Berkeley Engineering | BerkeleyHaas

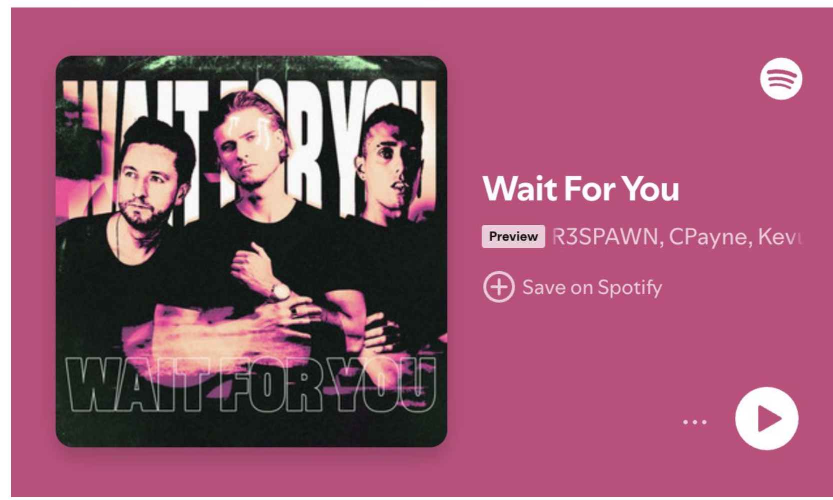
PROFESSIONAL CERTIFICATE IN MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

Let’s give everyone a couple of minutes to join…

Module 8

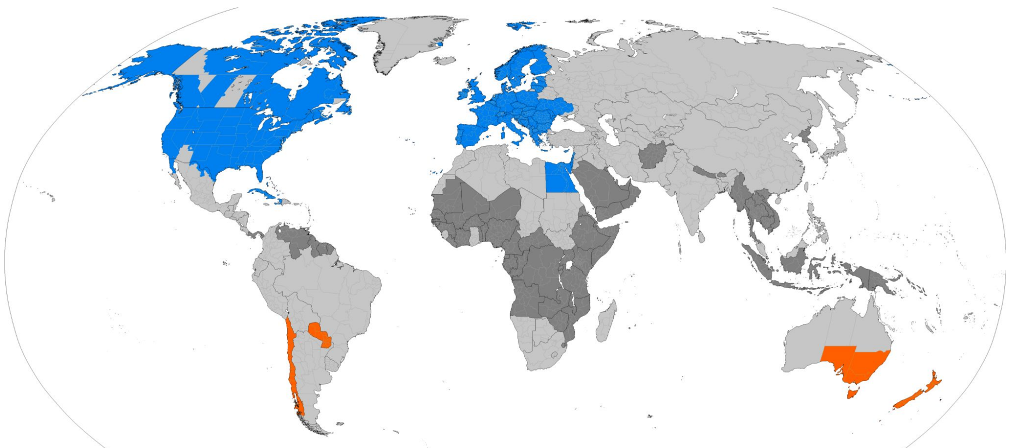
Feature Engineering and Overfitting

Office Hours with Viviana Márquez
October 24, 2024



https://open.spotify.com/track/63X08jZU1piJix8kvfoaq2?si=8e87e15cba5045b2

Daylight saving time!



Always check Canvas for the most up-to-date information regarding office hours!
Tool to convert to your timezone: https://www.worldtimebuddy.com/

Europe DST ends

North America DST ends

AGENDA

- Required activities for Module 8
- Content review Module 8: Feature Engineering and Overfitting
- Code
- Questions

Berkeley Engineering | Berkeley Haas

AGENDA

- Required activities for Module 8
- Content review Module 8: Feature Engineering and Overfitting
- Code
- Questions

Required Activities for Module 8

- Knowledge Check 8.1: Parabolic Model Fitting and Nonlinear Features
- Knowledge Check 8.2: Scikit-Learn Transformers
- Knowledge Check 8.3: Scikit-Learn Pipelines
- Codio Assignment 8.1: Scikit-Learn Pipeline
- Check 8.4: Order 0 Through 6 Models on Vehicle Data
- Knowledge Check 8.5: The Dangers of Overfitting
- Codio Assignment 8.2: Comparing Complexity and Variance
- Check 8.6: Overfitting and Validation
- Knowledge Check 8.7: Test Sets
- Codio Assignment 8.3: Evaluating Multiple Models

Berkeley Engineering | BerkeleyHaas

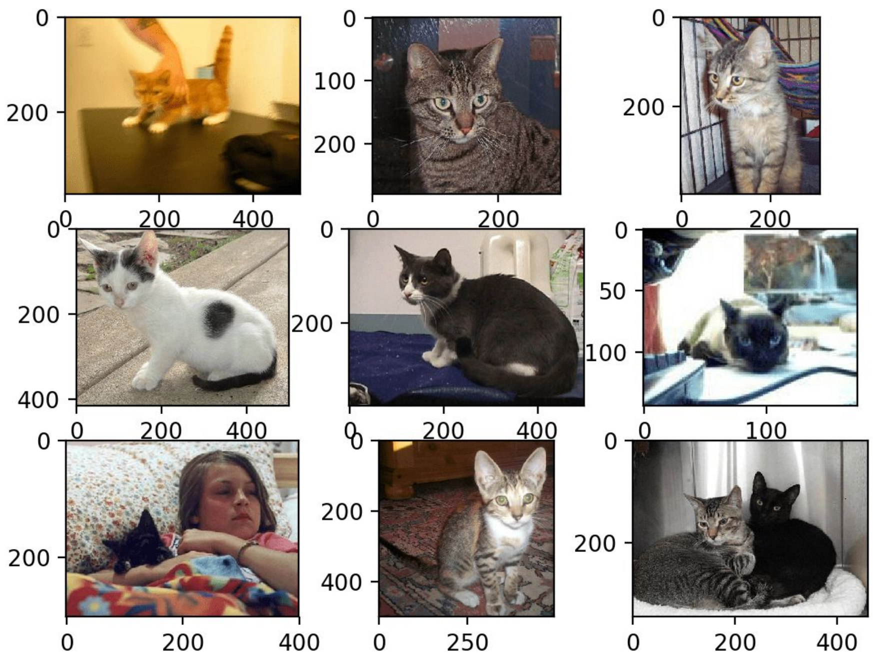
AGENDA

- ✅ Required activities for Module 8
- Content review Module 8: Feature Engineering and Overfitting
- Code
- Questions

Content review Module 8: Feature Engineering and Overfitting

- Quick recap Linear Regression
- Train/val/test datasets
- Sources of error in a model
- Bias-variance tradeoff
- Feature Engineering

What is Machine Learning?

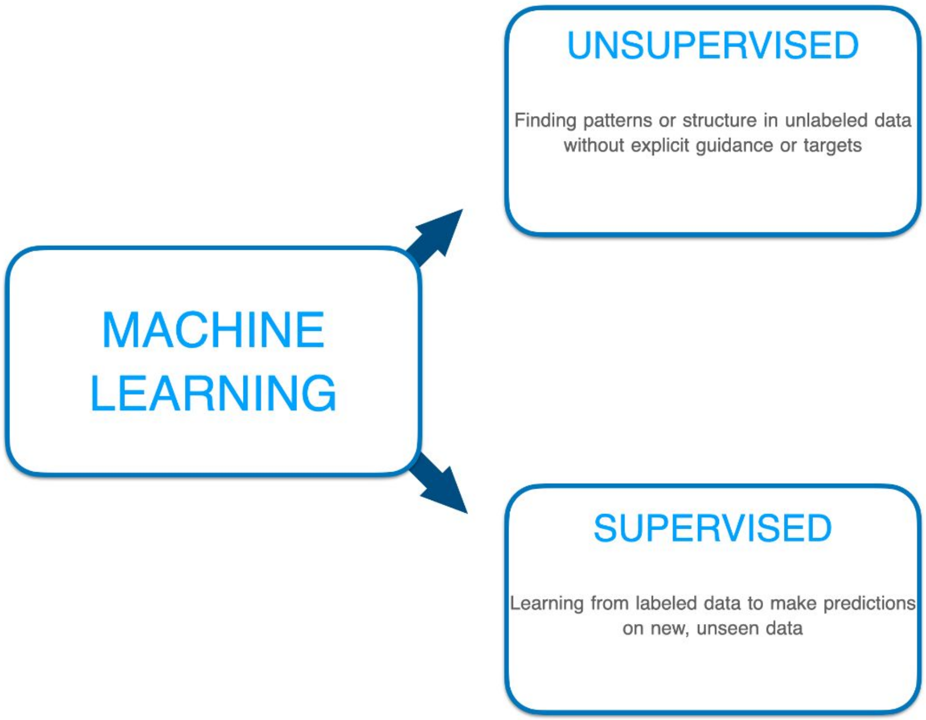


In machine learning, instead of explicitly programming a computer with specific instructions to perform a task, we provide it with large amounts of data and allow it to learn how to **generalize** from that data.

The Machine Learning landscape

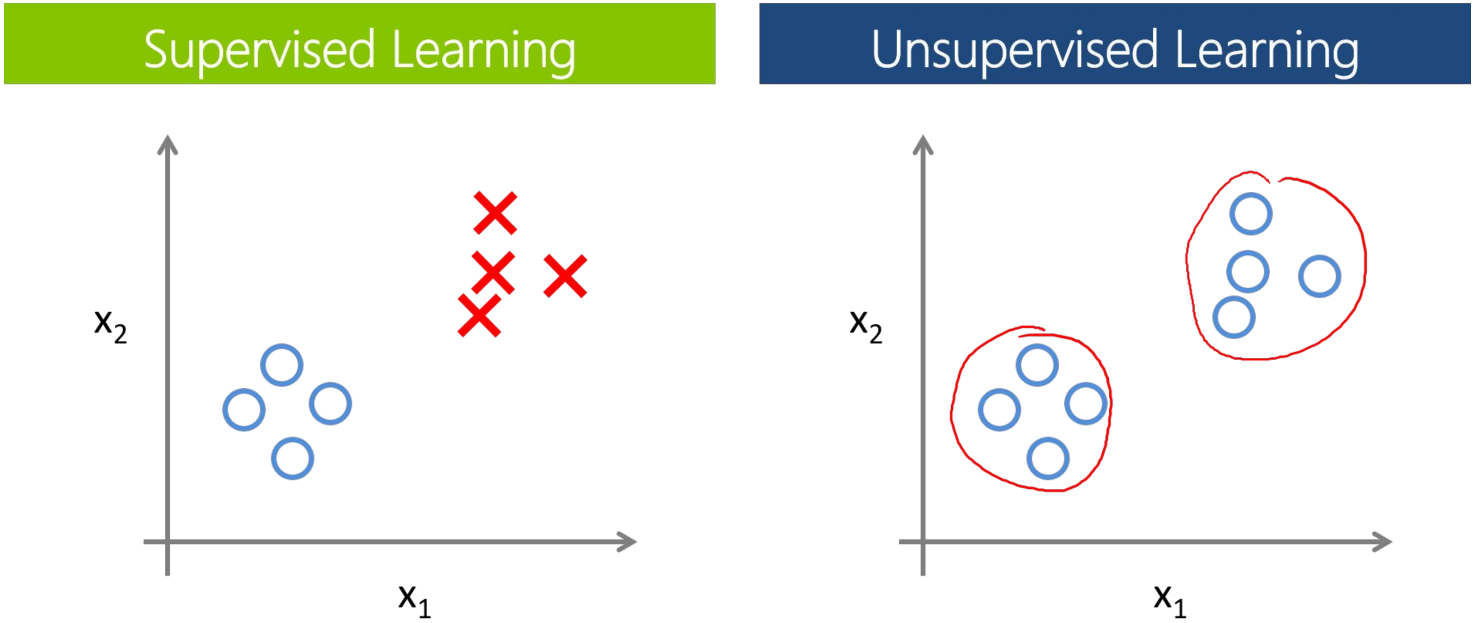
Which tool would you use to hit a nail?

First Question: Do we have labels?



