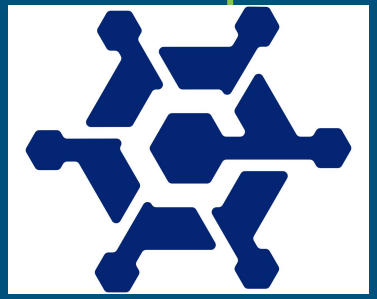


# CloudAEye Webinar series

## How to become a data-driven enterprise with AI

## Episode 2: Basics of ML Model Building

# Recap: Previous episode

Organization of the webinar series

- What to expect, topics covered, tools and contents provided

Data-driven Enterprise

- Data-driven organizations and the role of AI

Introduction to AI concepts

- AI/ML/DL/DS
- Introduction to basics of ML: supervised and unsupervised learning
- Supervised  and unsupervised learning basics: Regression, classification and Clustering etc.

Code examples

Summary and QA

# Agenda for this episode

- Deep dive into supervised ML models
  - Regression
  - Classification
  - Train and Test split
- ML Model basics
  - A simple regression example
  - Basic concepts: parameters, model fitting, bias and variance
  - Over and underfitting
  - Concepts of loss function and optimization
  - Regularization in ML models
- Code examples

# Supervised ML algorithms

Recap from last episode

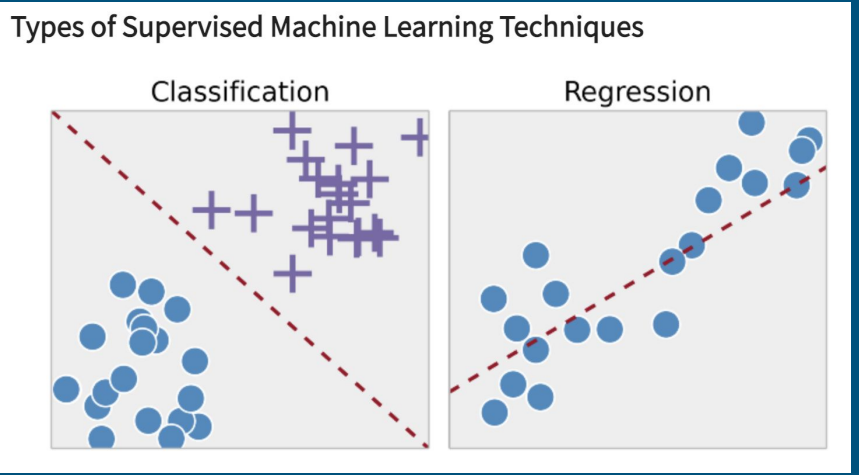
When the data is “labeled” or “tagged” for the correct outcome or value:

Input looks like (X, y)

Where X is the set of input features and y is the outcome variable that is provided

Based on y we have two classes of supervised learning:

1. Regression, where y is continuous ex: linear regression, tree-based regression, SVR
2. Classification, where y is discrete. ex: logistic regression, decision trees, SVM,  KNN, deep networks



# Supervised Learning: Model Building

We build model on

Available data (Training)

