

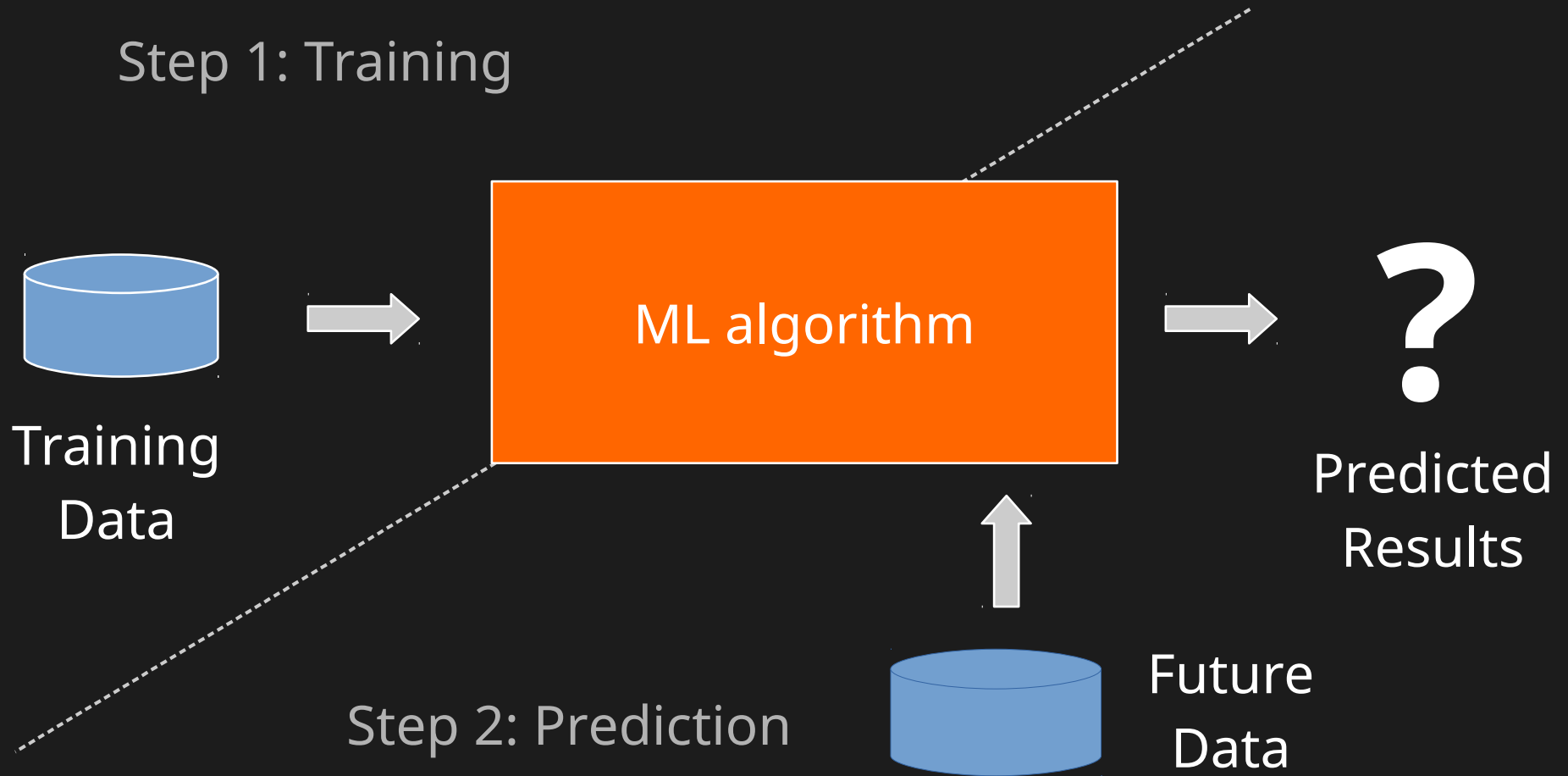
Supervised Learning

Concepts

Reminder: Supervised Learning

- Let the computer learn a task by providing (a lot of) examples
- Main applications:
 - Regression: predict a continuous variable
 - Classification: predict class affiliation

How does it work?