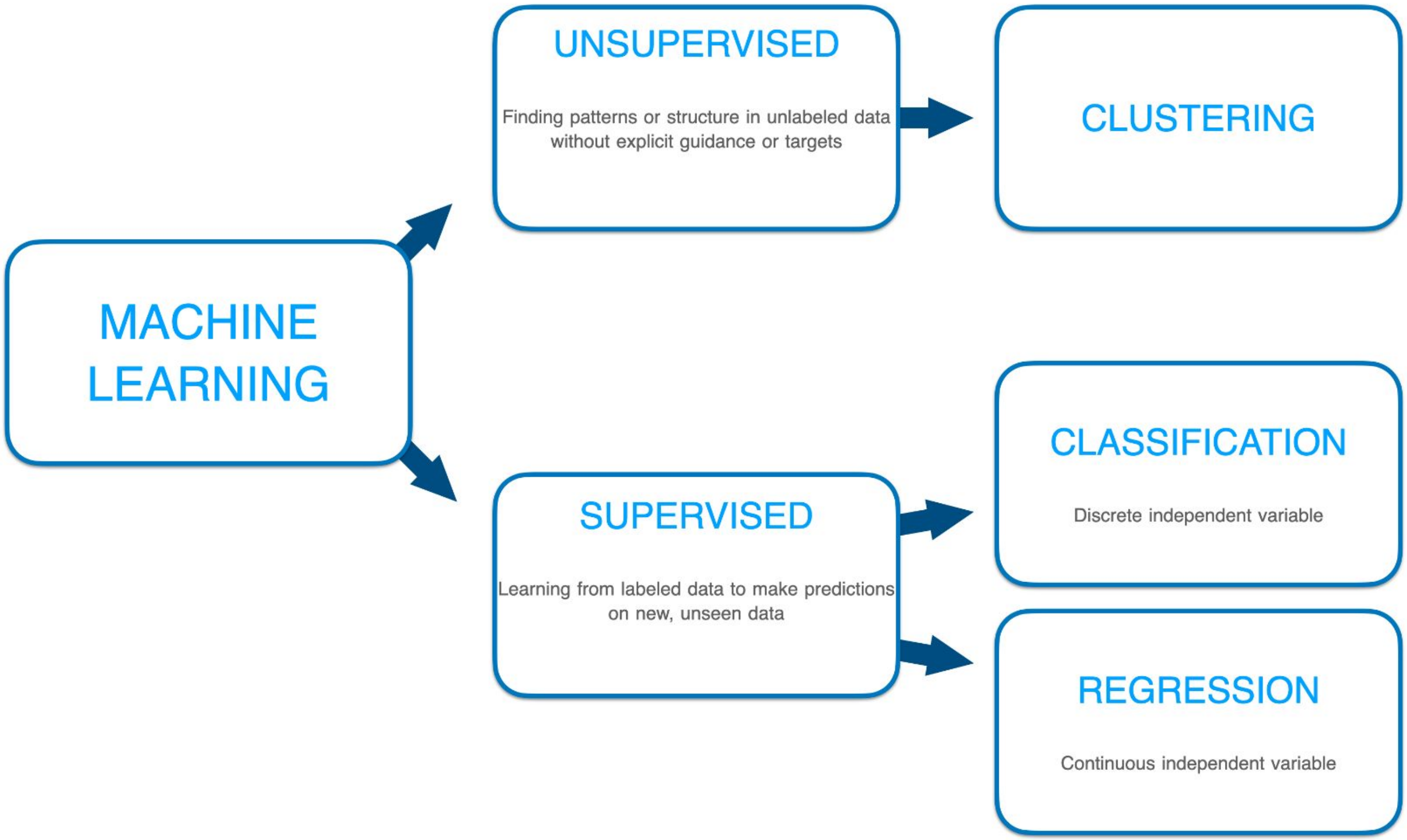
The Machine Learning landscape

Supervised learning vs Unsupervised learning

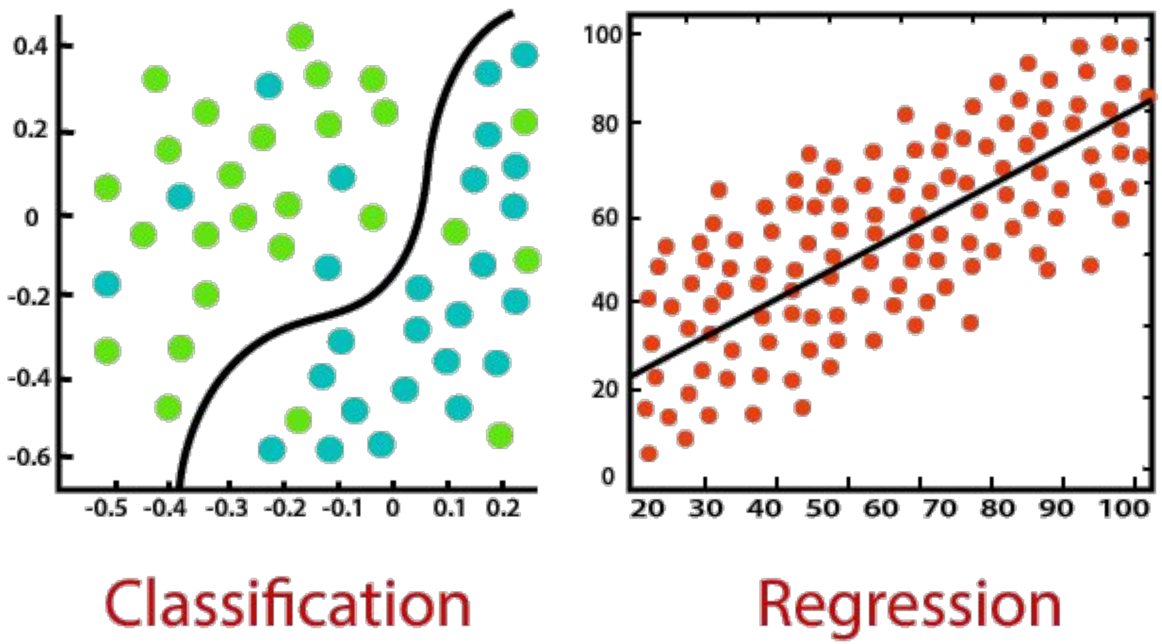


- **Supervised learning:** Problems with labels. Its aim is to predict data based on the labeled information.
- **Unsupervised learning:** Problems without labels. Its goal is to uncover patterns, structures, and relationships.

First Question: Do we have labels?
Second Question: Are our labels categorical or numerical?

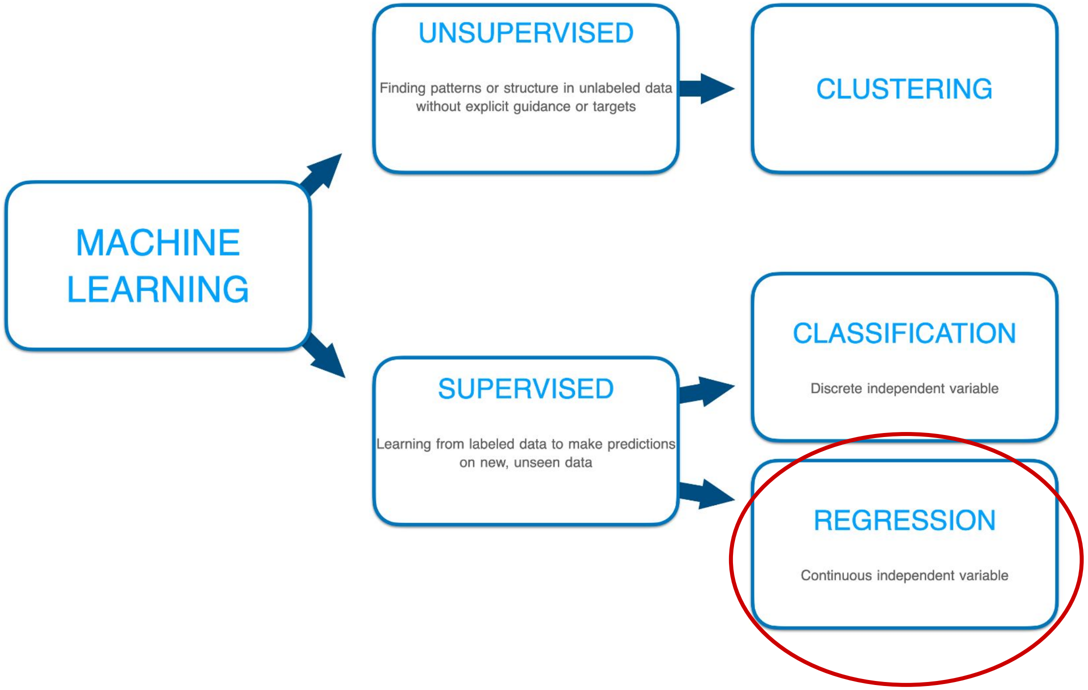


The Machine Learning landscape
Regression vs Classification



- **Regression:** Quantitative (continuous/numerical) target variable
- **Classification:** Qualitative (discrete/categorical) target variable

What type of data do we have?



Linear Regression

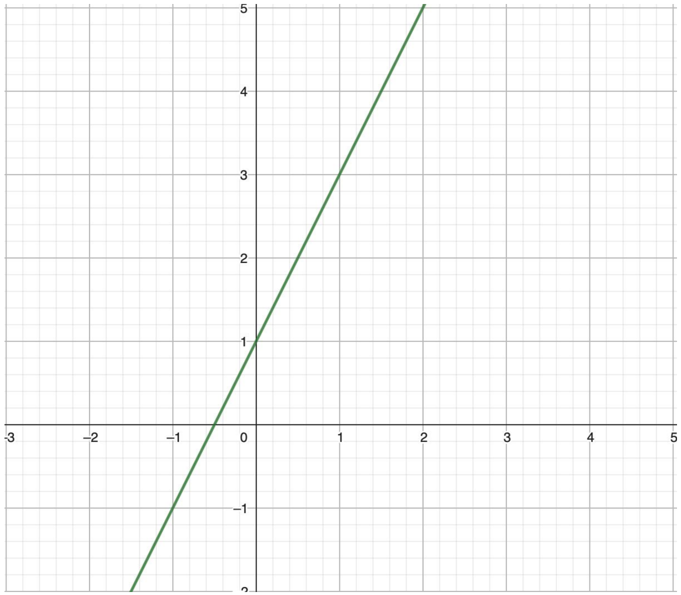
- Linear regression is one of the most well known machine learning models

- It is:
  - **Supervised** (has labels)
    - **Regression** (the labels are numerical values)

- The goal of a linear regression is to model the relationship between a dependent variable (target) and one or more independent variables (features) by fitting a linear equation

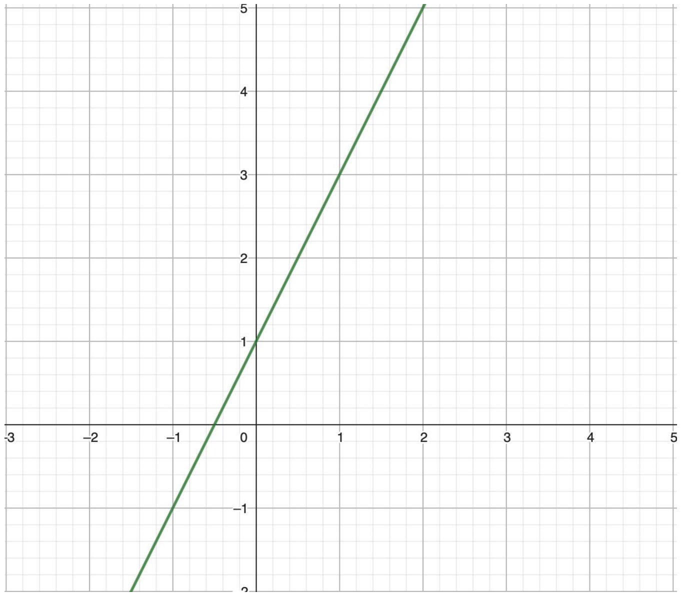
A tiny refresher

- What is the general equation of a straight line?



