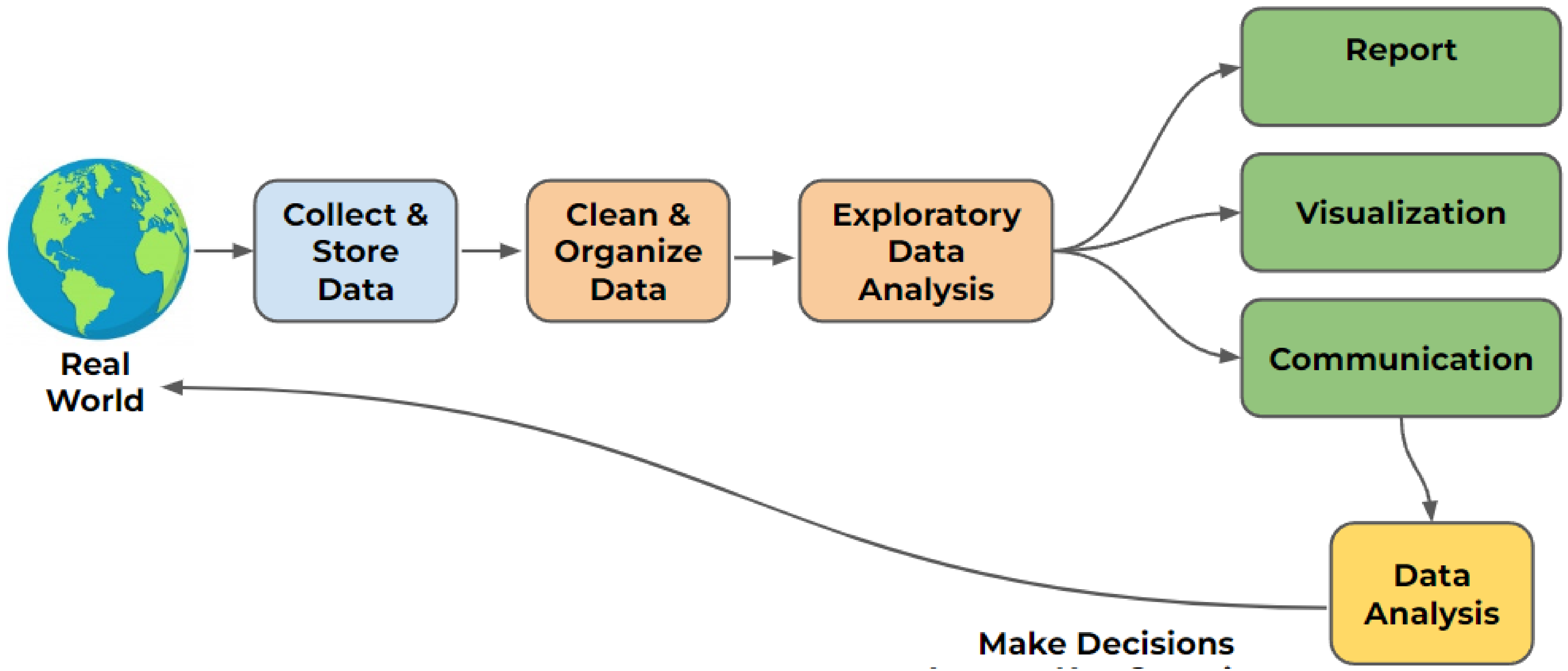
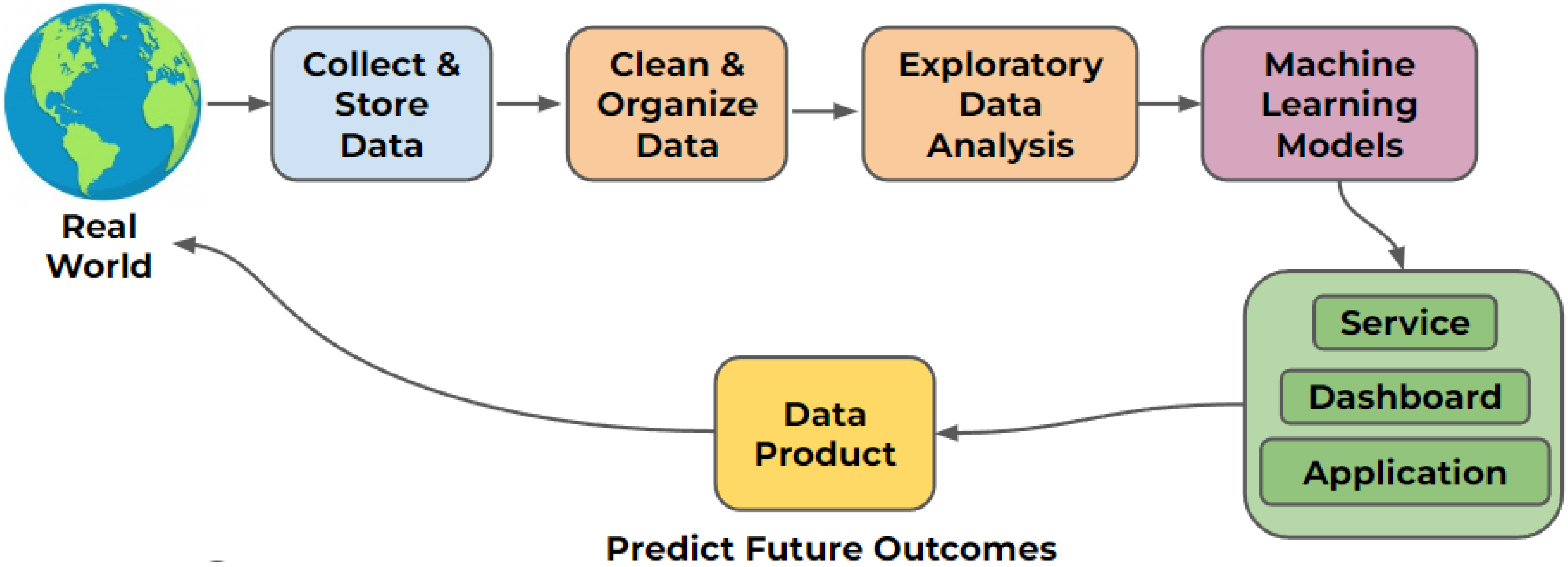
Applied AI From Scratch Using Python

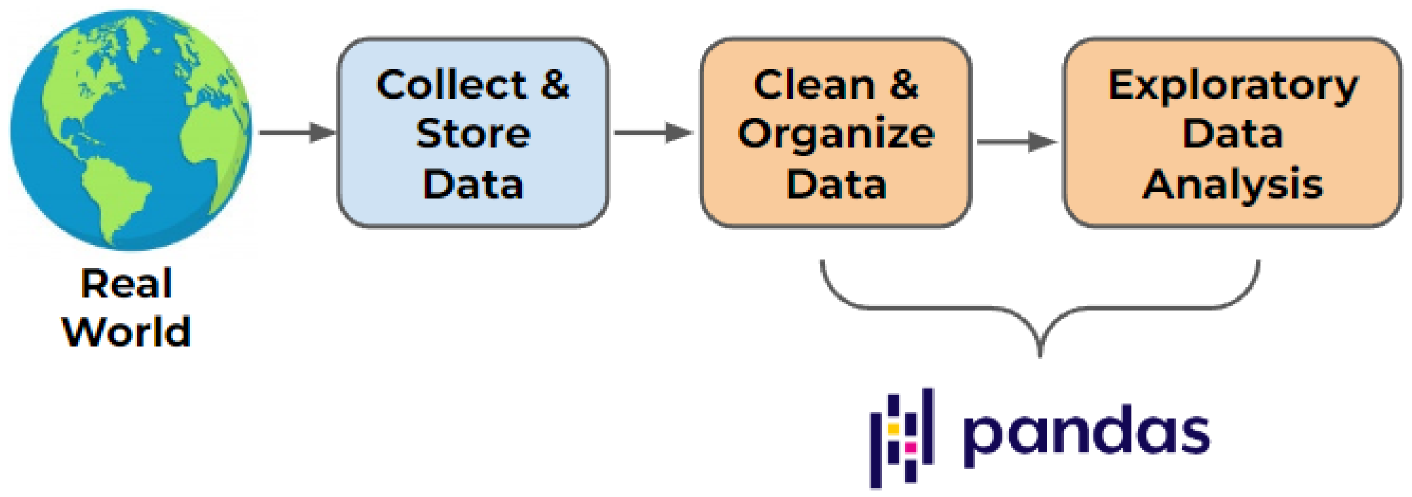
28-31 October 2024


Real World
Collect & Store Data
Clean & Organize Data
Exploratory Data Analysis
Report
Visualization
Communication
Data Analysis
Make Decisions


Real World
Collect & Store Data
Clean & Organize Data
Exploratory Data Analysis
Machine Learning Models
Service
Dashboard
Application
Data Product
Predict Future Outcomes

- What is NumPy?
  - Python library for creating N-dimensional arrays
  - Ability to quickly broadcast functions
  - Built-in linear algebra, statistical distributions, trigonometric, and random number capabilities

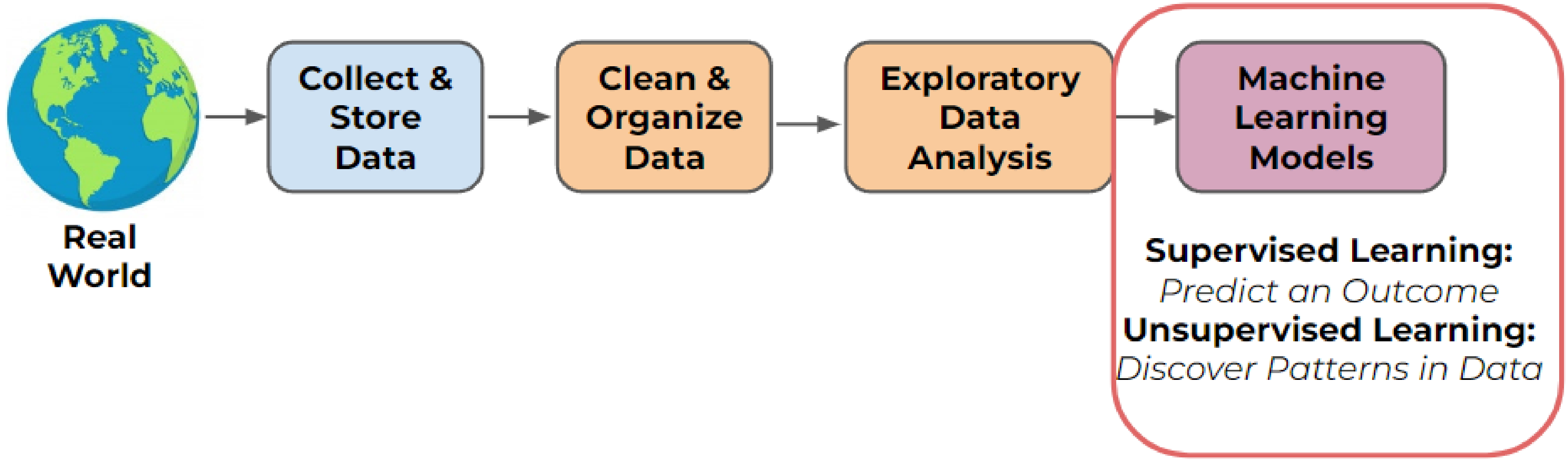
- Why use NumPy?
    - While NumPy structures look similar to standard Python lists, they are **much** more efficient!
    - The broadcasting capabilities are also extremely useful for quickly applying functions to our data sets.


Real World
Collect & Store Data
Clean & Organize Data
Exploratory Data Analysis
pandas

- Pandas is a library for Data Analysis.
- Extremely powerful table (DataFrame) system built off of NumPy.
- Fantastic documentation:
    - https://pandas.pydata.org/docs/

- What can we do with Pandas?
    - Tools for reading and writing data between many formats.
    - Intelligently grab data based on indexing,logic, subsetting, and more.
    - Handle missing data.
    - Adjust and restructure data.

- Series and DataFrames
- Conditional Filtering and Useful Methods
- Missing Data
- Group By Operations
- Combining DataFrames
- Text Methods and Time Methods
- Inputs and Outputs


Real World
Collect & Store Data
Clean & Organize Data
Exploratory Data Analysis
Machine Learning Models
Supervised Learning:
Predict an Outcome
Unsupervised Learning:
Discover Patterns in Data

- Our main goals in ML Overview section:
    - Problems solved by Machine Learning
    - Types of Machine Learning
        - Supervised Learning
        - Unsupervised Learning
    - ML Process for Supervised Learning

* Many other relevant topics will be discussed later in the course as we “discover” them, including:
    * Bias-Variance Trade-off
    * Cross-validation
    * Feature Engineering
    * Scikit-learn
    * Performance Metrics and much more!

- Machine learning in general is the study of statistical computer algorithms that improve automatically through data.
- This means unlike typical computer algorithms that rely on human input for what approach to take, ML algorithms infer best approach from the data itself.

- Machine learning is a subset of Artificial Intelligence.
- ML algorithms are not explicitly programmed on which decisions to make.
- Instead the algorithm is designed to infer from the data the most optimal choices to make.

- What kinds of problems can ML solve?
  - Credit Scoring
  - Insurance Risk
  - Price Forecasting
  - Spam Filtering
  - Customer Segmentation
  - Much more!

- Structure of ML Problem framing:
    - Given **features** from a data set **obtain** a desired **label**.
    - ML algorithms are often called "estimators" since they are estimating the desired **label** or output.

- How can ML be so robust in solving all sorts of problems?
- Machine learning algorithms rely on data and a set of statistical methods to learn what features are important in data.

- Simple Example:
    - Predict the price a house should sell at given its current features (Area,Bedrooms,Bathrooms,etc...)

- House Price Prediction
    - Typical Algorithm
        - Human user defines an algorithm to manually set values of importance for each feature.

- House Price Prediction
    - ML Algorithm
        - Algorithm automatically determines importance of each feature from existing data

- Why machine learning?
    - Many complex problems are only solvable with machine learning techniques.
    - Problems such as spam email or handwriting identification require ML for an effective solution.

- Why not just use machine learning for everything?
    - Major caveat to effective ML is good data.
    - Majority of development time is spent cleaning and organizing data, **not** implementing ML algorithms.

- There are two main types of Machine Learning we will cover in upcoming sections:
    - Supervised Learning
    - Unsupervised Learning

- Supervised Learning
    - Using **historical** and **labeled** data, the machine learning model predicts a value.
- Unsupervised Learning
    - Applied to **unlabeled** data, the machine learning model discovers possible patterns in the data.

- Supervised Learning
    - Requires **historical labeled** data:
        - Historical
            - Known results and data from the past.
        - Labeled
            - The desired output is known.

- Supervised Learning
  - Two main label types
    - Categorical Value to Predict
      - Classification Task
    - Continuous Value to Predict
      - Regression Task

- Supervised Learning
    - Classification Tasks
        - Predict an assigned category
            - Cancerous vs. Benign Tumor
            - Fulfillment vs. Credit Default
            - Assigning Image Category
                - Handwriting Recognition

- Supervised Learning
    - Regression Tasks
        - Predict a continuous value
            - Future prices
            - Electricity loads
            - Test scores

- Unsupervised Learning
    - Group and interpret data without a label.
    - Example:
        - Clustering customers into separate groups based off their behaviour features.

- Unsupervised Learning
    - Major downside is because there was no historical “correct” label, it is much harder to evaluate performance of an unsupervised learning algorithm.

- Machine Learning Sections
    - We first focus on supervised learning to build an understanding of machine learning capabilities.
    - Then shift focus to unsupervised learning for clustering and dimensionality reduction.

- Machine Learning Pathway

Real World

Collect & Store Data → **Clean & Organize Data** → **Exploratory Data Analysis** → **Machine Learning Models**

Jupyter, NumPy, Pandas, Matplotlib, Seaborn

Supervised Learning: *Predict an Outcome*

Unsupervised Learning: *Discover Patterns in Data*

Scikit-learn

Machine Learning

- Machine Learning Pathway

Real Worl d

Collect & Store Data

Clean & Organi ze Data

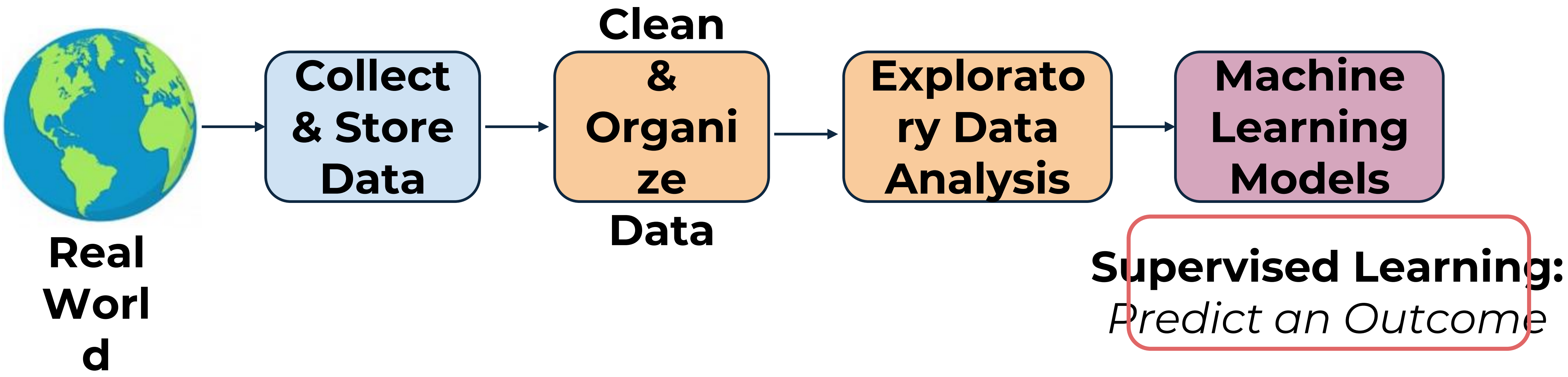
Explorato ry Data Analysis

Machine Learning Models

Supervised Learning:
Predict an Outcome

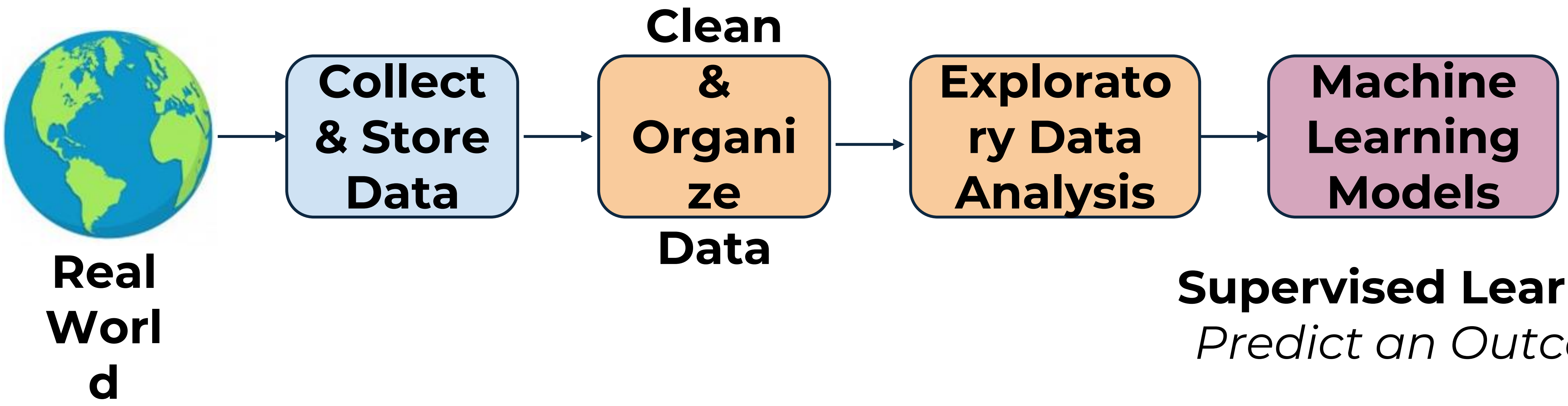
Machine Learning

- ML Process : Supervised Learning Tasks



Machine Learning

- **Predict price a house should sell at.**



Supervised Learning:
Predict an Outcome

Machine Learning

- **Supervised** Machine Learning Process
- Start with collecting and organizing a data set based on history:

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000

Machine Learning

- **Historical labeled** data on previously sold houses.

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000

Machine Learning

- If a new house comes on the market with a known Area, Bedrooms, and Bathrooms: *Predict what price should it sell at.*

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000

Machine Learning

- Data Product:
  - Input house features
  - Output predicted selling price

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000

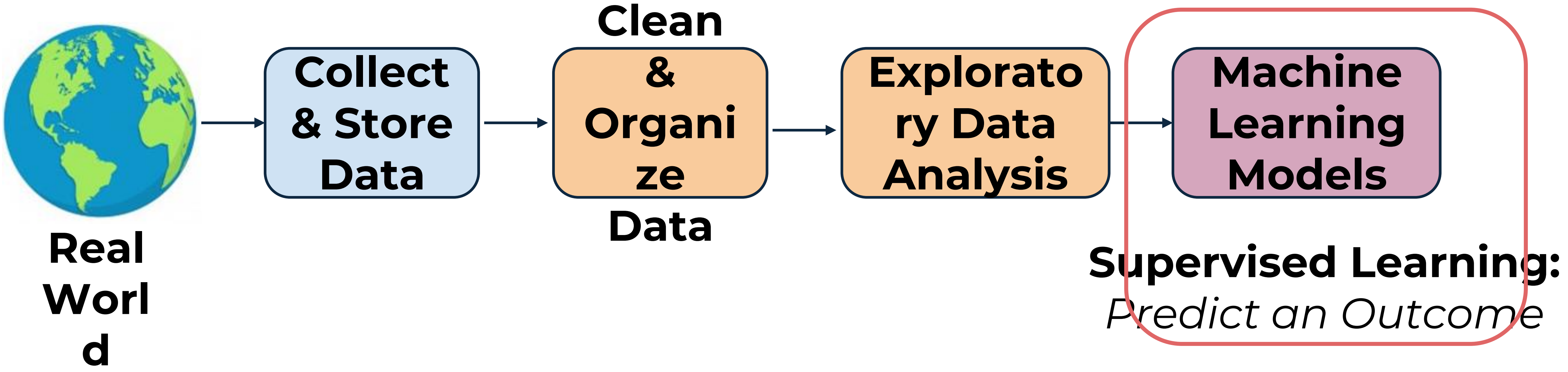
Machine Learning

- Using **historical, labeled** data predict a future outcome or result.