Stationary assumption:
training and future data are independent and identically distributed (i.i.d.)

Data: Nomenclature

Features/Attributes (input variables)

Targets/Labels
(output variables)

Examples/Instances

Weight	Height	Exterior	Wings?	Lika- bility	N _{legs}
0.1	0.1	feath.	true	1	2
3.5	0.3	fur	false	1	4
12.0	0.7	fur	false	1	4
500	1.8	skin	false	2	4
800	3.0	fur	true	3	4
2.5	0.5	fur	false	1	4
...

Data
Type:

continuous

continuous

categorical

binary

ordinal

integer

Pet?	Type
true	bird
true	cat
true	dog
false	rhinoceros
false	chimera
true	cat
...	...

binary

categorical
(multi-class)

classes of label "Type"

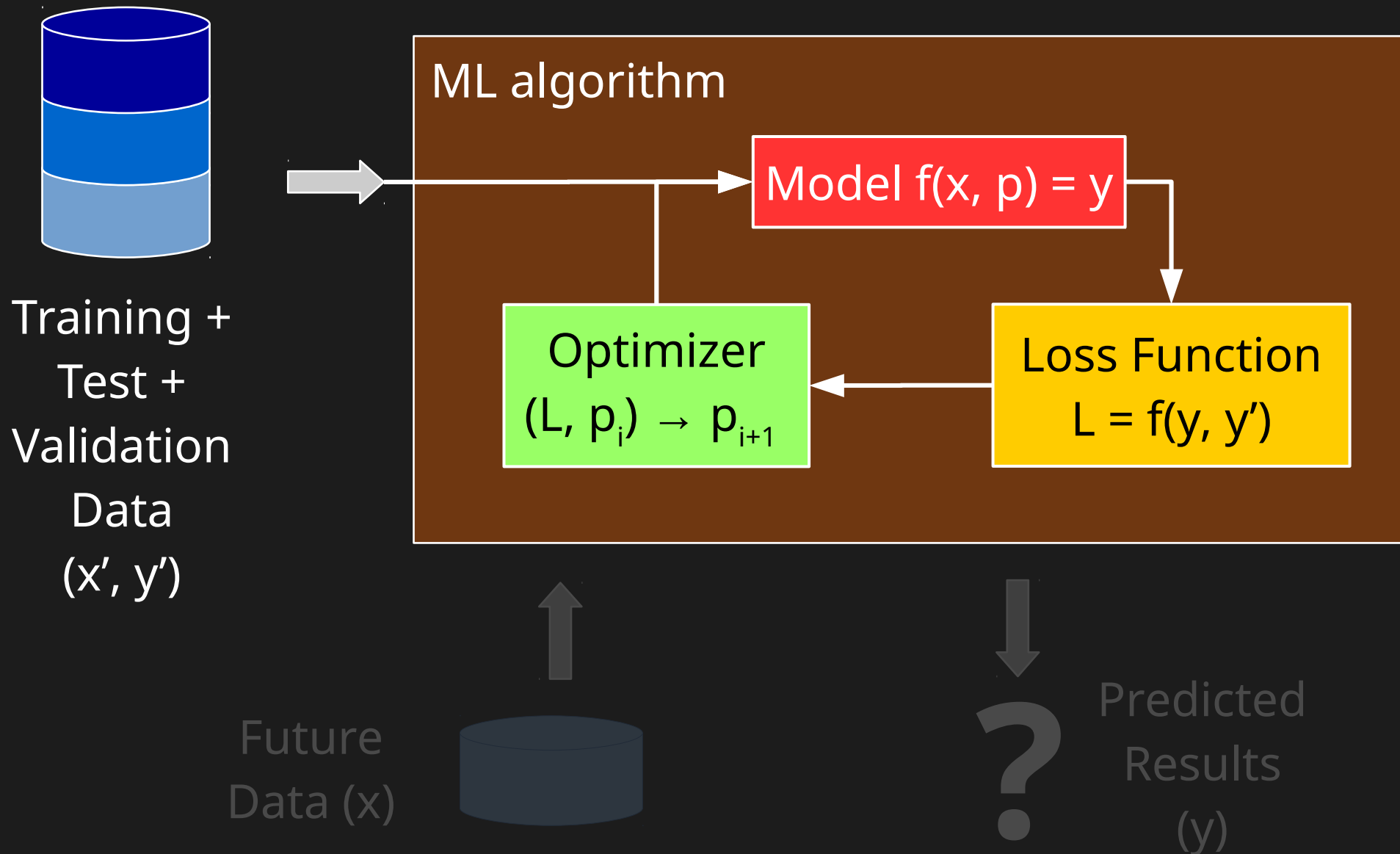
Data Preparation

- Keep in mind that not all data can be handled easily by machines:
 - **Continuous data** (floats, integers): fine!
 - **Categorical data**: nope.
→ one-hot encoding
 - **Binary data**: fine, but has to be numerical
 - **Ordinal data**: fine, but has to be numerical
- One-hot encoding can exponentially raise the number of features in your data set