A tiny refresher

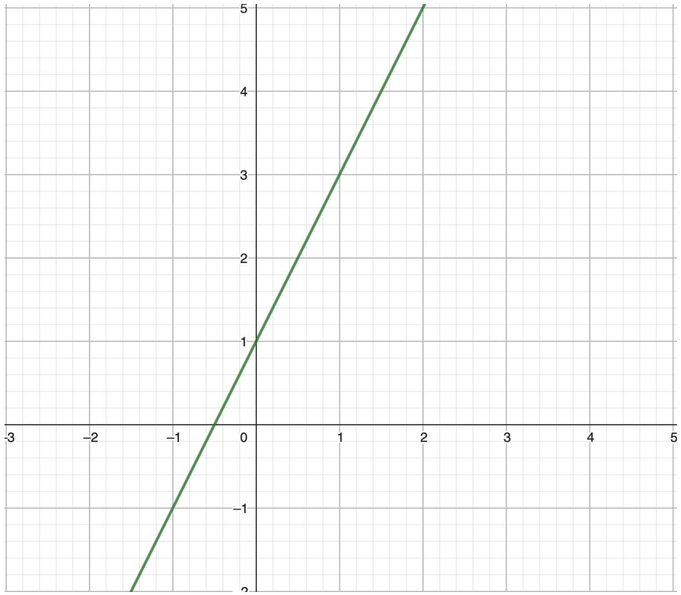
- What is the general equation of a straight line?



$$y = mx + b$$

A tiny refresher

- What is the general equation of a straight line?

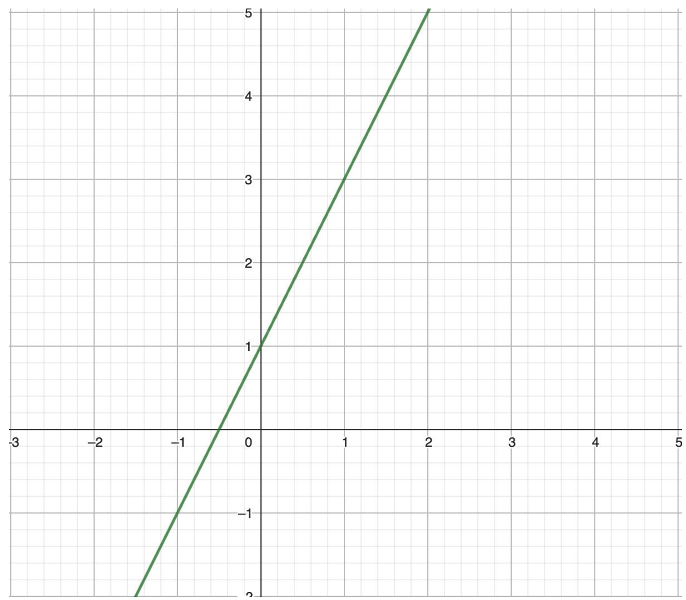


$$y = mx + b$$

Slope
(how steep the line is)

A tiny refresher

- What is the general equation of a straight line?



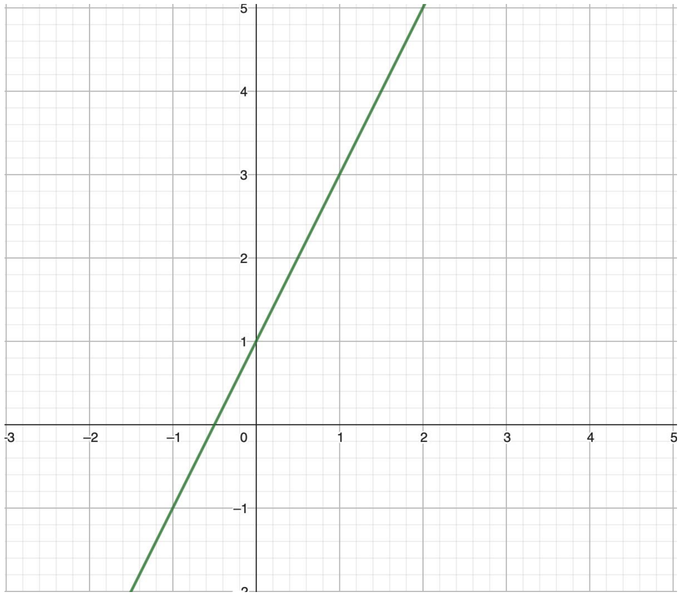
$$y = mx + b$$

Slope
(how steep the line is)

y-intercept

A tiny refresher

- What is the general equation of a straight line?



$$y = mx + b$$

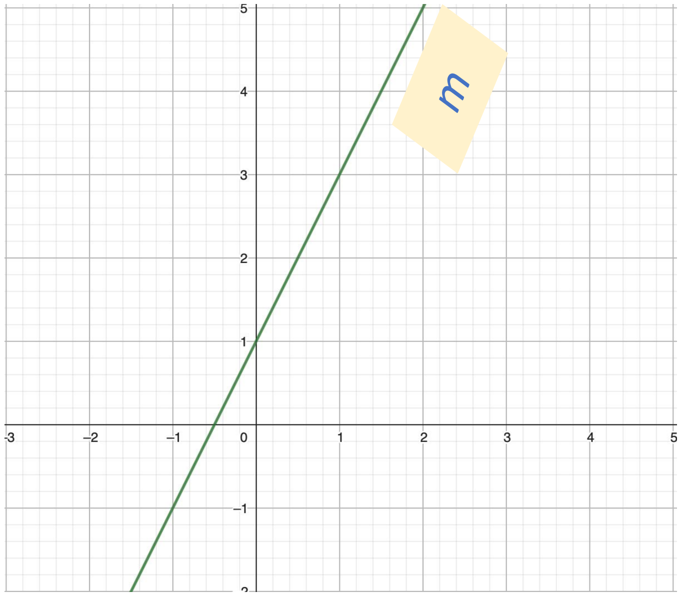
Slope
(how steep the line is)

y-intercept

$$y =$$

A tiny refresher

- What is the general equation of a straight line?



$$y = mx + b$$

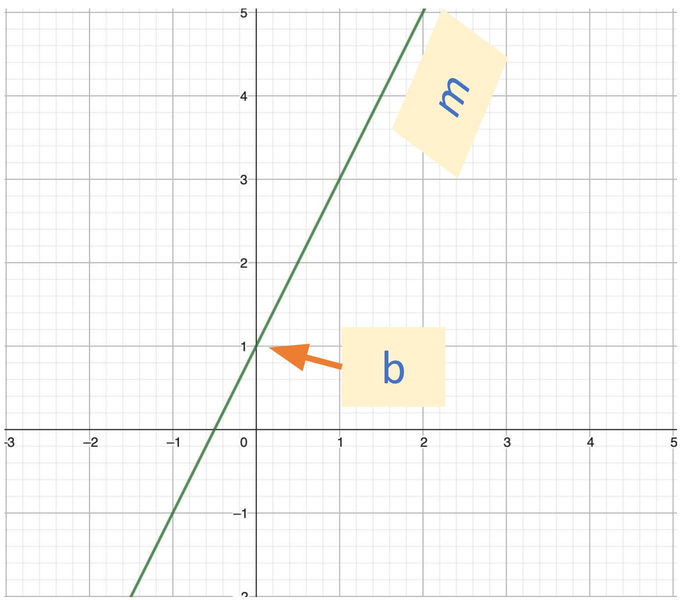
Slope
(how steep the line is)

y-intercept

$$y = 2x$$

A tiny refresher

- What is the general equation of a straight line?



$$y = mx + b$$

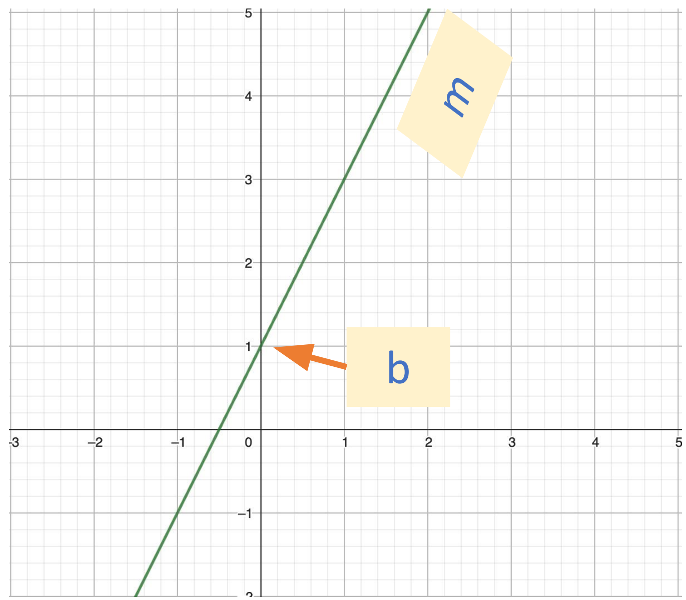
Slope
(how steep the line is)

y-intercept

$$y = 2x + 1$$

A tiny refresher

- What is the general equation of a straight line?



Input

$$y = mx + b$$

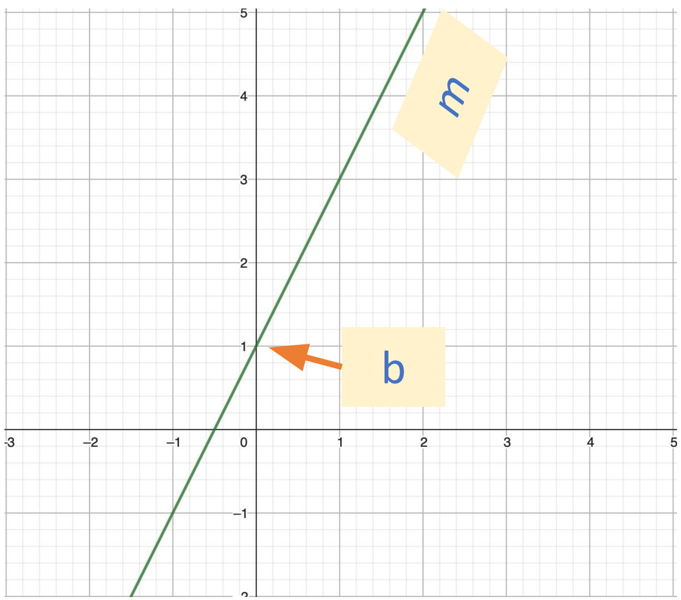
Slope
(how steep the line is)

y-intercept

$$y = 2x + 1$$

A tiny refresher

- What is the general equation of a straight line?



Output (Target)

Input (Features)

$$y = mx + b$$

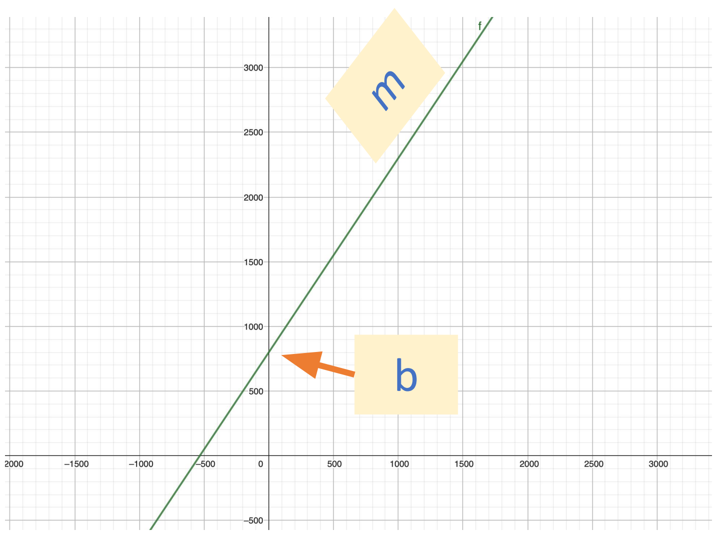
Slope (how steep the line is)

y-intercept

$$y = 2x + 1$$

A tiny refresher

- Example: Predict monthly rent based on the sq ft of an apartment



Output
(rent-dollars)