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000

Machine Learning

- **Predict price a house should sell at.**



Machine Learning

- **Predict price a house should sell at.**

Machine Learning Models

Supervised Learning:
Predict an Outcome

Machine Learning

- **Predict price a house should sell at.**

Machine Learning Models

Supervised Learning:
Predict an Outcome

Machine Learning

- **Predict price a house should sell at.**

Machine Learning Models

Supervised Learning:
Predict an Outcome

Data

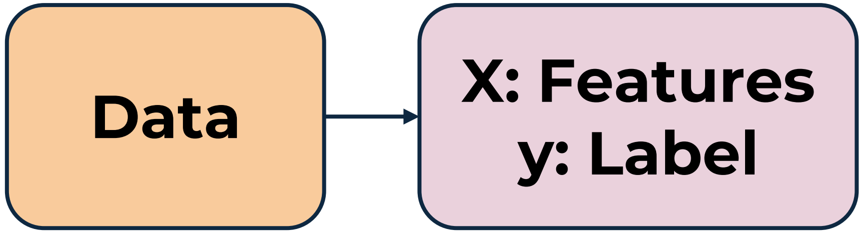
Machine Learning

- **Supervised** Machine Learning Process

Data

Machine Learning

- **Supervised** Machine Learning Process



Machine Learning

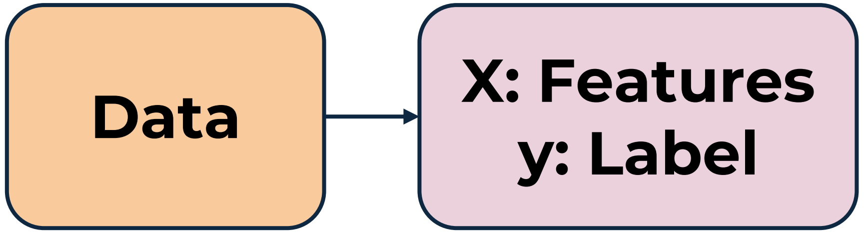
- **Supervised** Machine Learning Process

Data → **X: Features**
y: Label

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Machine Learning

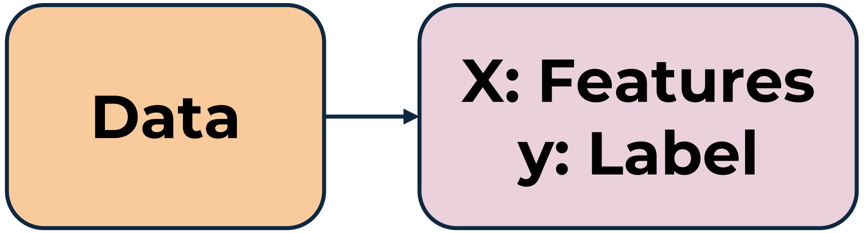
- **Label** is what we are trying to predict



Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Machine Learning

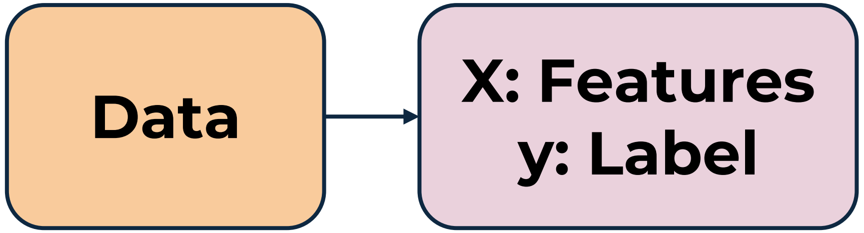
- **Label** is what we are trying to predict



Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Machine Learning

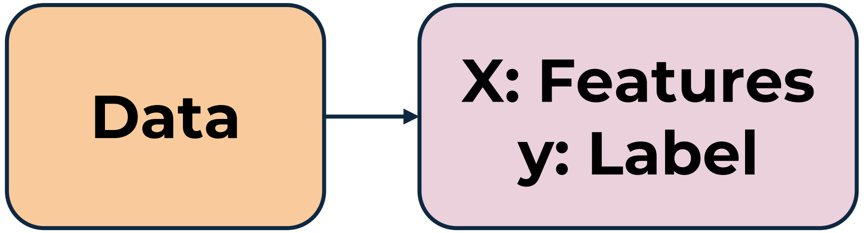
- **Features** are known characteristics or components in the data



Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Machine Learning

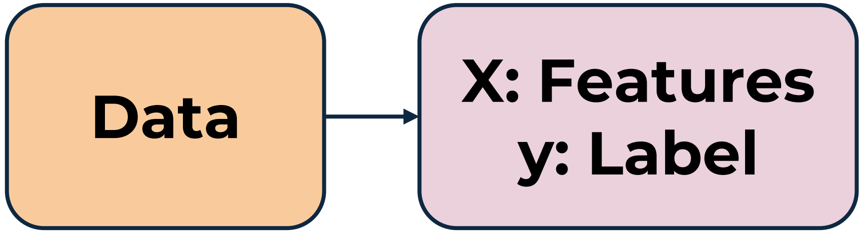
- **Features** are known characteristics or components in the data



Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Machine Learning

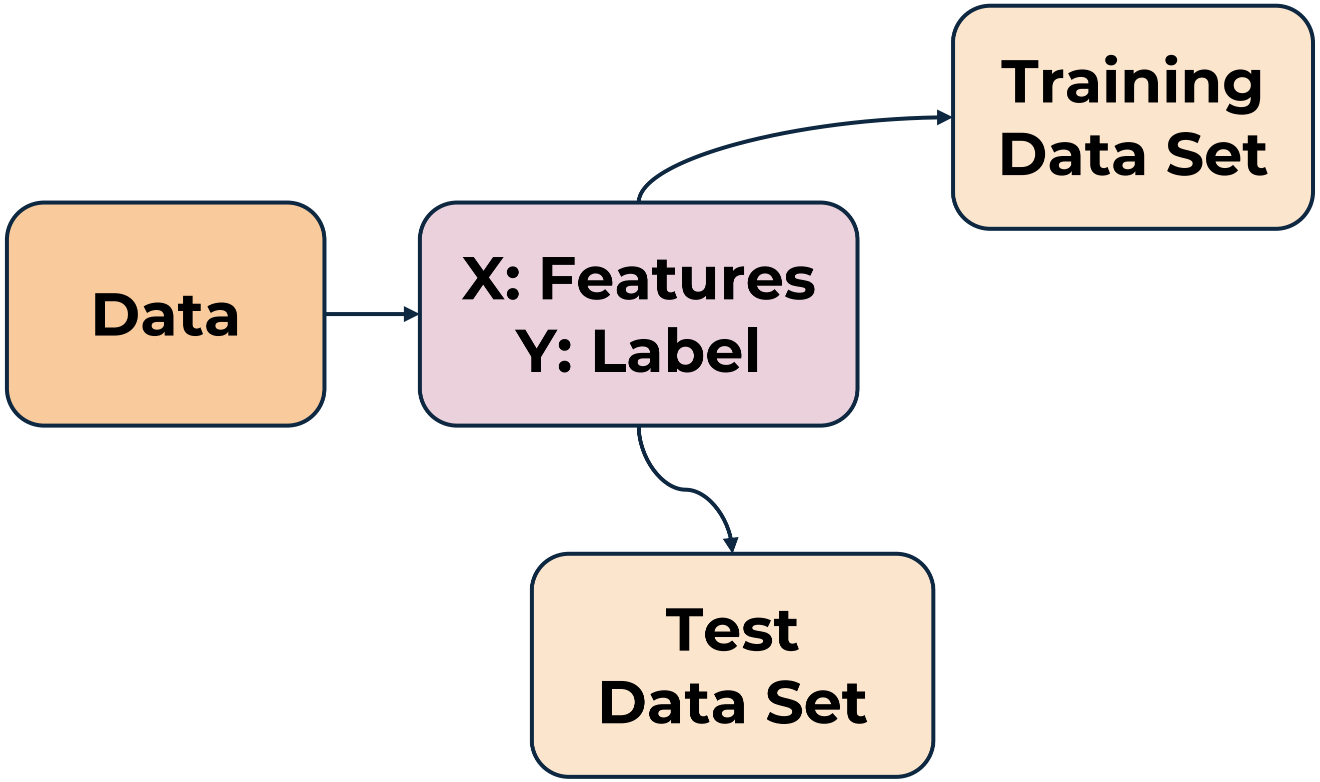
- **Features** and **Label** are identified according to the problem being solved.



Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

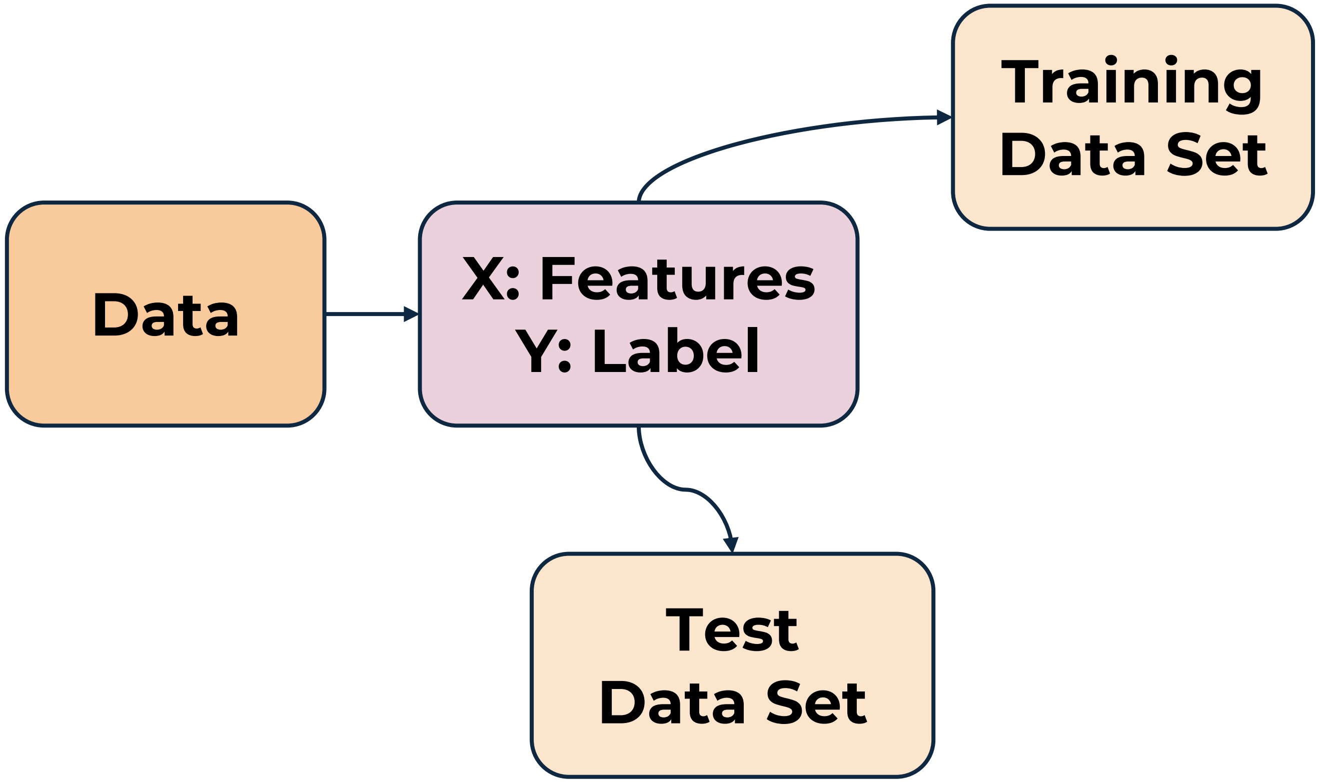
Supervised Machine Learning Process

- Split data into training set and test set



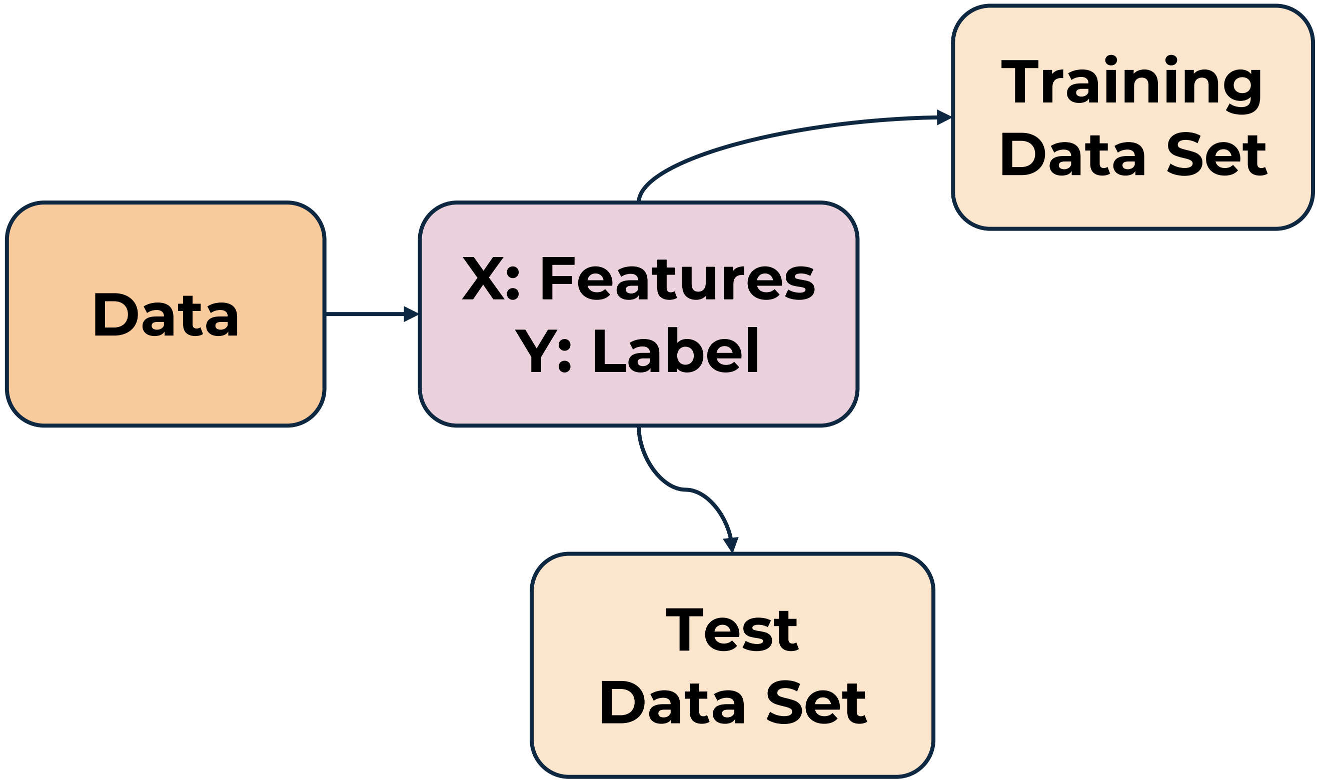
Supervised Machine Learning Process

- Later on we will discuss cross-validation



Supervised Machine Learning Process

- Why perform this split? How to split?



Supervised Machine Learning Process

- Why perform this split? How to split?

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000

Supervised Machine Learning Process

- How would you judge a human realtor's performance?

Area m^2	**Bedrooms**	**Bathrooms**	**Price**
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	[illegible]	[illegible]	$550,000

Supervised Machine Learning Process

- Ask a human realtor to take a look at historical data...

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Supervised Machine Learning Process

- Then give her the features of a house and ask her to predict a selling price.

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Supervised Machine Learning Process

- But how would you measure how accurate her prediction is? What house should you choose to test on?

Area m^2	**Bedrooms**	**Bathrooms**	**Price**
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Supervised Machine Learning Process

- You can’t judge her based on a new house that hasn’t sold yet, you don’t know it’s true selling price!

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Supervised Machine Learning Process

- You shouldn't judge her on data she's already seen, she could have **memorized** it!

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	[illegible]	[illegible]	$550,000

Supervised Machine Learning Process

- Thus the need for a Train/Test split of the data, let’s explore further…

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Supervised Machine Learning Process

- We already organized the data into Features (X) and a Label (y)

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Supervised Machine Learning Process

- Now we will split this into a training set and a test set:

TRAIN

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Supervised Machine Learning Process

- Now we will split this into a training set and a test set:

	Area m^2	Bedrooms	Bathrooms	Price
TRAIN	200	3	2	$500,000
	190	2	1	$450,000
TEST	230	3	3	$650,000
	180	1	1	$400,000
	210	2	2	$550,000

Supervised Machine Learning Process

- Notice how we have 4 components

	Area m^2	Bedrooms	Bathrooms	Price	
X TRAIN	200	3	2	$500,000	Y TRAIN
	190	2	1	$450,000	
X TEST	230	3	3	$650,000	Y TEST
	180	1	1	$400,000	
	210	2	2	$550,000	

Supervised Machine Learning Process

- Let's go back to fairly testing our human realtor....

Area m^2	**Bedrooms**	**Bathrooms**	**Price**
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
[illegible]	[illegible]	[illegible]	[illegible]

Supervised Machine Learning Process

- Let's go back to fairly testing our human realtor….

	Area m^2	Bedrooms	Bathrooms	Price
TRAIN	200	3	2	$500,000
	190	2	1	$450,000
TEST	230	3	3	$650,000
	180	1	1	$400,000
	210	2	2	$550,000

Supervised Machine Learning Process

- Let her study and learn on the training set getting access to both X and y.

TRAIN

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000

Supervised Machine Learning Process

- After she has “learned” about the data, we can test her skill on the test set.

TEST

Area m^2	Bedrooms	Bathrooms
180	1	1
210	2	2

Supervised Machine Learning Process

- Provide only the X test data and ask for her predictions for the sell price.

TEST

Area m^2	Bedrooms	Bathrooms
180	1	1
210	2	2

Supervised Machine Learning Process

- This is new data she has never seen before! She has also never seen the real sold price.

TEST

Area m^2	Bedrooms	Bathrooms
180	1	1
210	2	2

Supervised Machine Learning Process

- Ask for predictions per data point.

Predictions	Area m^2	Bedrooms	Bathrooms
$410,000	180	1	1
$540,000	210	2	2

Supervised Machine Learning Process

- Then bring back the original prices.

Predictions	Area m^2	Bedrooms	Bathrooms	Price
$410,000	180	1	1	$400,000
$540,000	210	2	2	$550,000

Supervised Machine Learning Process

- Finally compare predictions against true test price.

Predictions
$410,000
$540,000

Price
$400,000
$550,000

Supervised Machine Learning Process

- This is often labeled as $\hat{y}$ compared again y

$\hat{y}$

Predictions
$410,000
$540,000

y

Price
$400,000
$550,000

Supervised Machine Learning Process

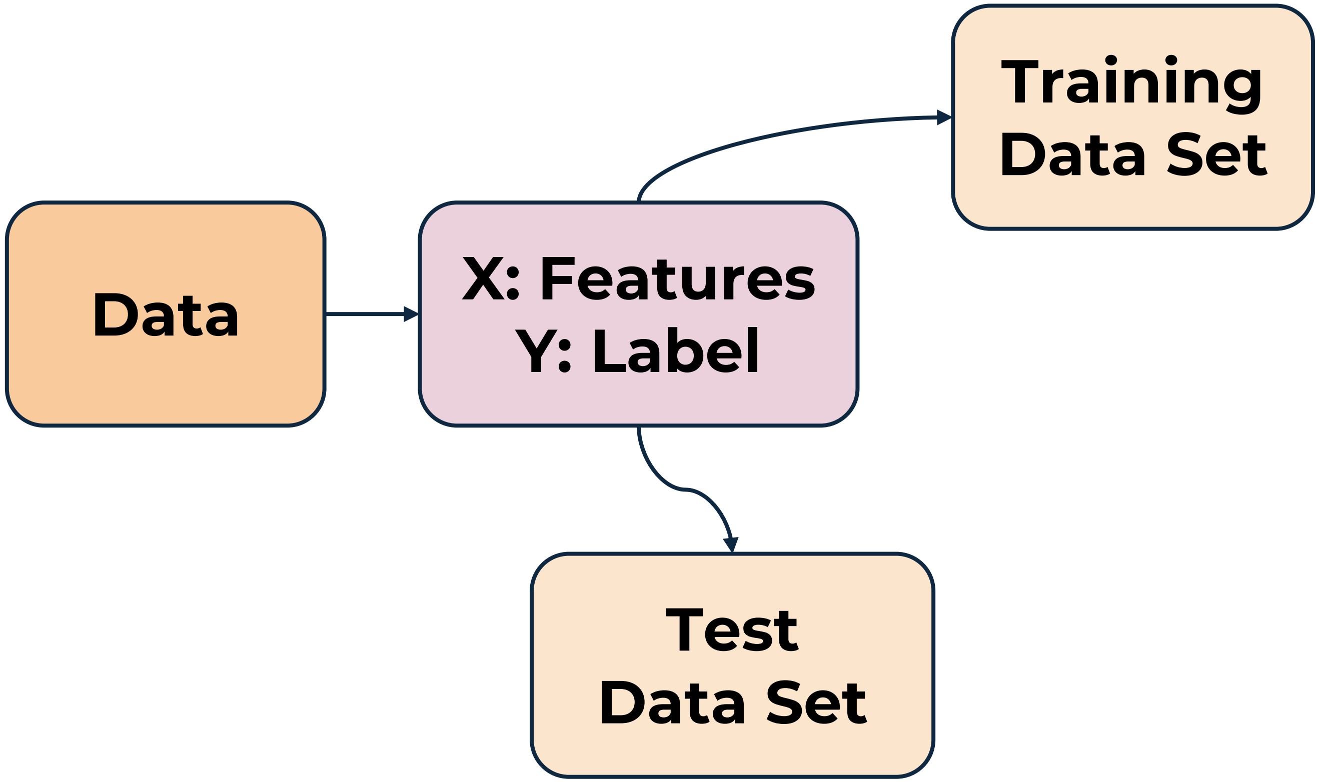
- Later on we will discuss the many methods of evaluating this performance!

