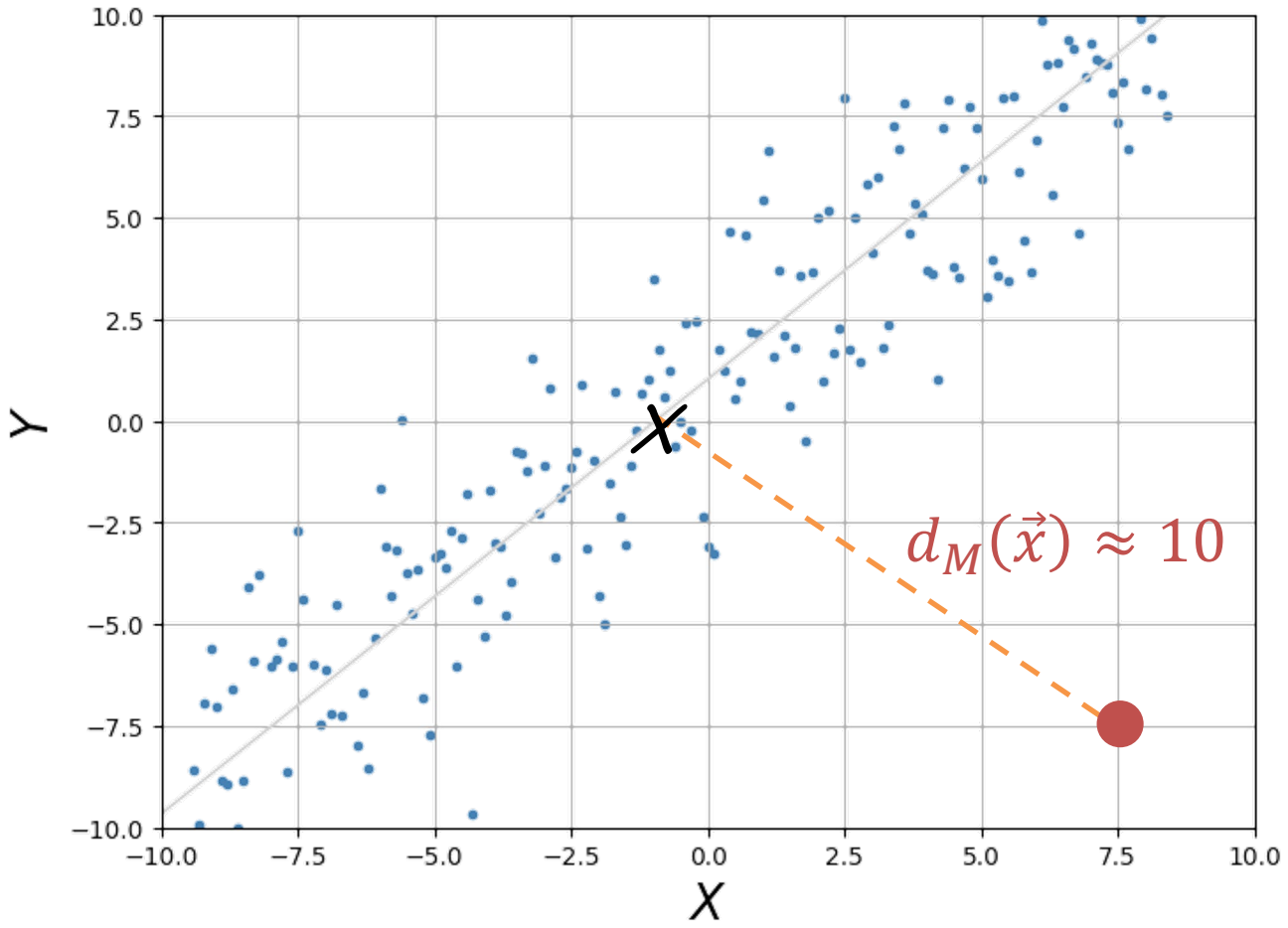
$$d_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})}$$



Covariance Matrix
(It is like σ but for $\geq 2D$.)



Outliers: Mahalanobis Distance



Q: Is ● an outlier?

A: Yes, it is now. $d_M(\vec{x}) > 3$.

California Housing Dataset

- California Housing Dataset: There are 8 features, which include median income, housing age, average rooms and bedrooms per household, block population, average occupancy, and geographical coordinates (latitude and longitude) of the block group.

$$\vec{w}^T = \begin{bmatrix} 4.33\times10^{-1} & (\$100\text{k}/\$1\text{k}) \\ 9.29\times10^{-3} & (\$100\text{k}/\text{Year}) \\ -9.86\times10^{-2} & (\$100\text{k}/\#\ \text{Rooms}) \\ 5.93\times10^{-1} & (\$100\text{k}/\#\ \text{Bedooms}) \\ -7.56\times10^{-6} & (\$100\text{k}/\text{Person}) \\ -4.74\times10^{-3} & (\$100\text{k}/\text{Person}) \\ -4.21\times10^{-1} & (\$100\text{k}/\text{Degree}) \\ -4.34\times10^{-1} & (\$100\text{k}/\text{Degree}) \end{bmatrix}$$