Input
(square footage)

$$y = 1.5x + 800$$

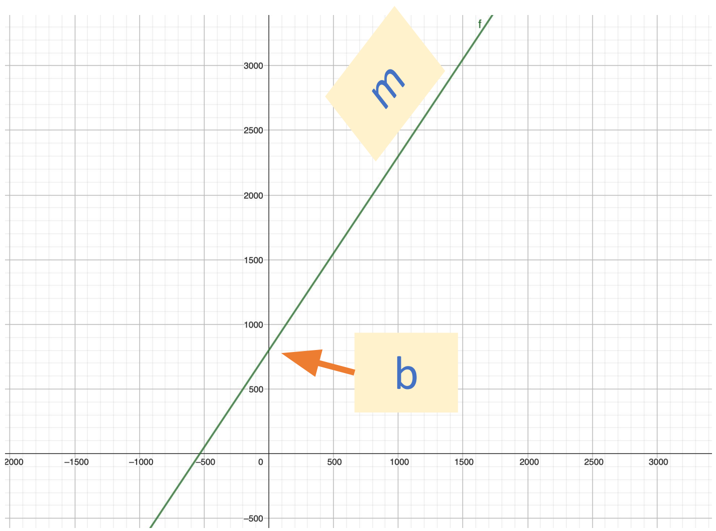
Slope (steepness)
For every additional square foot, the rent increases by $1.50

y-intercept
The rent of an apartment when the footage is is zero (can be thought of as a baseline)

A tiny refresher

How can we find the best values for the slope and the y-intercept? 🤔

- Example: Predict monthly rent based on the sq ft of an apartment



Output (rent-dollars)

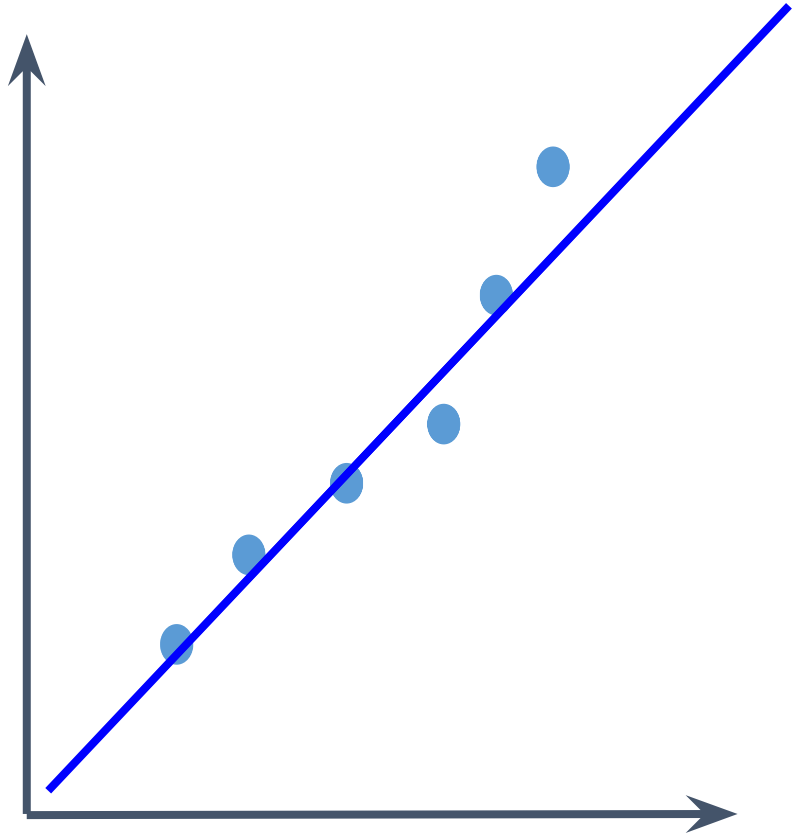
Input (square footage)

$$y = 1.5x + 800$$

Slope (steepness)
For every additional square foot, the rent increases by $1.50

y-intercept
The rent of an apartment when the footage is is zero (can be thought of as a baseline)

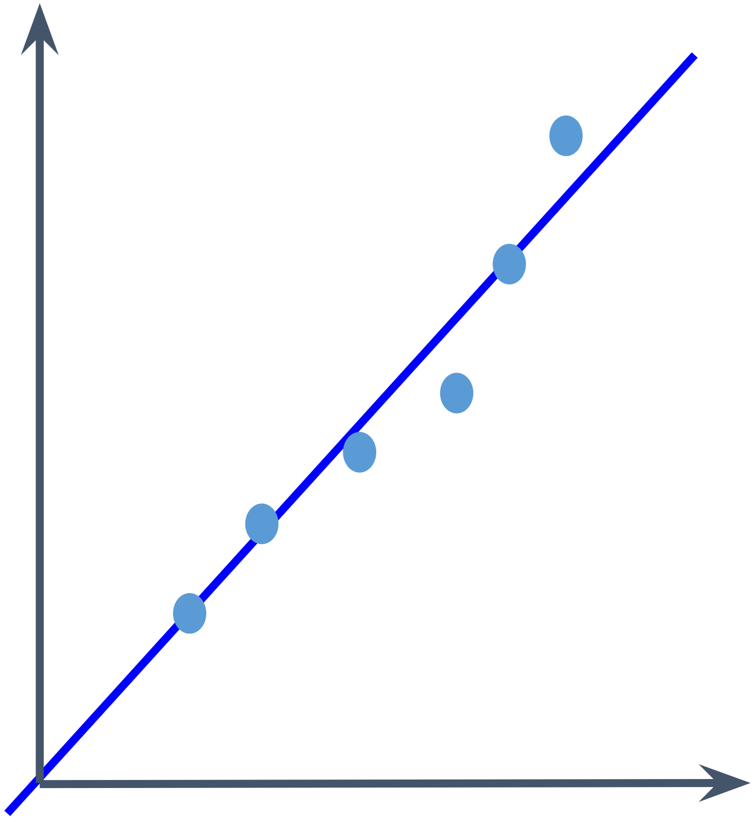
How can we find the best values for the slope and the y-intercept? 🤔



And that’s Linear Regression!

- Linear regression is one of the most well known machine learning models
- It is:
  - **Supervised** (has labels)
    - **Regression** (the labels are numerical values)
- The algorithm tries to find the best-fitting straight line (in simple regression) or hyperplane (in multiple regression) that models the relationship between a dependent variable (target) and one or more independent variables (features)
- This "best fit" is often determined by minimizing the difference (or error) between the predicted values and the actual observed values, a process known as Ordinary Least Squares (OLS)
- Simple, interpretable, very fast, and the best for linear relationships
- Usually a lower bound on performance, but often form the foundation for other, more powerful techniques
- If we combine multiple linear models and add a twist to it (an activation function), we get a neural network; those are incredibly useful and powerful! 🔥🔥🔥

How can we find the best values for the slope and the y-intercept? 🤔

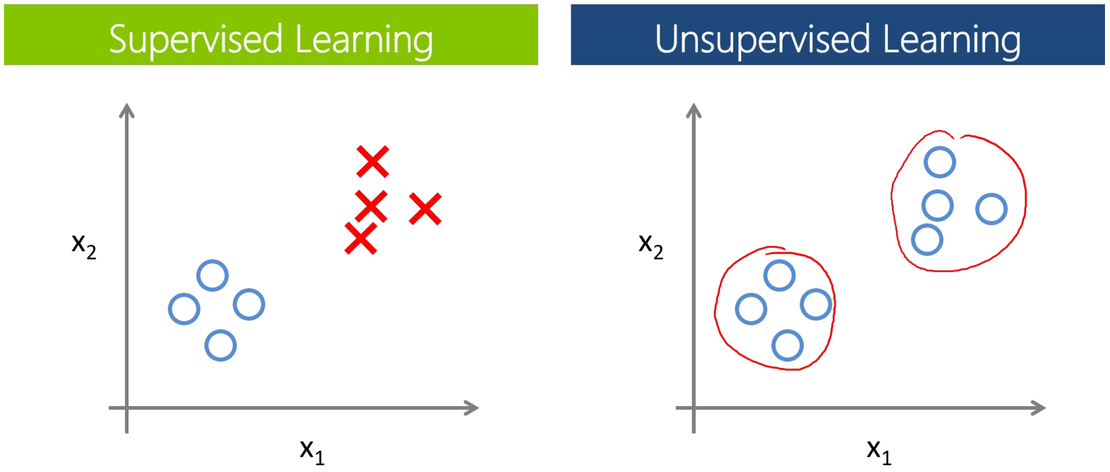


```
lm = LinearRegression()
lm.fit(X, y)
```

It's so easy using code!

The Machine Learning landscape

Supervised learning vs Unsupervised learning



- **Supervised learning:** Problems with labels
- **Unsupervised learning:** Problems without labels

Train/val/test data sets

Is my model any good?

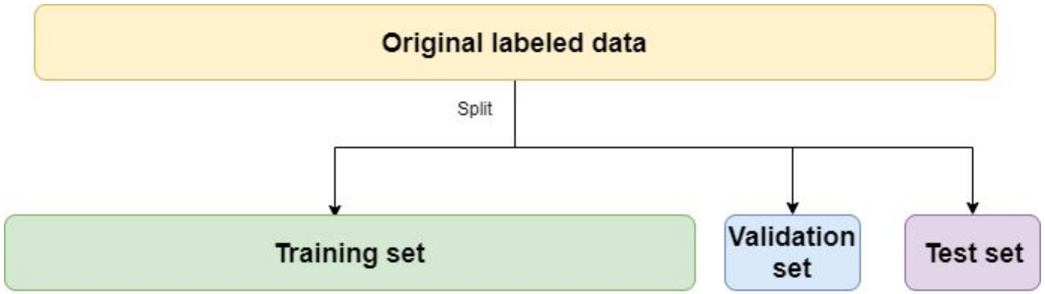
Train/val/test data sets

Is my model any good?

- A good model makes useful predictions on unknown, future data (it **generalizes**)
- It might not be *very* accurate, but if it's better than a coin flip, it might still be useful

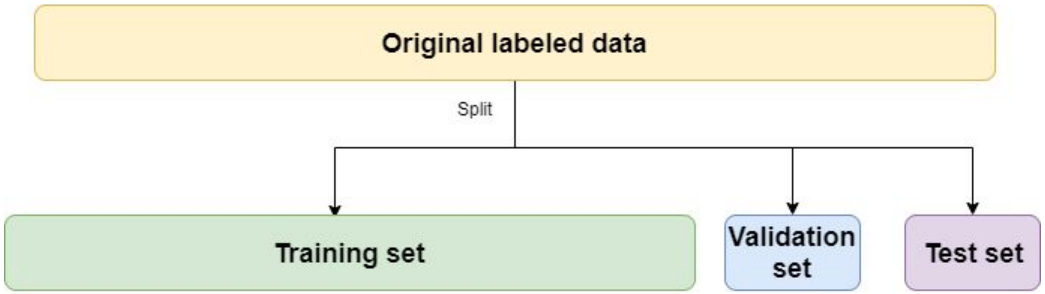
Train/val/test data sets

In order to make sure the model is learning to generalize, we split our dataset



- **Training data set:** Used to train machine learning model
- **Validation data set:** Used to tune the model hyperparameters
- **Test data set:** Evaluate the model's performance on unseen data, our best proxy on how the model will perform in the real world

Train/val/test data sets



- In general, putting 80% of the data in the training set, 10% in the validation set, and 10% in the test set is a good split to start with
- Most of the time, the data should be split randomly to be representative of the whole population

Train/val/test data sets

- It is important to split the data into train/val/test **BEFORE** doing any feature engineering or modeling to prevent **data leakage** and ensure that the model is truly evaluated on unseen data

Train/val/test data sets

- It is important to split the data into train/val/test **BEFORE** doing any feature engineering or modeling to prevent **data leakage** and ensure that the model is truly evaluated on unseen data

Data leakage: When information from val or test is used to inform the model during training or feature engineering

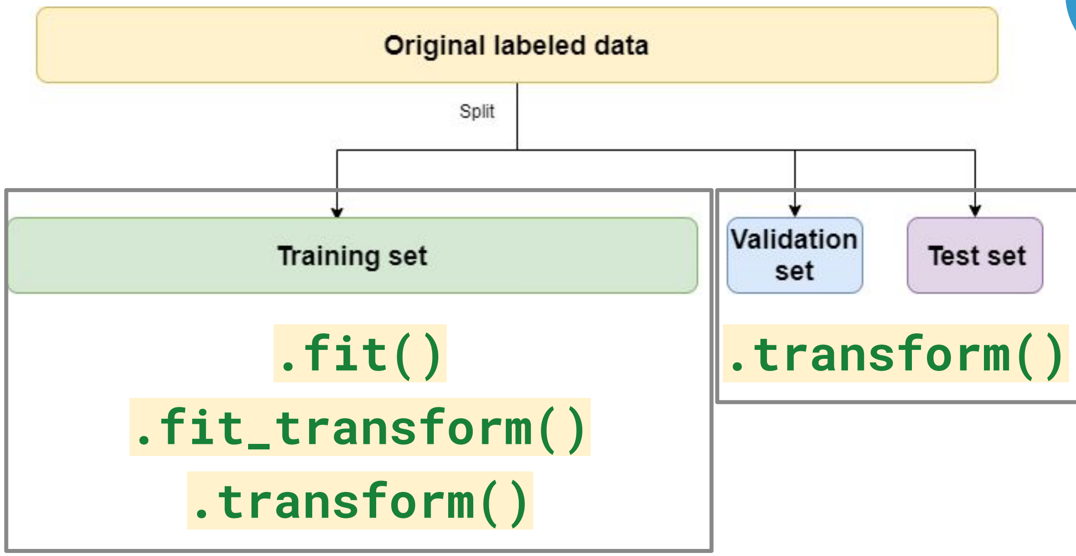
Train/val/test data sets

- It is important to split the data into train/val/test **BEFORE** doing any feature engineering or modeling to prevent **data leakage** and ensure that the model is truly evaluated on unseen data

Data leakage: When information from val or test is used to inform the model during training or feature engineering

No peeking!!!
The test data set is used **ONLY** after you think you have the best model. It's the only true measure of generality.