Predictions
$410,000
$540,000

Price
$400,000
$550,000

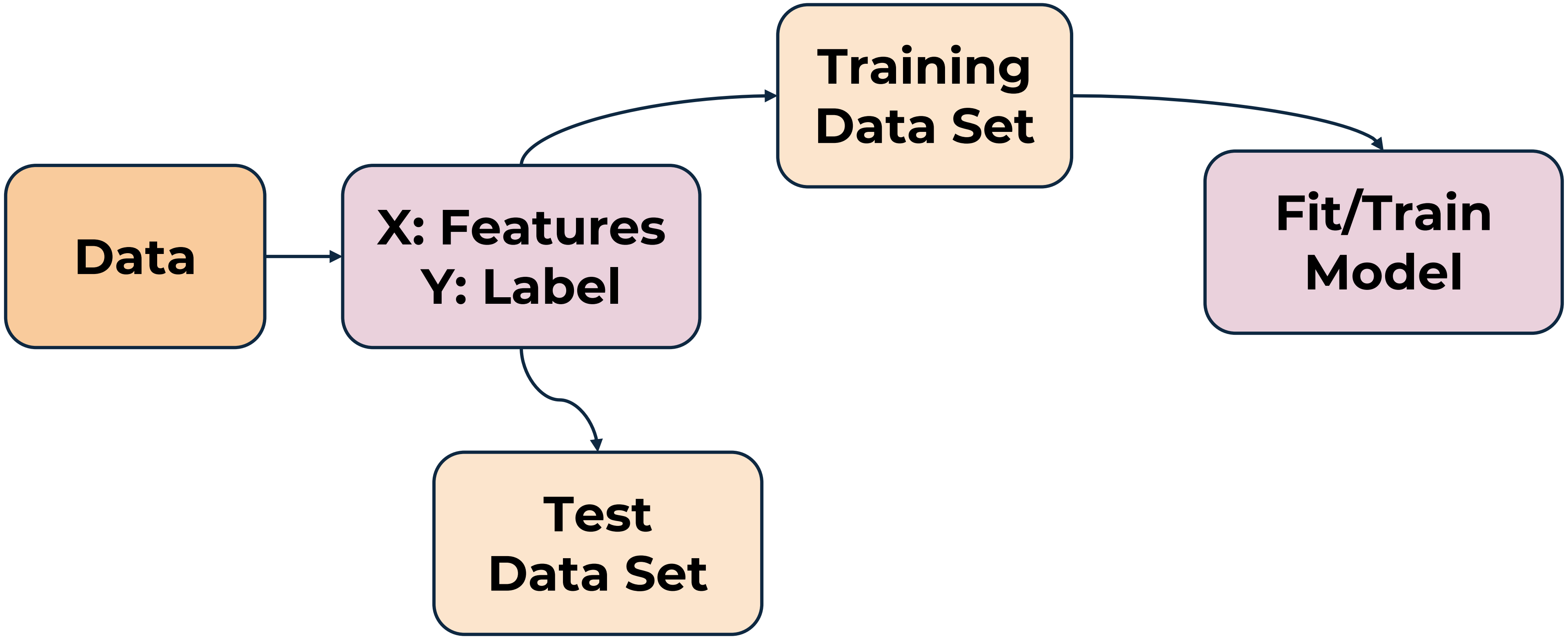
Supervised Machine Learning Process

- Split Data



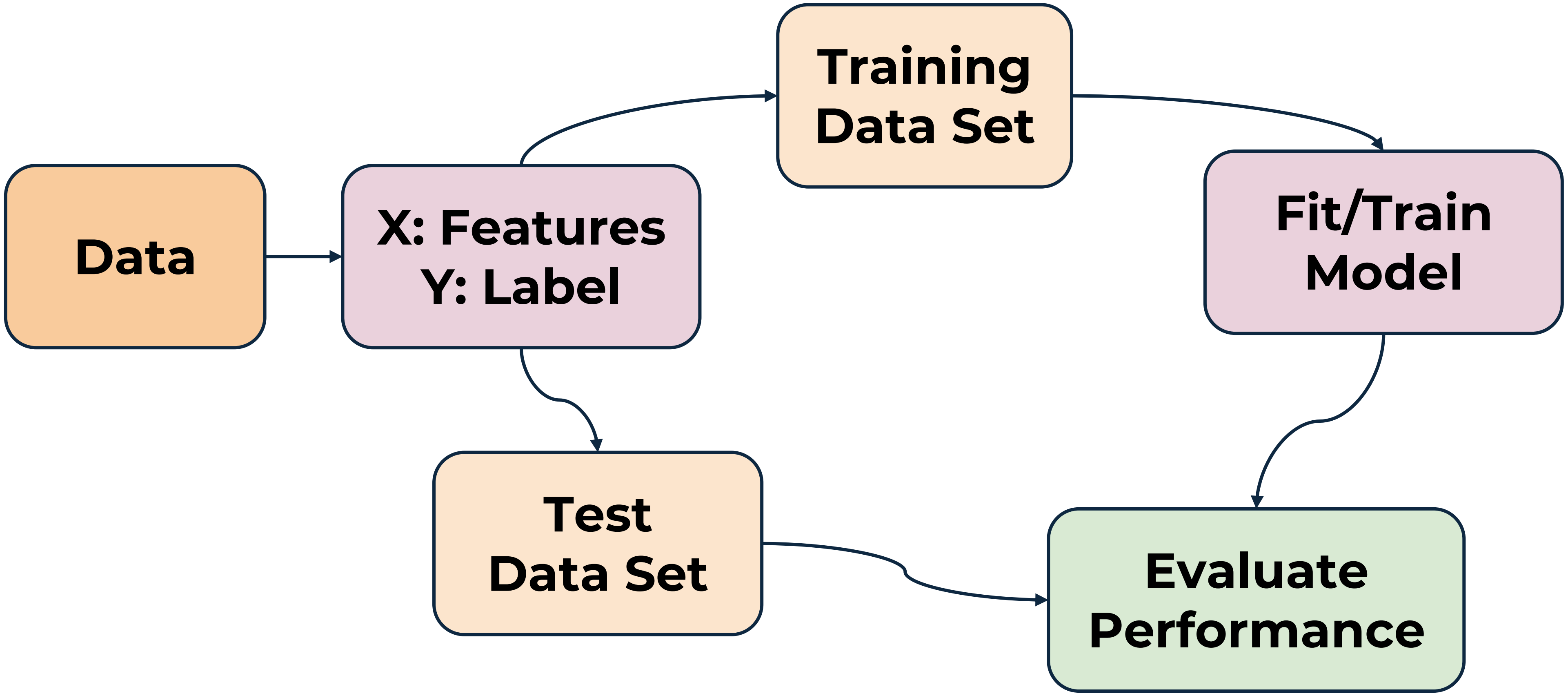
Supervised Machine Learning Process

- Split Data, Fit on Train Data



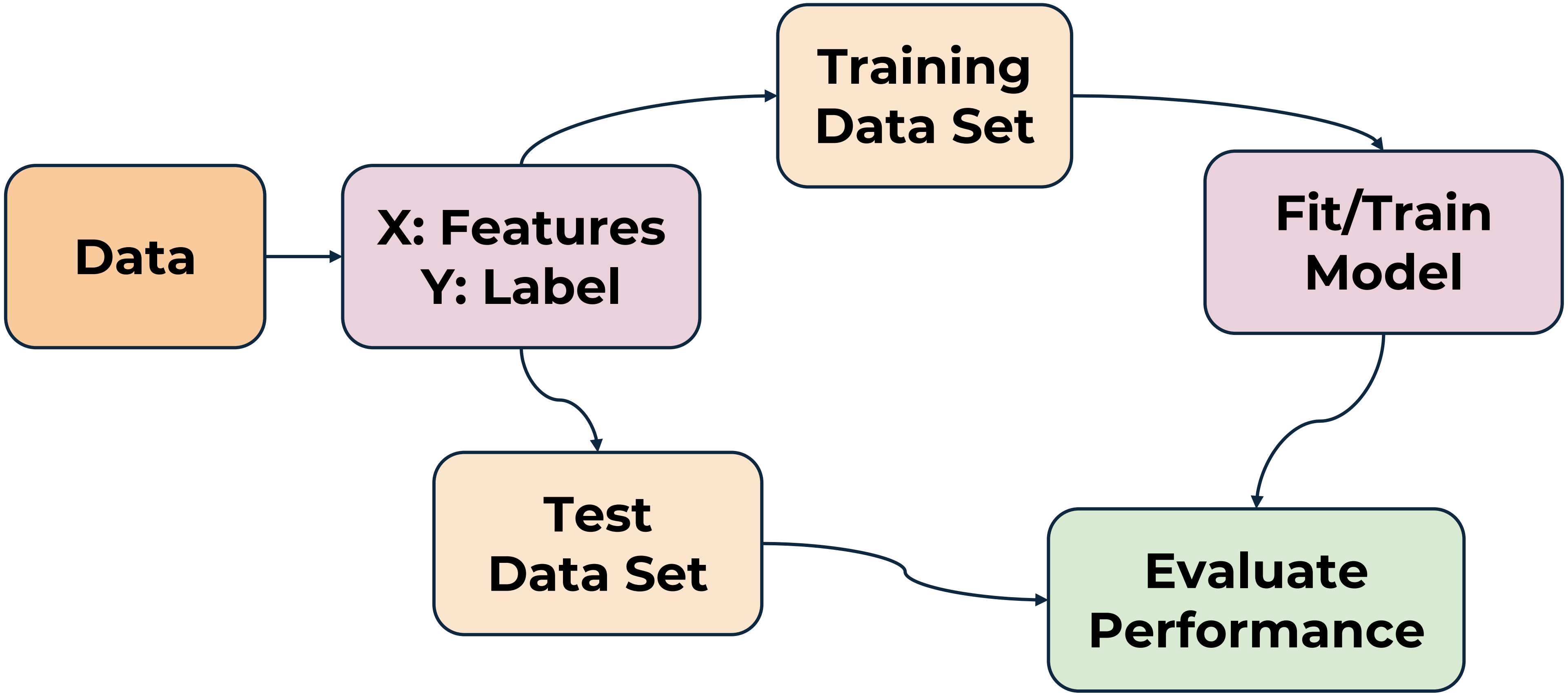
Supervised Machine Learning Process

- Split Data, Fit on Train Data,Evaluate Model



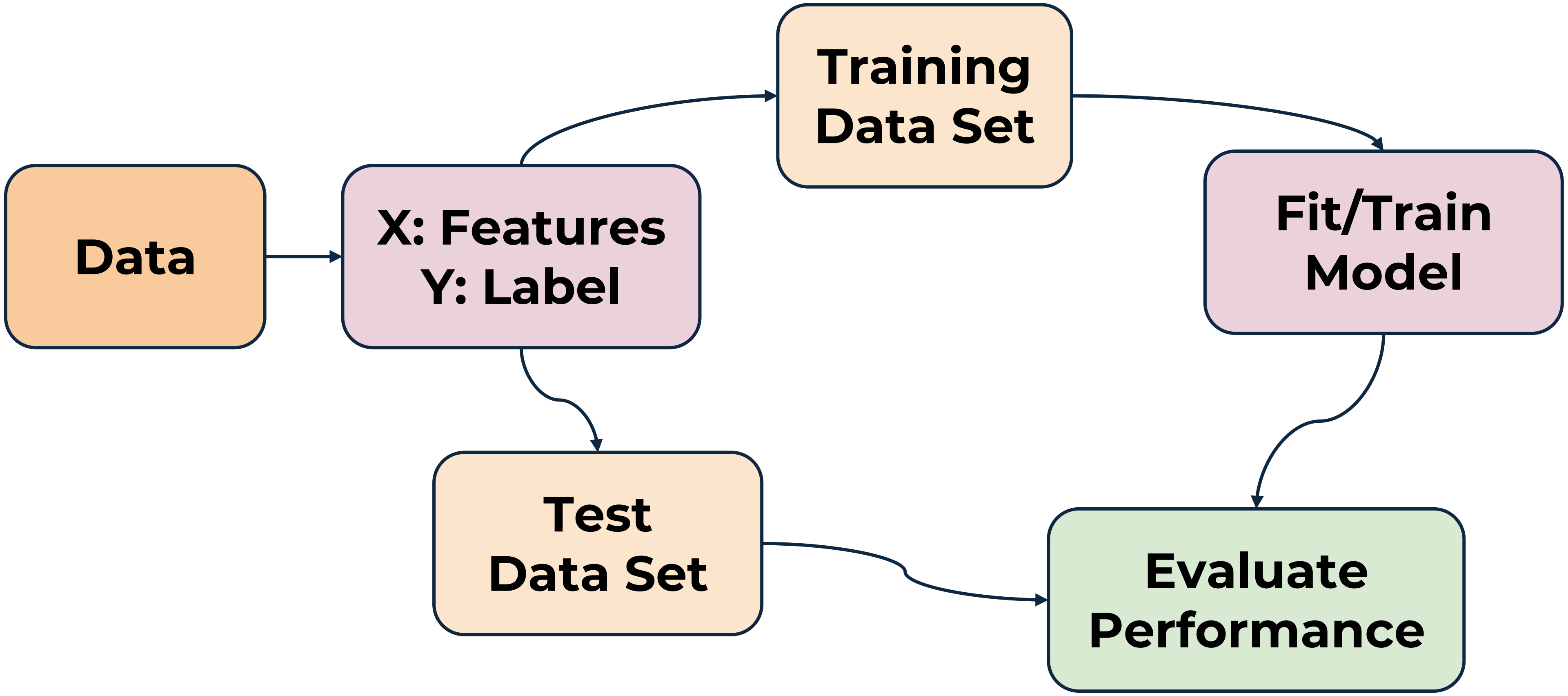
Supervised Machine Learning Process

- What happens if performance isn't great?



