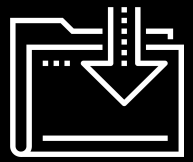
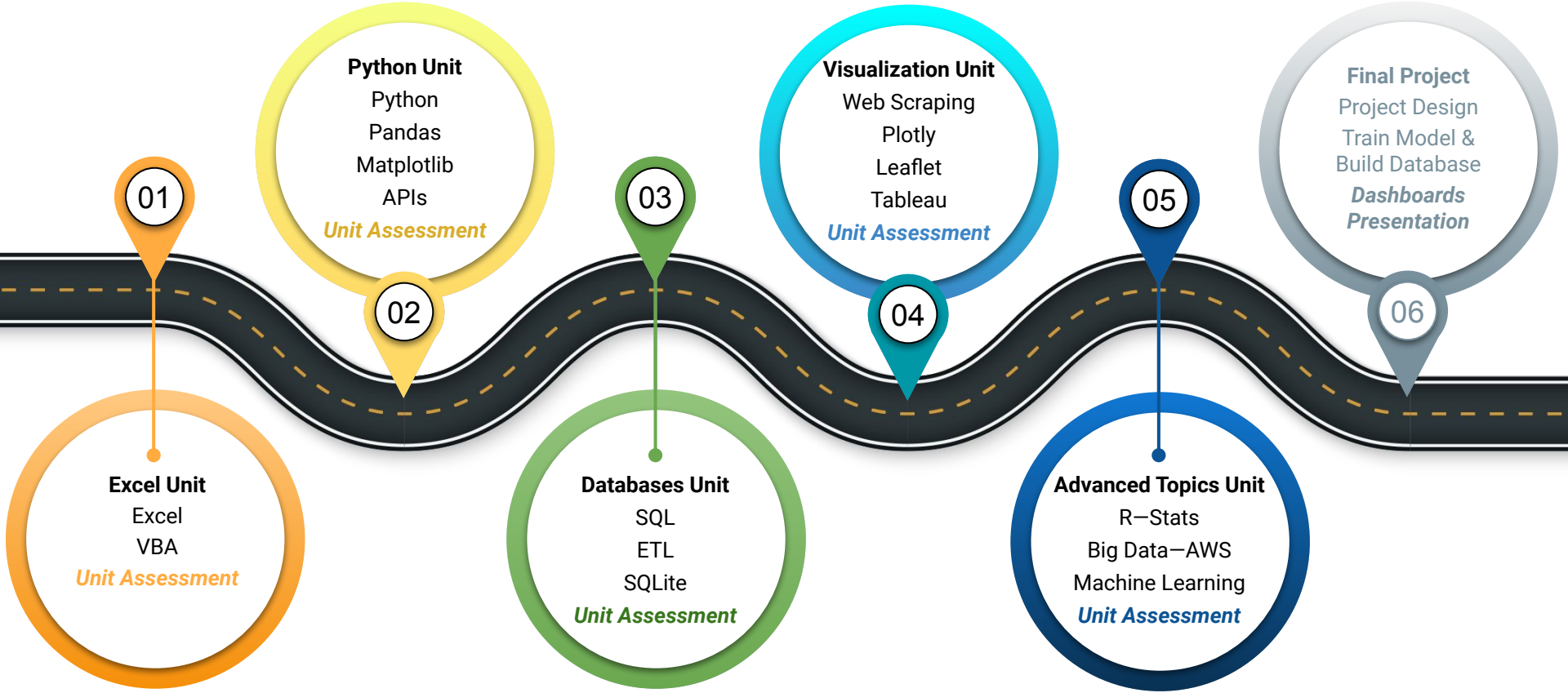
# Introduction to Supervised Machine Learning

Data Boot Camp

Lesson 17.1



# The Big Picture



# Pro Tip:

Now that we're nearing the end of the course, take some time to review the READMEs in your old assignments and make sure they're ready to be seen by potential hiring managers!

Module 17

# This Week: Supervised Machine Learning

# This Week: Supervised Machine Learning

By the end of this week, you'll know how to:

- Explain how a machine learning algorithm is used in data analytics
- Create training and test groups from a given data set
- Implement the logistic regression, decision tree, and random forest algorithms and interpret their results
- Compare the advantages and disadvantages of each supervised learning algorithm
- Determine which supervised learning algorithm is best for a given dataset or scenario
- Use ensemble and resampling techniques to improve model performance

# This Week's Challenge

Using the skills learned throughout the week, you will evaluate three machine learning models using different algorithms to determine which is better at predicting credit risk.

# Career Connection

How will you use this module's content in your career?

Module 17

# How to Succeed This Week

# Pro Tip:

Machine Learning is a really exciting tool!
Today, we'll build an understanding of the basics,
so consider this a starting point for your future learning.

Module 17

# Today’s Agenda

# Today's Agenda

By completing today's activities, you'll learn the following skills:

01 Creating a machine learning environment.

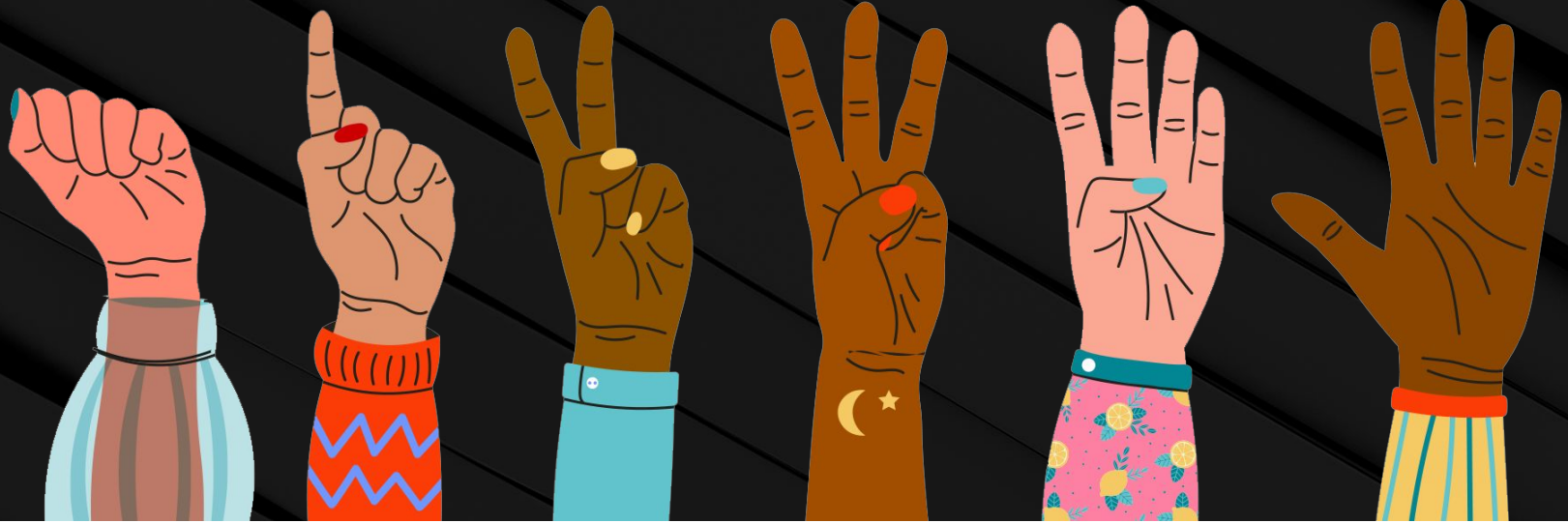
02 Preparing, transforming, and splitting data, and fit a model to the data.

03 Performing linear and logistic regression.

**Make sure you've downloaded any relevant class files!**

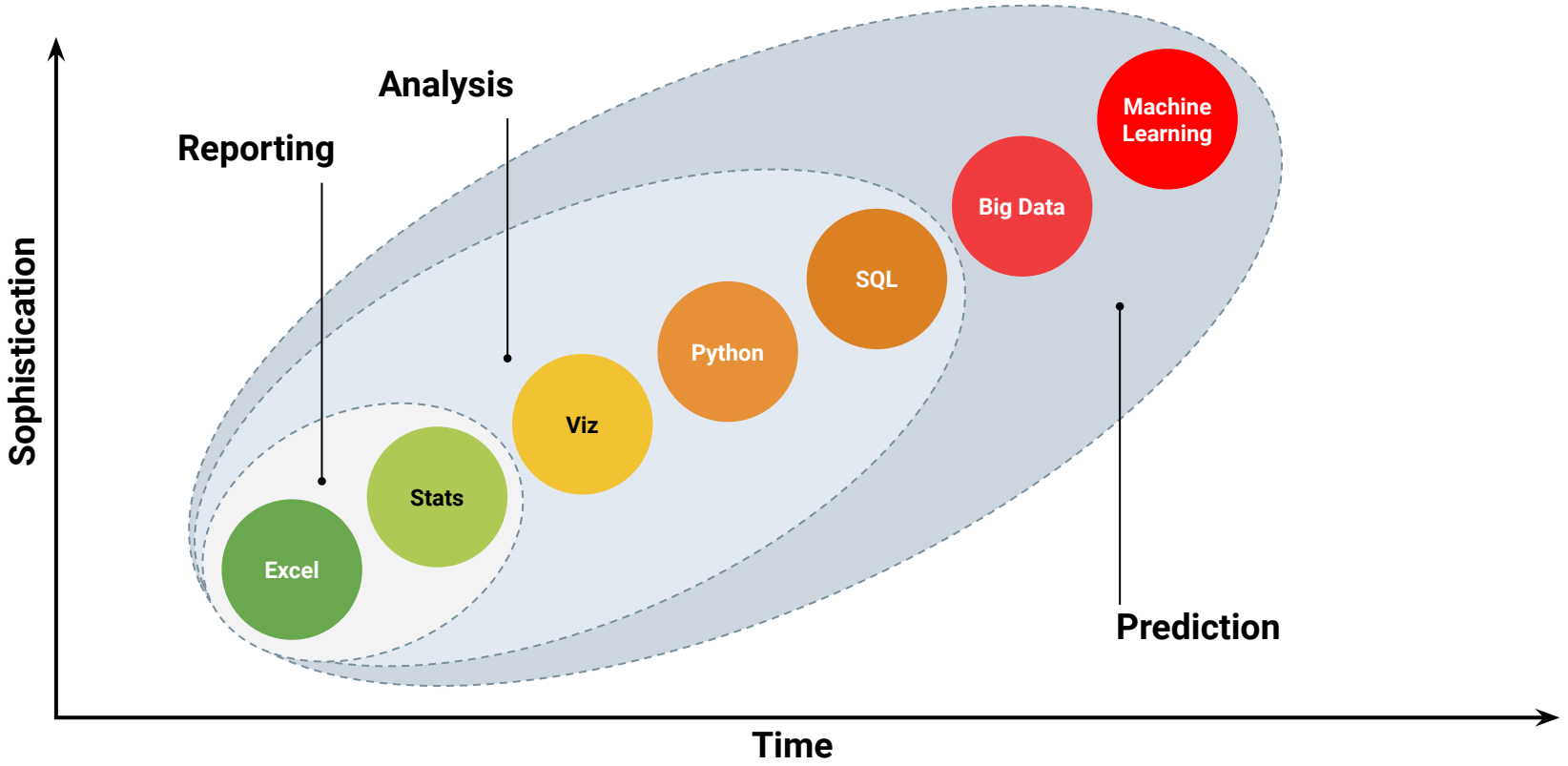
# FIST TO FIVE:

How comfortable do you feel with this topic?



# Where does machine learning fit in data analytics?

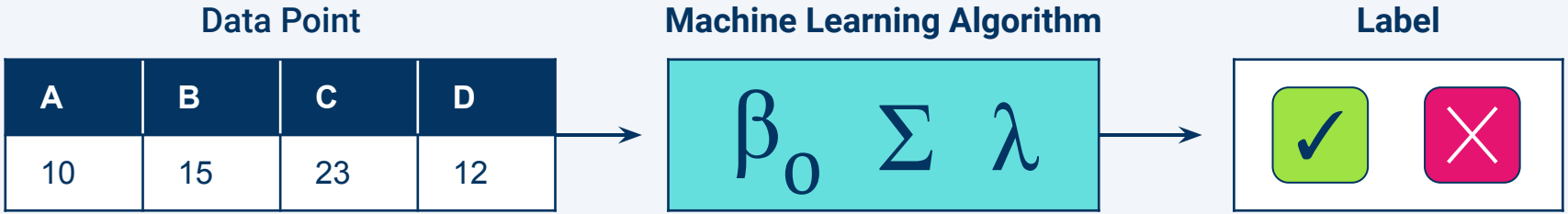
# Machine Learning (ML) is Predictive Analytics