Train/val/test data sets

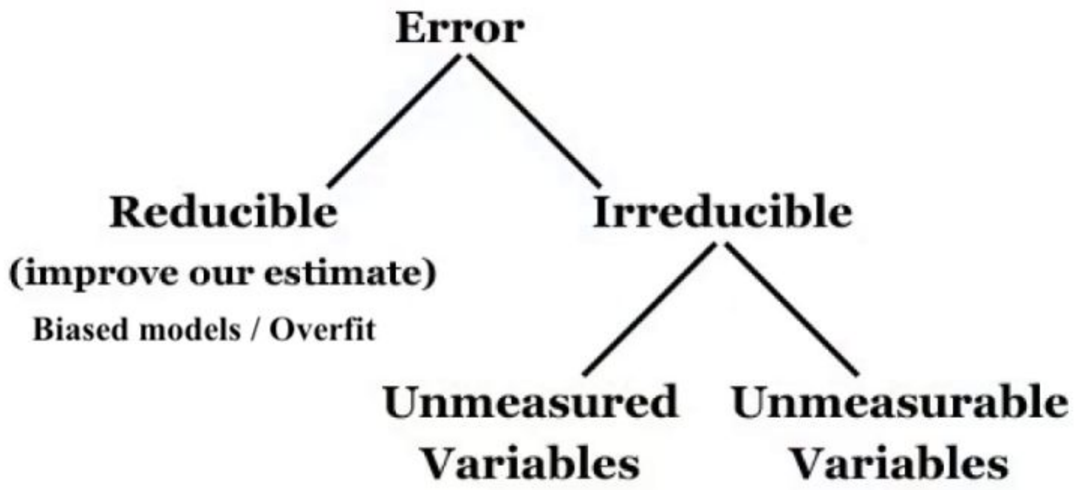


- **.fit():**
  This is the learning step, where the model or transformer observes the data to derive key parameters (e.g., coefficients in linear regression or decision boundaries in classification or the mean and variance in scaling). **ONLY** using on the train dataset. This is because the model should not learn from the val/test data—its purpose is to evaluate how well the model generalizes to unseen data.

- **.transform():**
  This step applies the previously learned transformation to new data (such as the validation or test set) without recalculating any parameters. Used in: Train/Val/Test sets to ensure the same transformation process is applied consistently.

- **.fit_transform():**
  Combined operation to save time when you want to learn and apply transformations simultaneously. **ONLY** using on the train dataset.

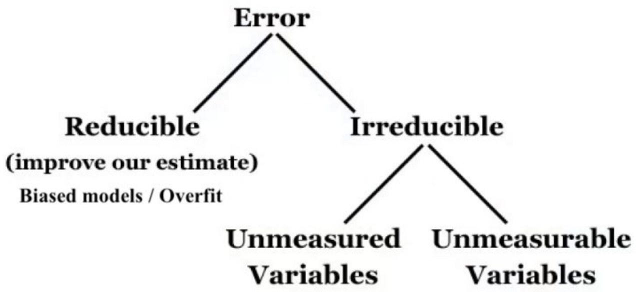
Sources of prediction error



Sources of prediction error

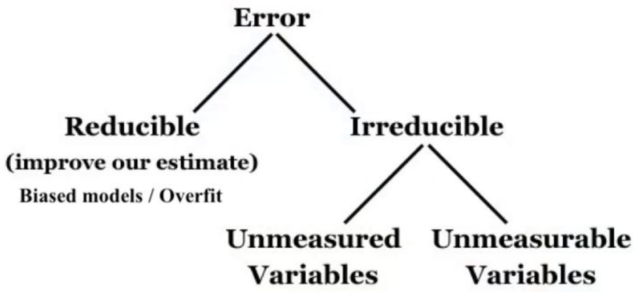
We are given ***(X,y)*** training data and we fit a model ***f(X)***

Err = (f(Xi)-yi)^2 from a single observation in the test case



Sources of prediction error

We are given ***(X,y)*** training data and we fit a model ***f(X)***



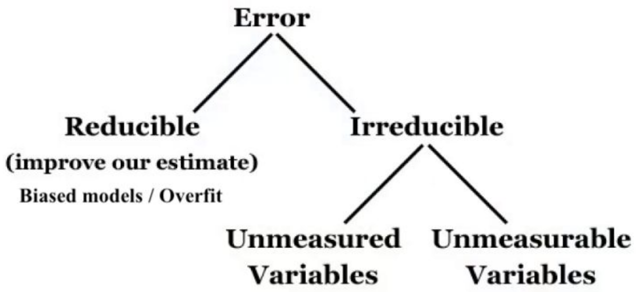
Err = (f(Xi)-yi)^2 from a single observation in the test case

There are three sources of errors in that ***Err*** number:

1. Noisy **X** or **y** data, such as inconsistent **X→y**
2. Model underfitting or bias: Too weak or simple
3. Model overfitting: Model too specific to training data

Sources of prediction error

We are given ***(X,y)*** training data and we fit a model ***f(X)***



Err = (f(Xi)-yi)^2 from a single observation in the test case

There are three sources of errors in that ***Err*** number:

1. Noisy **X** or **y** data, such as inconsistent **X→y**
2. Model underfitting or bias: Too weak or simple
3. Model overfitting: Model too specific to training data

Conceptually: Err = "noise" + "bias" + "overfitting”

Stats nerds: Err = Irreducible Error + Bias^2 + Variance

Sources of prediction error

1. Noise can lead to inconsistent data

- Imagine you have two observations such as:
  $[18,1,9] \rightarrow 91$
  $[18,1,9] \rightarrow 99$

- No model can predict two different *y* values for the same *x* vector

- Model will have *Err>0* no matter what

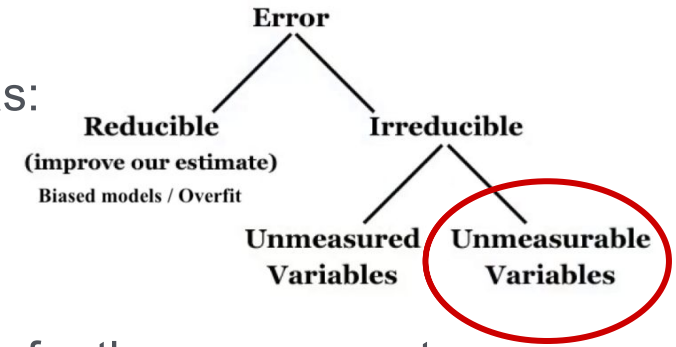
- We know this as the **irreducible error**

- Noise comes from faulty sensors, typos, self-reporting issues, etc…

- 🤷 Nothing we can do about the irreducible error

Sources of prediction error

1. Noise can lead to inconsistent data



- Imagine you have two observations such as:
  [18,1,9] → 91
  [18,1,9] → 99
- No model can predict two different *y* values for the same *x* vector
- Model will have *Err>0* no matter what
- We know this as the **irreducible error**
- Noise comes from faulty sensors, typos, self-reporting issues, etc…
- 🤷 Nothing we can do about the irreducible error

Sources of prediction error

1. Noise can lead to inconsistent data

- What if inconsistent training observations, such as:

  [18,1,9] → 91

  [18,1,9] → 99

- were really just missing a variable we don't have?

  [18,1,9,10] → 91

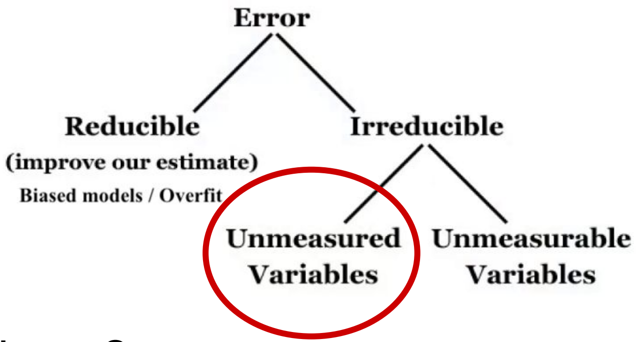
  [18,1,9,7] → 99

- Example: Two apartment observations look identical: 2bd & 1bath, but they have very different price only because we lack "awesome views" vars

- Missing variables are called *exogenous* variables

Sources of prediction error

1. Noise can lead to inconsistent data



- What if inconsistent training observations,

  [18,1,9] → 91

  [18,1,9] → 99

- were really just missing a variable we don't have?

  [18,1,9,10] → 91

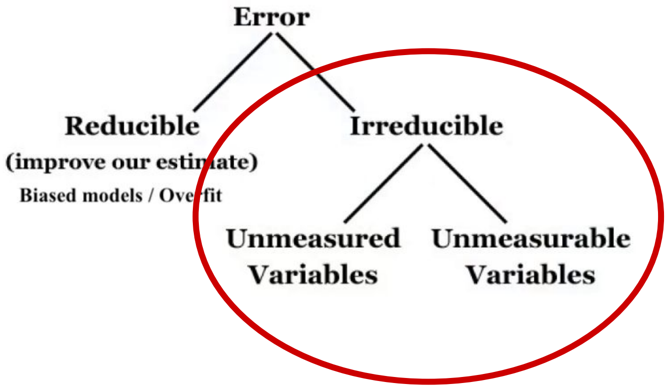
  [18,1,9,7] → 99

- Example: Two apartment observations look identical: 2bd & 1bath, but they have very different price only because we lack "awesome views" vars

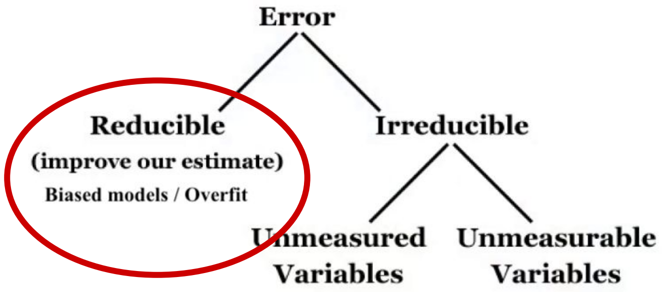
- Missing variables are called *exogenous* variables

Sources of prediction error

1. Noise can lead to inconsistent data



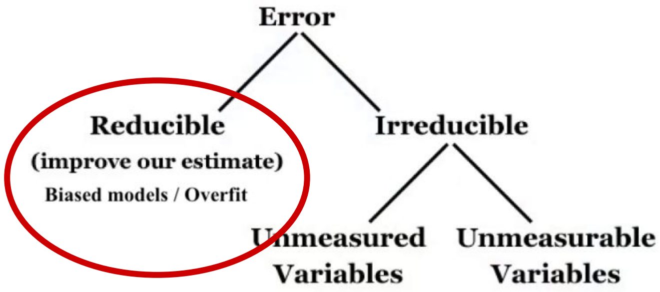
Sources of prediction error



There are three sources of errors in that ***Err*** number:

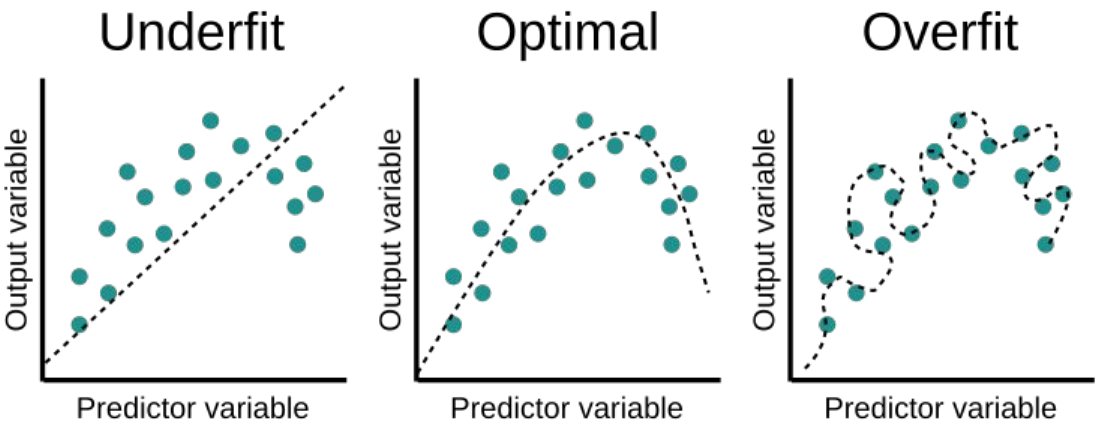
1. Noisy **X** or **y** data, such as inconsistent **X**→**y**
2. Model underfitting or bias: Too weak or simple
3. Model overfitting: Model too specific to training data

Sources of prediction error

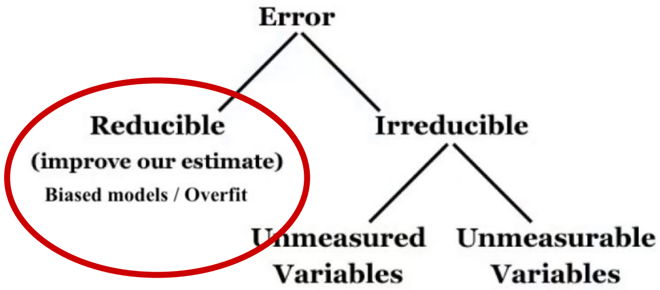


There are three sources of errors in that ***Err*** number:

1. Noisy **X** or **y** data, such as inconsistent **X**→**y**
2. Model underfitting or bias: Too weak or simple
3. Model overfitting: Model too specific to training data



Sources of prediction error



There are three sources of errors in that ***Err*** number:

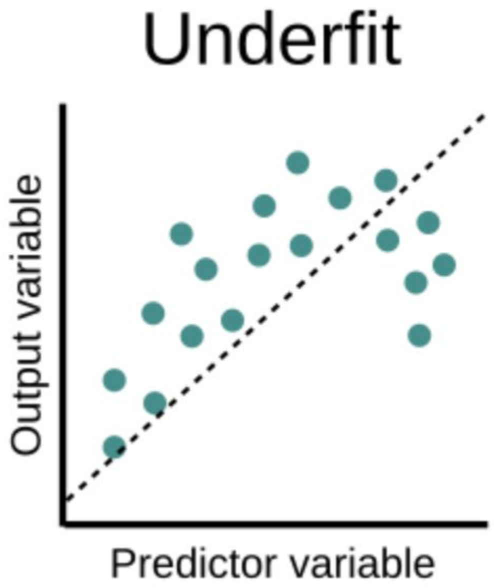
1. Noisy **X** or **y** data, such as inconsistent **X**→**y**
2. Model underfitting or bias: Too weak or simple
3. Model overfitting: Model too specific to training data

Underfit

Optimal

Overfit

Overly simple models lead to biased models



- **Bias** is the error rate of your model on the training set

- Bias is how much your model **underfits** the training data

Overly simple models lead to biased models

How do you compute bias?

$$Bias[\hat{f}(x)] = E[\hat{f}(x) - y]$$

Expected difference between predicted and observed

Overly simple models lead to biased models

👮‍♀️ Check-in

A model that has a good ability to fit the training data has ________ bias

Overly simple models lead to biased models

👮‍♀️ Check-in

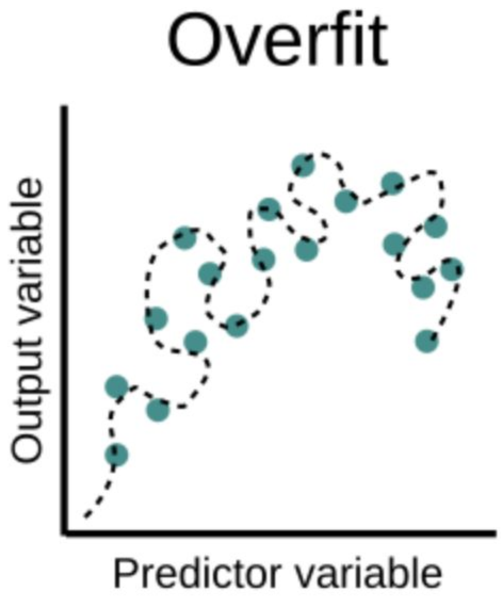
A model that has a good ability to fit the training data has **LOW** bias

- We want to minimize bias

- Models with high bias:
    - Fail to capture meaningful patterns in data
    - Under-fit training data

- How to decrease bias? Make model more complex!
    - Add more parameters
    - Pick a different model

Overly complex models can overfit



- **Variance** is the amount a model's prediction will change if a different training data is used, ie, small changes in the training data can result in large changes in the estimated model
- **Variance** in a model is the flexibility to learn patterns in the observed data
- Variance is how much your model **overfits** the training data

Overly complex models can overfit

How do you compute variance?

$$Var[\hat{f}(x)] = E[\hat{f}(x)^2] - (E[\hat{f}(x)])^2$$

Intuitively, how much the algorithm will move around its mean

Overly complex models can overfit

👮‍♀️ Check-in

A model that is strongly influenced by the specifics of the training data has ________ variance

Overly complex models can overfit

👮‍♀️ Check-in

A model that is strongly influenced by the specifics of the training data has **HIGH** variance

- We want to minimize variance. A model that has a good ability to predict test data has **low** variance.

- The more complex the model is, the more data points it will "capture". However, complexity will make the model "move" more to "capture" the data points, and hence its variance will be larger

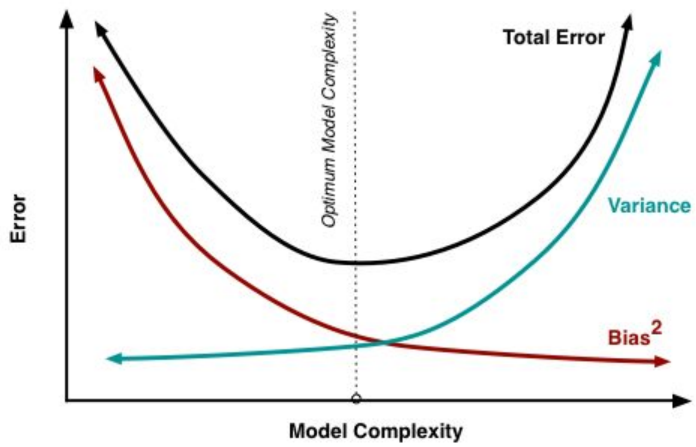
- How to decrease variance? Make model less complex!
  - A larger training set tends to decrease variance, ie, reduces the chance of overfitting --> increases the chance of generalization
  - Regularization

Bias-variance trade-off

So... we need to decrease both bias and variance (to avoid underfitting and overfitting)

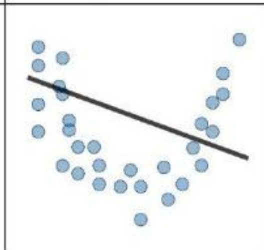
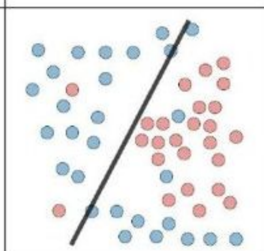
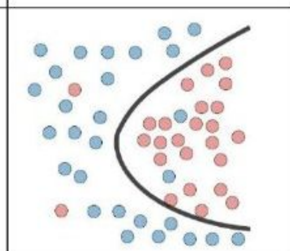
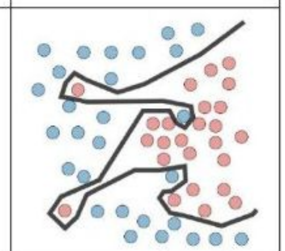
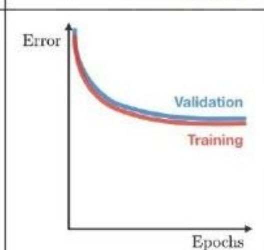
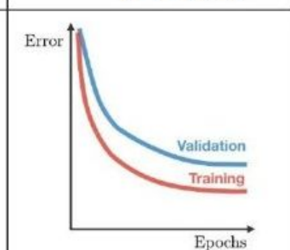
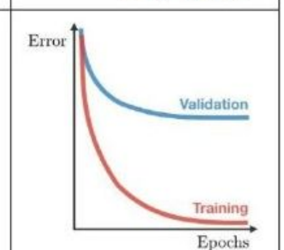

Low Bias
Low
Variance

Bias-variance tradeoff



Let the validation dataset be your guide to approximate complexity (typically using cross-validation)

- Collect a lot of data
- Engineer good features
- Pick a complex algorithm
- Train the specific model until validation scores starts to go down (smart early stopping)

	Underfitting	Just right	Overfitting
Symptoms	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Regression			
Classification			
Deep learning			
Remedies	- Complexify model - Add more features - Train longer		- Regularize - Get more data

Summary

- **Bias** is the error rate of your model on the training set. Bias is how much your model **underfits** the training data

- **Variance** is the amount a model's prediction will change if a different training data is used. Variance is how much your model **overfits** the training data

- We want to always **minimize both bias and variance**, but when one goes down, the other one goes up (more/less complex model)– hence **bias-variance tradeoff**


THE BEST WAY TO
EXPLAIN OVERFITTING
imgflip.com


My model
with 99%
accuracy
Me
Overfitting


NOT SURE IF GOOD MODEL...
...OR JUST OVERFITTING
memegenerator.net


A GOOD EXAMPLE OF
OVERFITTING

Is my model any good?

Performance metrics

- Measurements used to evaluate how well a machine learning model is performing.

- The choice of performance metrics depends on the nature of the problem, whether it's a classification, regression, clustering task, and so on.

Metrics for regression models	**Metrics for classification models**
In a regression problem, you're trying to predict a continuous outcome variable (like the price of a house). Performance metrics in regression evaluate the difference between the true and predicted values, often based on errors.	In a classification problem, you're trying to predict which category or class an observation belongs to (like spam vs. non-spam emails). Performance metrics in classification evaluate how well the model can correctly classify the observations.

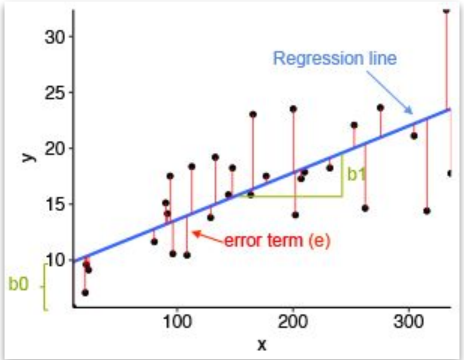
Performance metrics

- Measurements used to evaluate how well a machine learning model is performing.

- The choice of performance metrics depends on the nature of the problem, whether it's a classification, regression, clustering task, and so on.