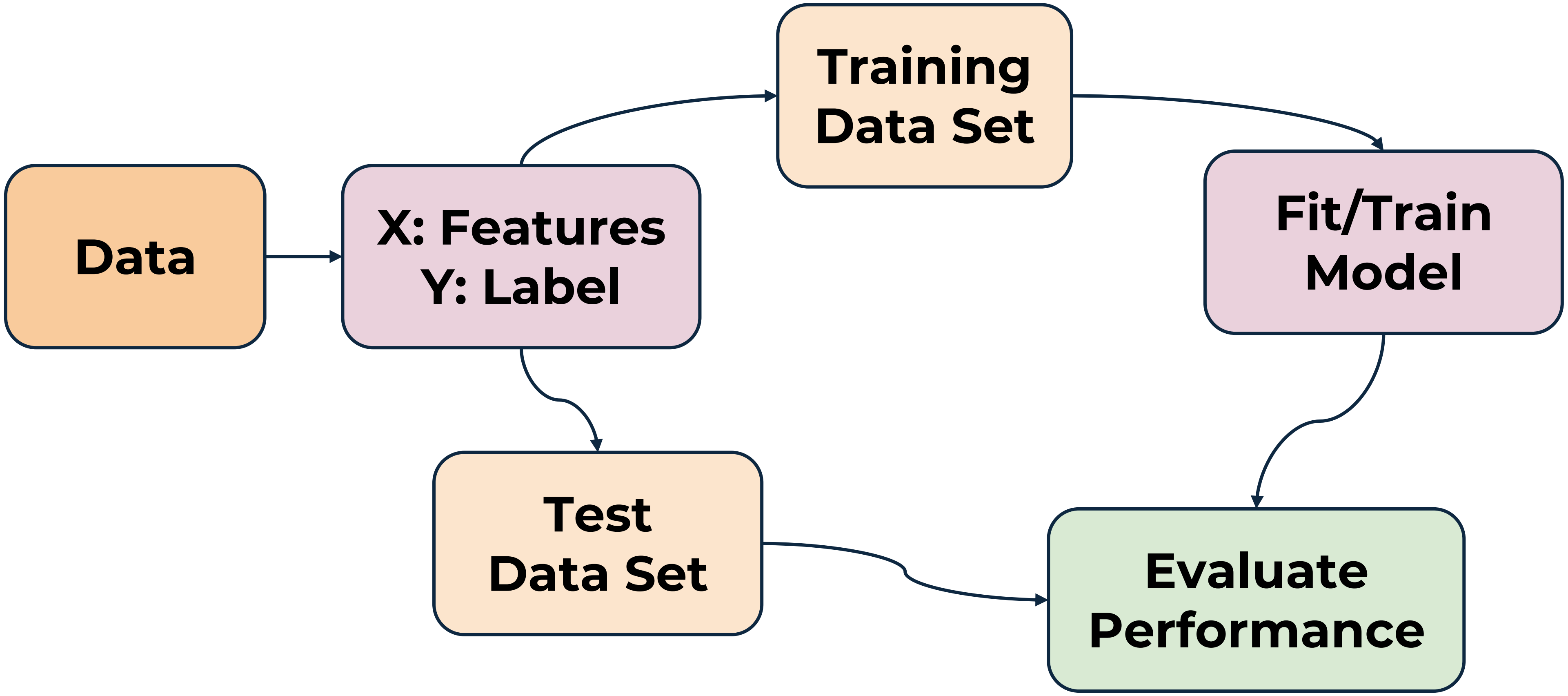
Supervised Machine Learning Process

- We can adjust model **hyperparameters**



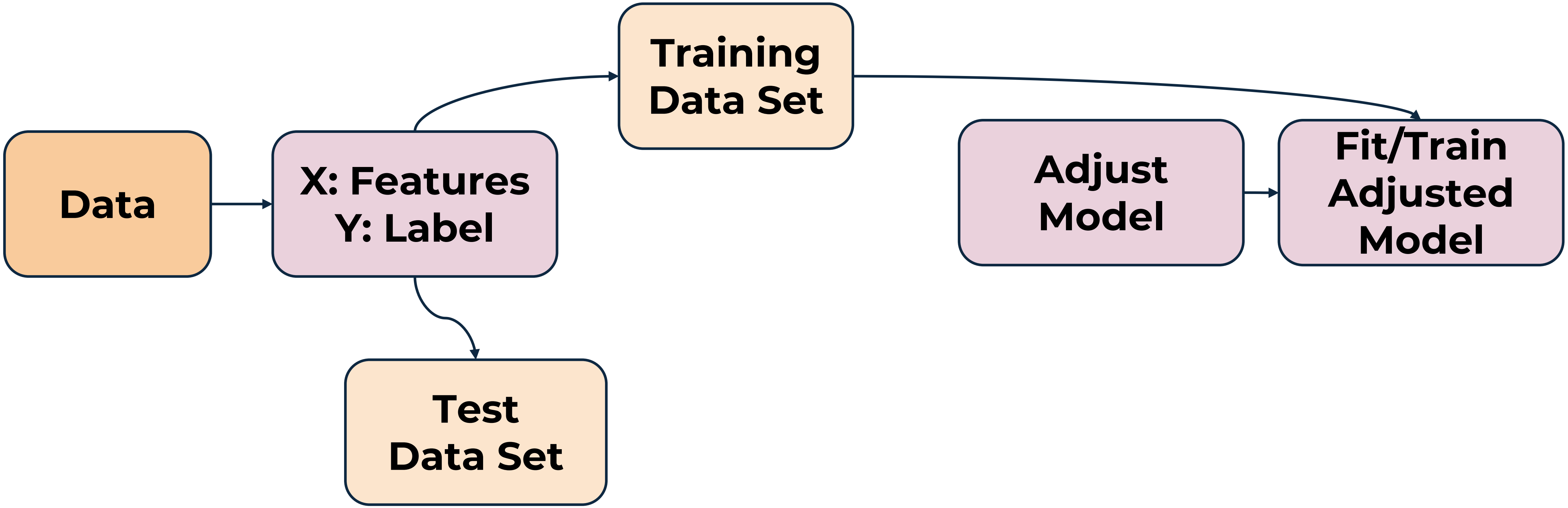
Supervised Machine Learning Process

- Many algorithms have adjustable values



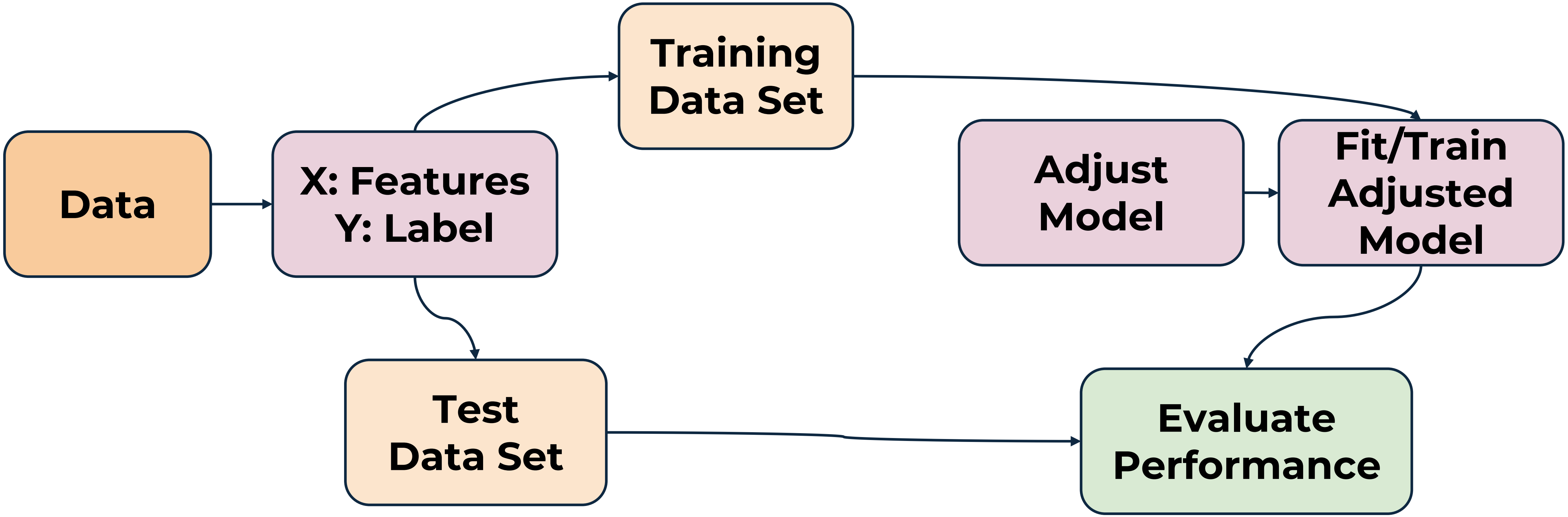
Supervised Machine Learning Process

- Many algorithms have adjustable values



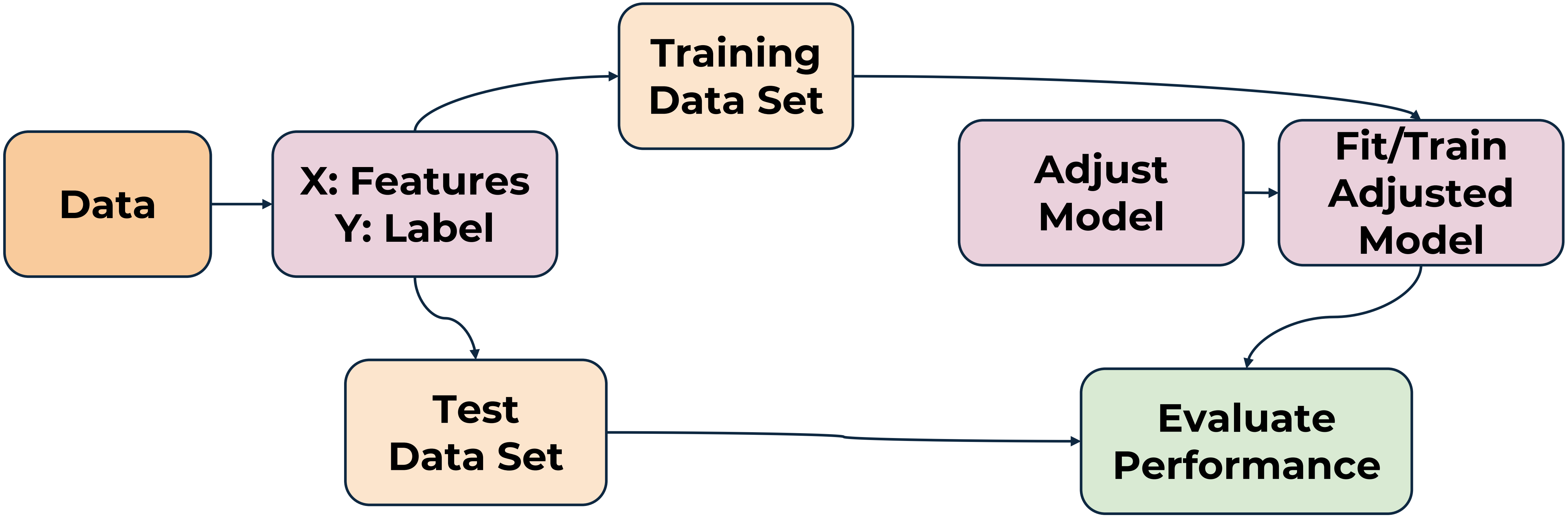
Supervised Machine Learning Process

- Evaluate adjusted model



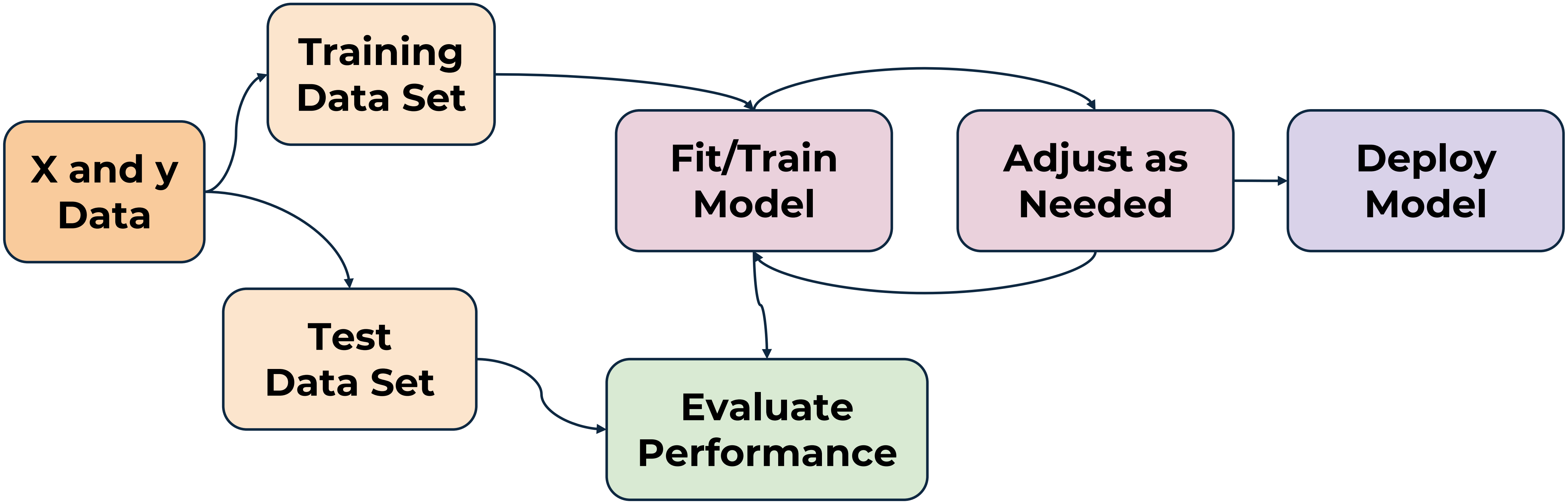
Supervised Machine Learning Process

- Can repeat this process as necessary



Supervised Machine Learning Process

- Full and Simplified Process



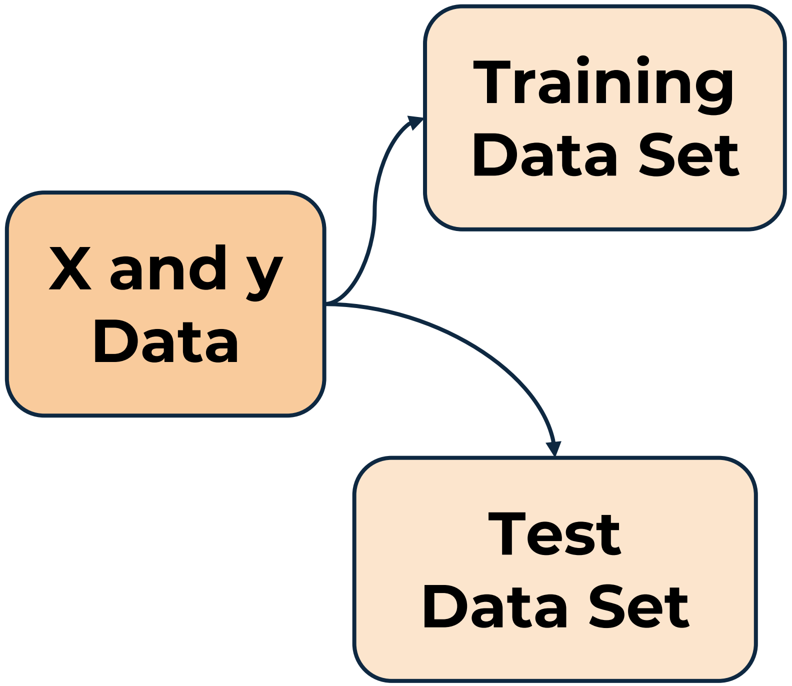
Supervised Machine Learning Process

- Get X and y data

X and y Data

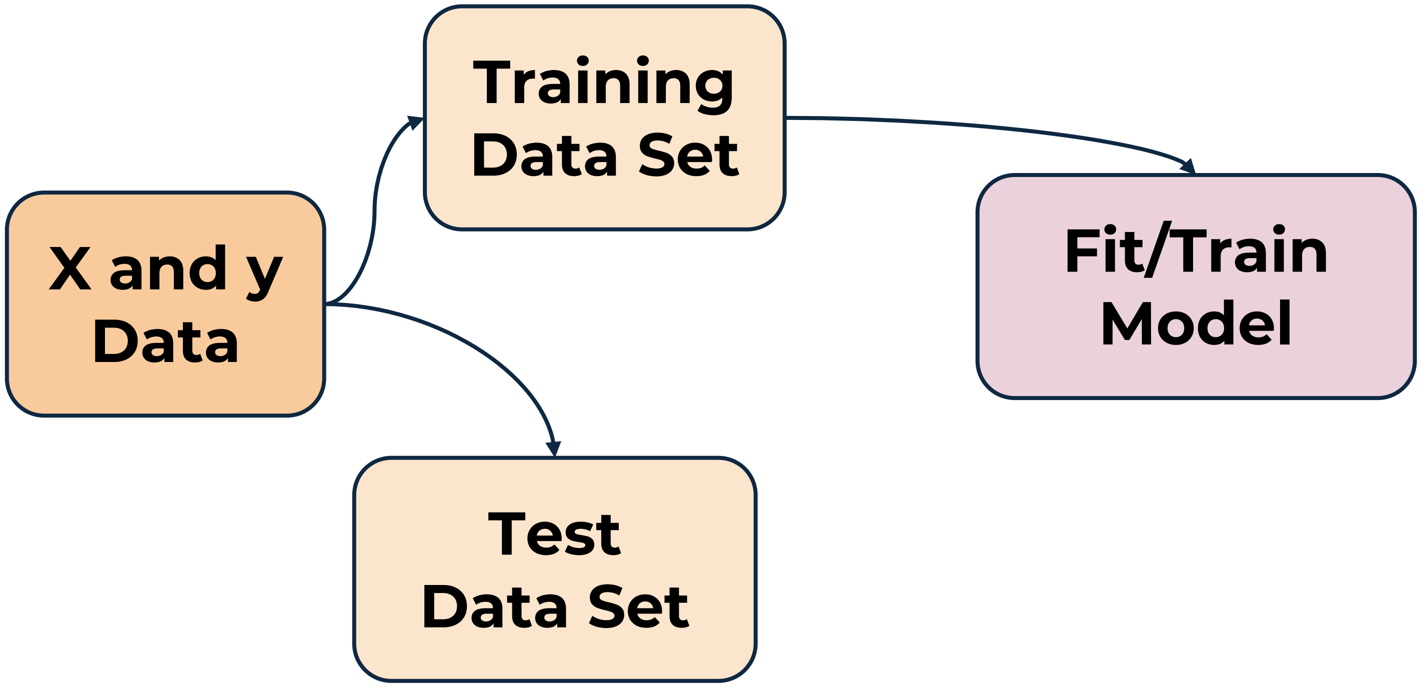
Supervised Machine Learning Process

- Split data for evaluation purposes



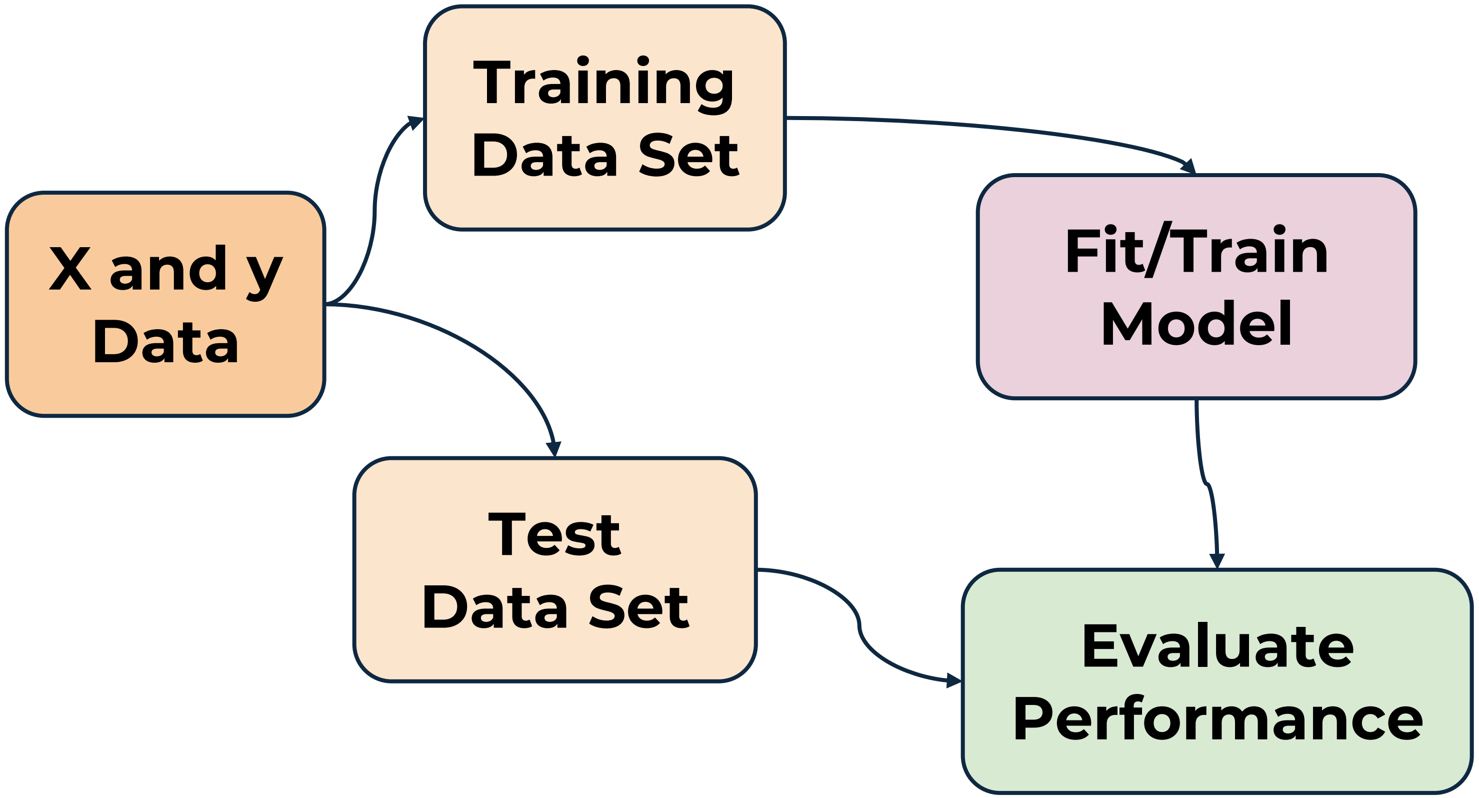
Supervised Machine Learning Process

- Fit ML Model on Training Data Set



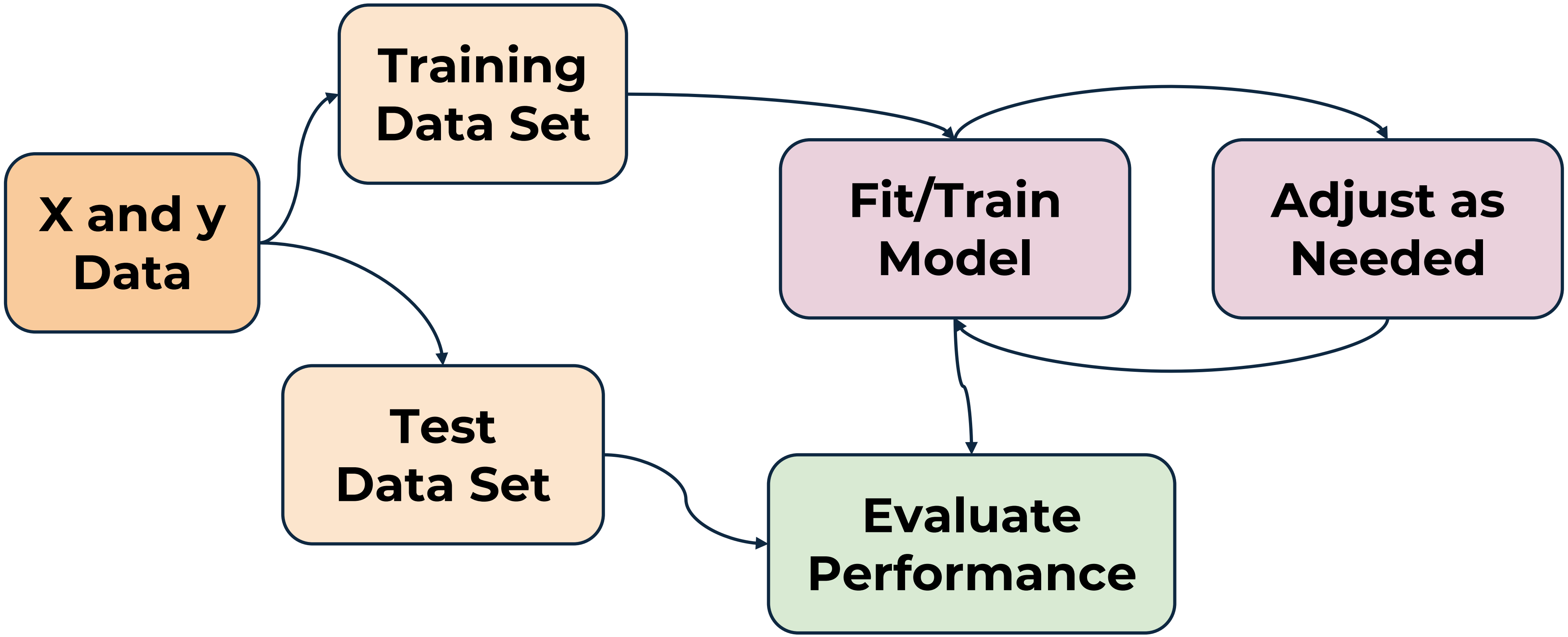
Supervised Machine Learning Process

- Evaluate Model Performance



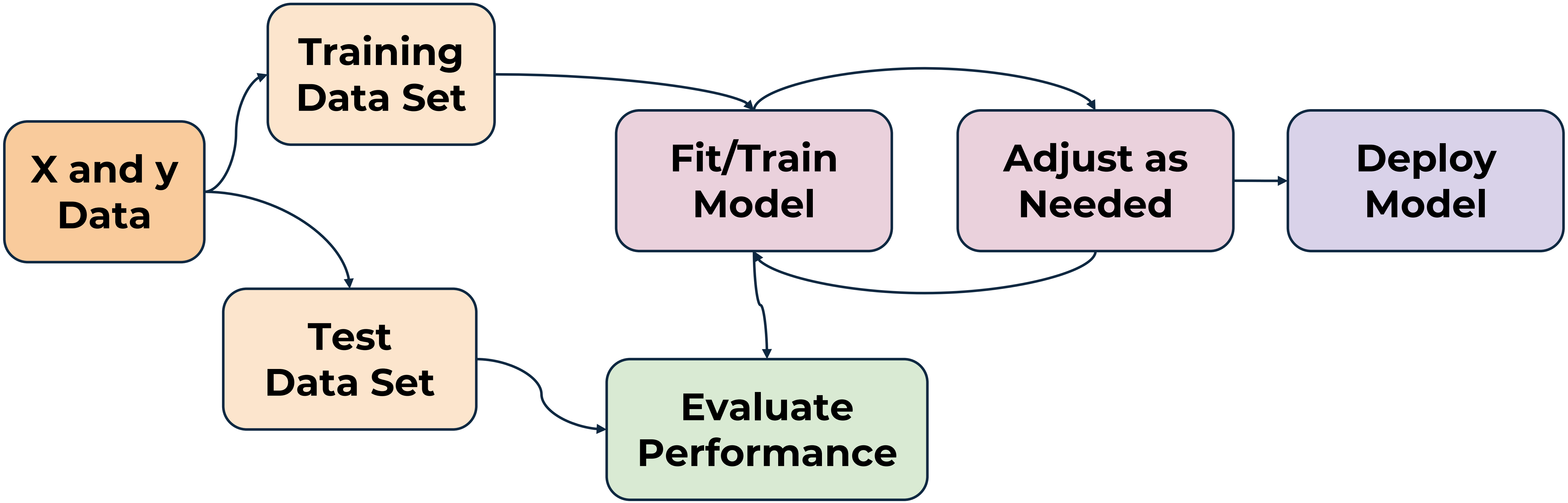
Supervised Machine Learning Process

- Adjust model hyperparameters as needed



Supervised Machine Learning Process

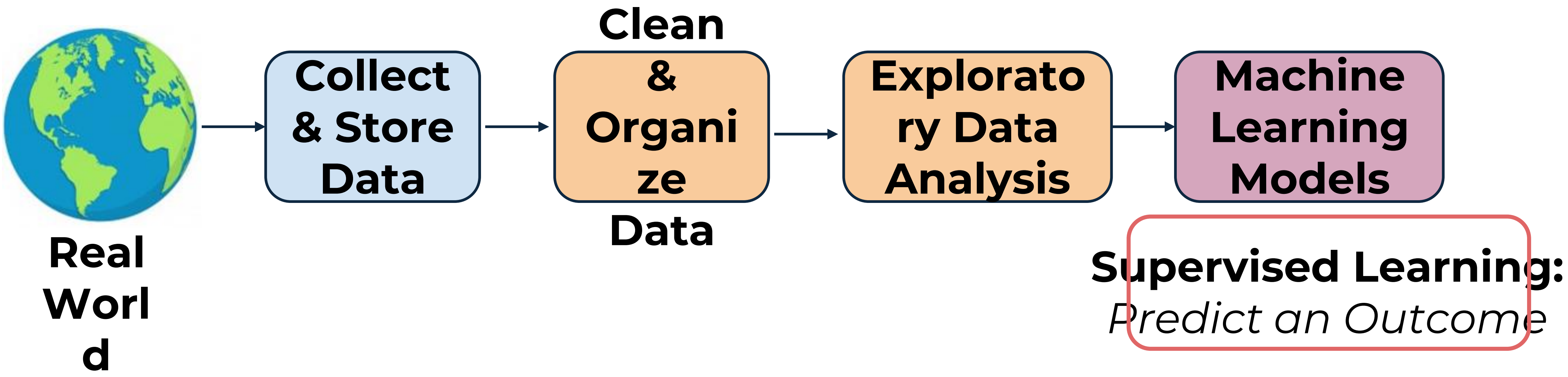
- Deploy model to real world



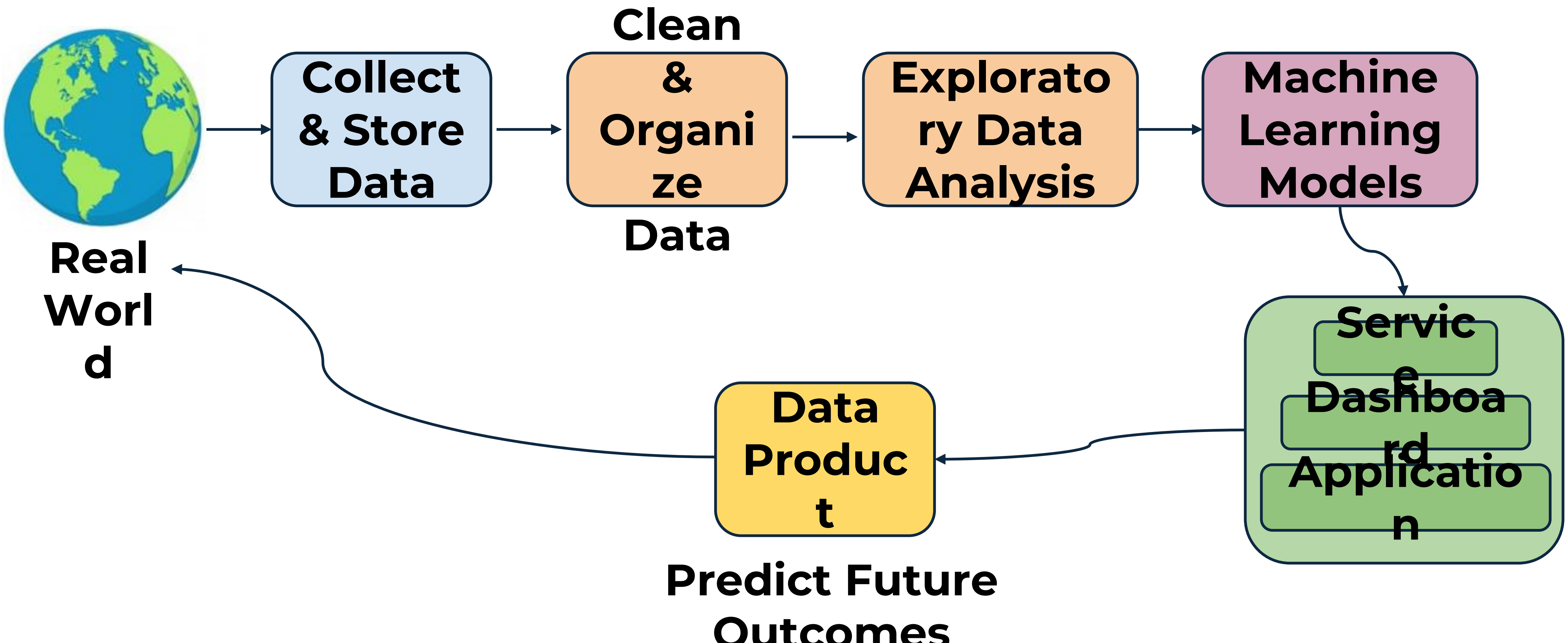
PIERIAN DATA

Machine Learning

- ML Process : Supervised Learning Tasks



ML Pathway


Collect & Store Data
Clean & Organize Data
Exploratory Data Analysis
Machine Learning Models
Service
Dashboard
Application
Data Product
Predict Future Outcomes
Real World

Evaluating Performance

CLASSIFICATION

Model Evaluation

- We just learned that after our machine learning process is complete, we will use performance metrics to evaluate how our model did.
- Let’s discuss classification metrics in more detail!

Model Evaluation

- The key classification metrics we need to understand are:
    - Accuracy
    - Recall
    - Precision
    - F1-Score

Model Evaluation

- But first, we should understand the reasoning behind these metrics and how they will actually work in the real world!

Model Evaluation

- Typically in any classification task your model can only achieve two results:
  - Either your model was **correct** in its prediction.
  - Or your model was **incorrect** in its prediction.

Model Evaluation

- Fortunately incorrect vs correct expands to situations where you have multiple classes.
- For the purposes of explaining the metrics, let’s imagine a **binary classification** situation, where we only have two available classes.

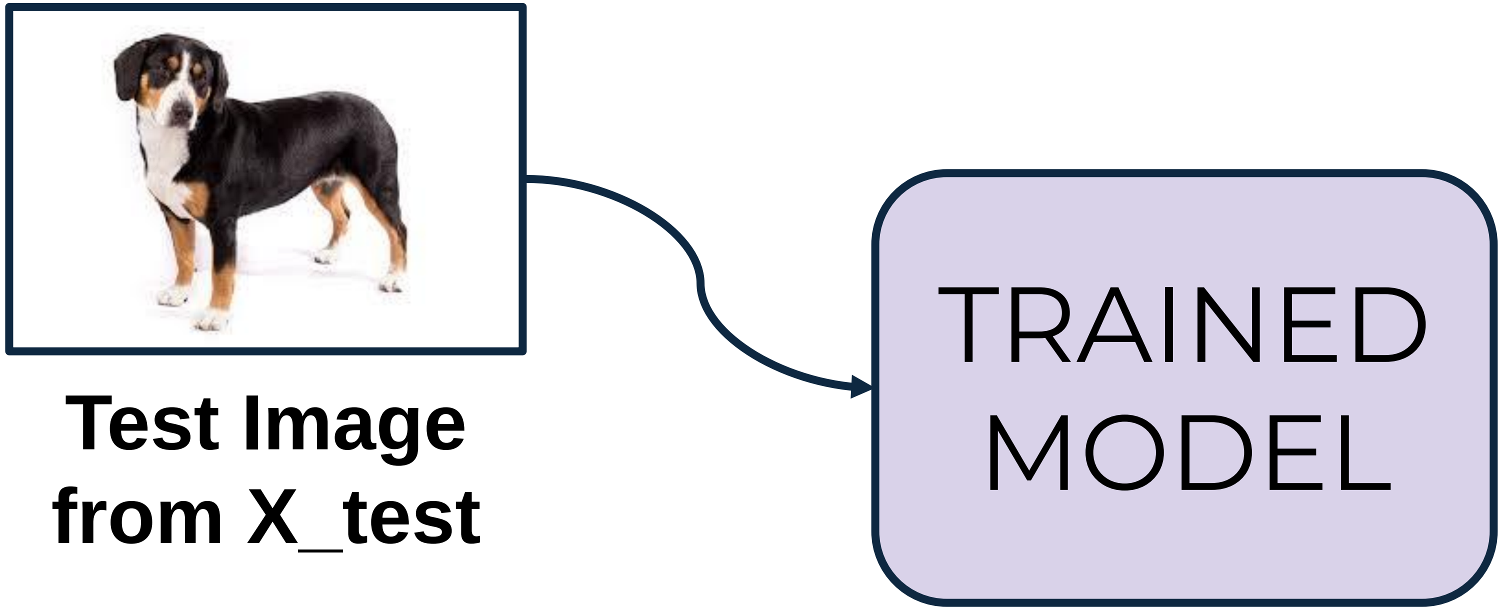
Model Evaluation

- In our example, we will attempt to predict if an image is a dog or a cat.
- Since this is supervised learning, we will first **fit/train** a model on **training data**, then **test** the model on **testing data**.
- Once we have the model’s predictions from the **X_test** data, we compare it to the **true y values** (the correct labels).