**Machine learning** algorithms are functions with internal parameters that apply labels to data points.

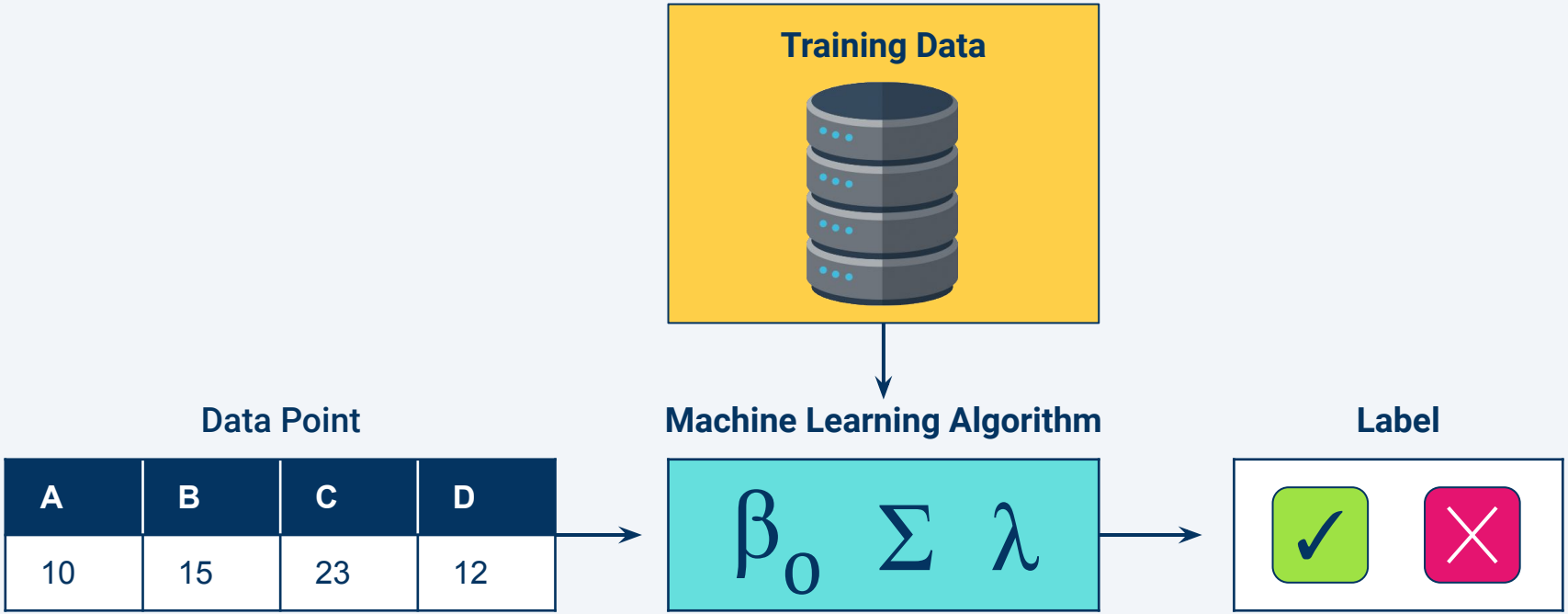
# Machine Learning Algorithms

Data points are assigned labels based on the internal parameters of the algorithm



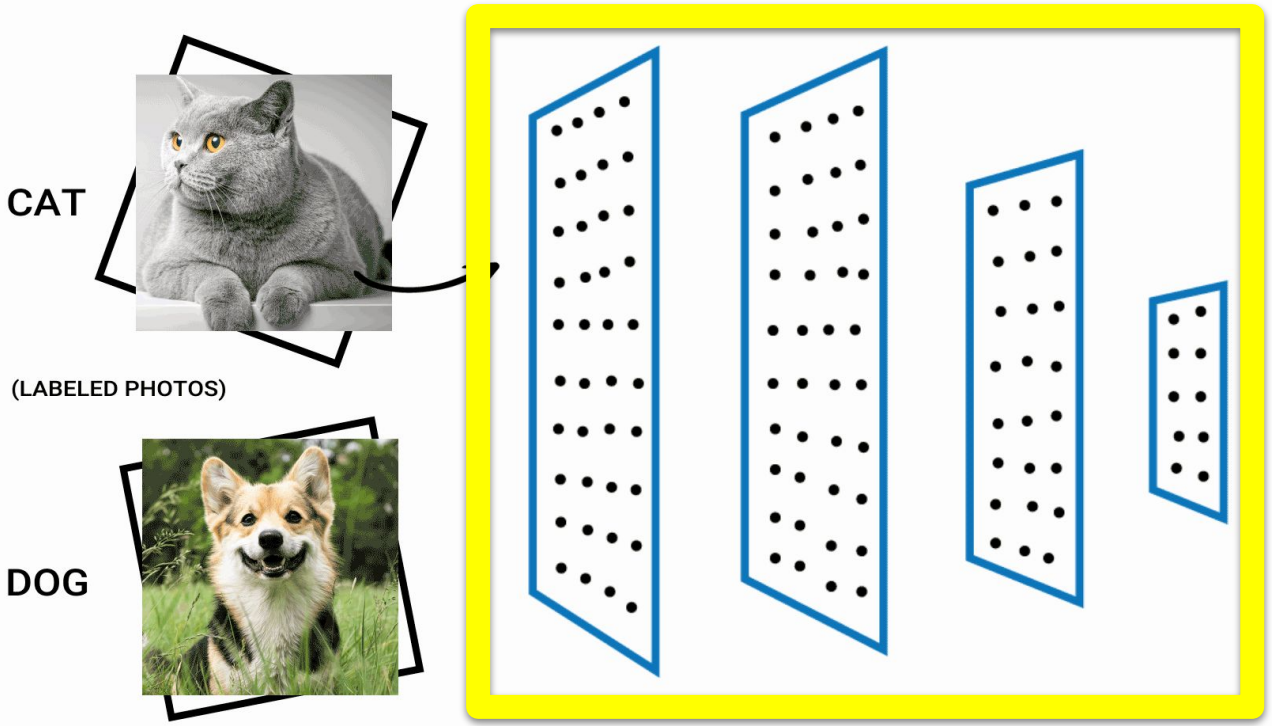
# But What Is It Learning?

Data points are assigned labels based on the internal parameters of the algorithm. This is the “learning” in machine learning.



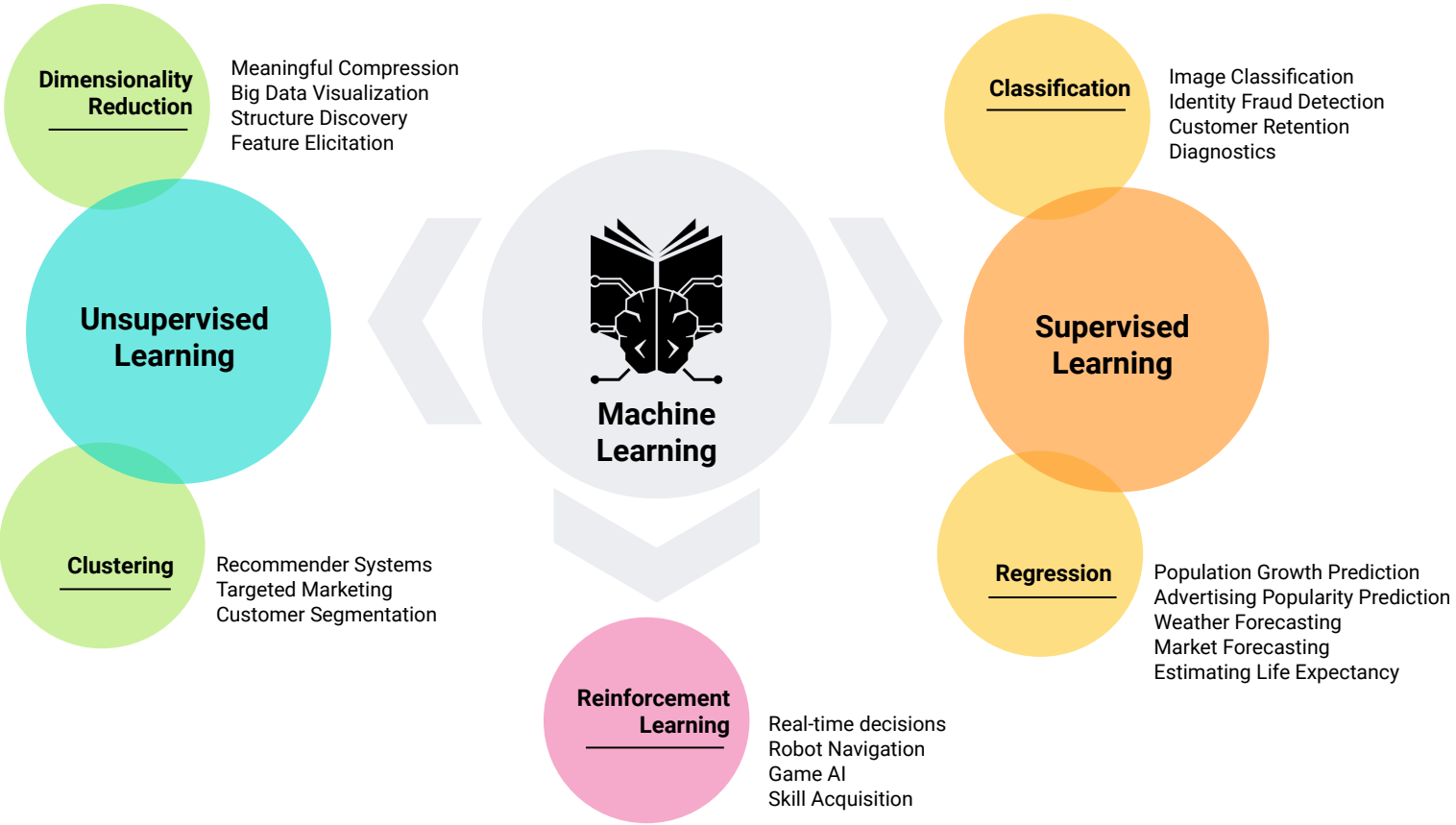
# But What Is It Learning?

Machine learning algorithms use training data to set their internal parameters.
**This is the “learning” in machine learning**.

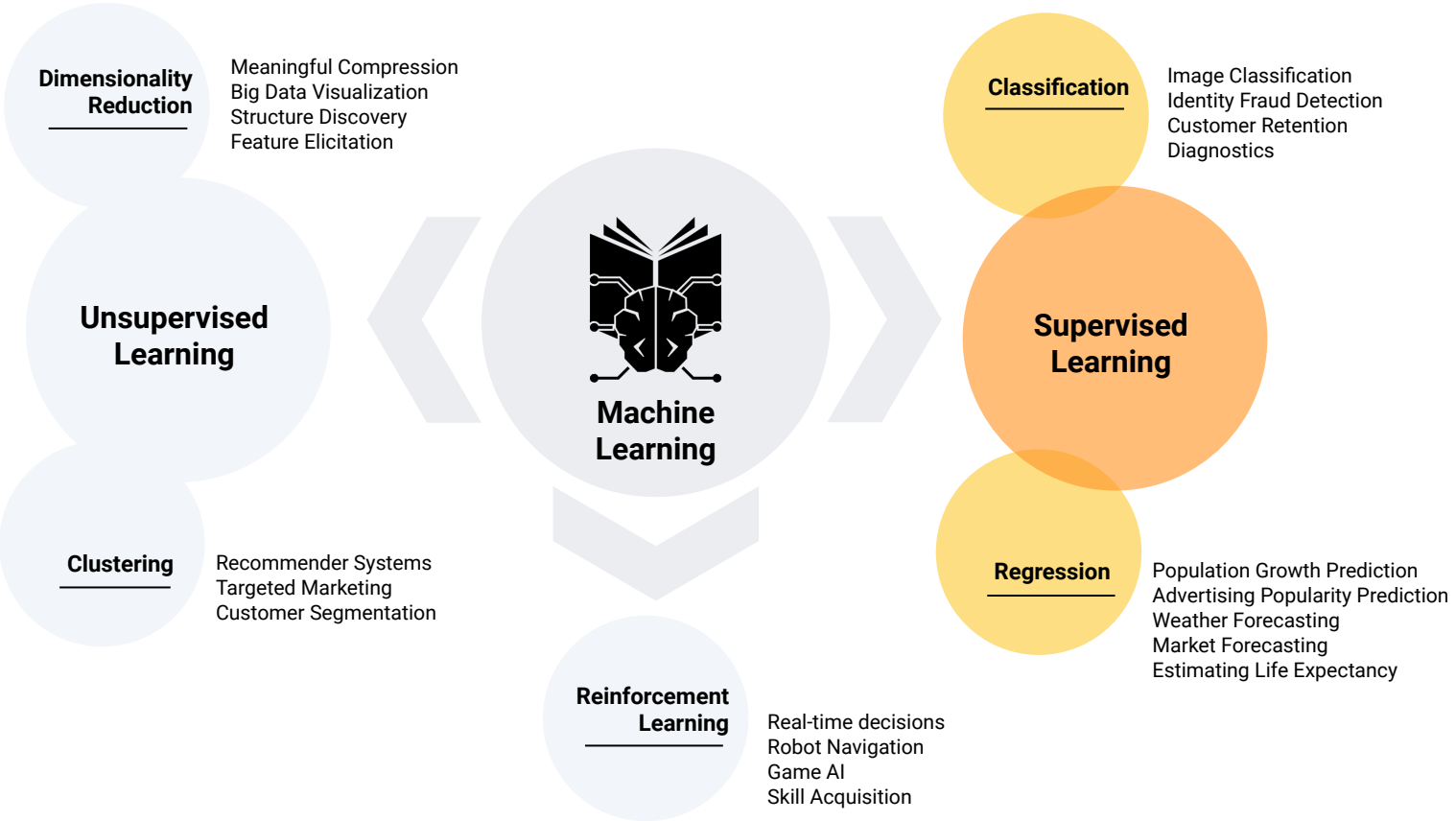


We can categorize machine learning into supervised learning, unsupervised learning, and reinforcement learning