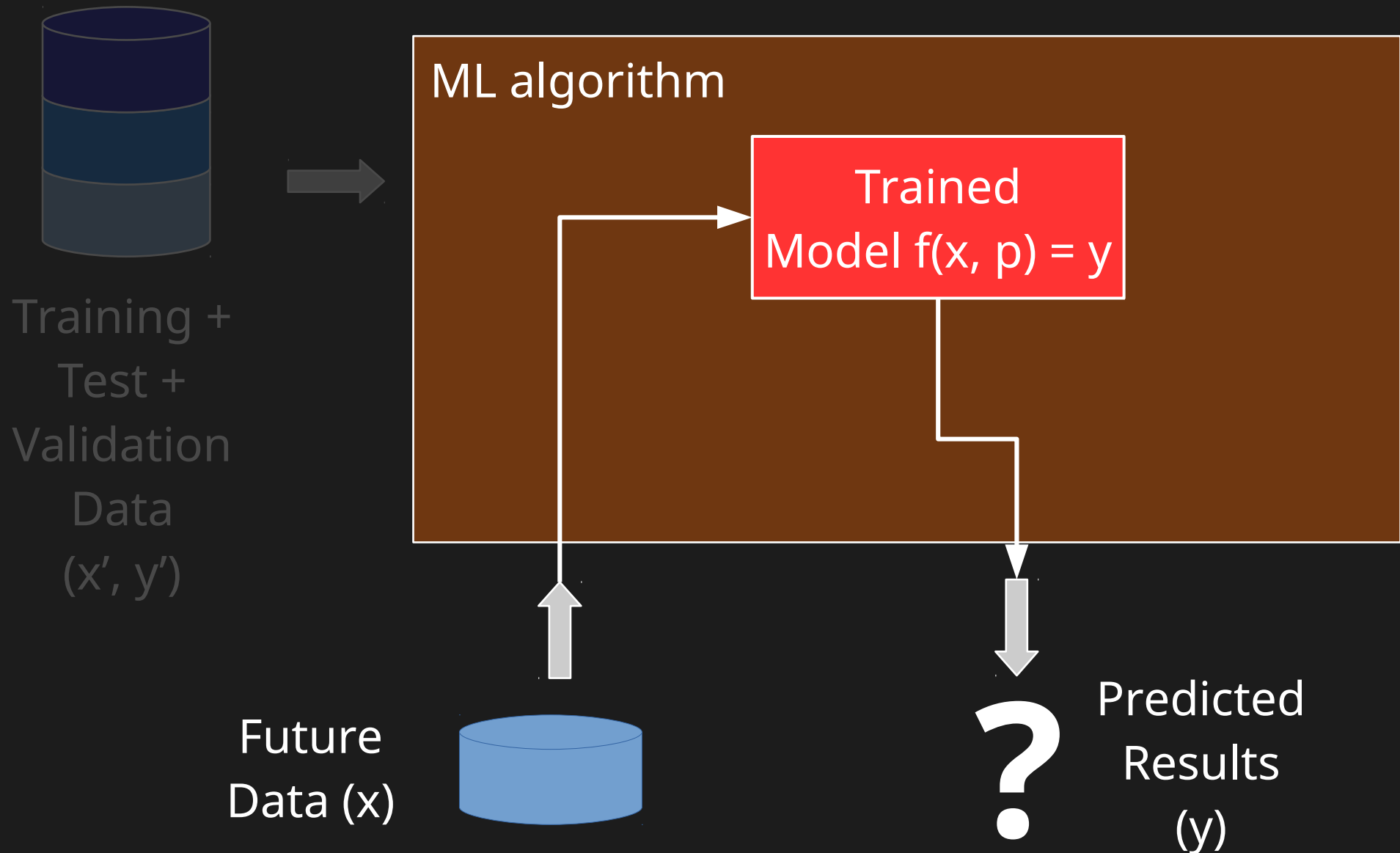
Data Samples

- **Training data:** features + labels
used as examples for training the ML model
- **Test data:** features + labels
used for evaluating the accuracy of the trained model; must be different from training data
- **Validation data:** features + labels
used for validating the model settings
- **Future data:** features
used to evaluate model and predict labels
- All data must be i.i.d.!

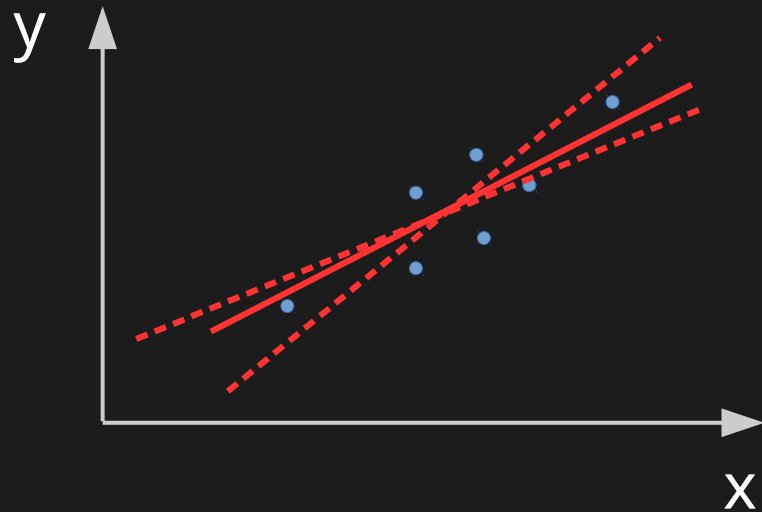
How does Supervised Learning work? Learning Stage:



How does Supervised Learning work? Prediction Stage:



Linear Regression as ML Example



Training Data $\{(x_1', y_1'), (x_2', y_2'), \dots\}$

Model $f(x_i, \vec{p}) = p_0 + p_1 \cdot x_i$

Parameters/
Weights $\vec{p} = (p_0 \ p_1)^T$

How do we fit the model to the data?

Loss Function

$$L(\vec{x}, \vec{y}') = \sum_i [f(x_i) - y_i']^2$$

Optimizer:

$$\frac{\partial L}{\partial \vec{p}} = \vec{0} \quad \rightarrow \quad \vec{p}$$