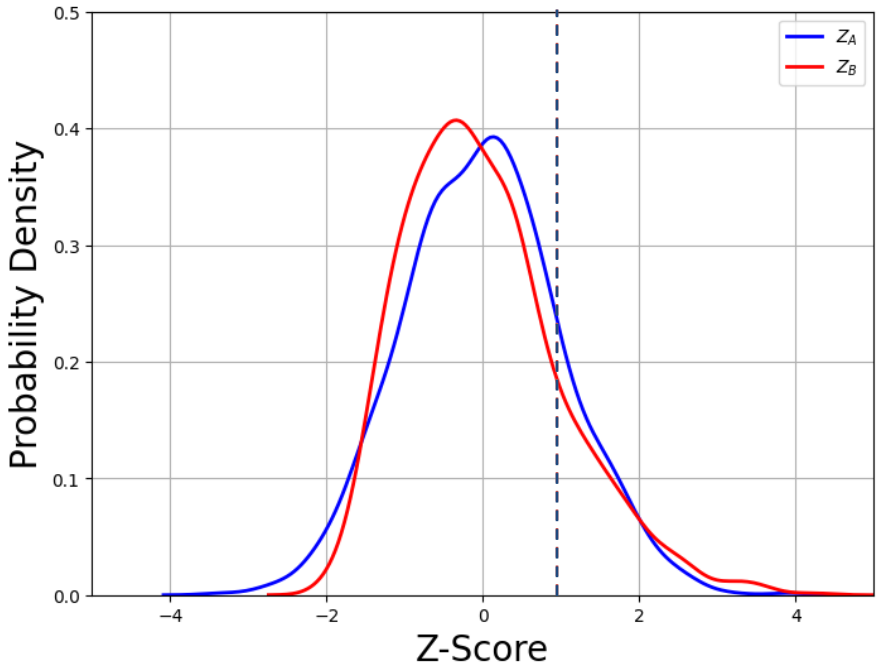
(Excl. Intercept Term)

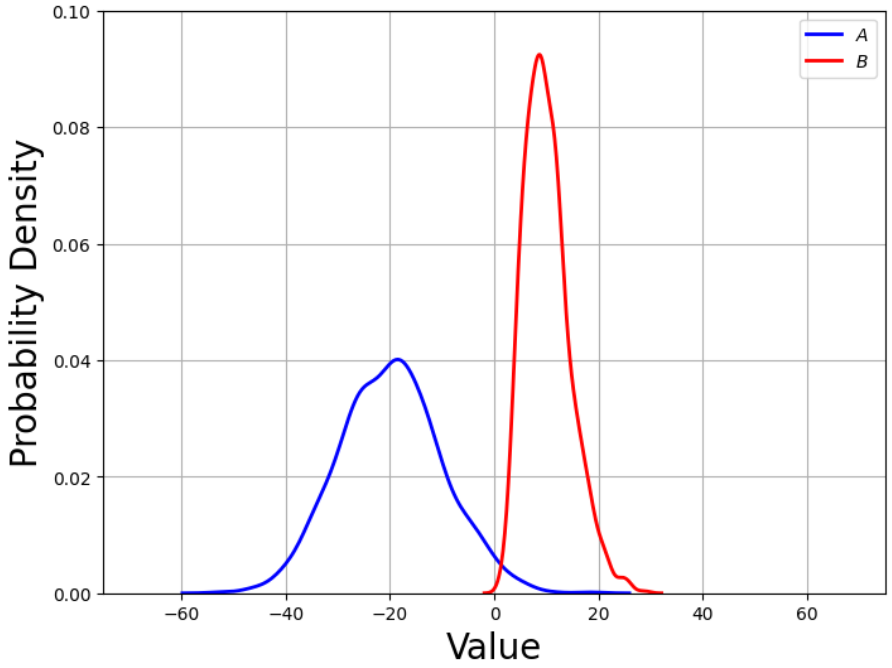
Q: How importance are the features comparing to each other?

A: We cannot tell since the weights are in different units and scales.

Standardising with Z Scores

$$Z = \frac{x - \mu}{\sigma}$$





We need an apple-to-apple comparison.

California Housing Dataset: Model Interpretation

- Standardisation ensures features with different units and scales are in the same scale and unit. This allows interpretation of model coefficients, in other words standardization enables features of importance>
- 8 features are median income, housing age, average rooms and bedrooms per household, block population, average occupancy, and geographical coordinates (latitude and longitude) of the block group.

$$\vec{w}^T = \begin{bmatrix} 0.826 & (\$100\text{k}) \\ 0.117 & (\$100\text{k}) \\ -0.248 & (\$100\text{k}) \\ 0.290 & (\$100\text{k}) \\ -0.008 & (\$100\text{k}) \\ -0.030 & (\$100\text{k}) \\ -0.900 & (\$100\text{k}) \\ -0.870 & (\$100\text{k}) \end{bmatrix}$$

(Excl. Intercept Term)

Q: How do we interpret these features?

A: Location is the most importance and then followed by the income.

Q: What are -0.900 and -0.870 telling us?

A: The areas towards the south and coastline are more expensive.

Summary

- Linear regression models the relationship between a dependent variable and one or more independent variables by fitting a linear equation. The model predicts the dependent variable using a weighted sum of the independent variables (features).
- The closed-form solution for linear regression, also known as the Normal Equation, is derived by setting the gradient of the error function (sum of squared errors) to zero. This results in the formula $\vec{w} = (X^T X)^{-1} X^T Y$, which directly calculates the optimal parameters that minimize error.
- Outliers can significantly distort linear regression models as they exert undue influence on the fitted line, often leading to skewed predictions and biased estimates. Detecting and handling outliers is critical for robust model performance.
- Model interpretation is enhanced through the clear relationship between inputs and outputs, making linear regression easy to interpret, especially when feature scaling is applied. Feature scaling ensures that all features contribute equally to the model's predictions, improving performance and interpretability when features vary significantly in scale.