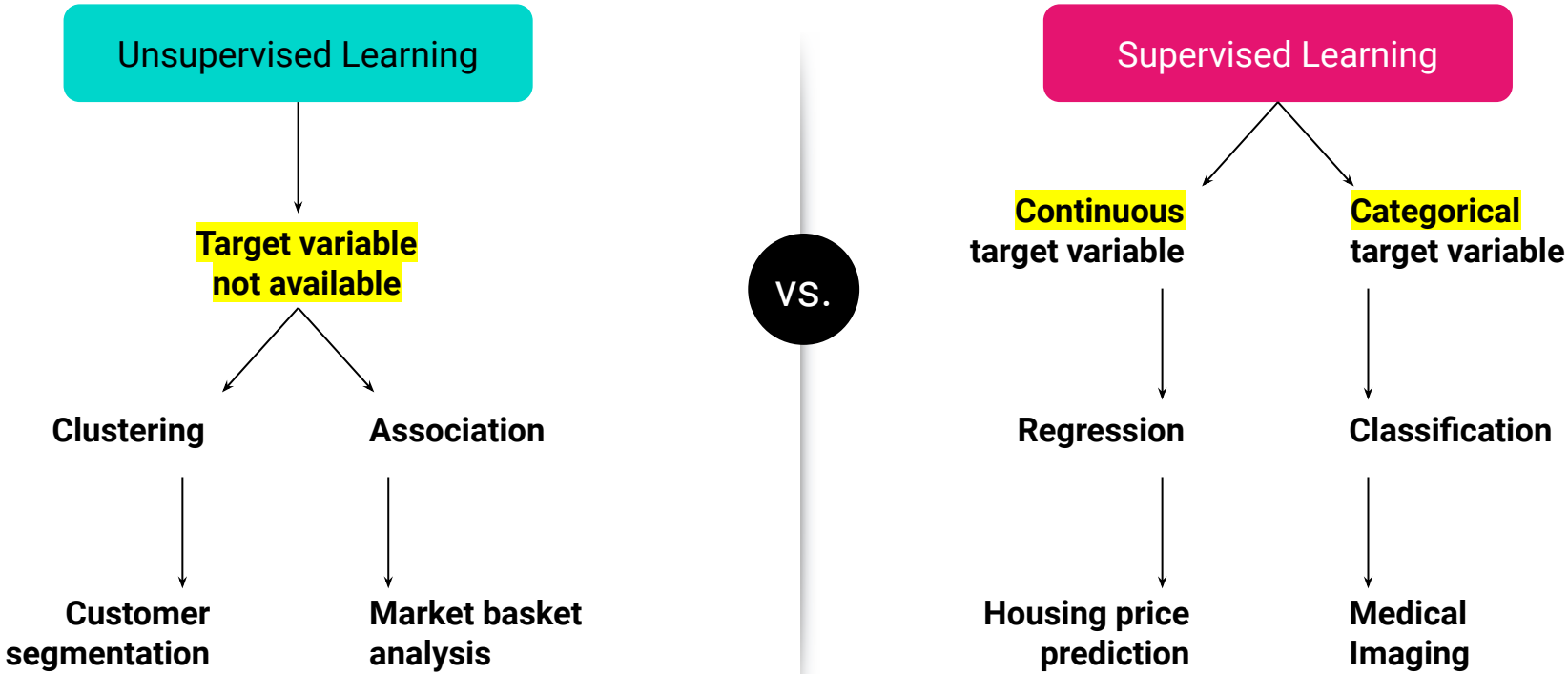
# Machine Learning (Categories)



# Machine Learning (Supervised Learning Subcategories)



# Unsupervised Learning vs. Supervised Learning



**Supervised learning**: Algorithms for which the potential outcomes are knowable in advance (e.g., category or numeric range) and can be used to correct the model's predictions.

# Machine Learning (Supervised)

Using data such as credit score, credit history, income, etc., we are trying to predict whether an individual is a credit risk or not.

**Known Category:**
“Credit Risk” vs. “Not Credit Risk”

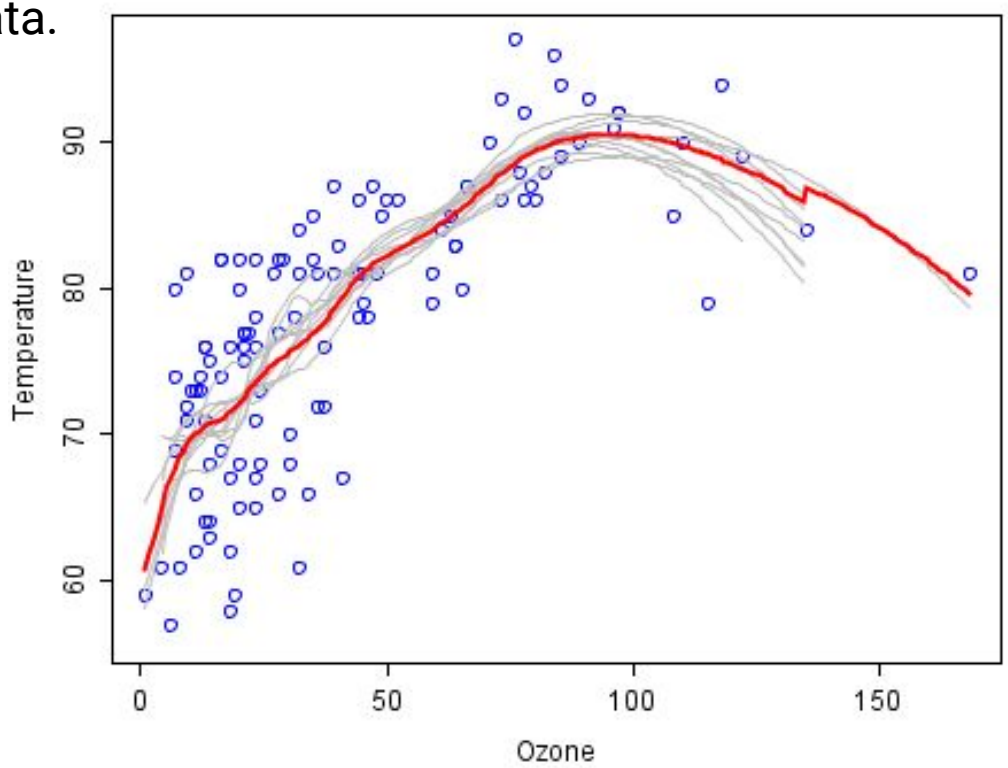
# Machine Learning (Supervised)

Using features such as number of bedrooms, square feet, etc., we are trying to predict the market value of a house.

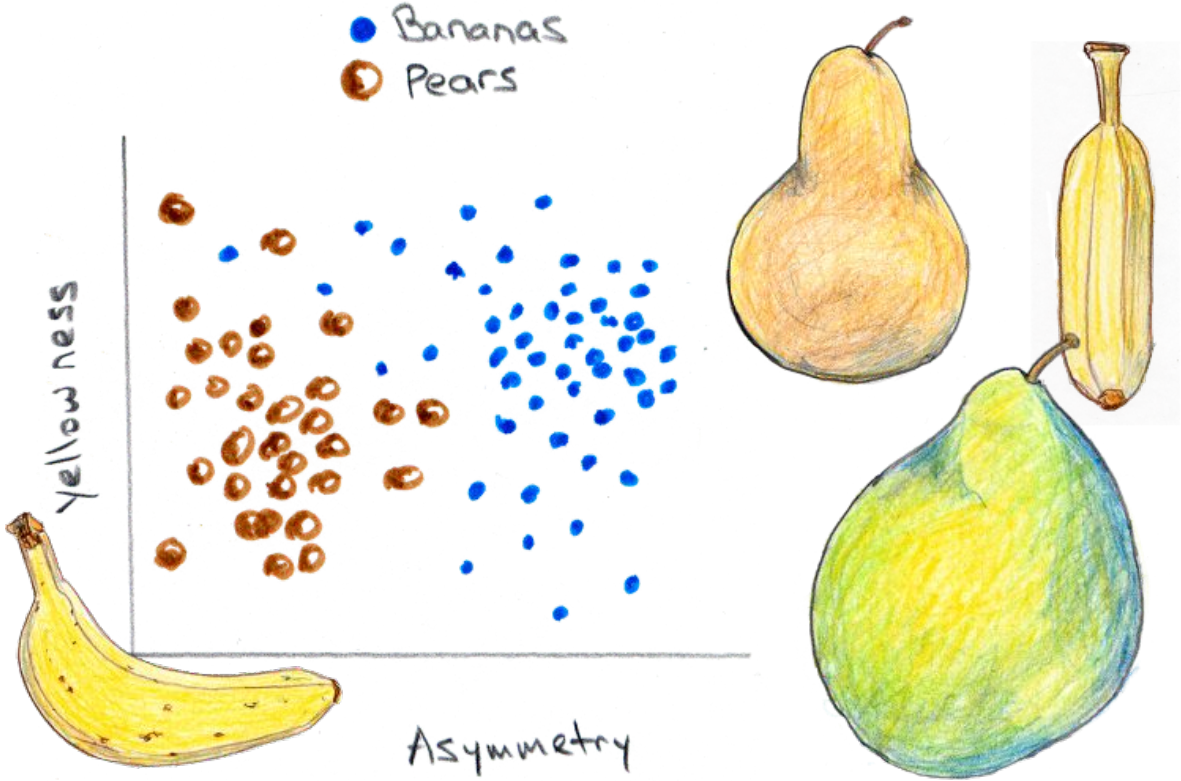
**Numeric Range:**
$50,000—$500,000

# Machine Learning (Regression)

We'll be revisiting regression to predict the location of data points based on old data.

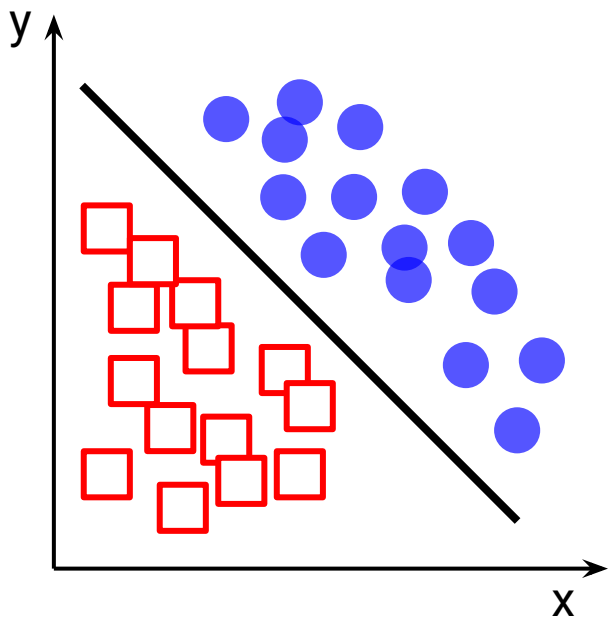


# Machine Learning (Classification)

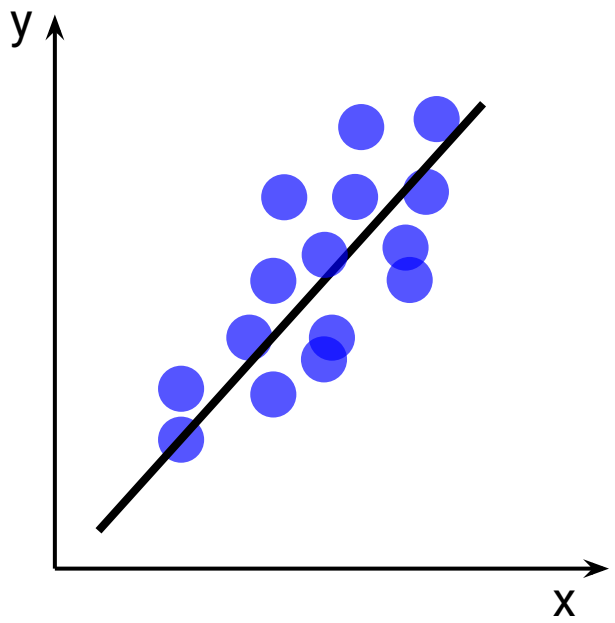


# Classification vs. Regression

## Classification



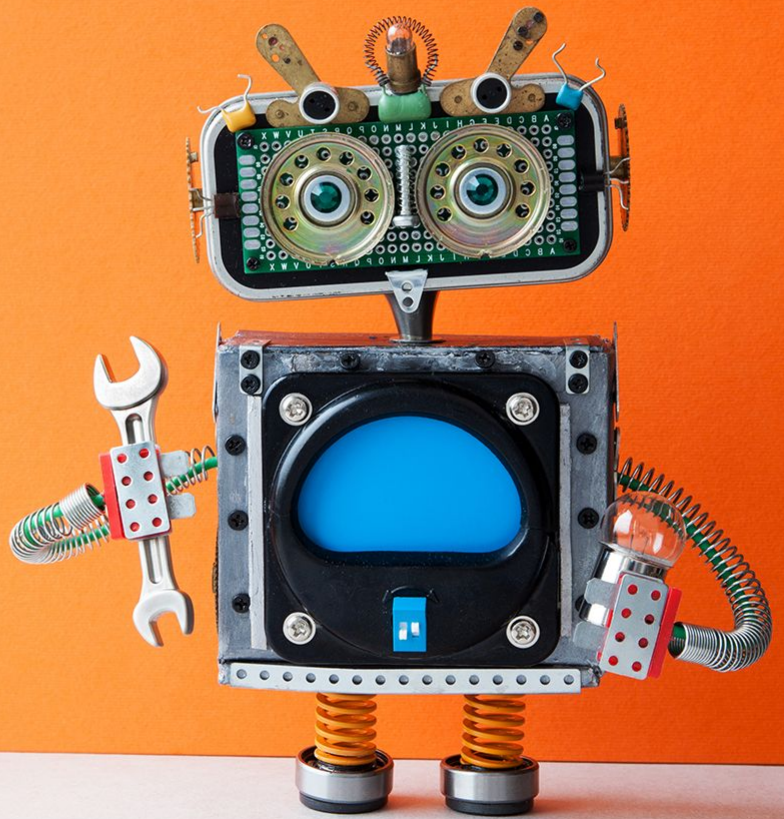

## Regression



Classification algorithms are used for discrete labels, while regression algorithms are used for continuous labels.