Model Evaluation

TRAINED MODEL

Model Evaluation



Test Image from X_test

Model Evaluation

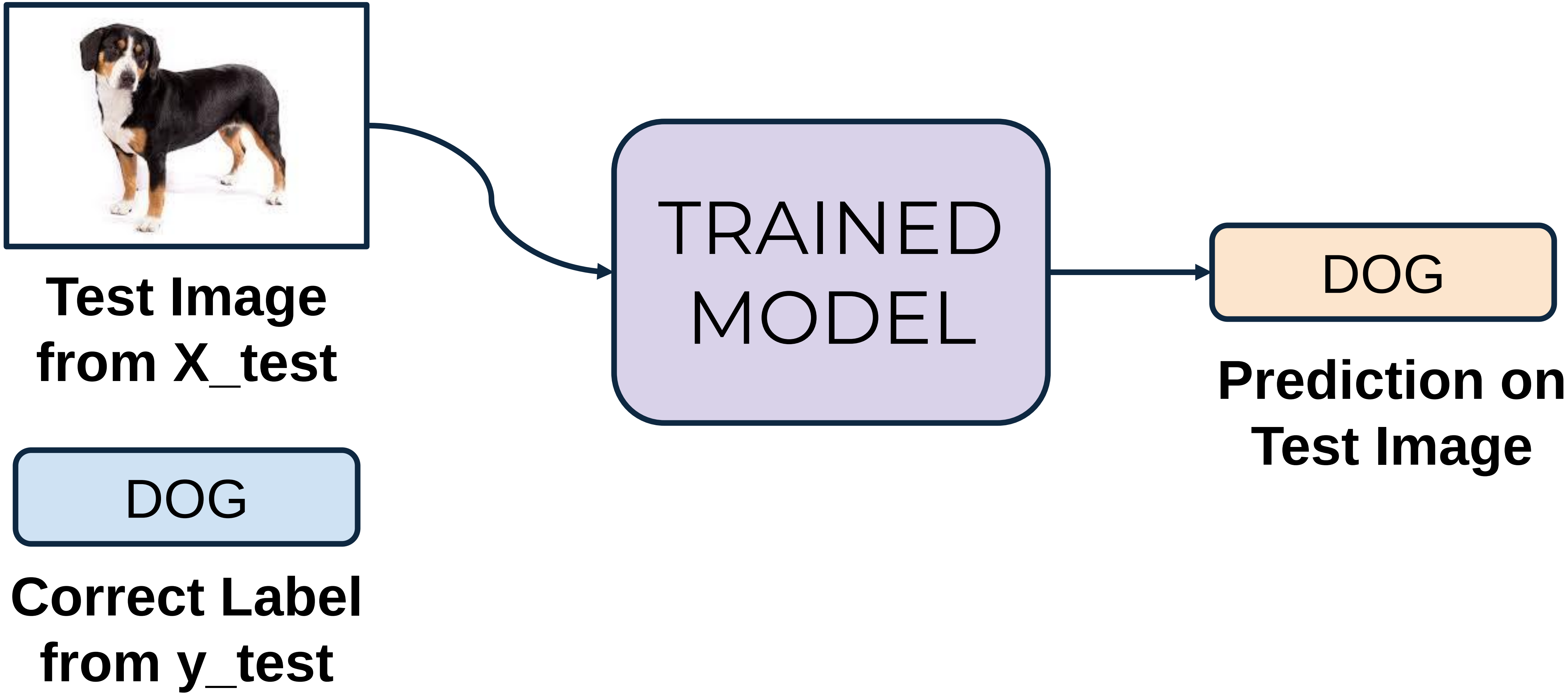
Test Image from X_test

TRAINED MODEL

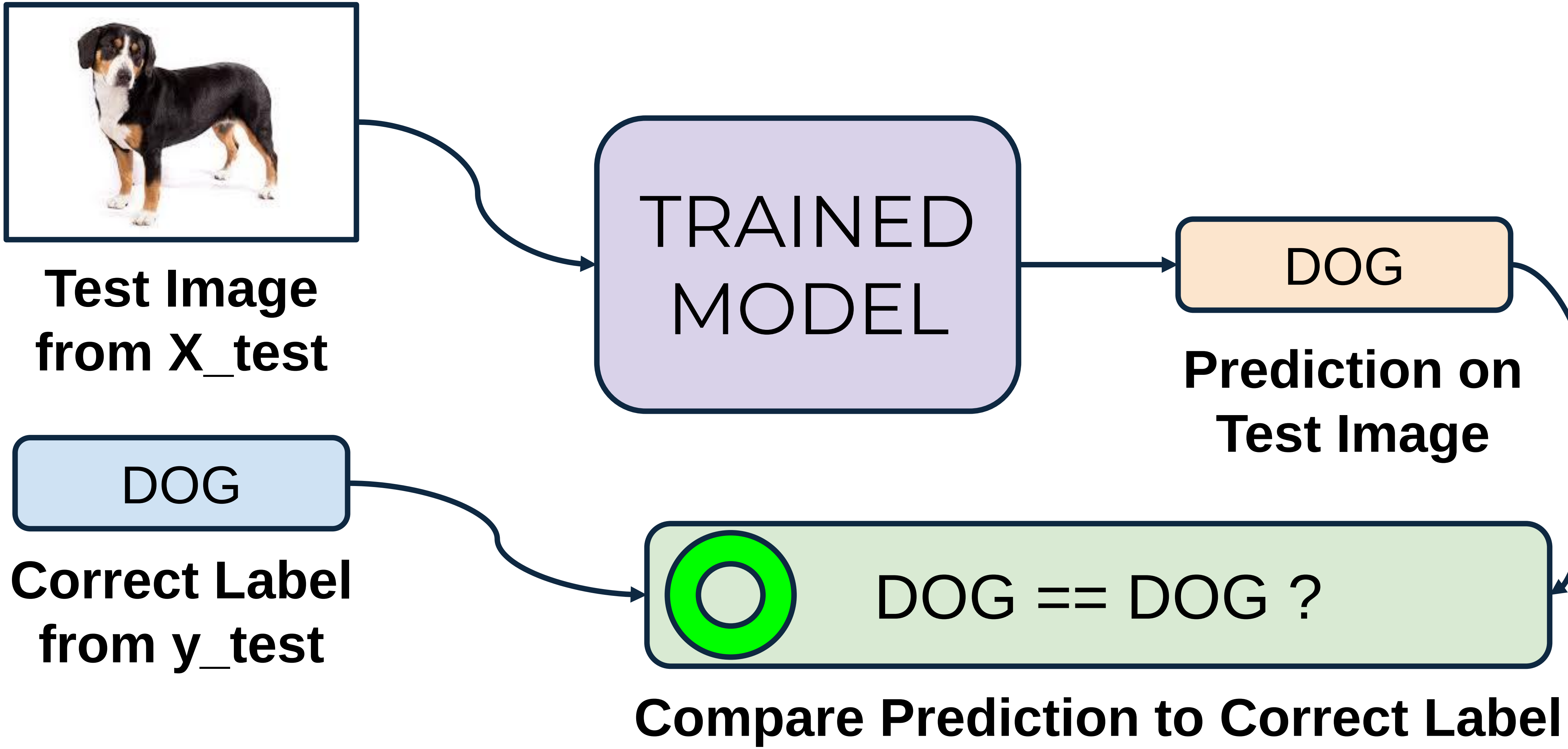
DOG

Correct Label from y_test

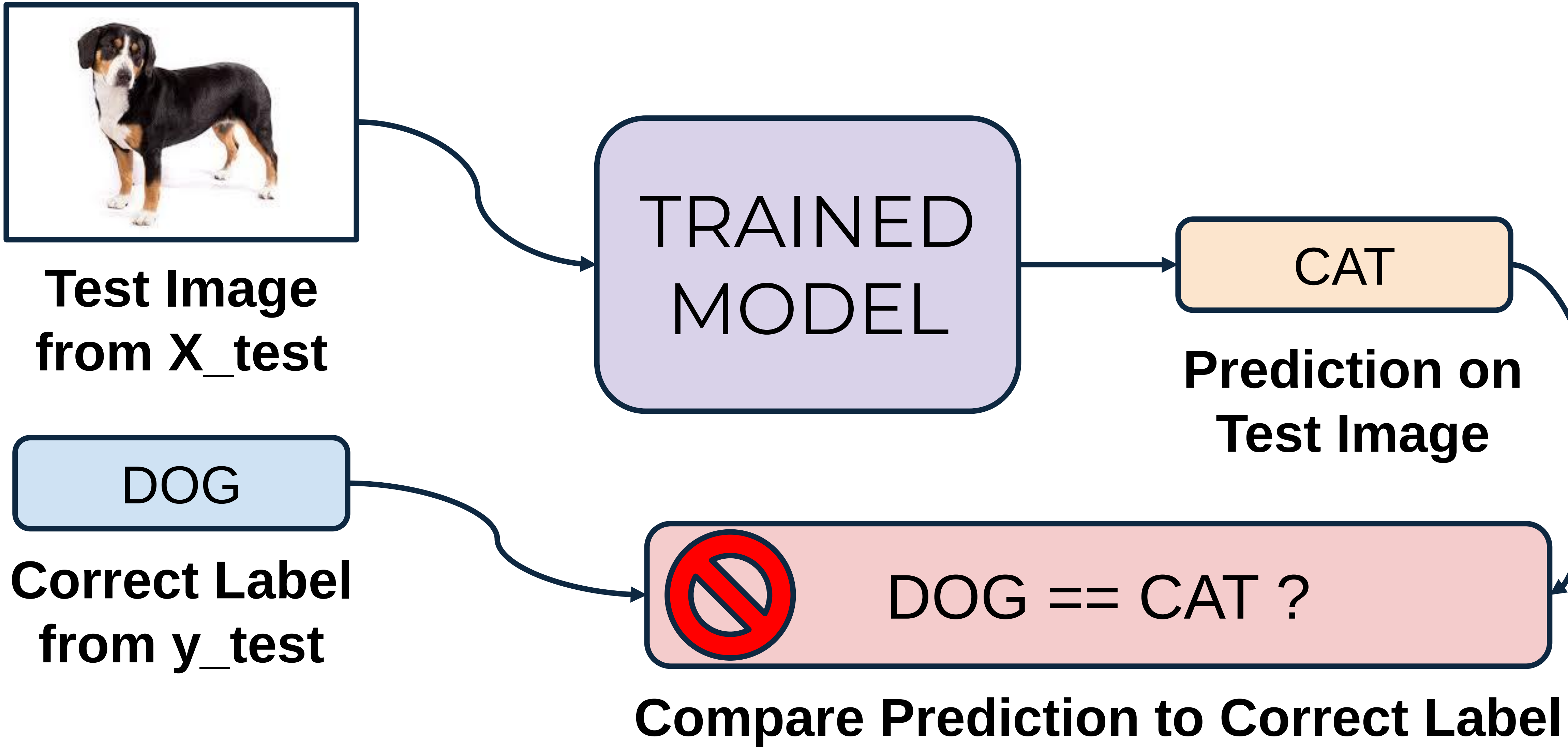
Model Evaluation



Model Evaluation



Model Evaluation



Model Evaluation

- We repeat this process for all the images in our X test data.
- At the end we will have a count of correct matches and a count of incorrect matches.
- The key realization we need to make, is that **in the real world**, **not all incorrect or correct matches hold equal value!**

Model Evaluation

- Also in the real world, a single metric won't tell the complete story!
- To understand all of this, let's bring back the 4 metrics we mentioned and see how they are calculated.
- We could organize our predicted values compared to the real values in a **confusion matrix.**

Model Evaluation

- Accuracy
    - Accuracy in classification problems is the **number of correct predictions** made by the model divided by the **total number of predictions.**

Model Evaluation

- Accuracy
  - For example, if the X_test set was 100 images and our model **correctly** predicted 80 images, then we have **80/100.**
  - **0.8** or **80% accuracy.**

Model Evaluation

- Accuracy
  - Accuracy is useful when target classes are well balanced
  - In our example, we would have roughly the same amount of cat images as we have dog images.

Model Evaluation

- Accuracy
  - Accuracy is **not** a good choice with **unbalanced** classes!
  - Imagine we had 99 images of dogs and 1 image of a cat.
  - If our model was simply a line that always predicted **dog** we would get 99% accuracy!

Model Evaluation

- Accuracy
  - Imagine we had 99 images of dogs and 1 image of a cat.
  - If our model was simply a line that always predicted **dog** we would get 99% accuracy!
  - In this situation we'll want to understand **recall** and **precision**

Model Evaluation

- Recall
  - Ability of a model to find all the relevant cases within a dataset.
  - The precise definition of recall is the number of true positives **divided by** the number of true positives plus the number of false negatives.

Model Evaluation

- Precision
  - Ability of a classification model to identify only the relevant data points.
  - Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives.

Model Evaluation

- Recall and Precision
  - Often you have a trade-off between Recall and Precision.
  - While recall expresses the ability to find all relevant instances in a dataset, precision expresses the proportion of the data points our model says was relevant actually were relevant.

Model Evaluation

- F1-Score
  - In cases where we want to find an optimal blend of precision and recall we can combine the two metrics using what is called the F1 score.

Model Evaluation

- F1-Score
    - The F1 score is the harmonic mean of precision and recall taking both metrics into account in the following equation:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Model Evaluation

- F1-Score
  - We use the harmonic mean instead of a simple average because it punishes extreme values.
  - A classifier with a precision of 1.0 and a recall of 0.0 has a simple average of 0.5 but an F1 score of 0.

Model Evaluation

- We can also view all correctly classified versus incorrectly classified images in the form of a confusion matrix.

Confusion Matrix

		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**

Confusion Matrix

		predicted condition		
	total population	prediction positive	prediction negative	Prevalence $= \frac{\Sigma \text{ condition positive}}{\Sigma \text{ total population}}$
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)	True Positive Rate (TPR), Sensitivity, Recall, Probability of Detection $= \frac{\Sigma \text{ TP}}{\Sigma \text{ condition positive}}$
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**	False Positive Rate (FPR), Fall-out, Probability of False Alarm $= \frac{\Sigma \text{ FP}}{\Sigma \text{ condition negative}}$
	Accuracy $= \frac{\Sigma \text{ TP} + \Sigma \text{ TN}}{\Sigma \text{ total population}}$	Positive Predictive Value (PPV), Precision $= \frac{\Sigma \text{ TP}}{\Sigma \text{ prediction positive}}$	False Omission Rate (FOR) $= \frac{\Sigma \text{ FN}}{\Sigma \text{ prediction negative}}$	Positive Likelihood Ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$
		False Discovery Rate (FDR) $= \frac{\Sigma \text{ FP}}{\Sigma \text{ prediction positive}}$	Negative Predictive Value (NPV) $= \frac{\Sigma \text{ TN}}{\Sigma \text{ prediction negative}}$	Negative Likelihood Ratio (LR–) $= \frac{\text{FNR}}{\text{TNR}}$

Model Evaluation

- The main point to remember with the confusion matrix and the various calculated metrics is that they are all fundamentally ways of comparing the predicted values versus the true values.
- What constitutes "good" metrics, will really depend on the specific situation!

Model Evaluation

- Still confused on the confusion matrix?
- No problem! Check out the Wikipedia page for it, it has a really good diagram with all the formulas for all the metrics.
- Throughout the training, we'll usually just print out metrics (e.g. accuracy).

Model Evaluation

- Let's think back on this idea of:
  - What is a good enough accuracy?
- This all depends on the context of the situation!
- Did you create a model to predict presence of a disease?
- Is the disease presence well balanced in the general population? (Probably not!)

Model Evaluation

- Often models are used as quick diagnostic tests to have **before** having a more invasive test (e.g. getting urine test before getting a biopsy)
- We also need to consider what is at stake!

Model Evaluation

- Often we have a precision/recall trade off, We need to decide if the model will should focus on fixing False Positives vs. False Negatives.
- In disease diagnosis, it is probably better to go in the direction of False positives, so we make sure we correctly classify as many cases of disease as possible!

Model Evaluation

- All of this is to say, machine learning is not performed in a “vacuum”, but instead a collaborative process where we should consult with experts in the domain (e.g. medical doctors)

Evaluating Performance

REGRESSION

Evaluating Regression

- Let’s take a moment now to discuss evaluating Regression Models
- Regression is a task when a model attempts to predict continuous values (unlike categorical values, which is classification)

Evaluating Regression

- You may have heard of some evaluation metrics like accuracy or recall.
- These sort of metrics aren’t useful for regression problems, we need metrics designed for **continuous** values!

Evaluating Regression

- For example, attempting to predict the price of a house given its features is a **regression task.**
- Attempting to predict the country a house is in given its features would be a classification task.

Evaluating Regression

- Let’s discuss some of the most common evaluation metrics for regression:
    - Mean Absolute Error
    - Mean Squared Error
    - Root Mean Square Error

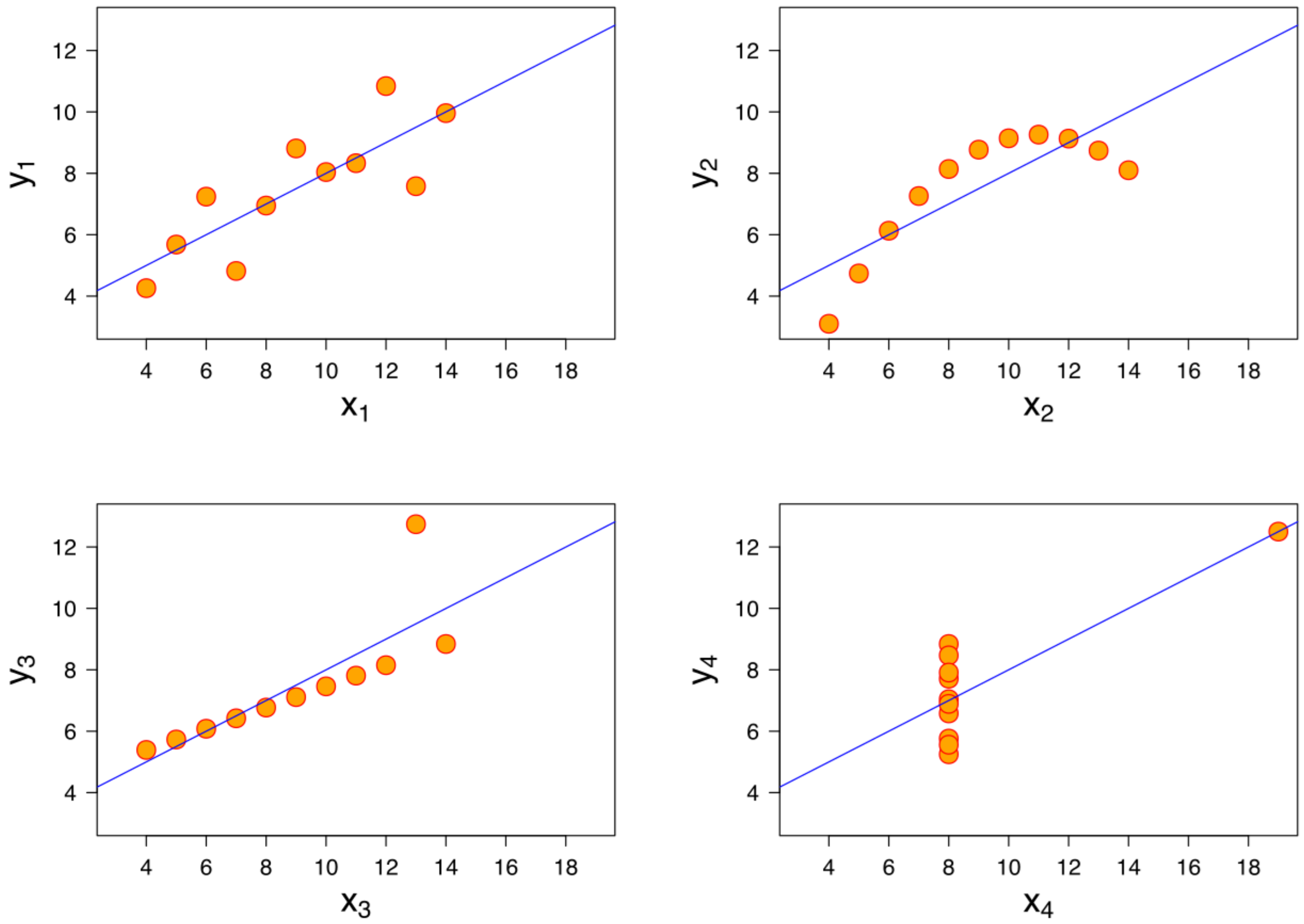
Evaluating Regression

- Mean Absolute Error (MAE)
  - This is the mean of the absolute value of errors.
  - Easy to understand

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

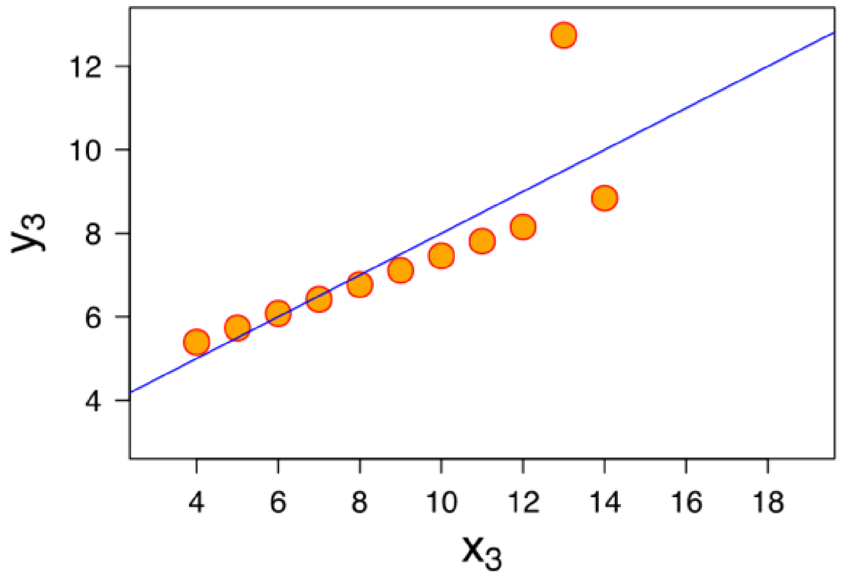
Evaluating Regression

- MAE won’t punish large errors however.



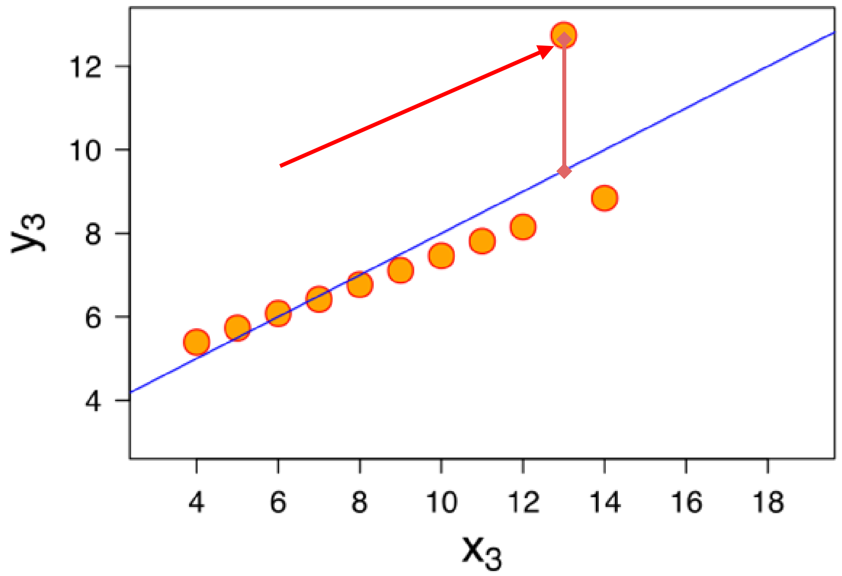
Evaluating Regression

- MAE won't punish large errors however.



Evaluating Regression

- We want our error metrics to account for these!



Evaluating Regression

- Mean Squared Error (MSE)
  - This is the mean of the squared errors.
  - Larger errors are noted more than with MAE, making MSE more popular.