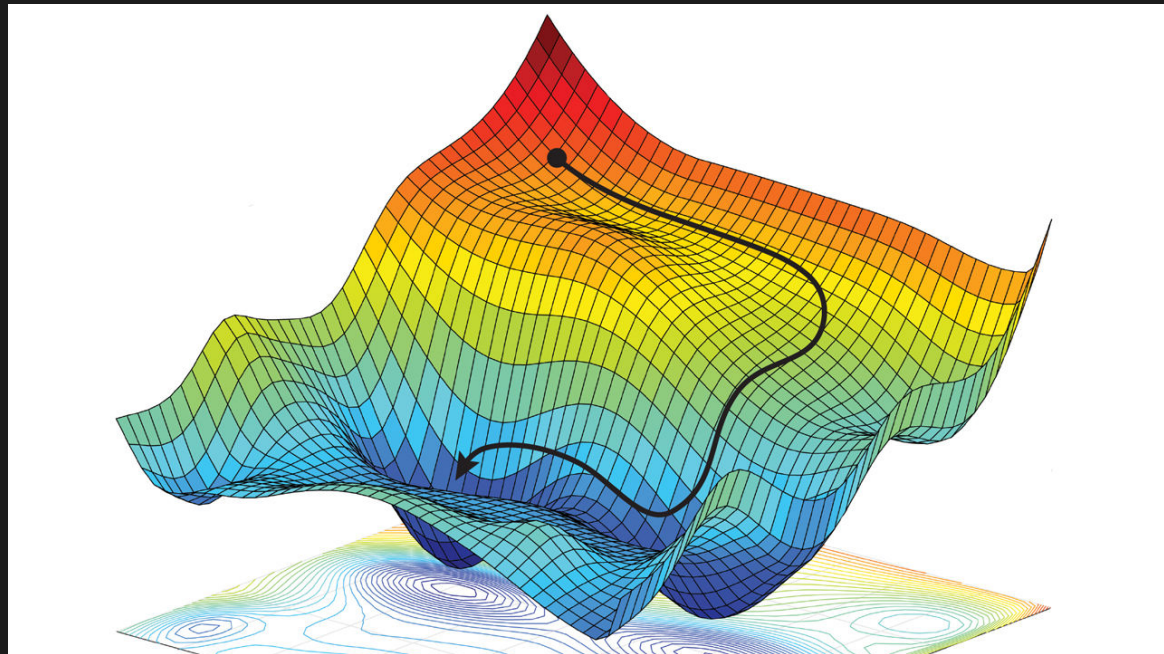
Loss Functions and Metrics

- Loss/Objective functions calculate loss
- Loss is used to improve the model parameters
- Metrics benchmark the model's success (**score**)
examples:
 - Accuracy (classification): $N_{\text{correct}}/N_{\text{total}}$
 - Root-Mean-Square-Error (regression)
- Regression metrics may be used as loss functions, but generally they are two different things

Optimizers

- In the case of linear regression, you can immediately solve for the best-fit parameters
- More complex (non-linear) models and loss functions require an iterative approach
- All-round optimizer: gradient descent

$$\vec{p}_{i+1} = \vec{p}_i - \alpha \nabla_p L(y, y')$$



Classification: Balance and Errors

- Accuracy is a simple metric, but can be misleading:

“1 in 1000 emails is spam.”