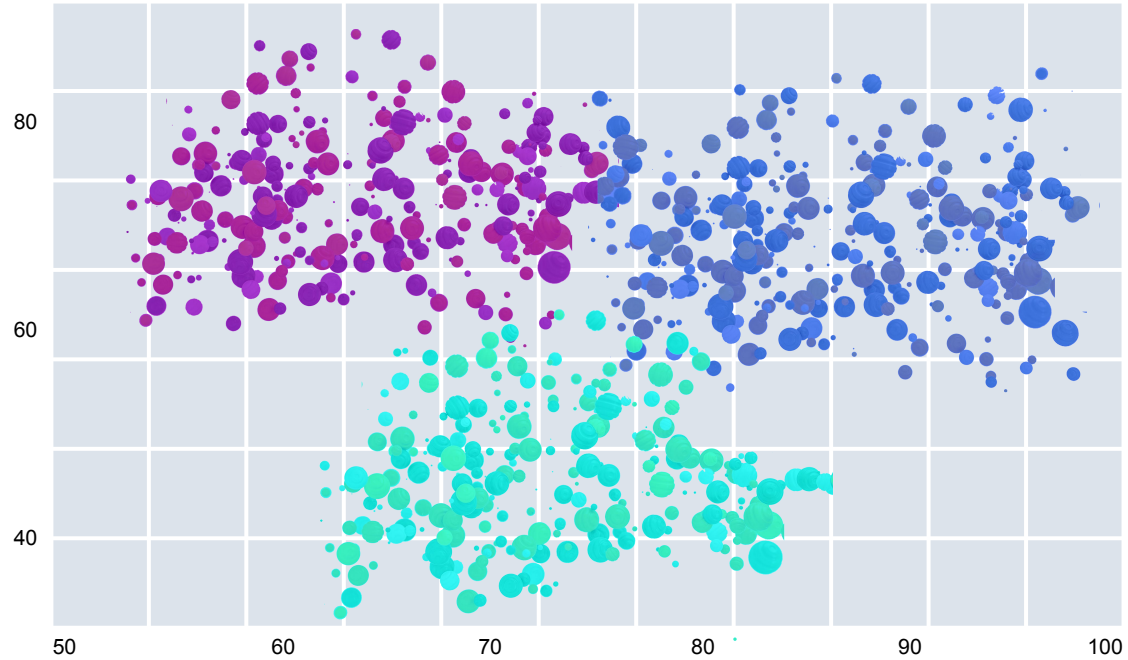
# Machine Learning (Unsupervised)

**Unsupervised Learning:** Algorithms for which the potential outcomes are unlabeled. Inferences are made directly from the data without feedback from known outcomes or labels.



# Machine Learning (Clustering)

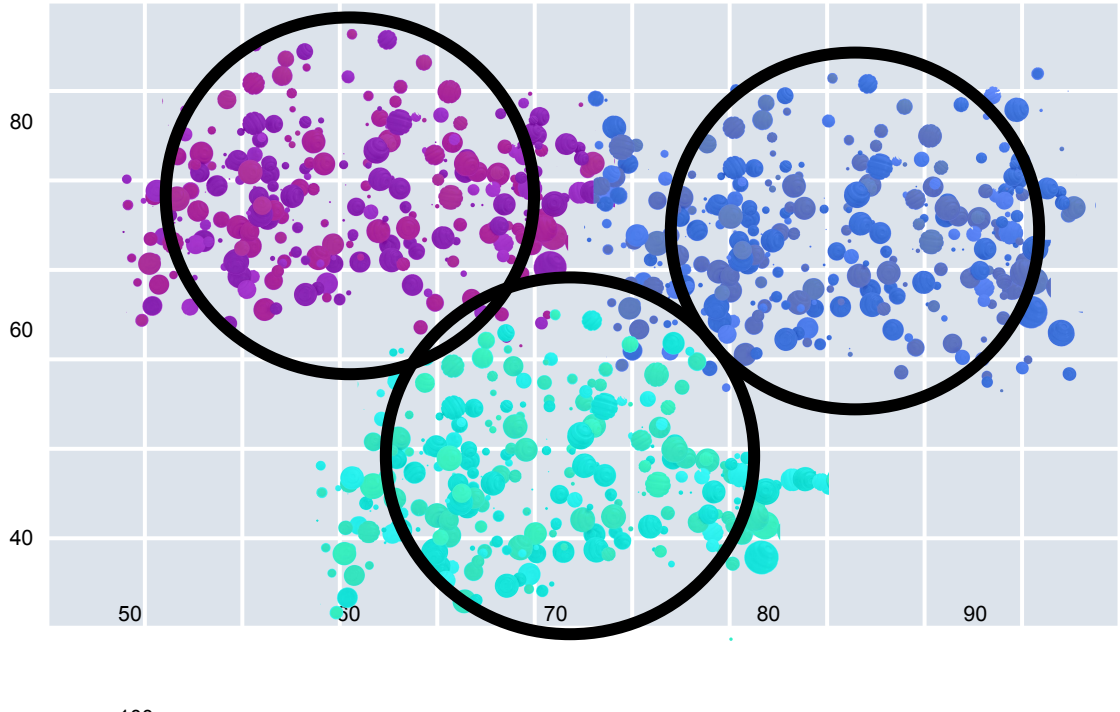
In this clustering problem, we expect our algorithm to find the groupings of data points based on location.



# Machine Learning (Clustering)

In this clustering problem, we expect our algorithm to find the groupings of data points based on location.

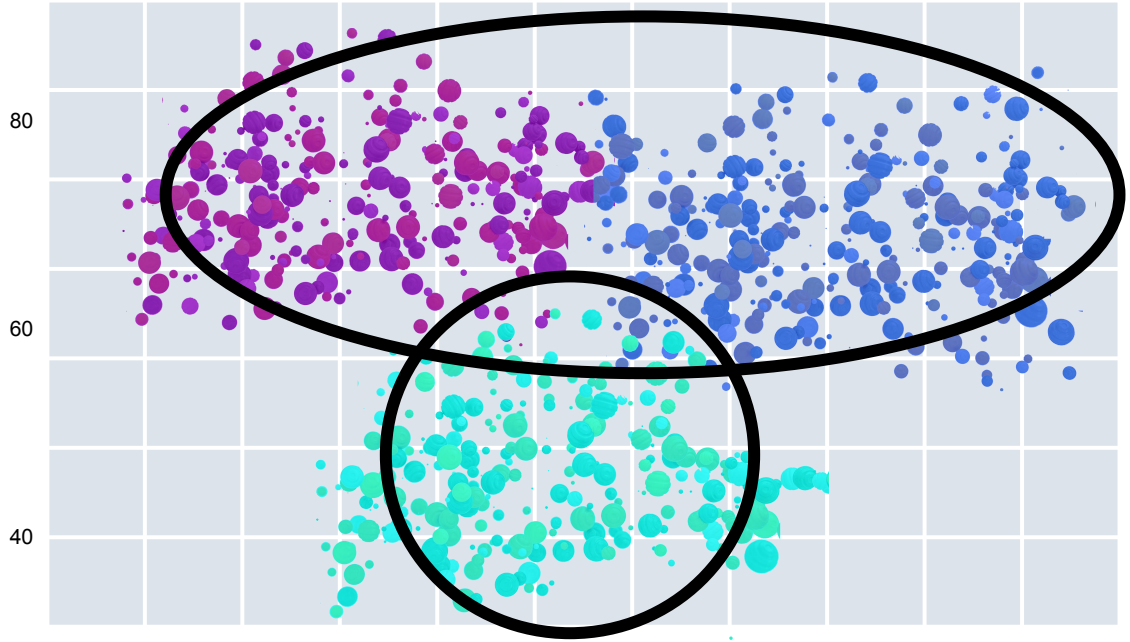
K=3



# Machine Learning (Clustering)

But the problem is more complex:

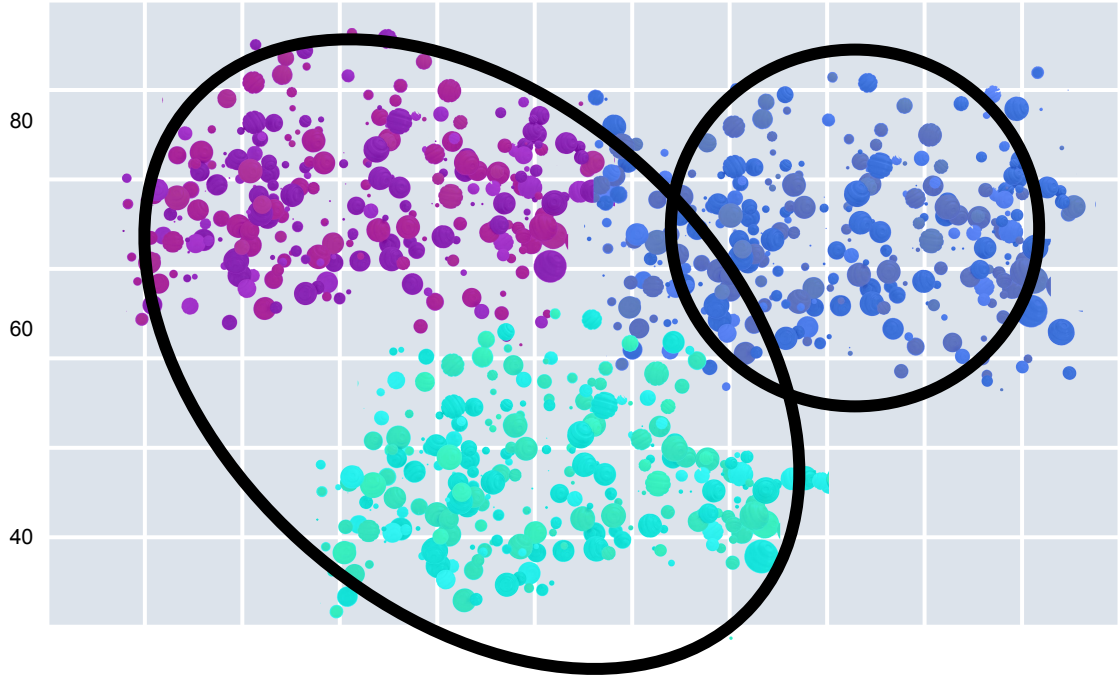
K=2



# Machine Learning (Clustering)

Perhaps the clusters are not where we think they are.

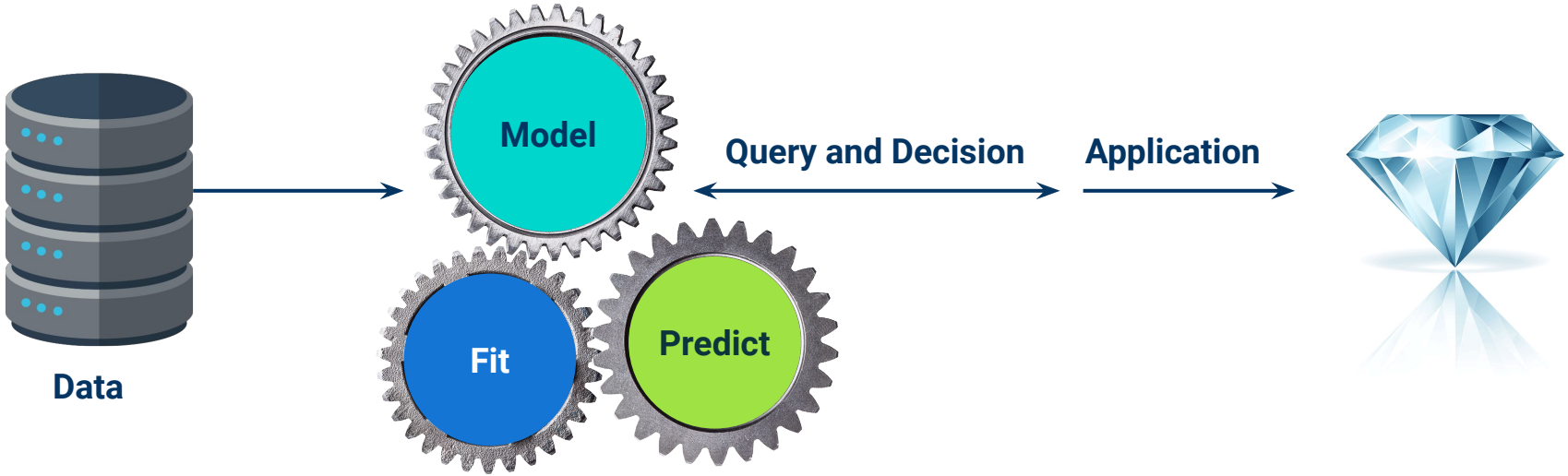
K=2



# Training and Predicting

# Training and Predicting

Regardless of the problem type, in machine learning, we follow a familiar paradigm: Model → Fit (Train) → Predict