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

Evaluating Regression

- Root Mean Square Error (RMSE)
  - This is the root of the mean of the squared errors.
  - Most popular (has same units as y)

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

Machine Learning

- Most common question from students:
    - “Is this value of RMSE good?”
- Context is everything!
- A RMSE of $10 is fantastic for predicting the price of a house, but horrible for predicting the price of a candy bar!

Machine Learning

- Compare your error metric to the average value of the label in your data set to try to get an intuition of its overall performance.
- Domain knowledge also plays an important role here!

Machine Learning

- Context of importance is also necessary to consider.
- We may create a model to predict how much medication to give, in which case small fluctuations in RMSE may actually be very significant.

Machine Learning with Python

Scikit Learn

We will be using the **Scikit Learn** package.

It’s the most popular machine learning package for Python and has a lot of algorithms built-in!

Scikit Learn

You’ll need to install it using:

conda install scikit-learn

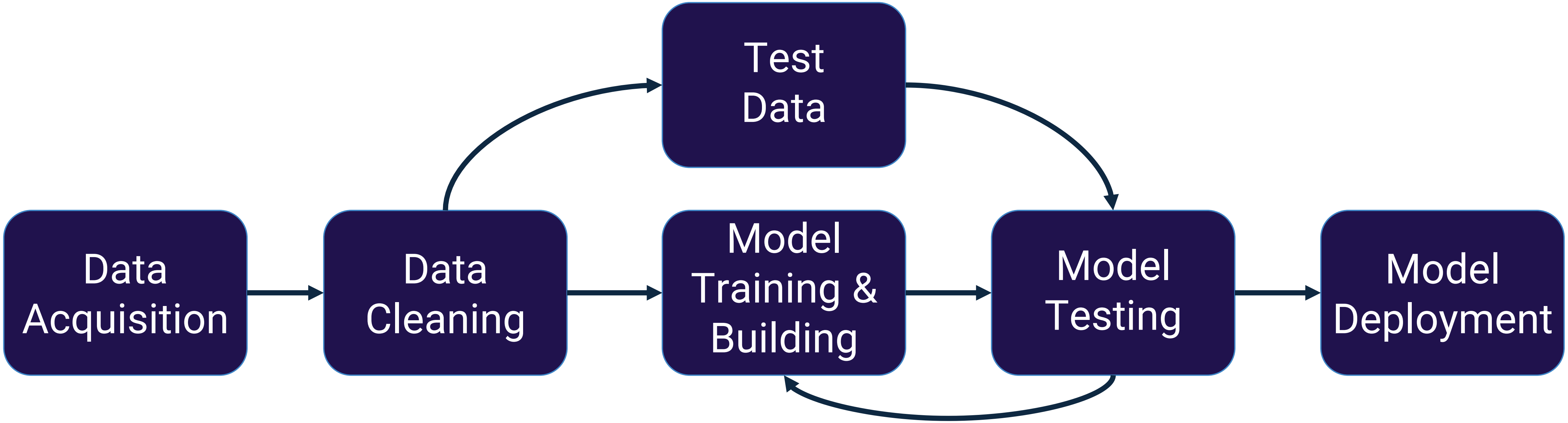
or

pip install scikit-learn

Scikit Learn

- Let’s talk about the basic structure of how to use Scikit Learn!
- First, a quick review of the machine learning process.

Machine Learning Process



Scikit Learn

- Now let's go over an example of the process to use SciKit Learn.
- Don't worry about memorizing any of this, we'll get plenty of practice and review when we actually start coding in subsequent lectures!

Scikit Learn

Every algorithm is exposed in scikit-learn via an "Estimator"

First you’ll import the model, the general form is:

```
from sklearn.family import Model
```

For example:

```
from sklearn.linear_model import LinearRegression
```

Scikit Learn

Estimator parameters: All the parameters of an estimator can be set when it is instantiated, and have suitable default values.

You can use Shift+tab in jupyter to check the possible parameters.

Scikit Learn

For example:

```
model = LinearRegression(normalize=True)
print(model)
```

LinearRegression(copy_X=True, fit_intercept=True, normalize=True)

Scikit Learn

Once you have your model created with your parameters, it is time to fit your model on some data!

But remember, we should split this data into a training set and a test set.

Scikit Learn

```
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(10).reshape((5, 2)), range(5)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]
```

Scikit Learn

```
>>> X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.3)
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]
```

Scikit Learn

Now that we have split the data, we can train/fit our model on the training data.

This is done through the model.fit() method:

```
model.fit(X_train,y_train)
```

Scikit Learn

- Now the model has been fit and trained on the training data.

- The model is ready to predict labels or values on the test set!

Scikit Learn

We get predicted values using the predict method:

```
predictions = model.predict(X_test)
```

Scikit Learn

We can then evaluate our model by comparing our predictions to the correct values.

The evaluation method depends on what sort of machine learning algorithm we are using (e.g. Regression,Classification, Clustering, etc.)

Scikit Learn

Scikit-learn strives to have a uniform interface across all methods, and we'll see examples of these below.

Given a scikit-learn *estimator* object named model, the following methods are available...

Scikit Learn

- Available in **all Estimators**
    - model.fit() : fit training data.
    - For supervised learning applications, this accepts two arguments: the data X and the labels y (e.g. model.fit(X, y)).
    - For unsupervised learning applications, this accepts only a single argument, the data X (e.g. model.fit(X)).

Scikit Learn

Available in **supervised estimators**

- model.predict() : given a trained model, predict the label of a new set of data. This method accepts one argument, the new data X_new (e.g. model.predict(X_new)), and returns the learned label for each object in the array.

Scikit Learn

Available in **supervised estimators**

- model.predict_proba() : For classification problems, some estimators also provide this method, which returns the probability that a new observation has each categorical label. In this case, the label with the highest probability is returned by model.predict().

Scikit Learn

Available in **supervised estimators**

- model.score() : for classification or regression problems, most estimators implement a score method. Scores are between 0 and 1, with a larger score indicating a better fit.

Scikit Learn

Available in **unsupervised estimators**

- model.predict() : predict labels in clustering algorithms.

Scikit Learn

Available in **unsupervised estimators**

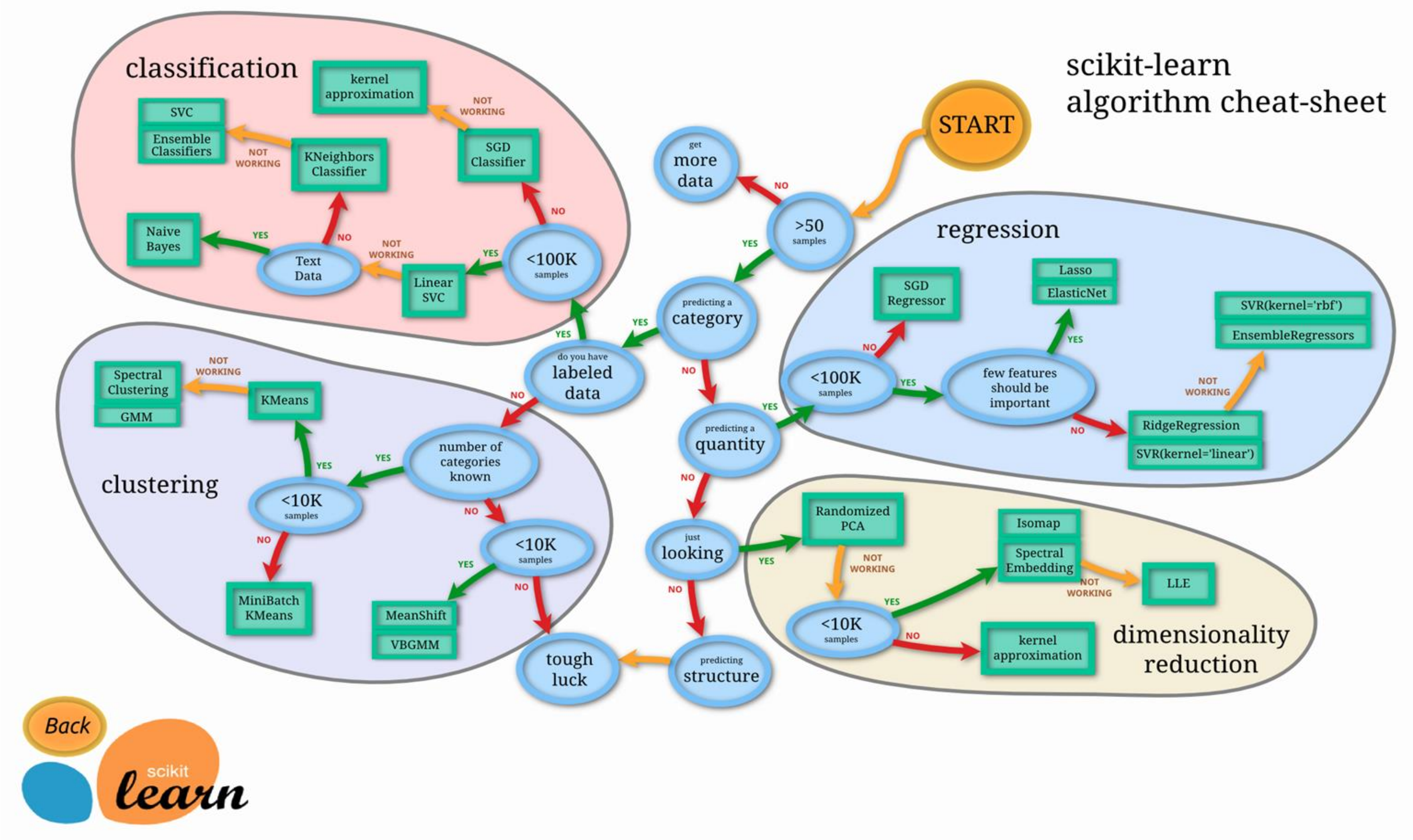
- model.transform() : given an unsupervised model, transform new data into the new basis. This also accepts one argument X_new, and returns the new representation of the data based on the unsupervised model.

Scikit Learn

Available in **unsupervised estimators**

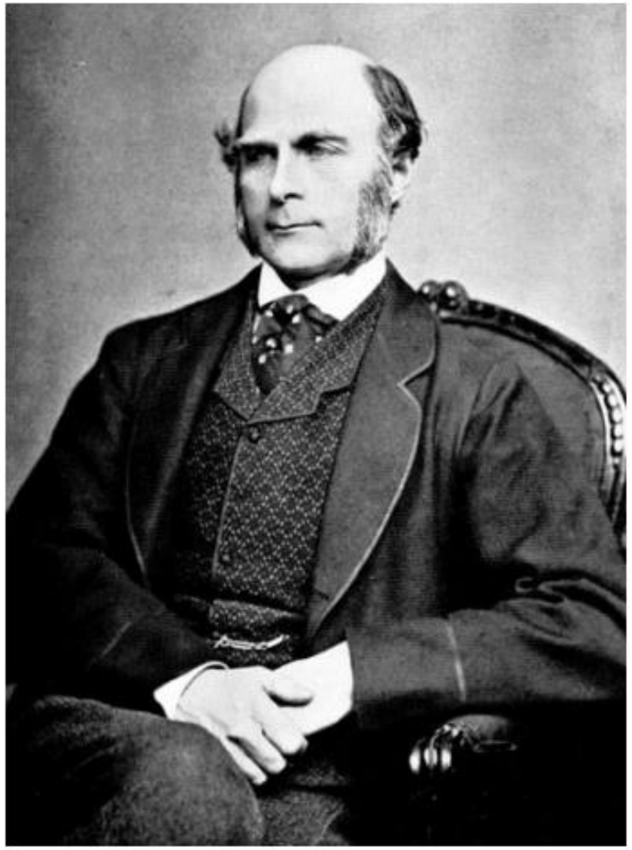
- model.fit_transform() : some estimators implement this method, which more efficiently performs a fit and a transform on the same input data.

Choosing an Algorithm



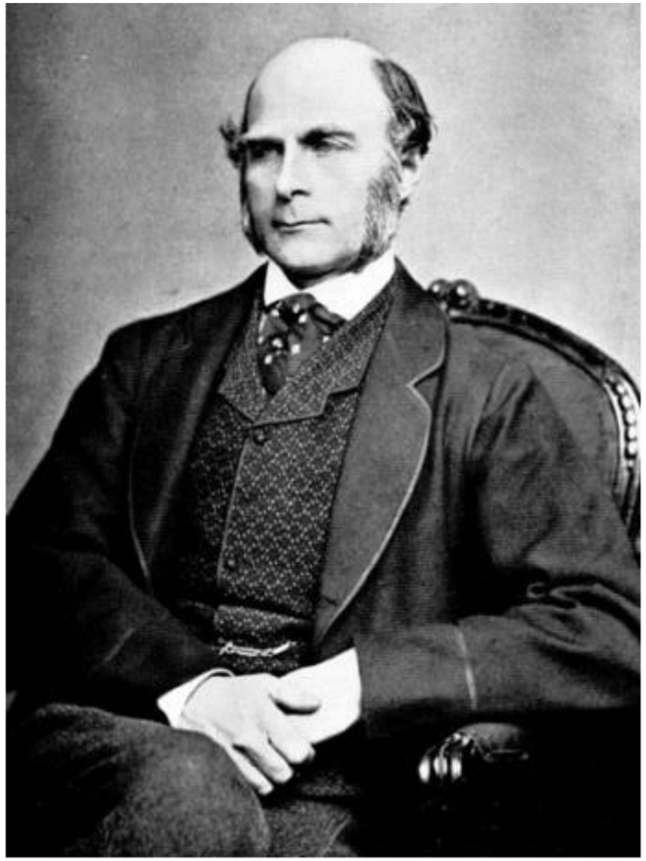
Introduction to Linear Regression

This all started in the 1800s with a guy named Francis Galton. Galton was studying the relationship between parents and their children. In particular, he investigated the relationship between the heights of fathers and their sons.



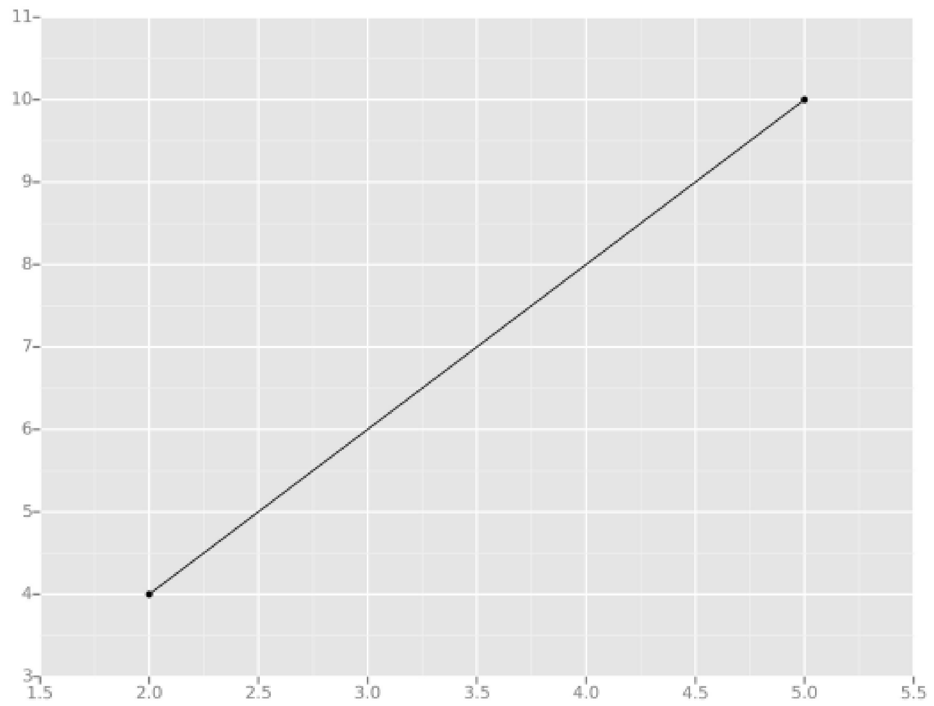
What he discovered was that a man's son tended to be roughly as tall as his father.

However Galton's breakthrough was that the son's height **tended to be closer to the overall average** height of all people.



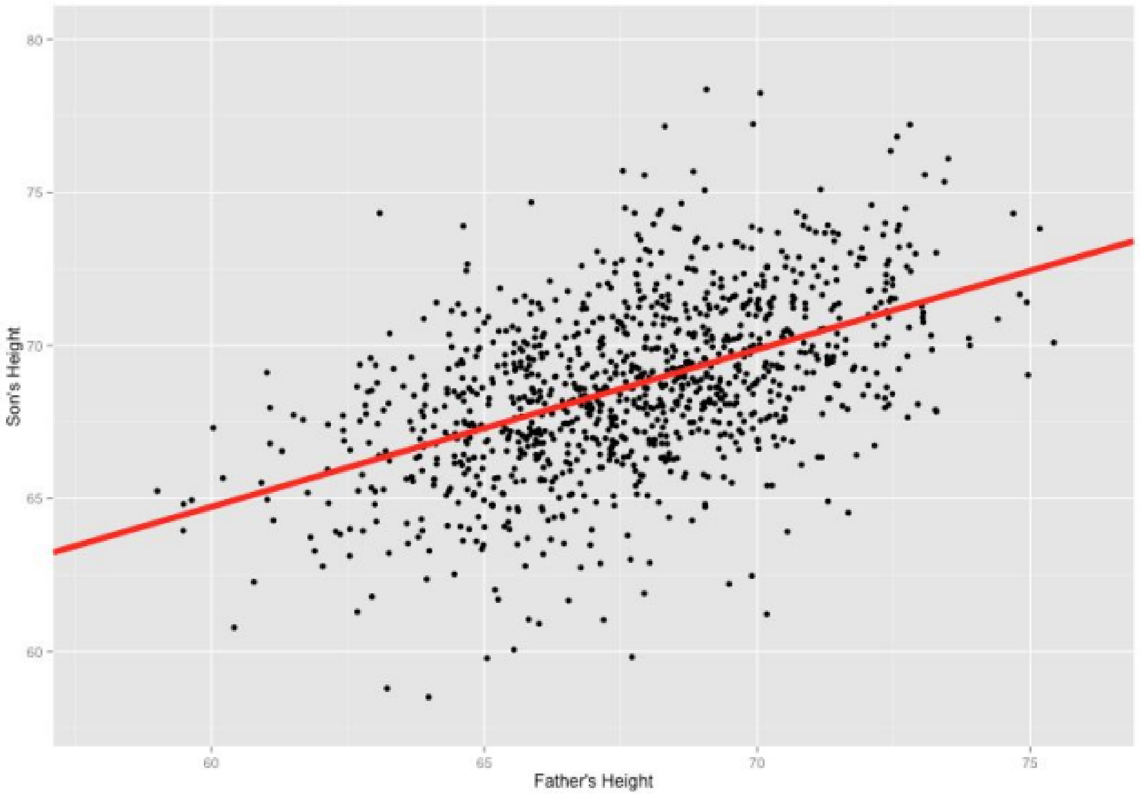
All we're trying to do when we calculate our regression line is draw a line that's as close to every dot as possible.

For classic linear regression, or "Least Squares Method", you only measure the closeness in the "up and down" direction

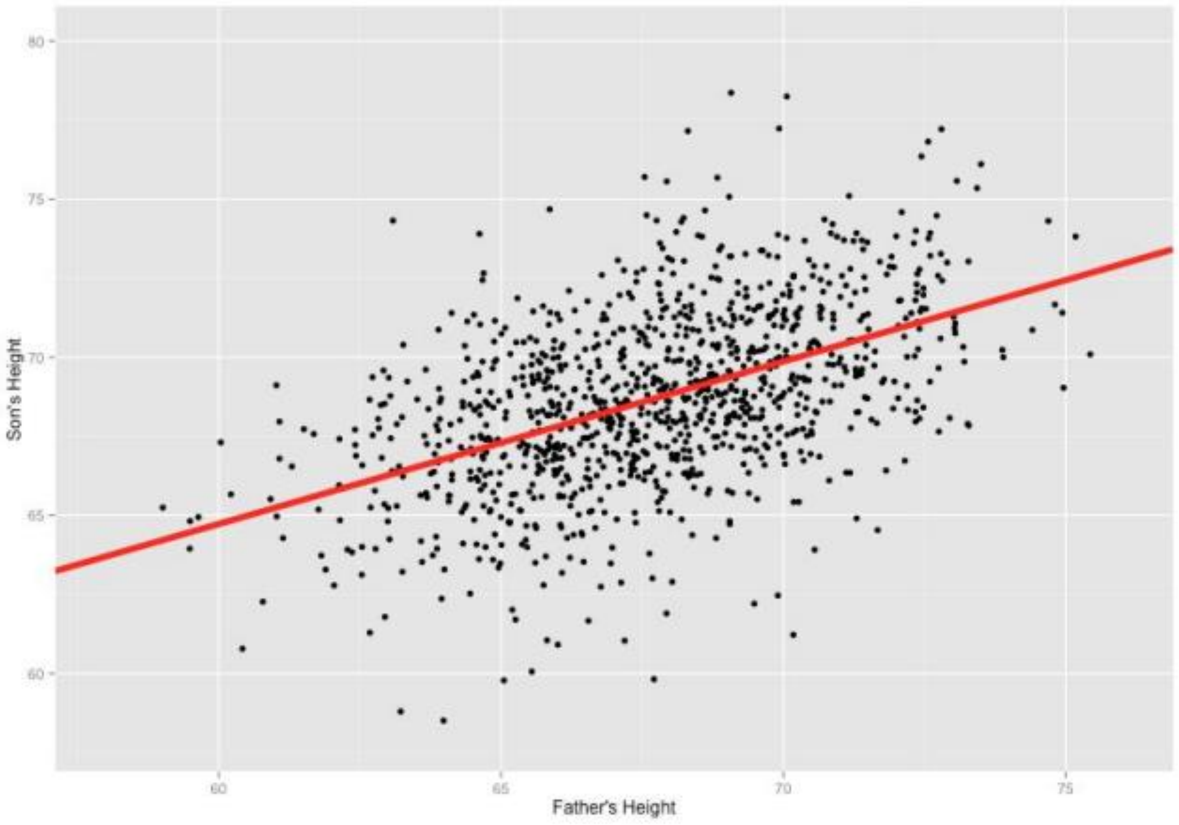


Our goal with linear regression is to **minimize the vertical distance** between all the data points and our line.

So in determining the **best line**, we are attempting to minimize the distance between **all** the points and their distance to our line.

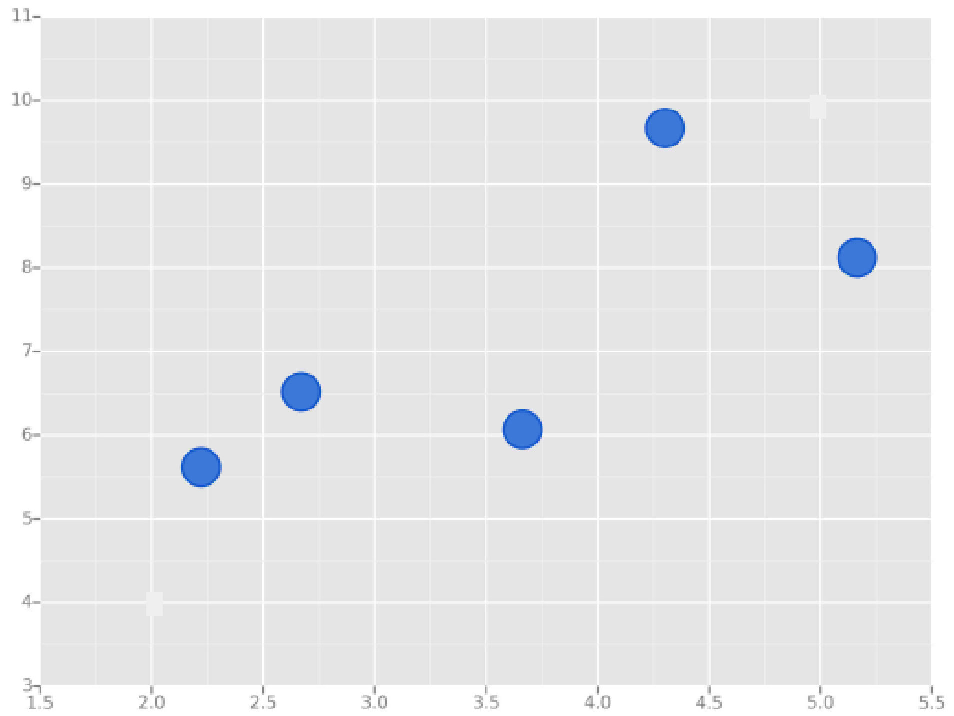


There are lots of different ways to minimize this, (sum of squared errors, sum of absolute errors, etc), but all these methods have a general goal of minimizing this distance.