Metrics for regression models	Metrics for classification models
• MAE (Mean Absolute Error) • Average Error • MAPE (Mean Absolute Percentage Error) • **RMSE (Root Mean Squared Error)** • SST (Total Sum of Squared Errors) • R-Squared • **Adjusted R-Squared** • and many more… (External link)	• Misclassification rate • **Accuracy** • Sensitivity • Specificity • **Recall** • **Precision** • **F1** • **ROC-AUC** • and many more… (Wiki link)

Performance metrics for regression models



Performance metrics in regression evaluate the difference between the true and predicted values, often based on errors.

MAE (Mean Absolute Error)

- MAE measures the average magnitude of the errors in a set of predictions, without considering their direction.

- It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

Divide by the total number of data points

Predicted output value

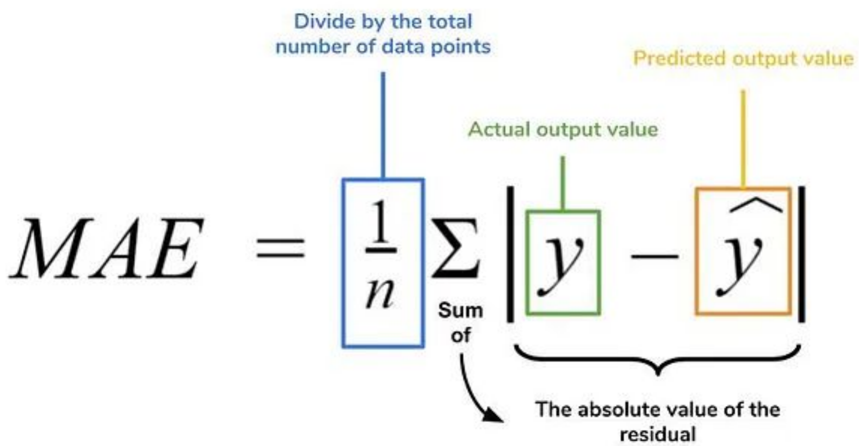
Actual output value

$$MAE = \boxed{\frac{1}{n}} \underset{\text{Sum of}}{\sum} \underbrace{\left| \boxed{y} - \boxed{\hat{y}} \right|}_{\text{The absolute value of the residual}}$$

MAE (Mean Absolute Error)

- MAE measures the average magnitude of the errors in a set of predictions, without considering their direction.

- It’s the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.



MSE (Mean Square Error)

- Both the mean squared error and the mean absolute error tell you how close a regression line is to a set of points.

- MSE is just like the MAE but squares the difference before summing them all instead of using the absolute value, therefore, it assigns more weight to the bigger errors.

Divide by the total number of data points

Actual output value

Predicted output value

$$MSE = \boxed{\frac{1}{n}}\sum\Big(\underbrace{\boxed{y} - \boxed{\hat{y}}}_{\text{The square of the difference between actual and predicted}}\Big)^2$$

MSE (Mean Square Error)

- Both the mean squared error and the mean absolute error tell you how close a regression line is to a set of points.

- MSE is just like the MAE but squares the difference before summing them all instead of using the absolute value, therefore, it assigns more weight to the bigger errors.

$$MSE = \overset{\text{Divide by the total number of data points}}{\boxed{\frac{1}{n}}} \Sigma \Big(\underbrace{\overset{\text{Actual output value}}{\boxed{y}} - \overset{\text{Predicted output value}}{\boxed{\hat{y}}}}_{\text{The square of the difference between actual and predicted}} \Big)^2$$

RMSE (Root Mean Square Error)

- It shares MSE's property of heavily penalizing larger errors (because it's based on the squared differences), but it's in the same units as the outcome variable, which can make it easier to interpret.

Divide by the total number of data points

Actual output value

Predicted output value

$$RMSE = \sqrt{\boxed{\frac{1}{n}} \Sigma \underbrace{\left(\boxed{y} - \boxed{\hat{y}}\right)^2}_{\text{The square of the difference between actual and predicted}}}$$

RMSE (Root Mean Square Error)

- It shares MSE's property of heavily penalizing larger errors (because it's based on the squared differences), but it's in the same units as the outcome variable, which can make it easier to interpret.

$$RMSE = \sqrt{\overset{\text{Divide by the total number of data points}}{\boxed{\frac{1}{n}}} \sum \underbrace{\left(\overset{\text{Actual output value}}{\boxed{y}} - \overset{\text{Predicted output value}}{\boxed{\hat{y}}}\right)^2}_{\text{The square of the difference between actual and predicted}}}$$

R-Squared

- R-squared, also known as the coefficient of determination, measures the proportion of the variance in the dependent variable that can be predicted from the independent variable(s).

- In other words, it shows how close the data are to the fitted regression line. An R-squared of 1 indicates that all changes in the dependent variable are completely explained by changes in the independent variable(s).

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

SSRES: the residual sum of squared errors

SSTOT: the total sum of squared errors

R-Squared

- R-squared, also known as the coefficient of determination, measures the proportion of the variance in the dependent variable that can be predicted from the independent variable(s).

- In other words, it shows how close the data are to the fitted regression line. An R-squared of 1 indicates that all changes in the dependent variable are completely explained by chang the independent variable(s).

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Adjusted R-Squared

- There's a potential problem with R^2: it never decreases as more variables are added to the model, even if those variables are only weakly associated with the response. This is where Adjusted R-squared comes in.

- Adjusted R-Squared also shows how well the regression line approximates the real data points but it also takes into account the number of predictors in the model.

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

N = the number of observations

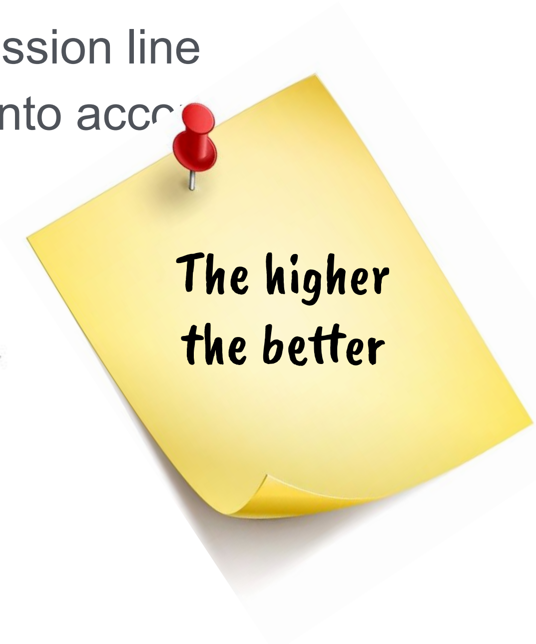
p = the number of predictors

Adjusted R-Squared

- There's a potential problem with R^2: it never decreases as more variables are added to the model, even if those variables are only weakly associated with the response. This is where Adjusted R-squared comes in.

- Adjusted R-Squared also shows how well the regression line approximates the real data points but it also takes into acc
the number of predictors in the model.

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N-1)}{N-p-1}$$



Which metric should I get started with?

- Among the many available metrics, these two are good starting points because they provide insights into different aspects:
    - **RMSE** (Root Mean Squared Error) measures the magnitude of the error in real units. (We want this to be low.)
    - **R^2** (R-squared) tells us how well the model explains the variability in the data. (We want this to be high, close to 1.)

- Example:
    - If the RMSE is 500 in a model that predicts house prices in thousands of dollars, this means the model has an average prediction error of $500,000. If the R^2 is 0.85, the model explains 85% of the variation in house prices.

Summary: performance metrics for regression models

- Cheat Sheet

Train/val/test data sets

- It is important to split the data into train/val/test **BEFORE** doing any feature engineering or modeling to prevent **data leakage** and ensure that the model is truly evaluated on unseen data

Data leakage: When information from val or test is used to inform the model during training or feature engineering

No peeking!!!
The test data set is used **ONLY** after you think you have the best model. It's the only true measure of generality.

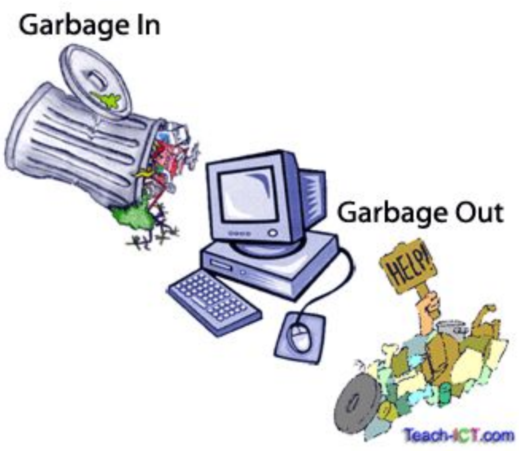
What We See

What Computers See

Feature Engineering

- Feature engineering is the process of selecting and transforming raw data into features that can be used to train machine learning models

- AKA creating/changing number of columns



Feature Engineering
Handling categorical attributes

Most machine learning models prefer to work with numbers

- **Ordinal encoder**
  Disadvantage: Assumes two nearby values are more similar to each other (useful for ordered categories such as "bad". "average", "good", "excellent")

- **One-Hot encoder**
  One binary category per attribute

Ordinal encoder

Example: X = [“Undergraduate”, “Masters”, “PhD”]

X
Undergraduate
Masters
PhD

X
1
2
3

One-Hot Encoding

Example: X = [“Red”, “Blue”, “Green”]

X
Red
Blue
Green

One-Hot Encoding

Example: X = [“Red”, “Blue”, “Green”]

X
Red
Blue
Green

X_red	X_blue	X_green
1	0	0
0	1	0
0	0	1

Feature Engineering
Feature Scaling

With few exceptions, machine learning algorithms don't perform well when the input of the numerical values have very different scales

Example in this dataset: total_rooms, median_income

- **MinMax**
    - AKA normalization
    - Values are shifted and rescaled so that they end up ranging from 0 to 1
    - How to calculate it? Subtracting the min val and diving by the difference between the min and the max

- **Standarization**
    - How to calculate it? Subtracts the mean value, then it divides the result by the standard deviation
    - It doesn't restrict the values to a specific range

MinMax

Example: X = [2, 10, 15]

$$= \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

2:

$$\frac{2-2}{15-2} = \frac{0}{13} = 0$$

10:

$$\frac{10-2}{15-2} = \frac{8}{13} \approx 0.615$$

15:

$$\frac{15-2}{15-2} = \frac{13}{13} = 1$$

X
2
10
15

X
0
0.615
1

Standardization

- **Media (μ):**

$$\mu = \frac{2+10+15}{3} = \frac{27}{3} = 9$$

- **Desviación estándar (σ):**

$$\sigma = \sqrt{\frac{(2-9)^2+(10-9)^2+(15-9)^2}{3}} = \sqrt{\frac{49+1+36}{3}} = \sqrt{\frac{86}{3}} \approx 5.387$$

Example: X = [2, 10, 15]

$$= \frac{X-\mu}{\sigma}$$

2:

$$\frac{2-9}{5.387} = \frac{-7}{5.387} \approx -1.299$$

10:

$$\frac{10-9}{5.387} = \frac{1}{5.387} \approx 0.186$$

15:

$$\frac{15-9}{5.387} = \frac{6}{5.387} \approx 1.114$$

X
2
10
15

X
-1.299
0.186
1.114

Feature Engineering

Included but not limited to:

- Handling categorical features: Ordinal encoder, OneHot encoder, etc.
- Feature scaling: MinMax, standarizaton, square root, log, etc.
- Feature extraction: Creating new features from existing ones
- Text processing (TF-IDF, Word2Vec, BERT, etc.)
- Extracting things that make sense (date→holiday)
- Dimensionality reduction (PCA, t-SNE, etc.)

Depends on your dataset!

Feature Engineering

Fit vs Fit_Transform vs Transform ⚠️

Feature Engineering

Fit vs Fit_Transform vs Transform ⚠️

FIT

- What it does: The fit method computes the necessary parameters needed to apply a transformation or learn from the data. However, it doesn't actually modify the data.

- When to use it: Use fit when you want to compute the parameters based on your data (like mean, standard deviation for standard scaling, or the min and max values for MinMax scaling), but you don't want to transform the data yet.