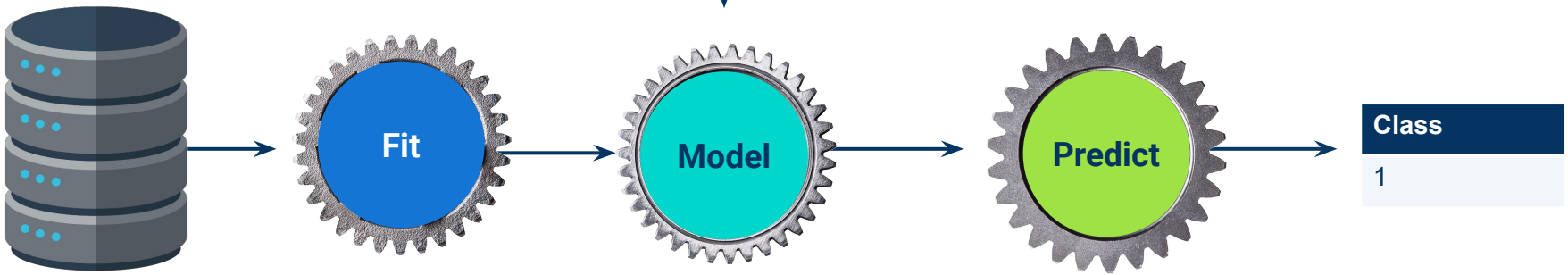


# Training and Predicting

Regardless of the problem type, in machine learning, we follow a familiar paradigm: Model → Fit (Train) → Predict

| A | B | C | Class |
|---|---|---|---|
| 11 | 16 | 22 | 1 |
| 10 | 8 | 4 | 2 |
| … | … | … | … |

| A | B | C | Class |
|---|---|---|---|
| 10 | 15 | 23 | ? |



# Questions?

# Linear Regression with scikit-learn

# What is the purpose of using linear regression?

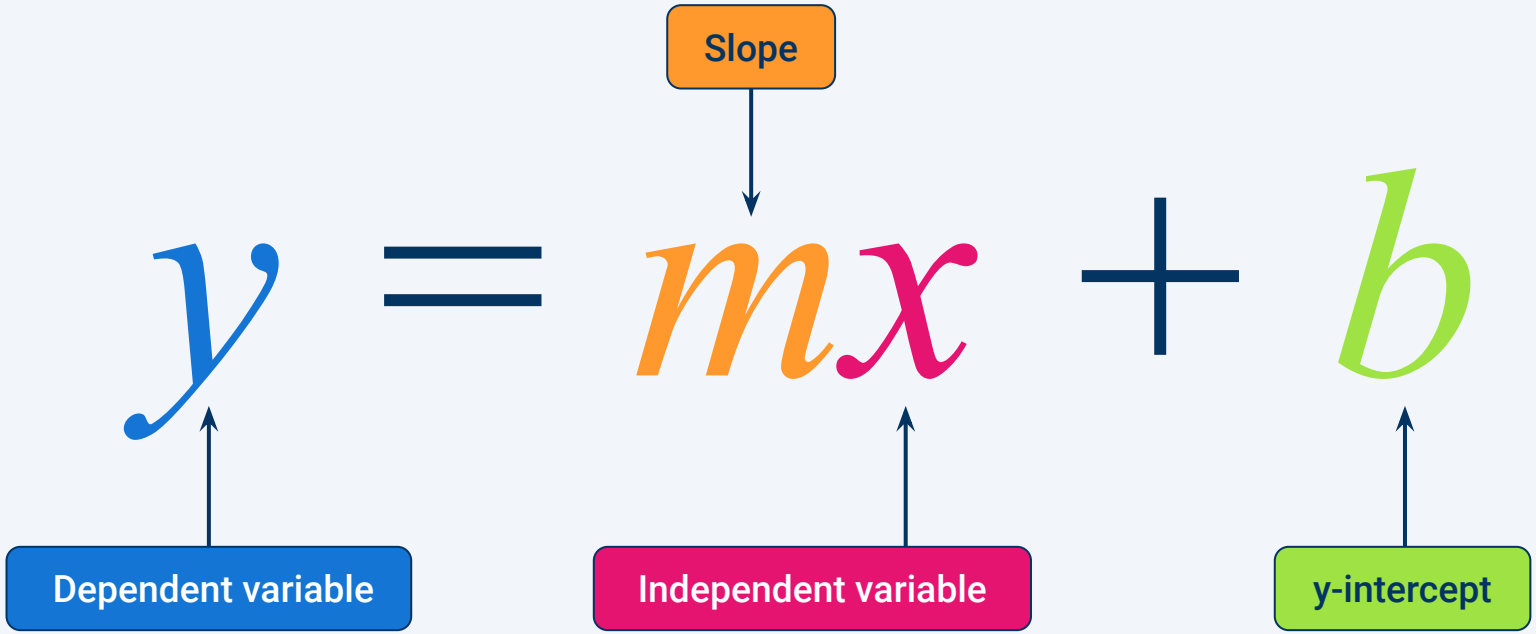
**Linear regression** tries to model and predict the relationship between a dependent variable and an independent variable.

In machine learning, the independent variable is also referred to as a feature or a factor.

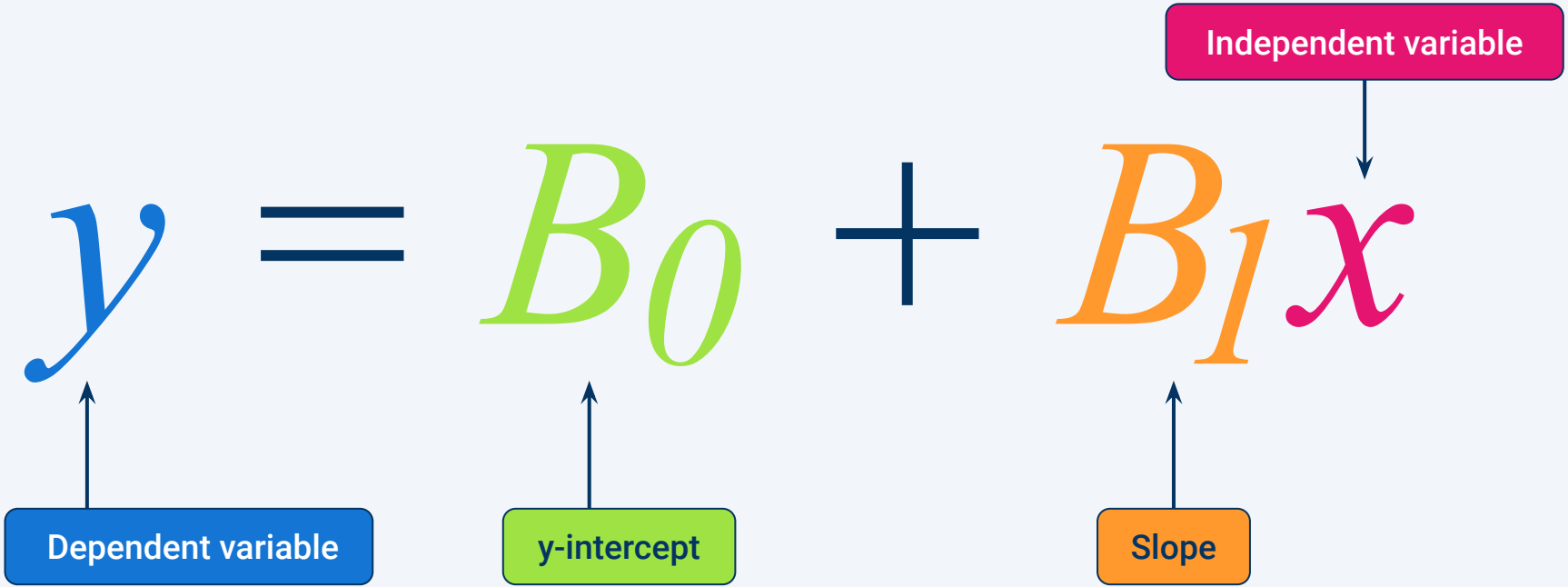
# What the purpose of using multiple linear regression?

**Multiple linear regression** tries to predict a dependent variable based on multiple independent variables.

# The Equation of a Line (Univariate)



# The Equation of a Line (Univariate) *in Greek!*



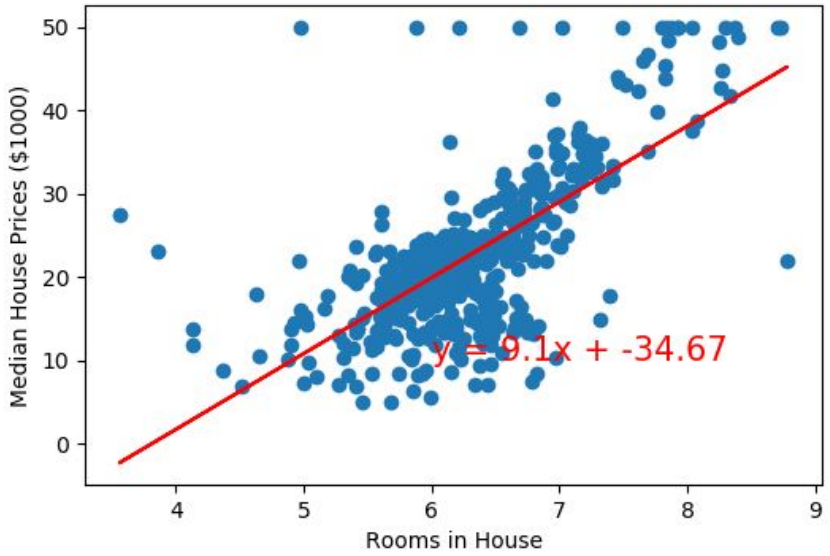
# Linear Regression

Linear Regression predicts a dependent variable based on values from an independent variable.

There are two basic types:

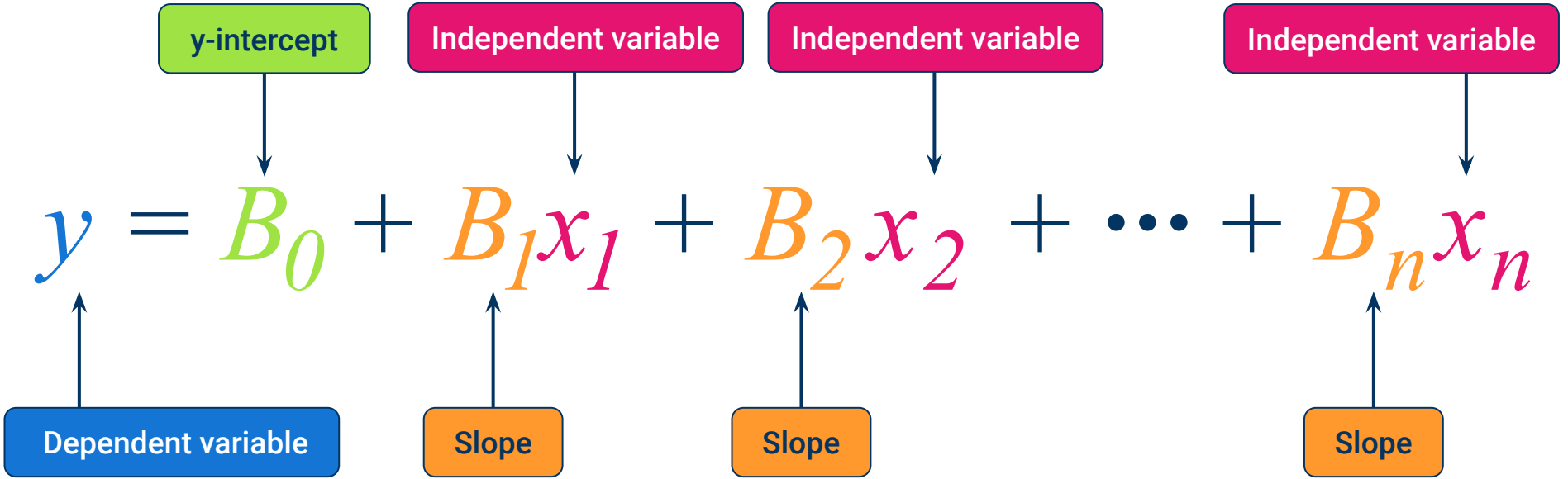
01 **Simple linear regression**

02 **Multiple linear regression**




Both types predict an independent variable using the linear equation.