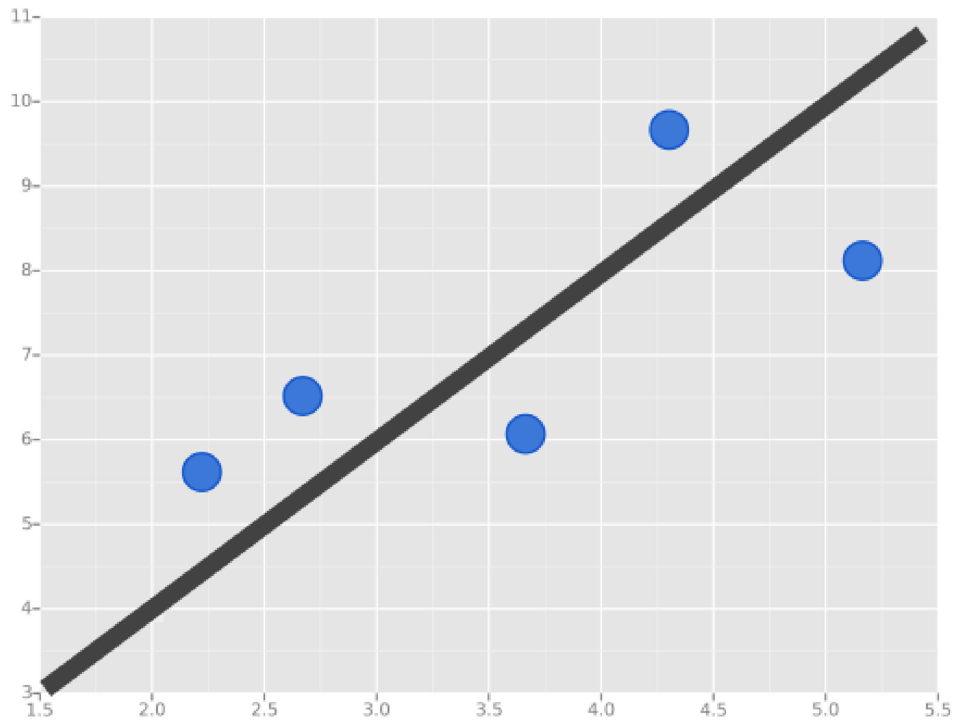
For example, one of the most popular methods is the least squares method.

Here we have blue data points along an x and y axis.



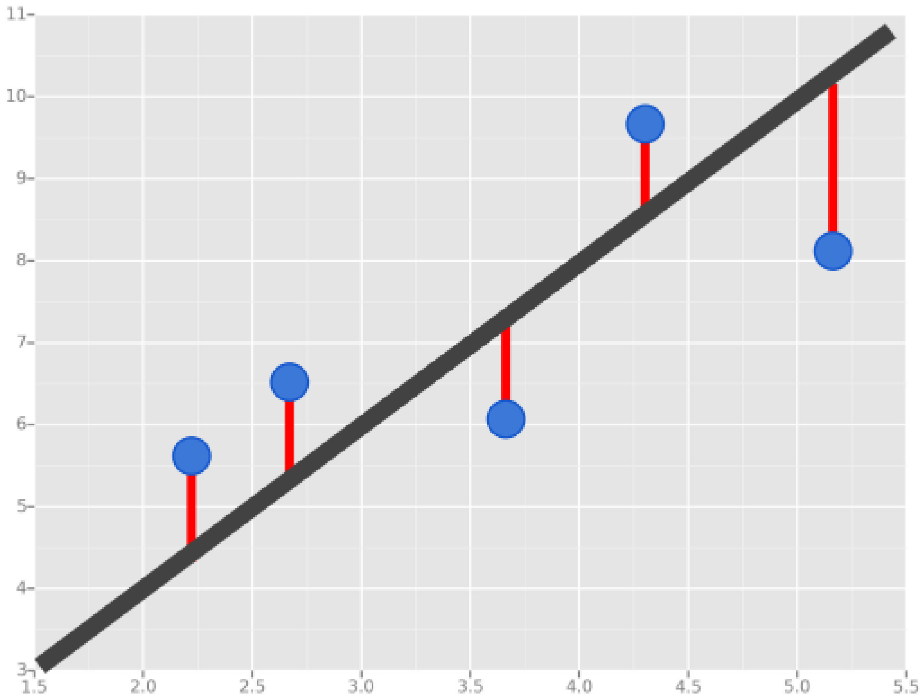
Now we want to fit a linear regression line.

The question is, how do we decide which line is the best fitting one?



We'll use the Least Squares Method, which is fitted by minimizing the ***sum of squares of the residuals.***

The residuals for an observation is the difference between the observation (the y-value) and the fitted line.



Bias-Variance Trade Off

Overfitting versus Underfitting

Overfitting and Underfitting

- In general, increasing model complexity in search for better performance leads to a **Bias-Variance trade-off**.
- We want to have a model that can generalize well to new unseen data, but can also account for variance and patterns in the known data.

Overfitting and Underfitting

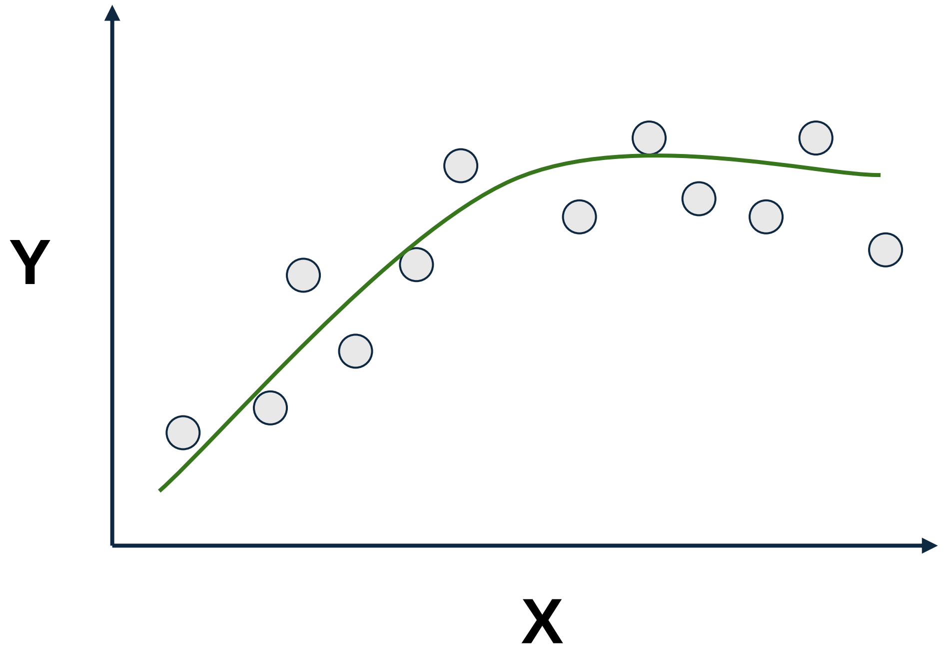
- Extreme bias or extreme variance both lead to bad models.
- We can visualize this effect by considering a model that underfits (high bias) or a model that overfits (high variance).
- Let’s start with a model that overfits to a dataset…

Overfitting and Underfitting

- **Overfitting**
  - The model fits too much to the noise from the data.
  - This often results in **low error on training sets but high error on test/validation sets.**

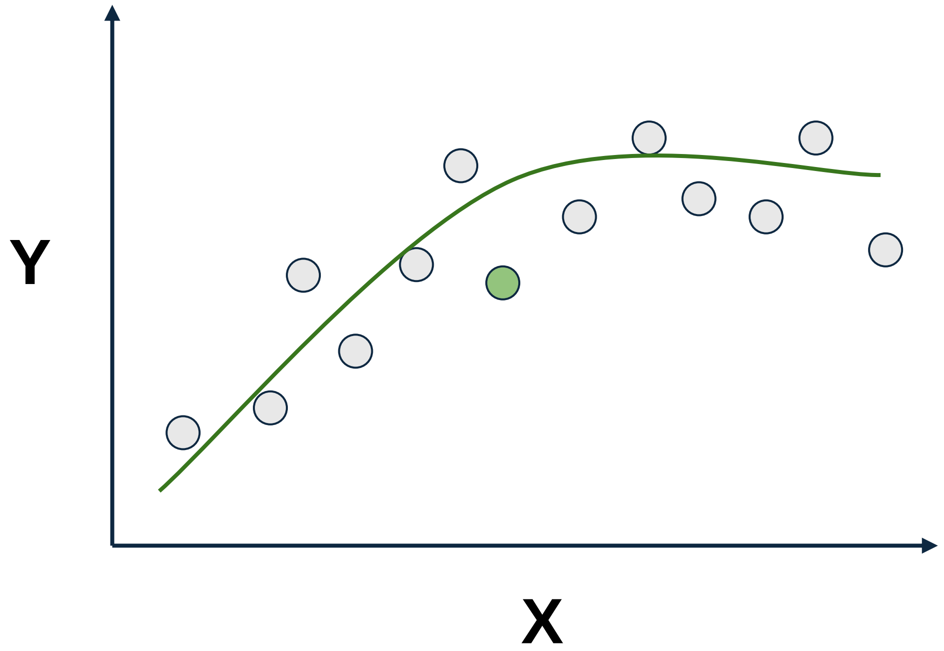
Overfitting and Underfitting

Good Model



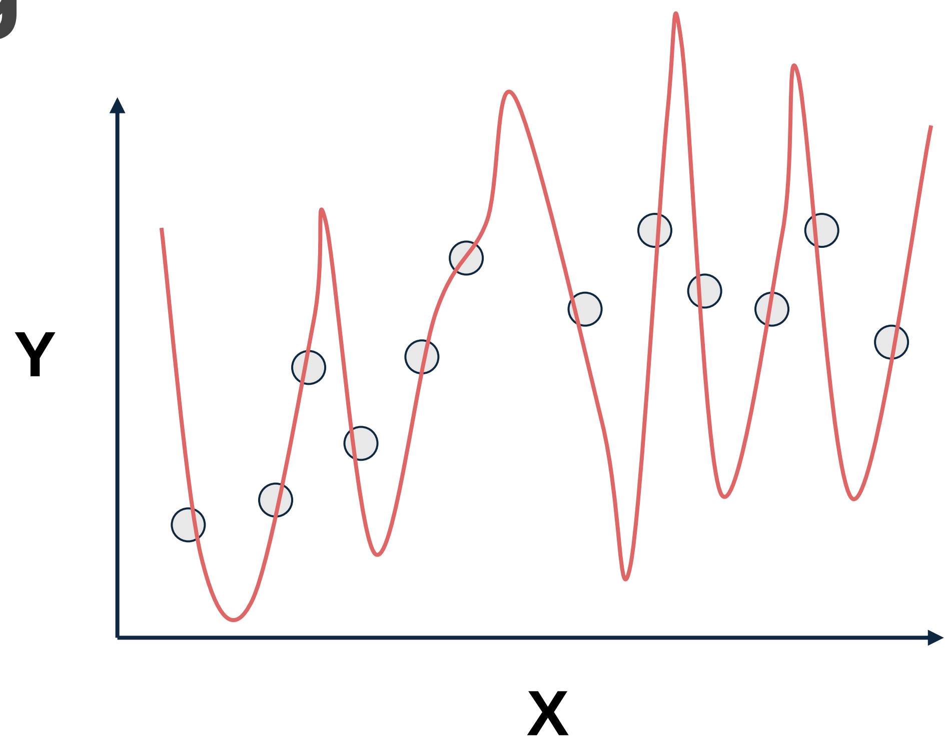
Overfitting and Underfitting

Good Model



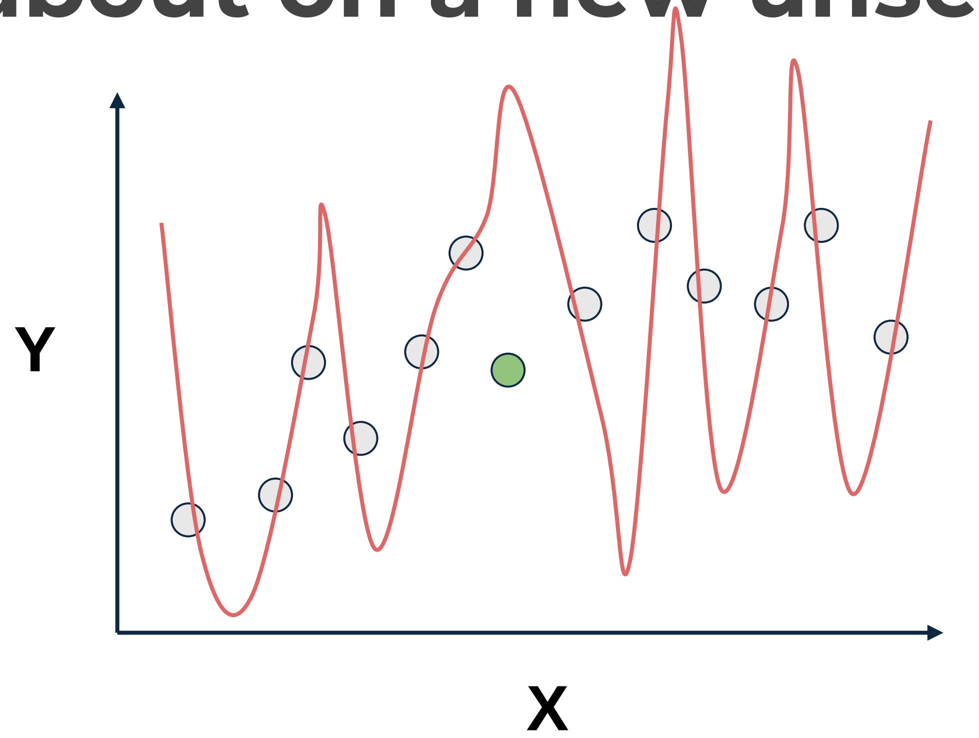
Overfitting and Underfitting

- **Overfitting**



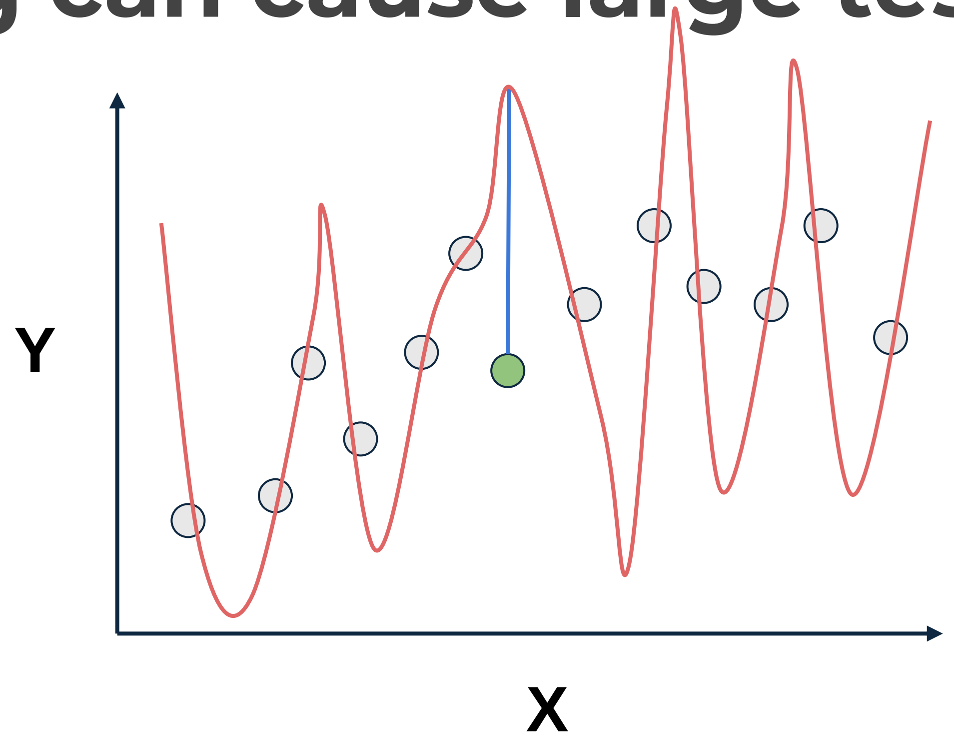
Overfitting and Underfitting

- **But what about on a new unseen data point?**



Overfitting and Underfitting

- **Overfitting can cause large test errors!**



Overfitting and Underfitting

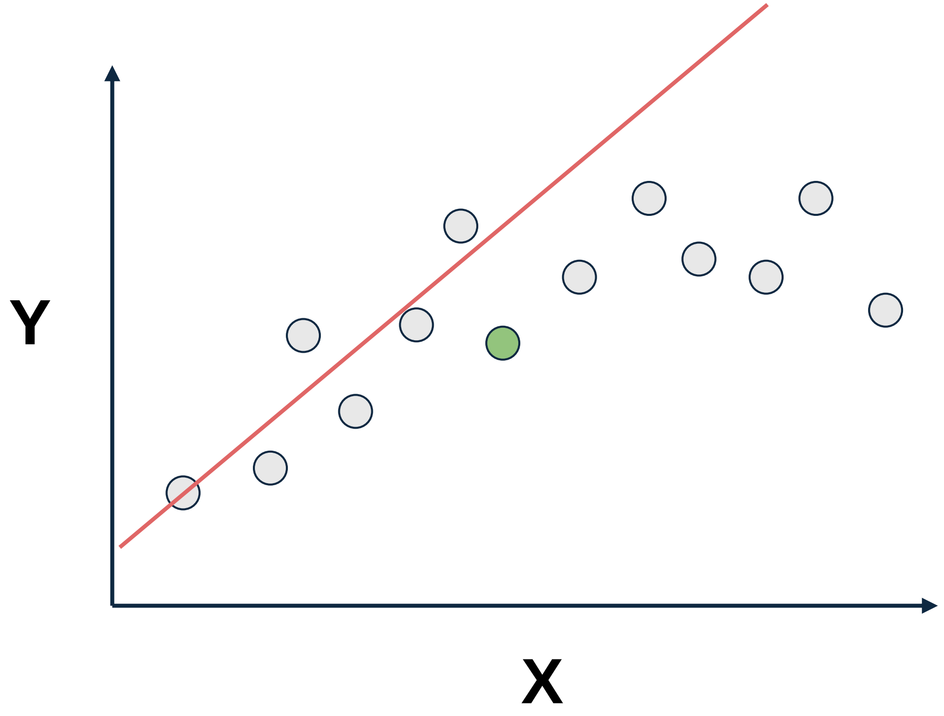
- **Overfitting**
  - Model is fitting too much to noise and variance in the training data.
  - Model will perform very well on training data, but have poor performance on new unseen data.

Overfitting and Underfitting

- **Underfitting**
  - Model does not capture the underlying trend of the data and does not fit the data well enough.
  - Low variance but high bias.
  - Underfitting is often a result of an excessively simple model.

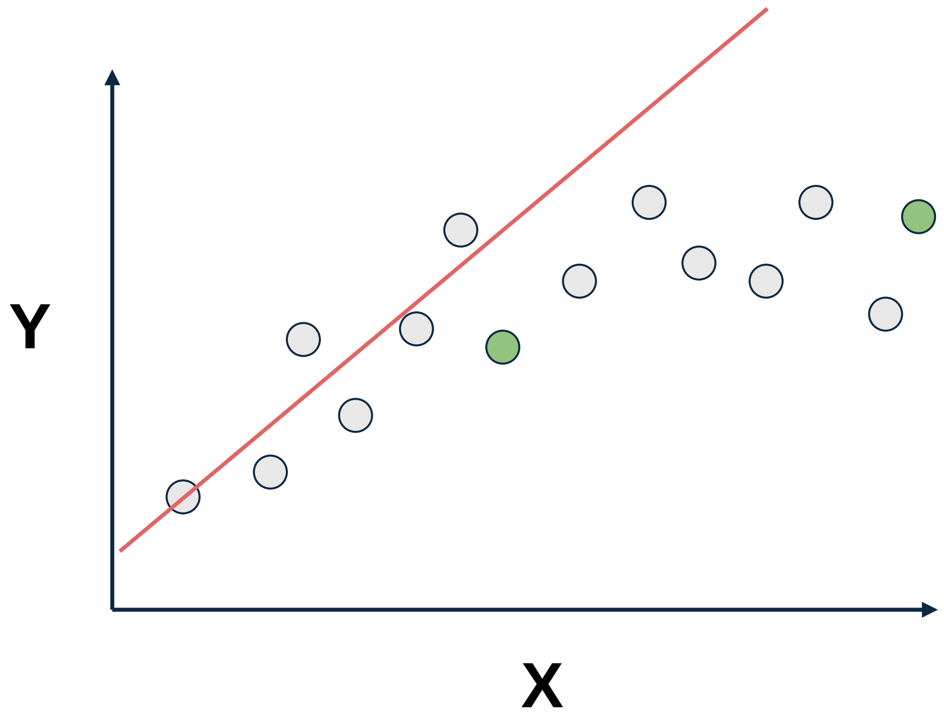
Overfitting and Underfitting

Underfitting



Overfitting and Underfitting

Underfitting



Overfitting and Underfitting

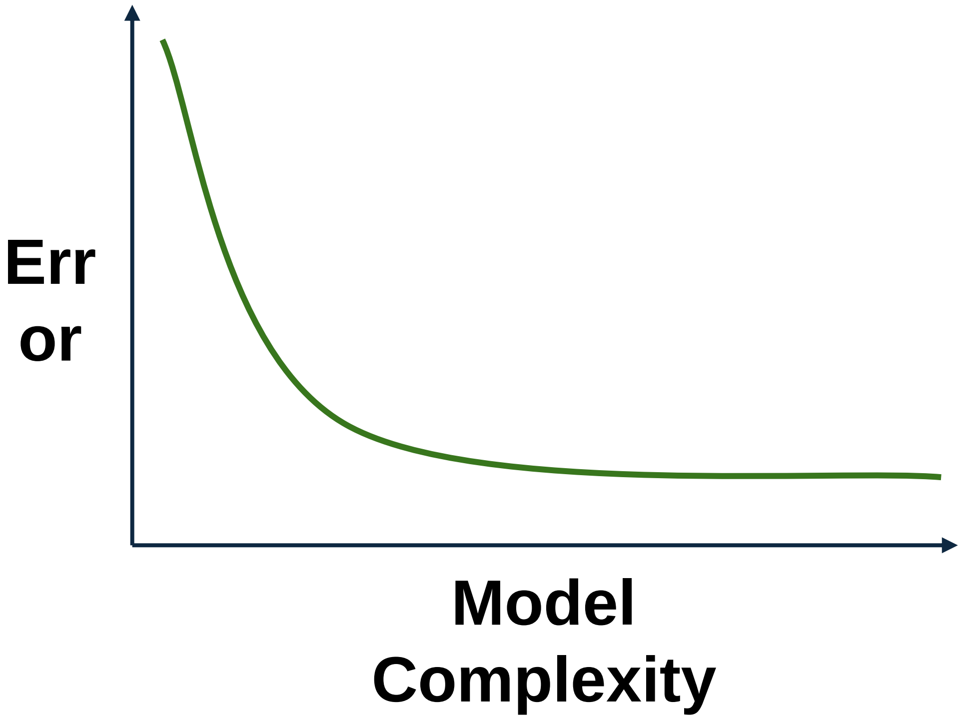
- **Underfitting**
  - Model has high bias and is generalizing too much.
  - Underfitting can lead to poor performance in both training and testing data sets.