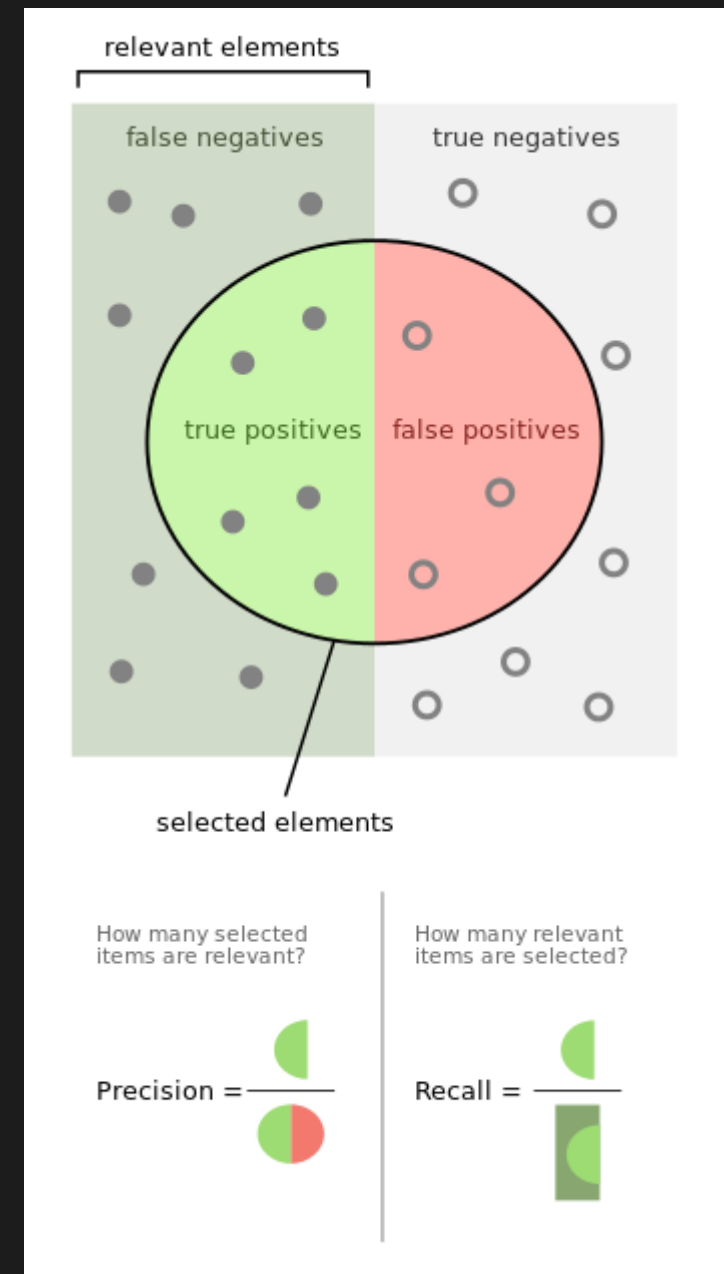
By assuming no email is spam you get a 99.9% accuracy. But that's not the idea...

- Accuracy is a bad metric if the data set (i.e., their classes) is unbalanced
- Have to account for application and different errors...

Classification: Balance and Errors

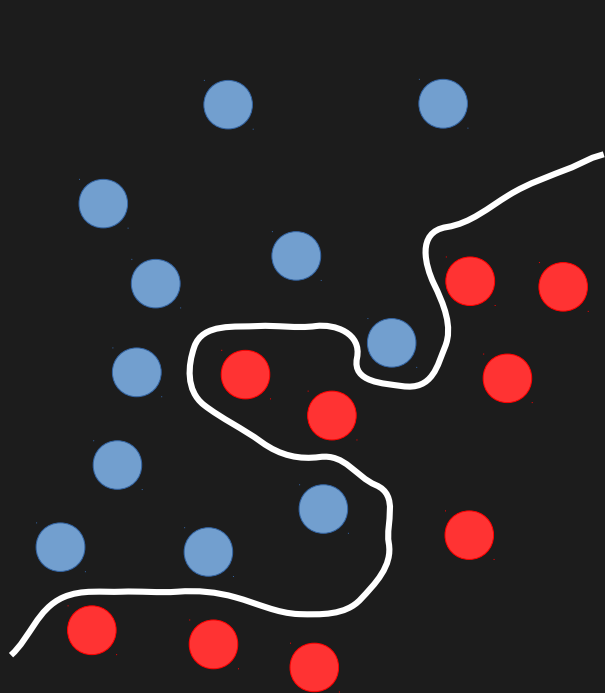
- Precision and Recall can be better metrics - depending on application:
 - spam filter: max precision
 - cancer diagnosis: max recall
- Good compromise:

$$F1 = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

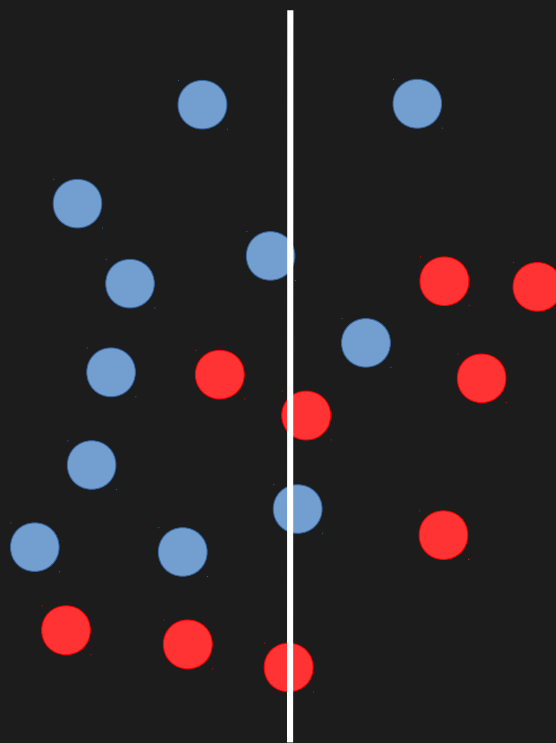


Generalization

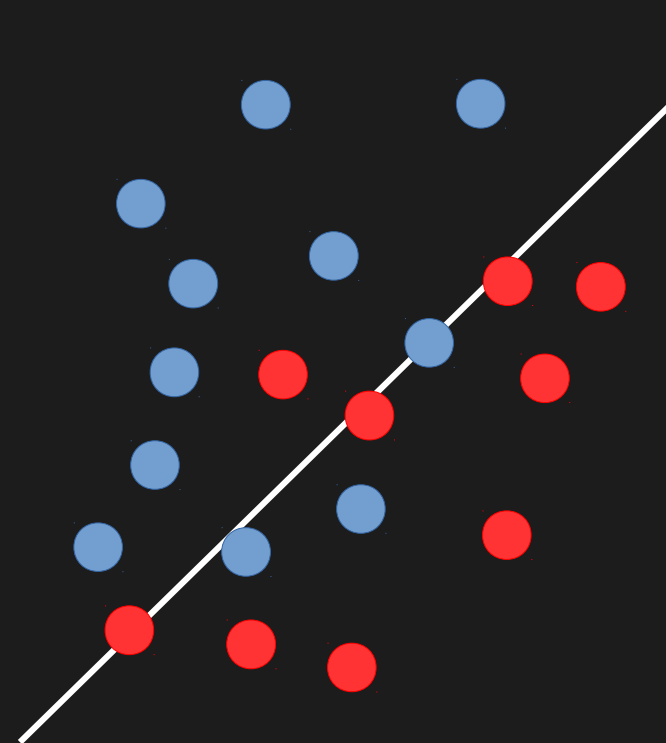
- What if you get a high training score, but a low test/validation sample score?



Low Generalization
Overfitting



High Generalization
Underfitting



Good Generalization
Good fit

Generalization and Regularization

- Overfitting: high training score, low test score
- Good fit: similar training and test scores
- Underfitting: low training and test scores
- Generalization can be enforced through regularization as part of the training process by imposing a penalty on the loss function (more on the details later)

Cross-Validation

- Data is typically expensive
- But we need training/test/validation data, all of which have to be exclusive samples

- We can recycle data!

1. Shuffle



available data

2. Split

training

test

validation

3. Train model and get scores

4. redo N times and pick best/mean score

Hyperparameters

- Hyperparameters are model parameters that are not learned; they are set by the user
- Hyperparameters need tuning by the user
- Expensive: run a full-blown training with different hyperparameter settings
- sklearn: `gridsearchcv` combines a gridsearch over hyperparameters with cross-validation

General Approach

- Prepare the data: Identify and format the features and targets, scale data
- Pick a suitable model: regression/classification, complexity
- Train the model (training + test data)
- Tune hyperparameters
- Check model performance (validation data)
- Predict future data

sklearn

- sklearn has implementations of most standard models available
- Training and hyperparameter tuning can be automated using `gridsearchcv` and pipelines
- More complex third-party models (e.g., `lightgbm`) often provide interfaces to the sklearn infrastructure and can be used like sklearn models
- More about this next time...