# The Equation of a Line (Multivariate)

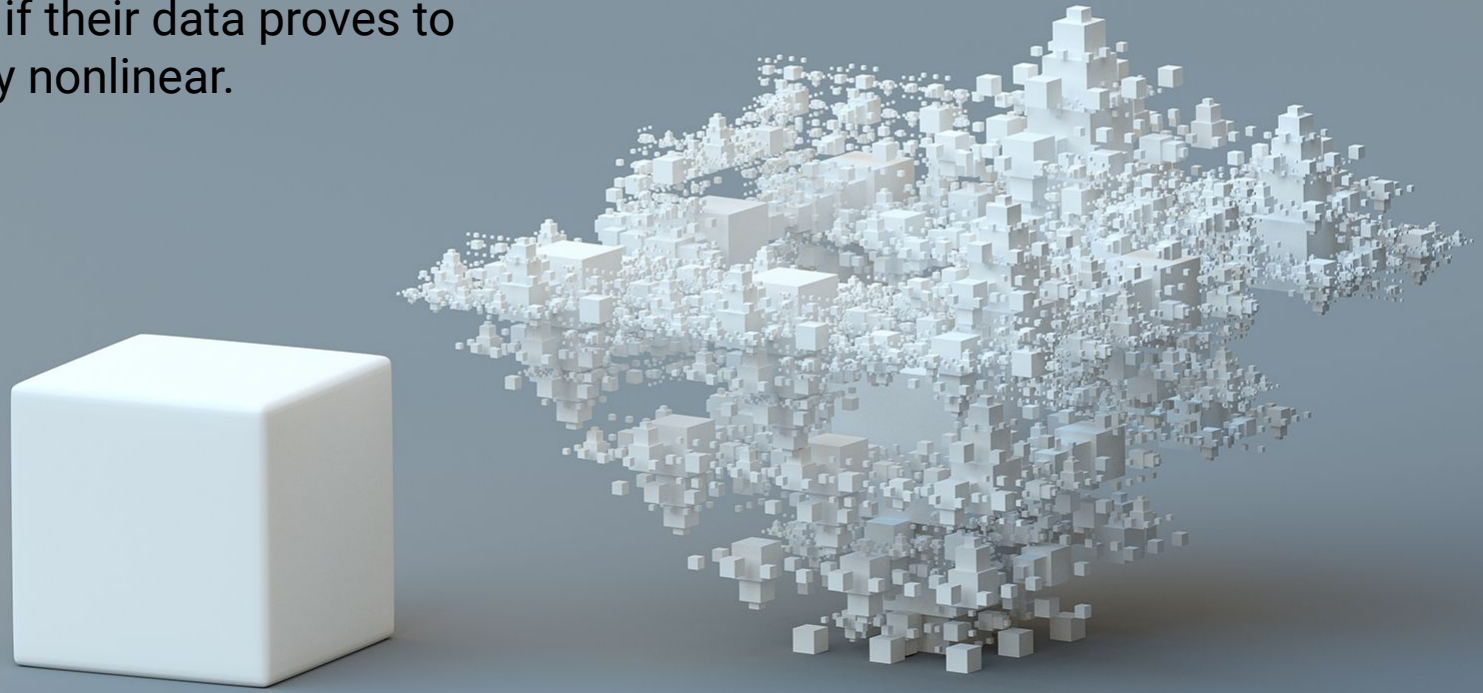


Linear regression is fast! If linear regression can solve a problem, then it may be more efficient and economical than using a more complex model such as deep learning.

# Linear Regression

Many data scientists start with a linear regression model.

Then, they move to a more complex model if their data proves to be truly nonlinear.



# Instructor Demonstration

## Linear Regression

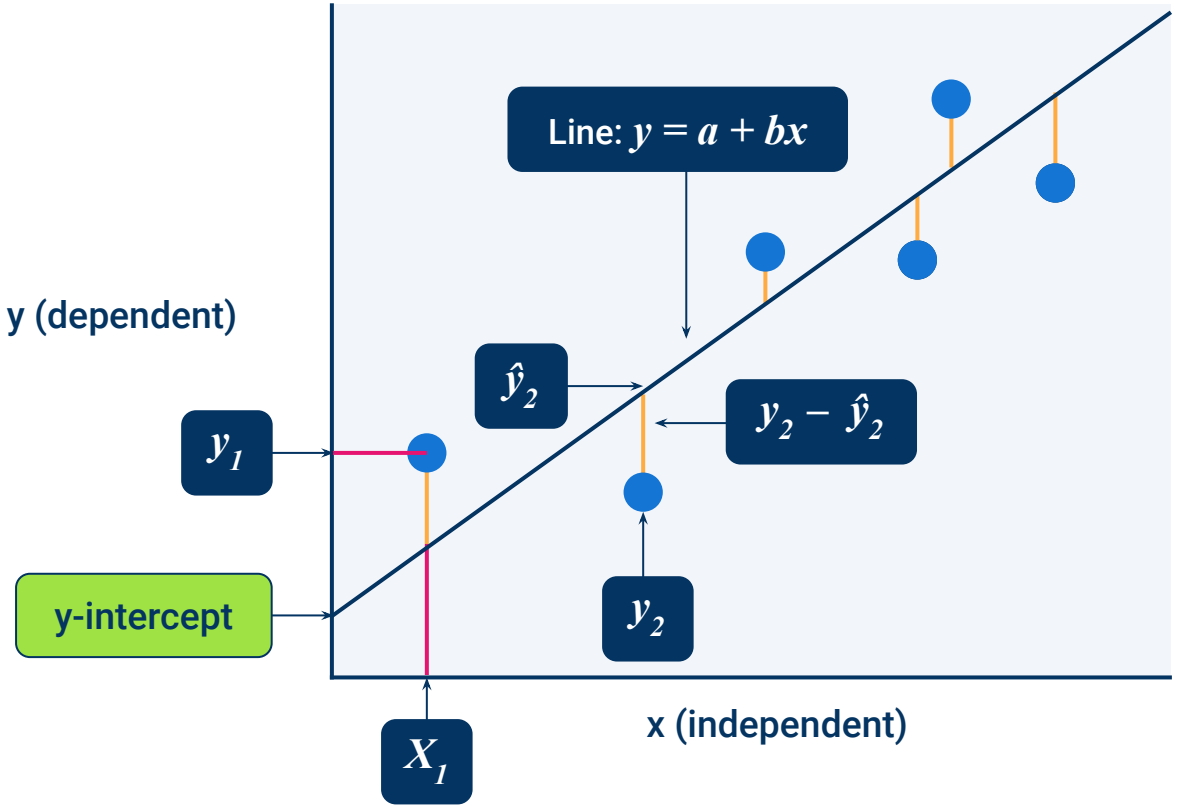
# Linear Regression

Linear data trends:

| | |
|---|---|
| **Positive trendC** | As the independent value `(x)` increases, the dependent value `(y)` increases. |
| **Negative trend** | As the independent value `(x)` increases, the dependent value `(y)` decreases. |
| **No trend** | As the independent value `(x)` increases, the dependent value `(y)` randomly increases and decreases to the point where there is no clear pattern in the data. |

# Linear Regression

Formula for univariate linear regression:




**Minimize:**

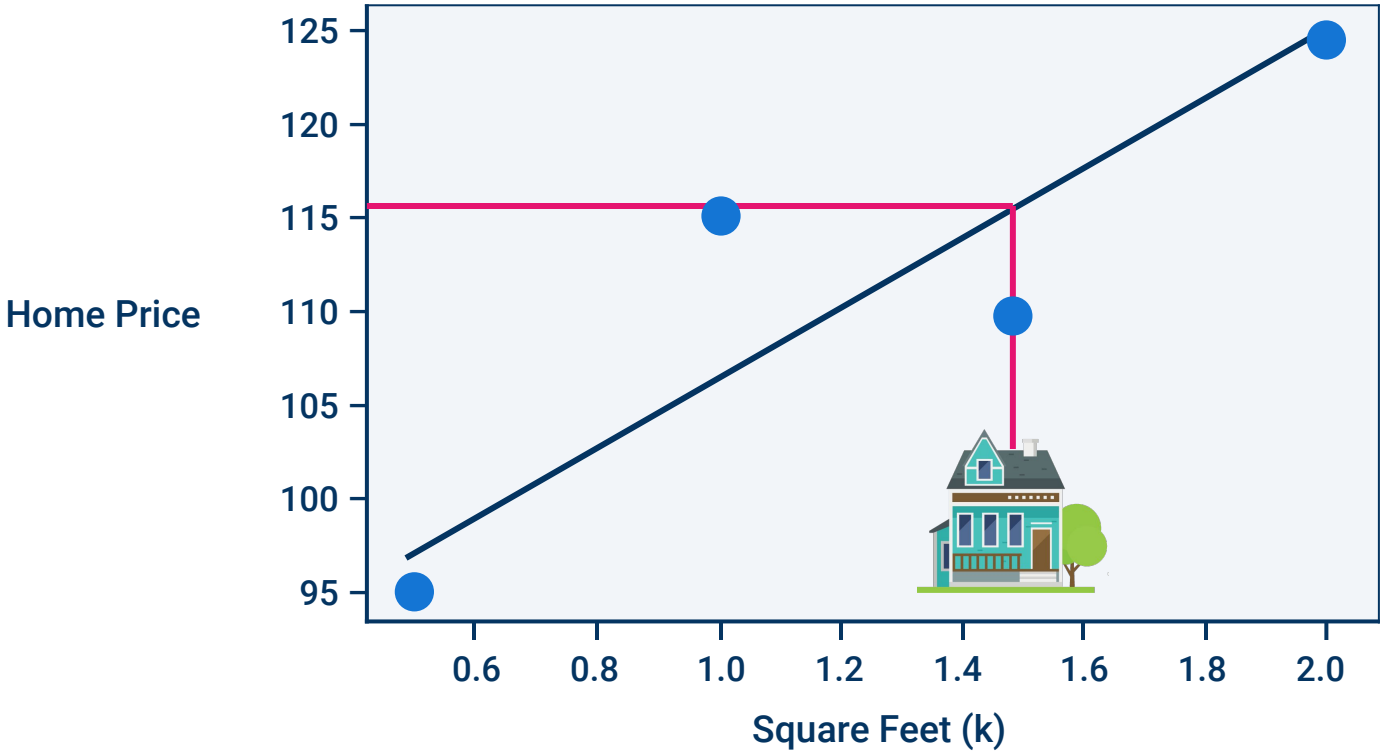
$$\sum_{i=1}^{n} (y_i - \hat{y}_i)2$$

**Least squares method:**

$$i = 1$$

# Linear Regression

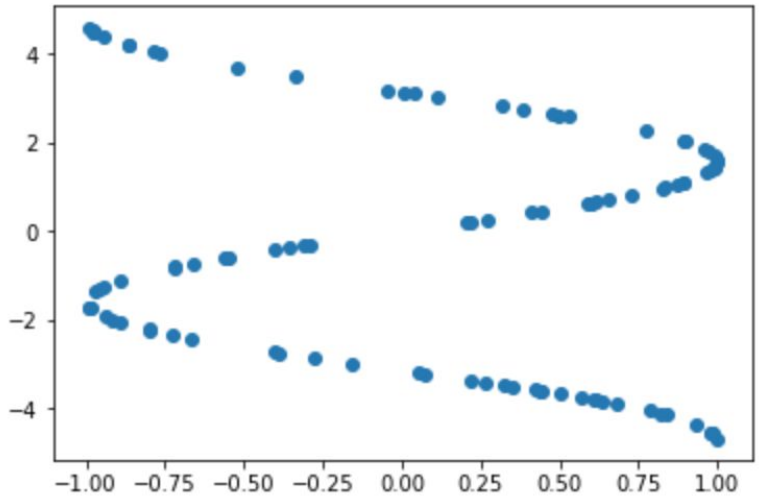
Using linear regression to predict the home price:



# Nonlinear Data

A plot of nonlinear data:

```
Out[4]: <matplotlib.collections.PathCollection at 0x11e319c18>
```



# Model-Fit-Predict Pattern

# Model-Fit-Predict Pattern

Many popular machine learning libraries follow the model-fit-predict pattern.

First, we'll import `LinearRegression` from sklearn and use it to create an instance of a model.

```
from sklearn.linear_model import LinearRegression
 model = LinearRegression()
```

# Model-Fit-Predict Pattern

Once we have a model instance, we need to fit the model to the data—this is the training process.

The goal of the training is to find the slope and the intercept that best represent the data (that is, to fit a line to the data).

```
model.fit(X, y)

print(model)
```

# Model-Fit-Predict Pattern

We can use the line to make predictions for new inputs because we have a model that can take any value of X and calculate a value for y that follows the trend of the original data.

```
In [7]: print('Weight coefficients: ', model.coef_)
        print('y-axis intercept: ', model.intercept_)
```

```
Weight coefficients:  [ 12.44002424]
y-axis intercept:  101.896225057
```

Our linear model now looks like this:

$$y = 101.896225057 + 12.44002424x$$

# Model-Fit-Predict Pattern

The format for passing values to `model.predict()` is a list of lists, as the following code shows:

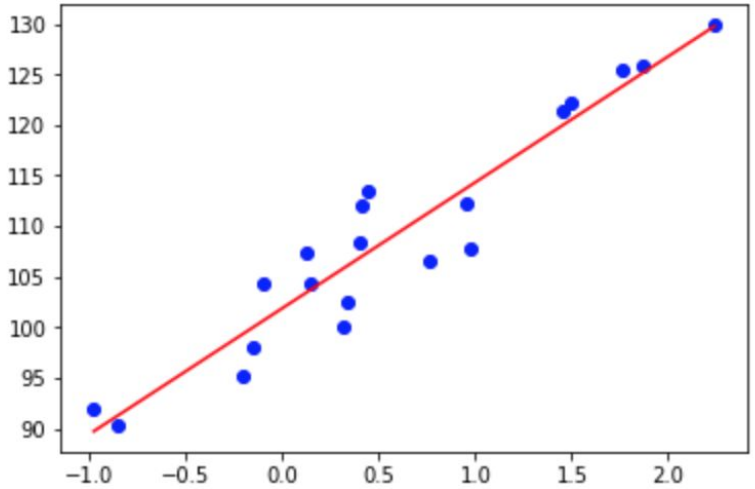
```
y_min_predicted = model.predict([[x_min]])

y_max_predicted = model.predict([[x_max]])
```

If we compare the first prediction to the original output value, the values should be very close because the model represents the trend of the original data.

# Model-Fit-Predict Pattern

The original data compared to the predicted minimum and maximum values:

```
Out[13]: [<matplotlib.lines.Line2D at 0x11e93fd30>]
```



# Multiple Linear Regression

Multiple linear regression is linear regression that uses multiple input features.