Overfitting and Underfitting

- **Overfitting versus Underfitting**
  - Overfitting can be harder to detect, since good performance on training data could lead to a model that appears to be performing well.

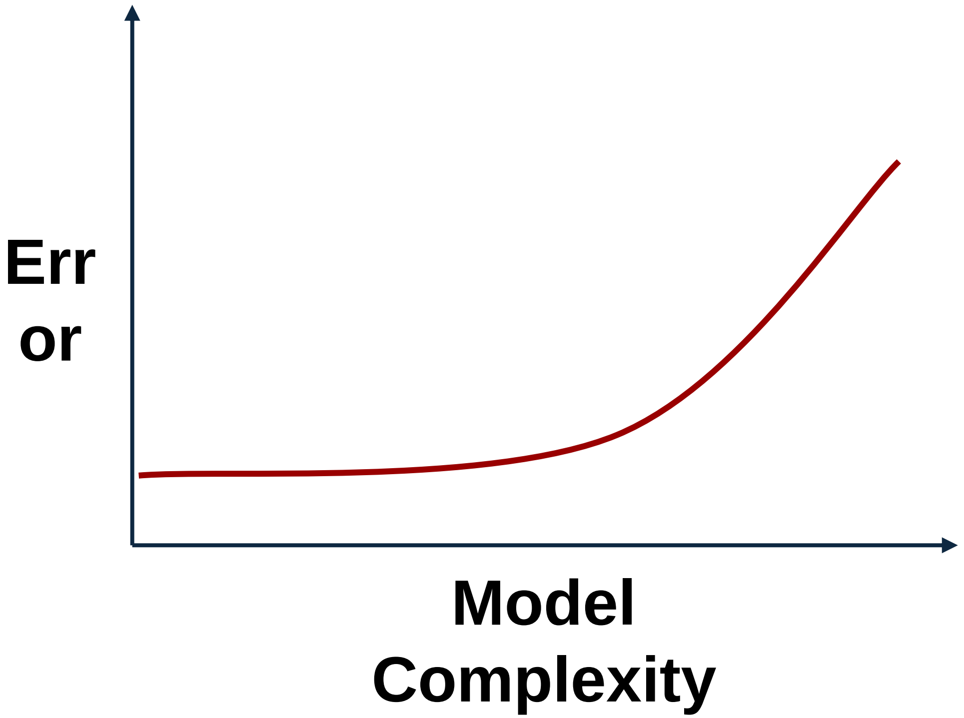
Overfitting and Underfitting

- Good Model



Overfitting and Underfitting

- Bad Model



Overfitting and Underfitting

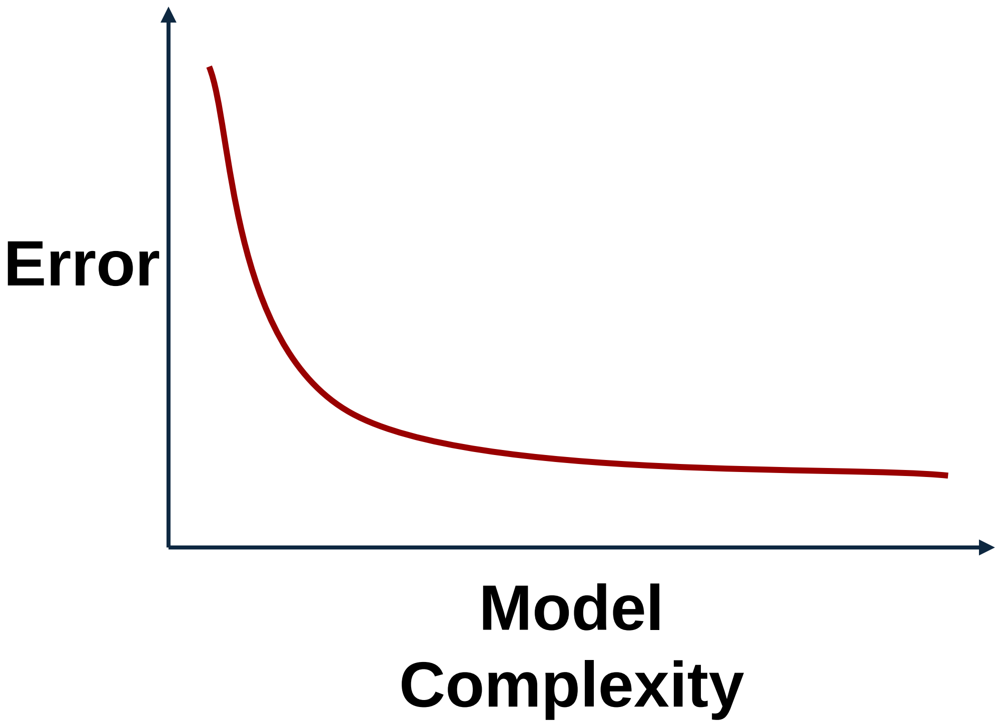
- When thinking about **overfitting** and **underfitting** we want to keep in mind the relationship of model performance on the training set versus the test/validation set.

Overfitting and Underfitting

- Let's imagine we split our data into a **training set** and a **test set**

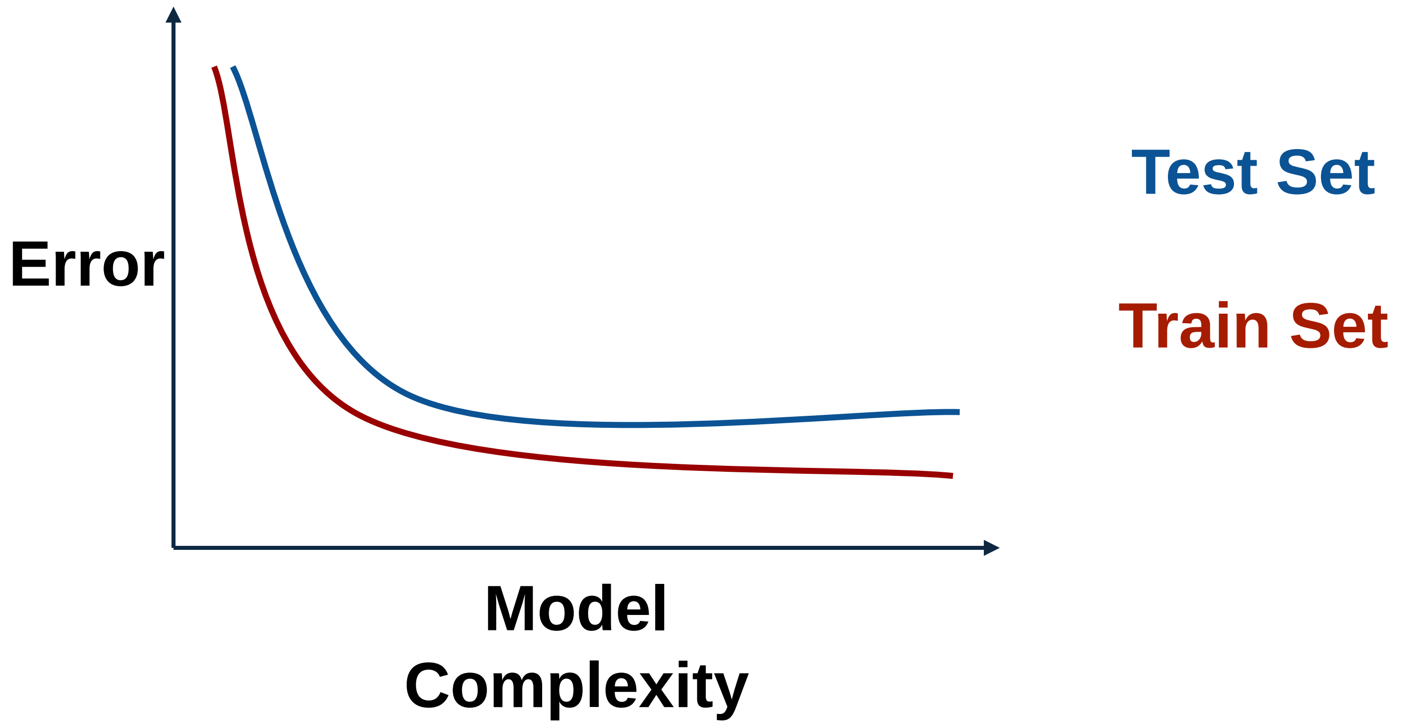
Overfitting and Underfitting

- We first see performance on the **training set**



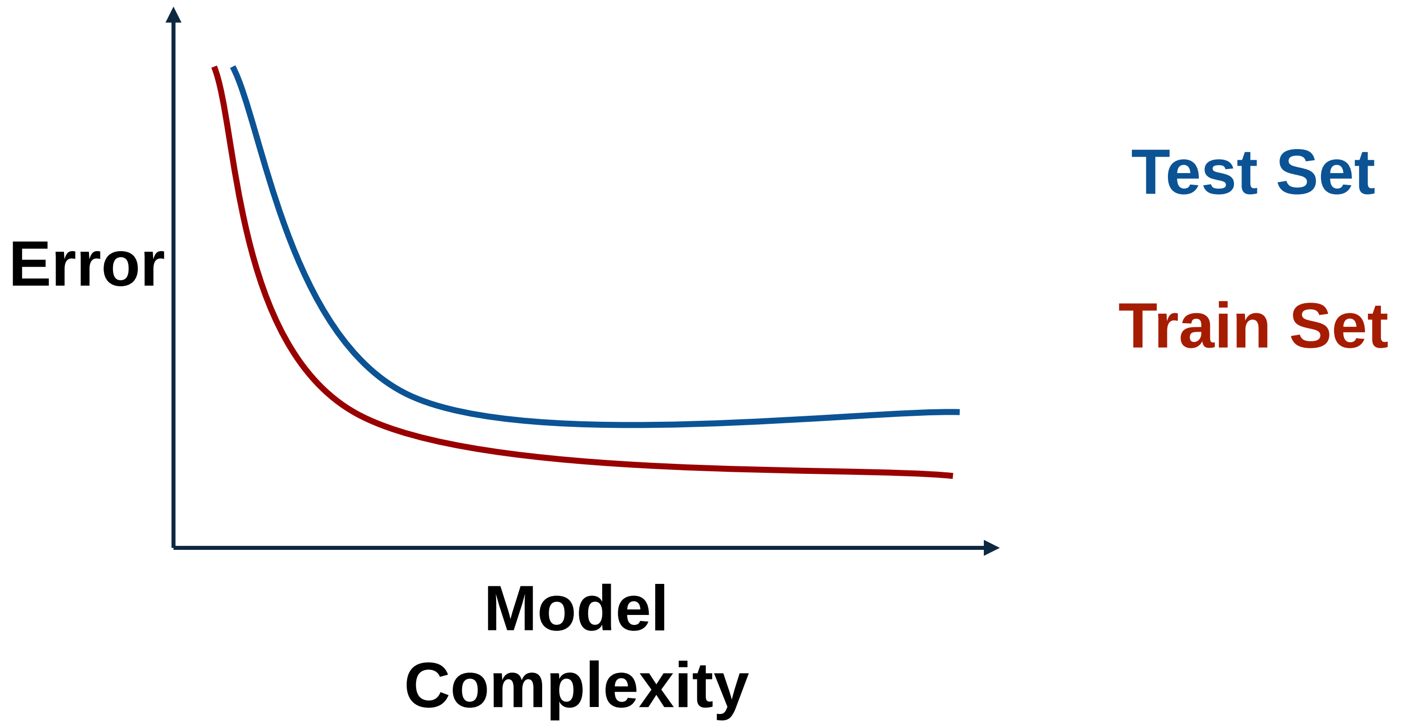
Overfitting and Underfitting

- Next we check performance on the **test set**



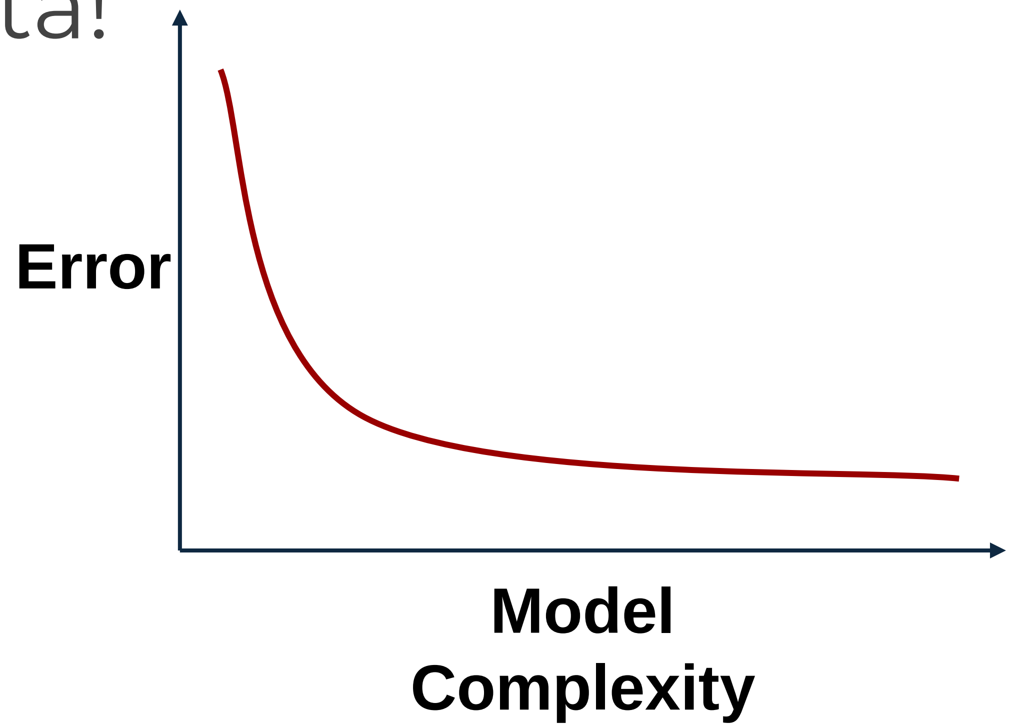
Overfitting and Underfitting

- Ideally the model would perform well on both, with similar behavior.



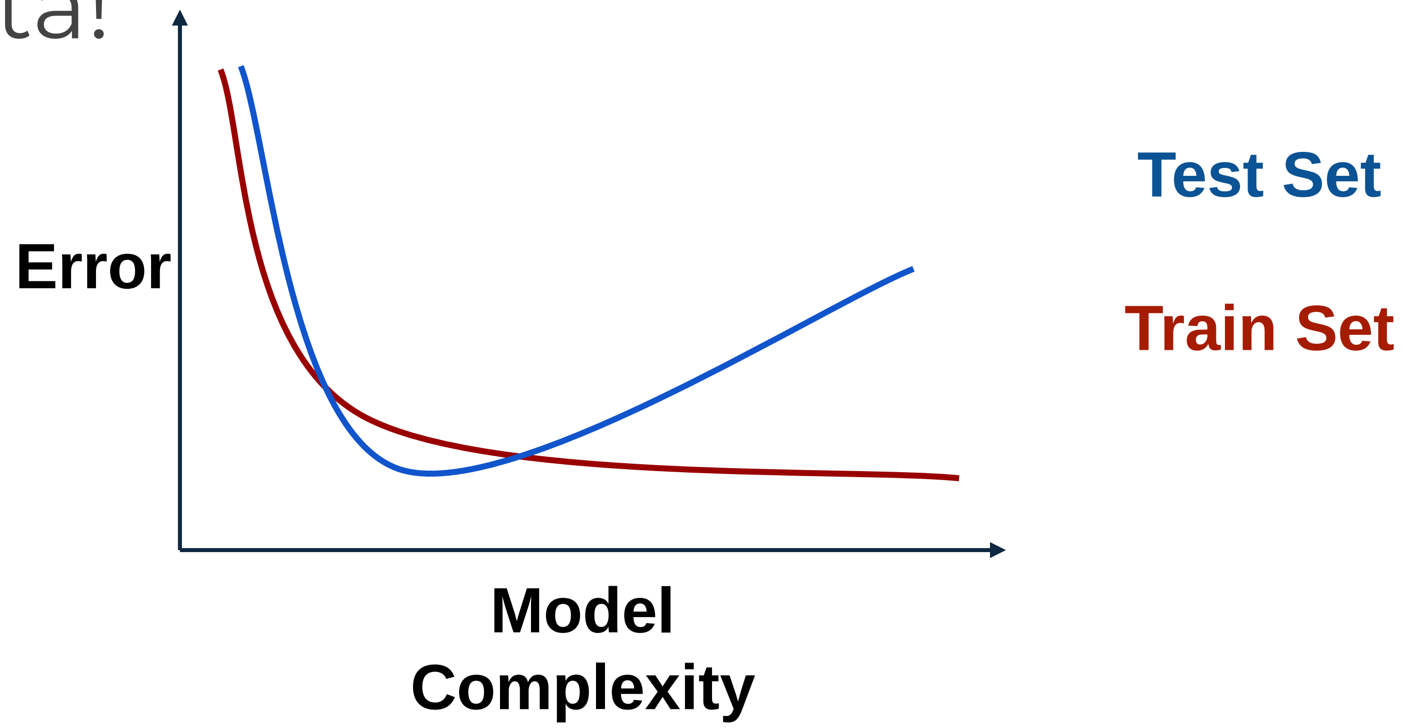
Overfitting and Underfitting

- But what happens if we overfit on the training data? That means we would perform poorly on new test data!



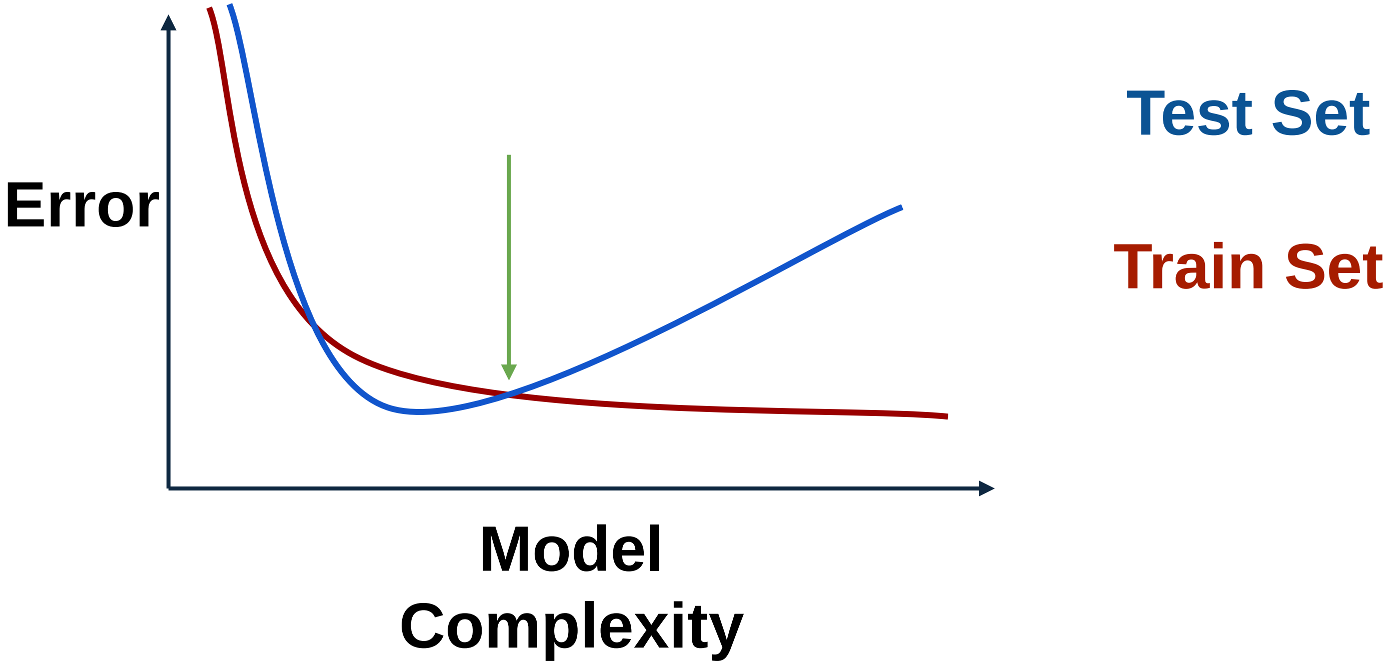
Overfitting and Underfitting

- But what happens if we overfit on the training data? That means we would perform poorly on new test data!



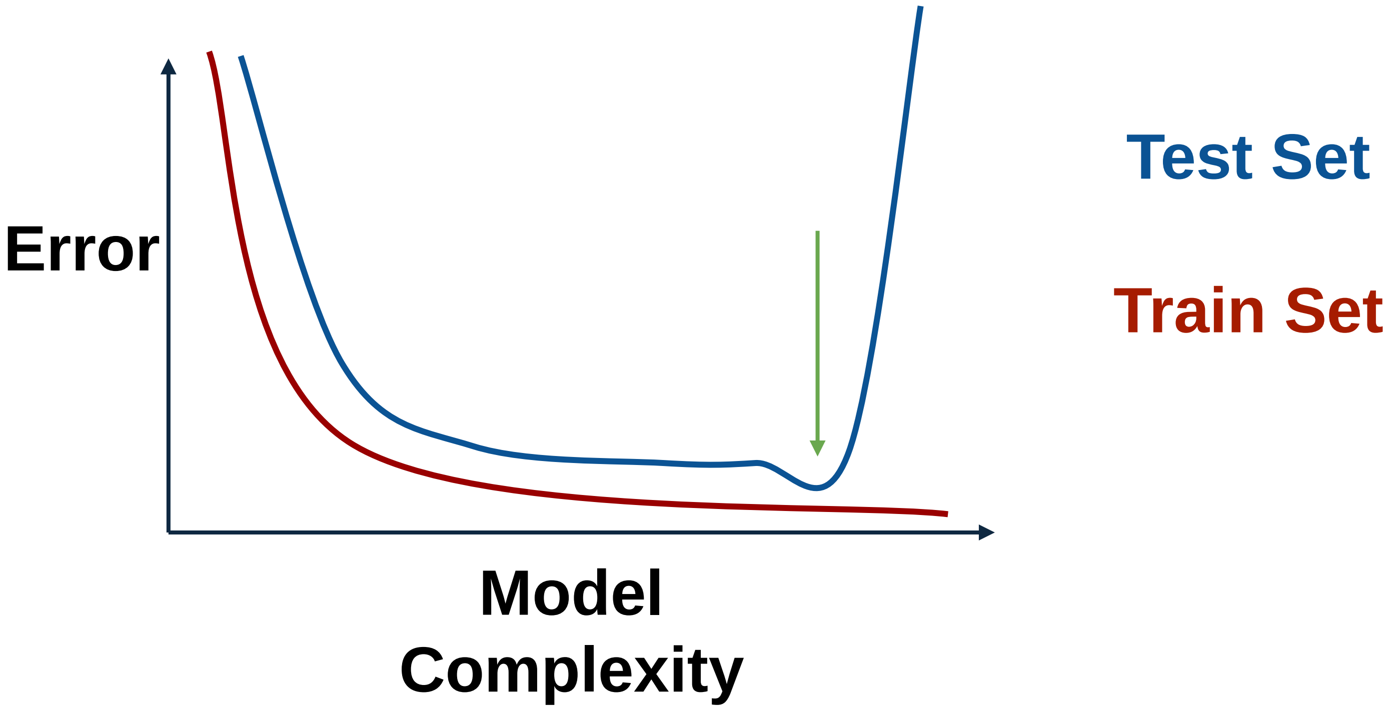
Overfitting and Underfitting

- This is a good indication too much complexity, you should look for the point to determine appropriate values!



Overfitting and Underfitting

- For certain algorithms this test error jump can be sudden instead of gradual.



Overfitting and Underfitting

- This means when deciding optimal model complexity **and** wanting to fairly evaluate our model's performance, we can consider both the train error and test error to select an ideal complexity.

Overfitting and Underfitting

- In the case of Polynomial Regression, complexity directly relates to degree of the polynomial, but many machine learning algorithms have their own hyperparameters that can increase complexity.