- Here, the scaler learns the mean and standard deviation from X_train, but X_train remains unchanged.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
```

Feature Engineering

Fit vs Fit_Transform vs Transform ⚠️

TRANSFORM

- What it does: The transform method applies a previously computed transformation to the data. It doesn't compute any new parameters; it simply uses the parameters computed during the fit phase.

- When to use it: Use transform when you want to apply a transformation to new or existing data using parameters learned during the fit phase.

- Here, both X_train and X_test are scaled using the mean and standard deviation learned from X_train during the fit phase.

```
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Feature Engineering

Fit vs Fit_Transform vs Transform ⚠️

FIT_TRANSFORM

- What it does: fit_transform is essentially a combination of fit and transform. It computes the necessary parameters from the data and then immediately applies the transformation.

- When to use it: Use fit_transform when you're sure you want to both compute the parameters and transform your data in one step. This can be more efficient than calling fit and transform separately.

- This achieves the same result as the two-step process above but in a single step.

```
X_train_scaled = scaler.fit_transform(X_train)
```

Feature Engineering

Fit vs Fit_Transform vs Transform ⚠️

IMPORTANT:

When splitting data into training and test sets, **always fit (or fit_transform) on the training data only**. This simulates a real-world scenario where your model has to generalize to unseen data. By fitting only on the training data, you avoid leaking information from the test set into the training process.

For instance, when scaling:

1. Use fit_transform (or fit and transform) on the training data.
2. Use transform on the test data (do not re-fit the scaler on test data).

This ensures that the test data is transformed based on the parameters learned from the training data, preserving the integrity of the evaluation process.

Example

X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

- Let's randomly split this dataset into train/test

```
X_train = [1, 2, 3, 6, 7, 8]
X_test = [4, 5, 9, 10]
```

Example

X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

- Let's randomly split this dataset into train/test

```
X_train = [1, 2, 3, 6, 7, 8]
X_test = [4, 5, 9, 10]
```

1. First step: Let's fit the scaler on the train set:
   `MinMax.fit(X_train)`

   Minimum value = 1
   Maximum = 8

Example

X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

- Let's randomly split this dataset into train/test

```
X_train = [1, 2, 3, 6, 7, 8]
X_test = [4, 5, 9, 10]
```

1. First step: Let's fit the scaler on the train set:
   `MinMax.fit(X_train)`

   Minimum value = 1
   Maximum = 8

2. Second step: Let's transform our train data

`X_train = 1`:

$$\frac{1-1}{8-1} = \frac{0}{7} = 0$$

`X_train = 2`:

$$\frac{2-1}{8-1} = \frac{1}{7} \approx 0.143$$

`X_train = 3`:

$$\frac{3-1}{8-1} = \frac{2}{7} \approx 0.286$$

`X_train = 6`:

$$\frac{6-1}{8-1} = \frac{5}{7} \approx 0.714$$

`X_train = 7`:

$$\frac{7-1}{8-1} = \frac{6}{7} \approx 0.857$$

`X_train = 8`:

$$\frac{8-1}{8-1} = \frac{7}{7} = 1$$

Example

X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

- Let's randomly split this dataset into train/test

```
X_train = [1, 2, 3, 6, 7, 8]
X_test = [4, 5, 9, 10]
```

1. First step: Let's fit the scaler on the train set: `MinMax.fit(X_train)`

   Minimum value = 1
   Maximum = 8

2. Second step: Let's transform our train data

3. Third step: Let's transform our test data

`X_train = 1`:

$$\frac{1-1}{8-1} = \frac{0}{7} = 0$$

`X_train = 2`:

$$\frac{2-1}{8-1} = \frac{1}{7} \approx 0.143$$

`X_train = 3`:

$$\frac{3-1}{8-1} = \frac{2}{7} \approx 0.286$$

`X_train = 6`:

$$\frac{6-1}{8-1} = \frac{5}{7} \approx 0.714$$

`X_train = 7`:

$$\frac{7-1}{8-1} = \frac{6}{7} \approx 0.857$$

`X_train = 8`:

$$\frac{8-1}{8-1} = \frac{7}{7} = 1$$

`X_test = 4`:

$$\frac{4-1}{8-1} = \frac{3}{7} \approx 0.429$$

`X_test = 5`:

$$\frac{5-1}{8-1} = \frac{4}{7} \approx 0.571$$

`X_test = 9`:

$$\frac{9-1}{8-1} = \frac{8}{7} \approx 1.143$$

`X_test = 10`:

$$\frac{10-1}{8-1} = \frac{9}{7} \approx 1.286$$

Berkeley Engineering | BerkeleyHaas

AGENDA

- ✅ Required activities for Module 8
- ✅ Content review Module 8: Feature Engineering and Overfitting
- Code
- Questions

Code

- https://drive.google.com/file/d/1ccaTiK4kli_WMikBP_Q20YPWgCUmkXwF/view?usp=sharing

Berkeley Engineering | BerkeleyHaas

AGENDA

- ✅ Required activities for Module 8
- ✅ Content review Module 8: Feature Engineering and Overfitting
- ✅ Code
- Questions

Berkeley Engineering | BerkeleyHaas

QUESTIONS?

Berkeley Engineering | BerkeleyHaas

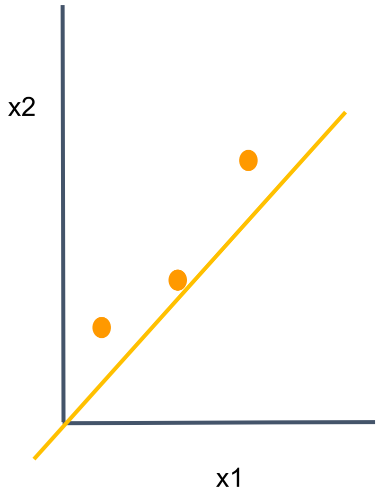
AGENDA

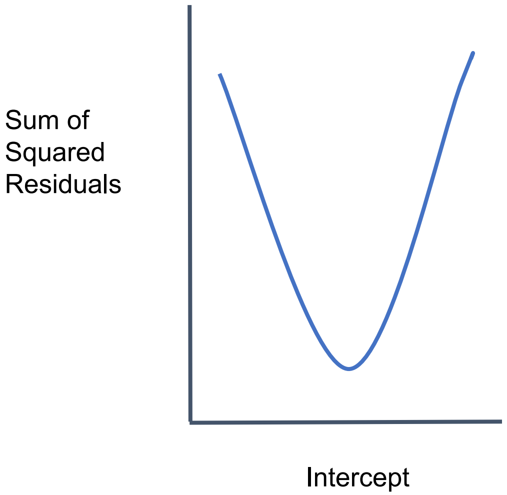
- ✅ Required activities for Module 8
- ✅ Content review Module 8: Feature Engineering and Overfitting
- ✅ Code
- ✅ Questions

APPENDIX

Objective of a Neural Network:

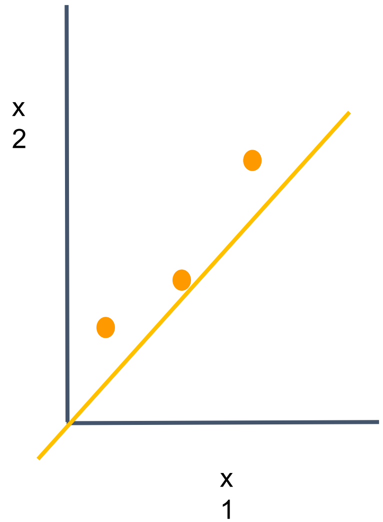
Find weights and biases that minimizes the loss function

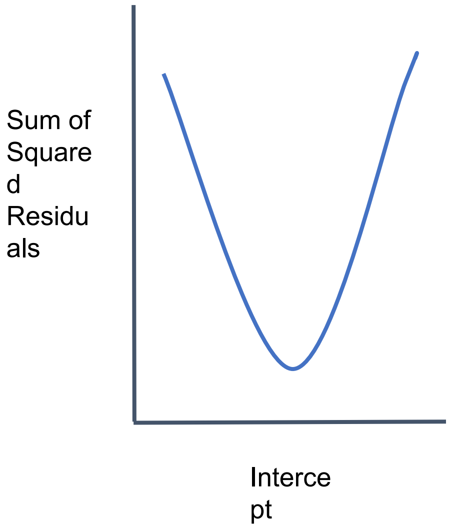




Objective of a Neural Network:

Find weights and biases that minimizes the loss function

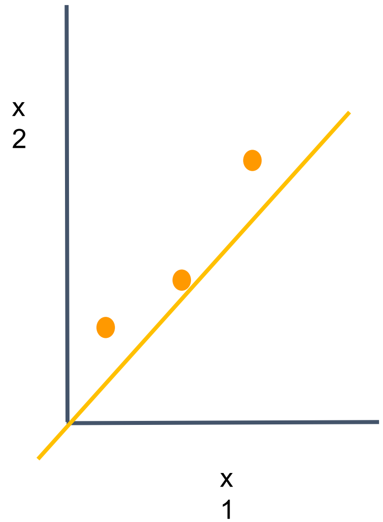


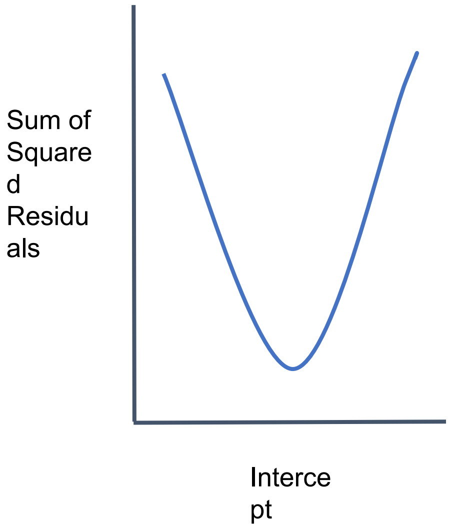


Derivative: Slope of a curve at a specific point

Objective of a Neural Network:

Find weights and biases that minimizes the loss function





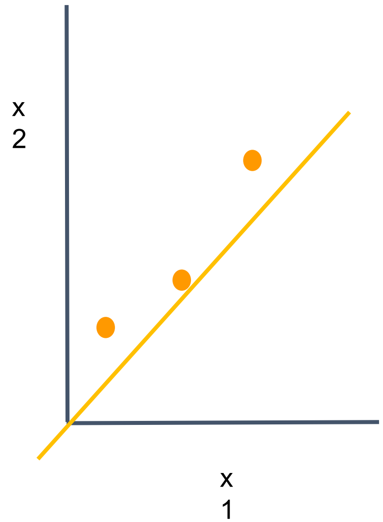
Derivative: Slope of a curve at a specific point

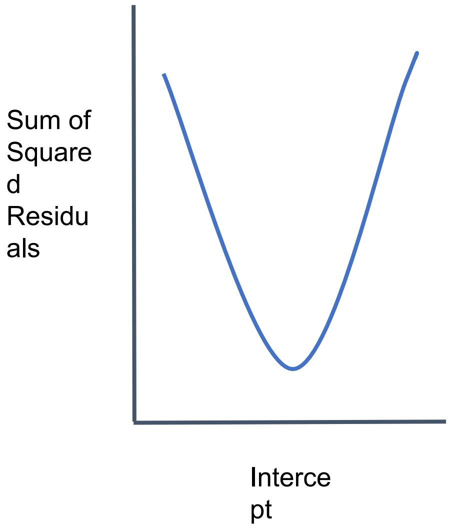
Gradient: Generalization of the derivative. It extends this idea to function of multiple variables.

A gradient is like an arrow that points in the direction of the steepest increase on a surface.

Objective of a Neural Network:

Find weights and biases that minimizes the loss function





If you imagine being on a hill and you want to know which way is uphill, the gradient would point in that direction. If you want to go downhill (like in gradient descent), you'd go in the opposite direction of the gradient.

Derivative: Slope of a curve at a specific point

Gradient: Generalization of the derivative. It extends this idea to function of multiple variables.

Gradient descent

Objective of a Neural Network:

Find weights and biases that minimizes the loss function

- The **gradient** of the loss gives you the direction of the steepest ascent
- To minimize, you move the opposite way (towards the steepest descent)