Multiple Linear Regression simply means that you have more than one feature variable.

$$Y_i = \theta_0 + \theta_1 X_{i1} + \theta_2 X_{i2} + \ldots + \theta_p X_{ip}$$

You can think of this as:

$$Y_i = Bias_0 + Weight_1 Feature_1 + Weight_2 Feature_2 + \ldots + Weight_p Feature_p$$

For the Housing Price example, you may have features like this:

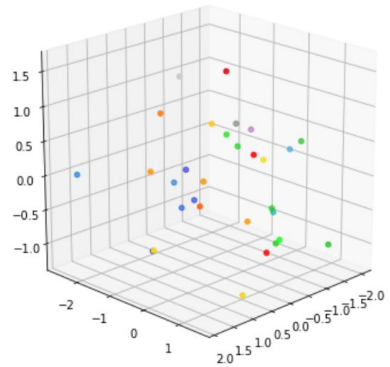
$Y_i$ = $Bias_0$ + $Weight_1$ sq_feet + $Weight_2$ num_bedrooms + $Weight_3$ num_bathrooms

With multiple linear regression, it becomes hard to visualize the linear trends in the data. We need to rely on our regression model to correctly fit a line.

# Multiple Linear Regression

Sklearn uses the ordinary least squares (OLS) method for fitting the line. Luckily, the API to the linear model is the same as before! We simply fit our data to our n-dimensional X array, as the following image shows:

In [3]:
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(1, figsize=(5, 5))
axes = Axes3D(fig, elev=20, azim=45)
axes.scatter(X[:,0], X[:,1], X[:,2], c=y, cmap=plt.cm.spectral)
plt.show()
```



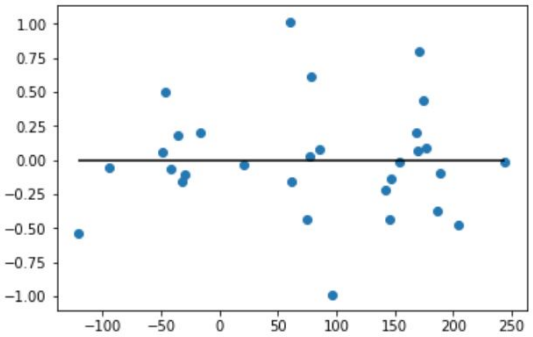
Sometimes, a linear regression model may not be an appropriate or decent fit for the data. To determine if the linear regression model is a decent fit for the data, we need to examine the residuals.

**Residuals** are the differences between the true values of $y$ and the predicted values of $y$.

# Residuals

When we plot residuals, if there is equal distribution above and below the *x*-axis, then the linear model provides a decent fit to the data.

In [5]:
```
predictions = model.predict(X)
# Plot Residuals
plt.scatter(predictions, predictions - y)
plt.hlines(y=0, xmin=predictions.min(), xmax=predictions.max())
plt.show()
```



# Questions?

# Time to Code

## Linear Regression

Suggested Time:

# Instructor Demonstration

## Quantifying Regression

# Metrics to Quantify Machine Learning Models

# Common Scoring Metrics

## $R^2$ ($R$ squared):

This is the baseline metric that many ML tools report on score. Higher $R^2$ values signify that the model is “highly predictive.”

An $R^2$ value of >0.90 means that our model roughly accounts for 90% of the variability of the data.

## MSE (mean squared error)

This measures the average of the squares of the errors or deviations.

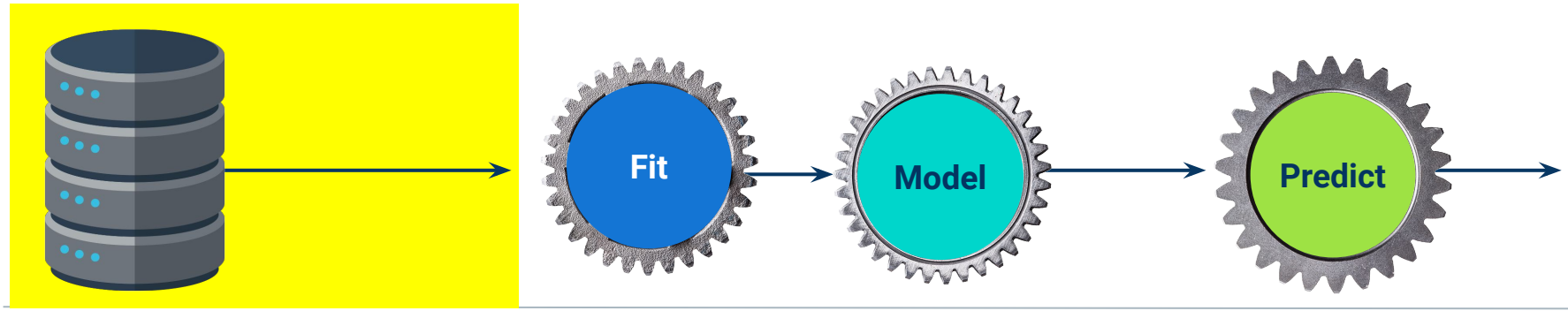
# Basic Premise of Validation Using Training and Testing Data

We will cut a slice of this data (80%) to build our model and then use it to predict the values for the remaining 20%.

| Full Data Set (Historic) | | | |
|---|---|---|---|
| N=1000 | | | |
| # bedrooms | # baths | Sq. feet (k) | Price (k) |
| 2 | 1 | 1 | 200 |
| 3 | 2 | 1.5 | 250 |
| … | … | … | … |

| Training Data Set | | | |
|---|---|---|---|
| N=800 | | | |
| # bedrooms | # baths | Sq. feet (k) | Price (k) |
| 4 | 3.5 | 3.2 | 450 |
| 2 | 2 | 1.5 | 220 |
| … | … | … | … |

| Testing Data Set | | | |
|---|---|---|---|
| N=200 | | | |
| # bedrooms | # baths | Sq. feet (k) | Price (k) |
| 1 | 1 | .5 | 60 |
| 5 | 3.5 | 4.2 | 780 |
| … | … | … | … |



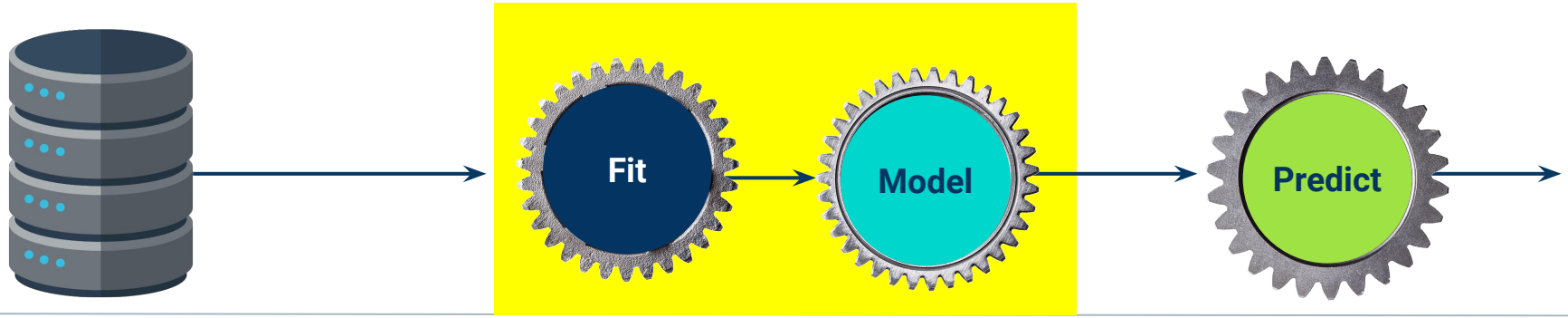
# Basic Premise of Validation Using Training and Testing Data

We use the training data to fit the model to the data. This is the training step where we build a model that can predict our output (home price) for a given set of features (# bedrooms, # baths, square feet). Once the model is trained, we can use the model to make predictions.

| Full Data Set (Historic) | | | |
|---|---|---|---|
| N=1000 | | | |
| # bedrooms | # baths | Sq. feet (k) | Price (k) |
| 2 | 1 | 1 | 200 |
| 3 | 2 | 1.5 | 250 |
| … | … | … | … |

| Training Data Set | | | |
|---|---|---|---|
| N=800 | | | |
| # bedrooms | # baths | Sq. feet (k) | Price (k) |
| 4 | 3.5 | 3.2 | 450 |
| 2 | 2 | 1.5 | 220 |
| … | … | … | … |

| Testing Data Set | | | |
|---|---|---|---|
| N=200 | | | |
| # bedrooms | # baths | Sq. feet (k) | Price (k) |
| 1 | 1 | .5 | 60 |
| 5 | 3.5 | 4.2 | 780 |
| … | … | … | … |



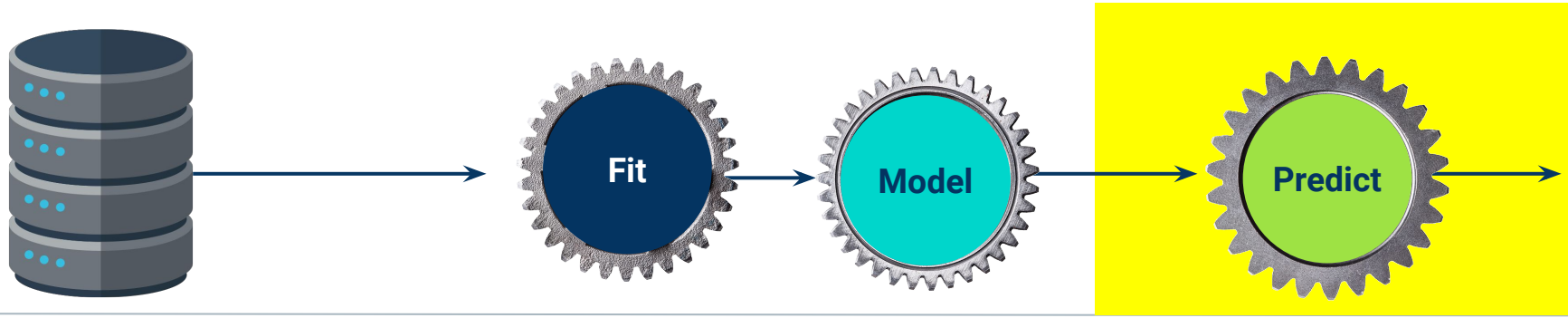
# Basic Premise of Validation Using Training and Testing Data

We use the test data to make new home price predictions. Then, we can compare the home price of our prediction against the actual price.

| Full Data Set (Historic) | | | |
|---|---|---|---|
| N=1000 | | | |
| # bedrooms | # baths | Sq. feet (k) | Price (k) |
| 2 | 1 | 1 | 200 |
| 3 | 2 | 1.5 | 250 |
| … | … | … | … |

| Training Data Set | | | |
|---|---|---|---|
| N=800 | | | |
| # bedrooms | # baths | Sq. feet (k) | Price (k) |
| 4 | 3.5 | 3.2 | 450 |
| 2 | 2 | 1.5 | 220 |
| … | … | … | … |

| Testing Data Set | | | |
|---|---|---|---|
| N=200 | | | |
| # bedrooms | # baths | Sq. feet (k) | Price (k) |
| 1 | 1 | .5 | 60 |
| 5 | 3.5 | 4.2 | 780 |
| … | … | … | … |



Based (roughly) on how often we are “correct,” we get a score for the model as a whole. If the model scores well, we can trust it for future use. We train the model on the training data and score the model based on data that it has never seen before (test data).

# Time to Code

## Brains!

Suggested Time:

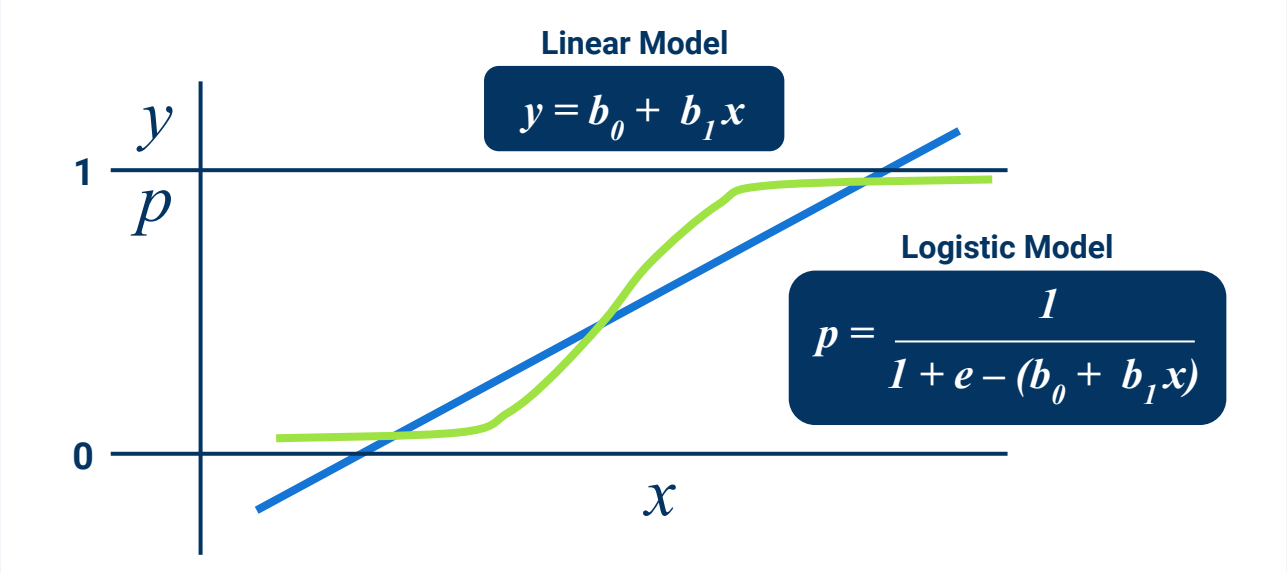
20 minutes

# Logistic Regression

**Logistic regression** is a classification algorithm used to predict a discrete set of classes or categories (for example, Yes/No, Young/Old, Happy/Sad).

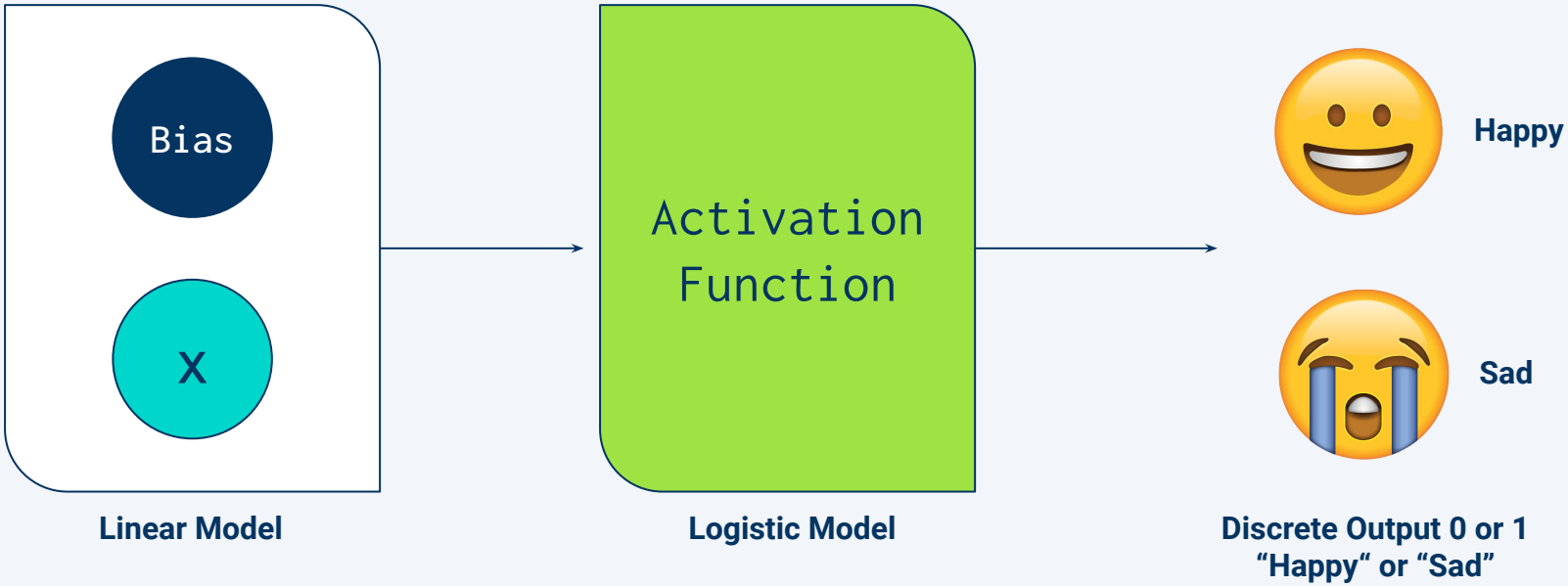
# Logistic Regression

Unlike linear regression, which outputs continuous numerical values (for example, age), logistic regression applies an activation function, such as the sigmoid function, to return a probability value of 0 or 1. This can then be mapped to a discrete class, like “Young” or “Old.”



# Logistic Regression

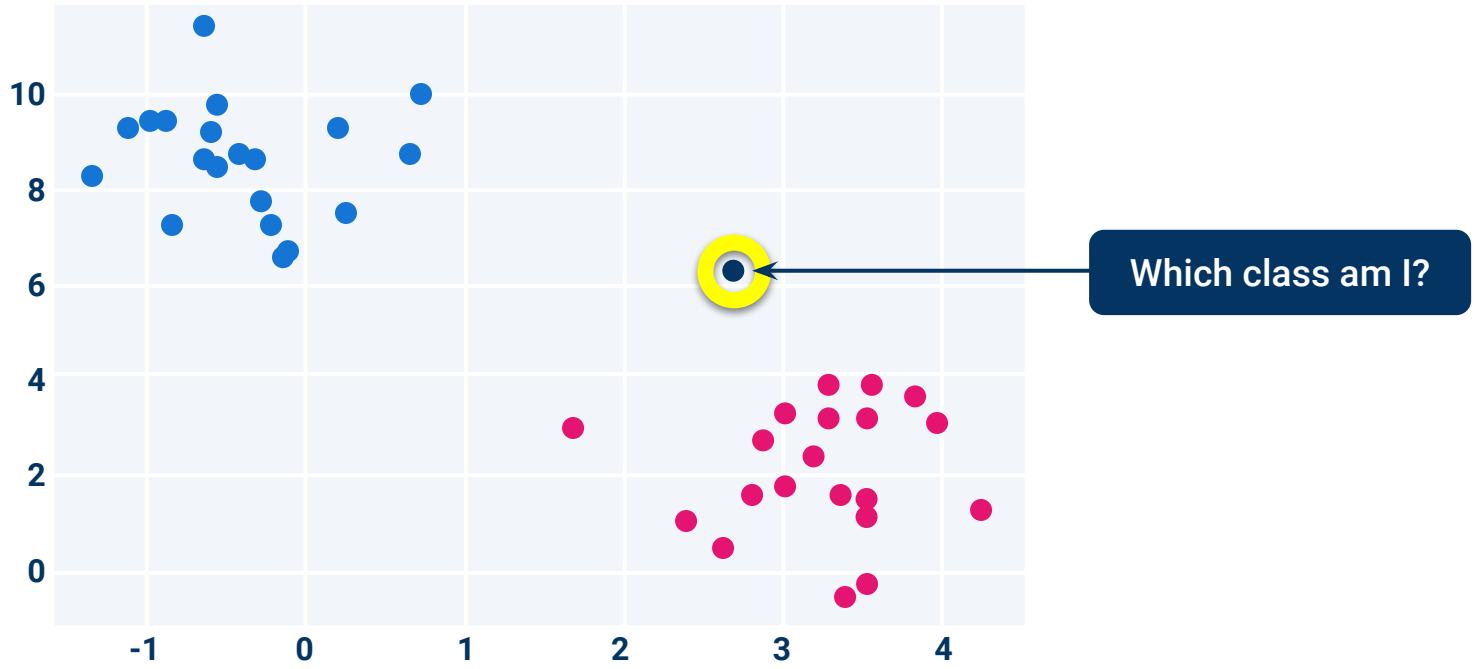
Logistic regression can be used to predict a discrete output or category.



We can use logistic regression to predict which category or class a new data point should belong to.

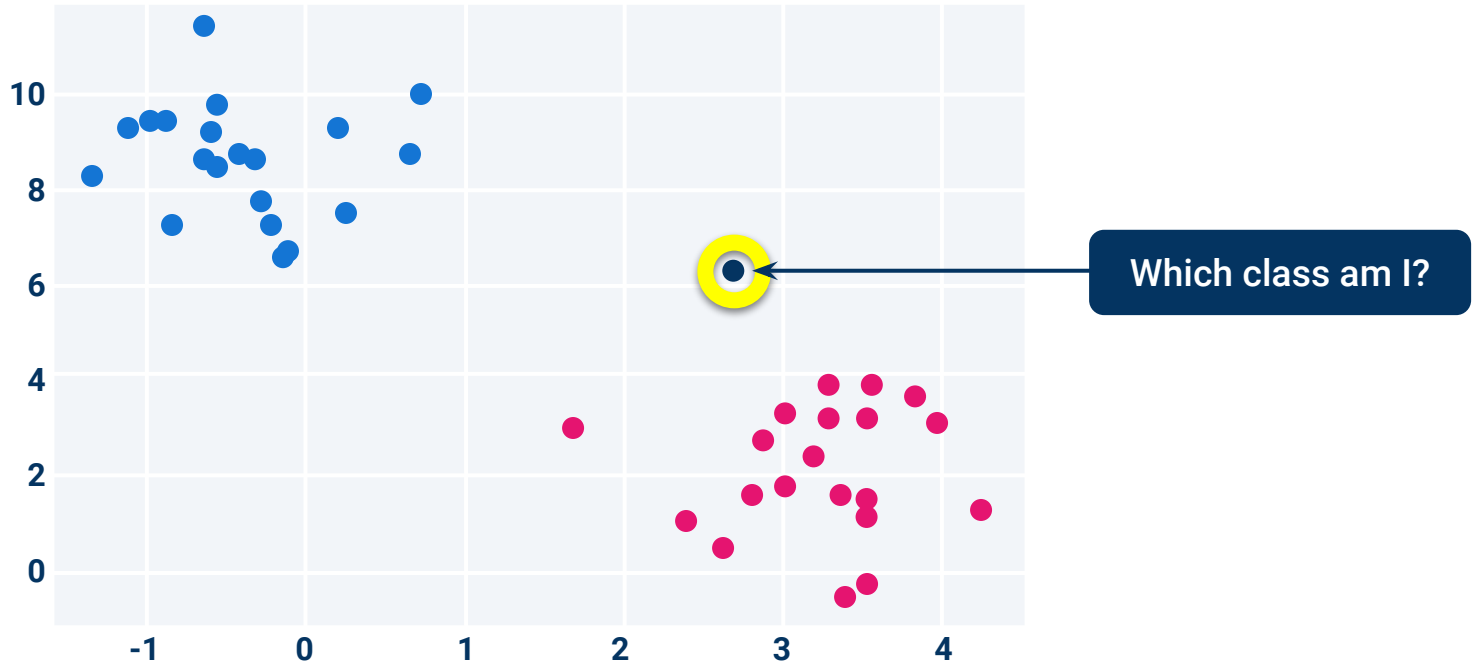
# Logistic Regression

For example, assume that we have two classes of data: a red class and a blue class. The data points in each class cluster together on a plot.



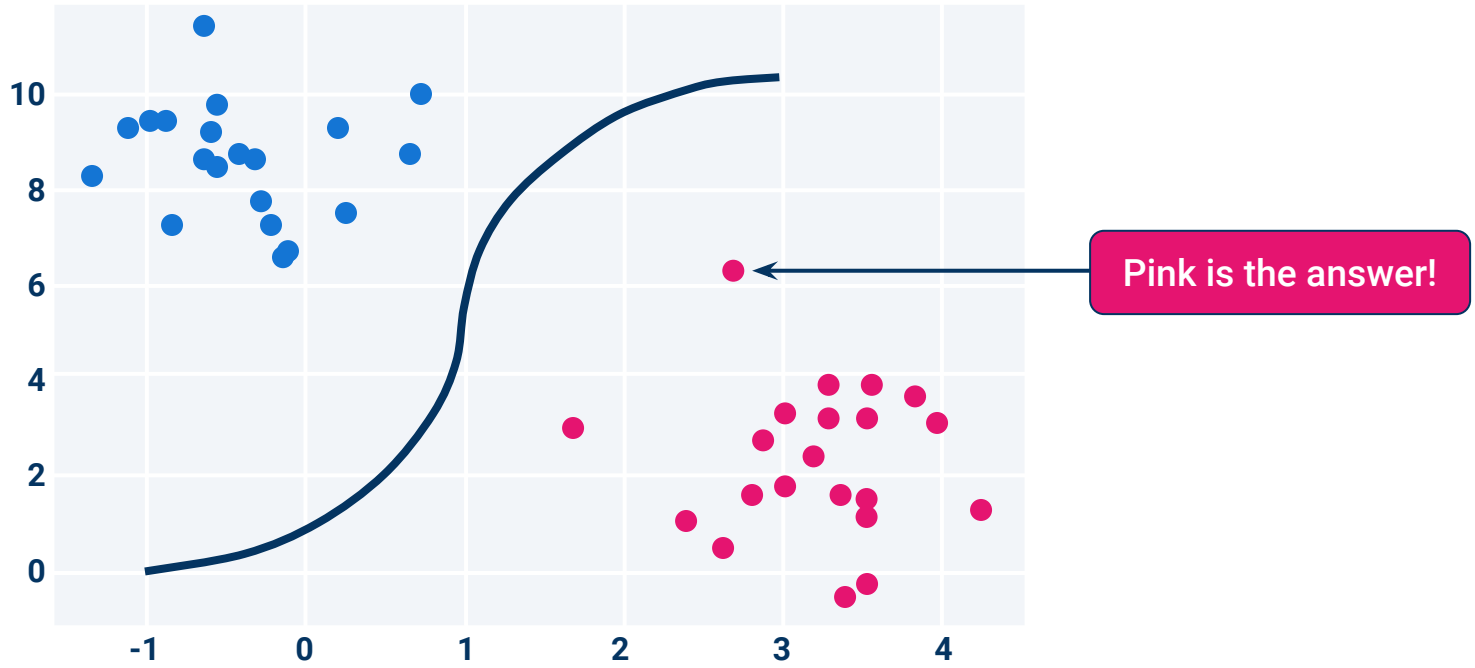
# Logistic Regression

Applying logistic regression gives us a line that separates the two classes on the plot.



# Logistic Regression

Now, we can predict which class a new data point should belong to—according to which side of the line it falls on.



# Instructor Demonstration

## Logistic Regression

# Activity: Counterfeit Catcher

In this activity, you will apply logistic regression to predict whether a particular bank note is counterfeit or legitimate by using computed features from digitized images.

Suggested Time:

15 minutes

# Activity: Counterfeit Catcher

## Instructions

01 Split your data into training and testing data.

02 Create a logistic regression model with sklearn.

03 Fit the model to the training data.

04 Make 10 predictions, and then compare them to the testing data labels.

05 Compute the accuracy score for the training and testing data separately.

# Time's Up! Let's Review.

# Questions?