Our goal is to build a model that works well on unseen data: **GENERALIZATION**

To ensure a good generalization, we mimic the process

Train/Test split

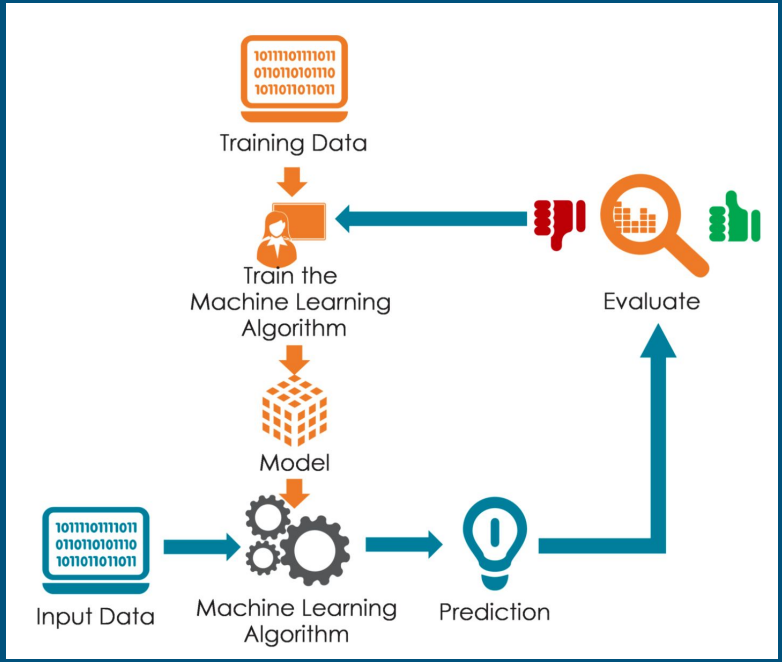
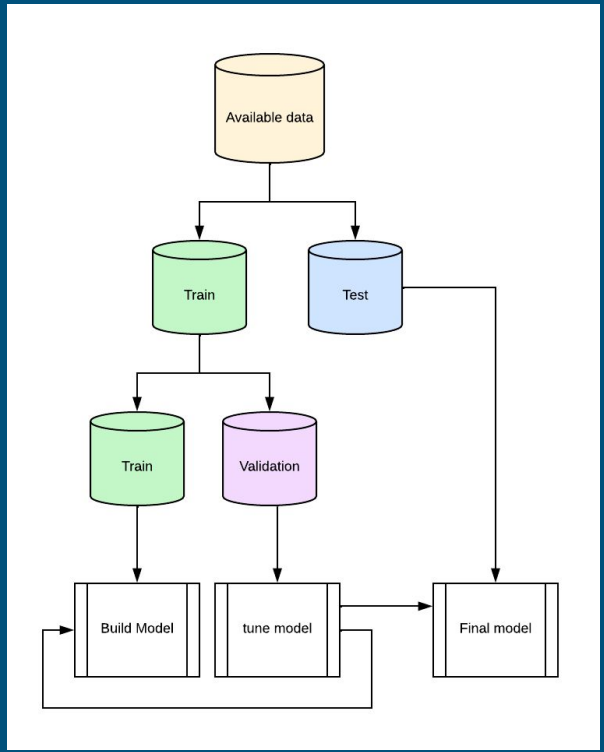



Image source: https://medium.com/@tekaround/train-validation-test-set-in-machine-learning-how-to-understand-6cdd98d4a764

# Model Building: Train-Test split

- Available data is divided into
  - Train (usually 80%)
  - Test (20%)
- Test data is kept aside (held out) and never seen by the algorithm during model building
- To build a better model we may need to tweak different settings (hyper parameters) so Train data is further divided into
  - Train (80%)
  - Validation (20%)



# Supervised ML: Basics of Model Building

- What is a model? How is it different from algorithm?

  An ML algorithm is the particular method, and the model is the result of applying that method

  $Y = f(X)$ here **f** is the model

  Example: a linear regression model, a decision tree model

- Parametric vs non-parametric models
  - A parametric model uses a set of parameters to represent the learning of the algorithms e.g., coefficients of regression or weights of neural networks etc
  - A nonpayment model doesn't explicitly use weights or parameters e.g., k-nearest neighbors

# Linear Regression

- Most basic form of supervised ML model

Given a n-dimensional vectors of numeric input variables $X_i$ (usually called “independent” or “explanatory” variables) and a numeric outcome variable $y_i$ (usually called“the dependent variable”):

• Find a n-dimensional vector β such that $\beta.x_i$ is a good estimate of $y_i$.

$$\hat{Y} = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

# Linear Regression: Least Square Solution

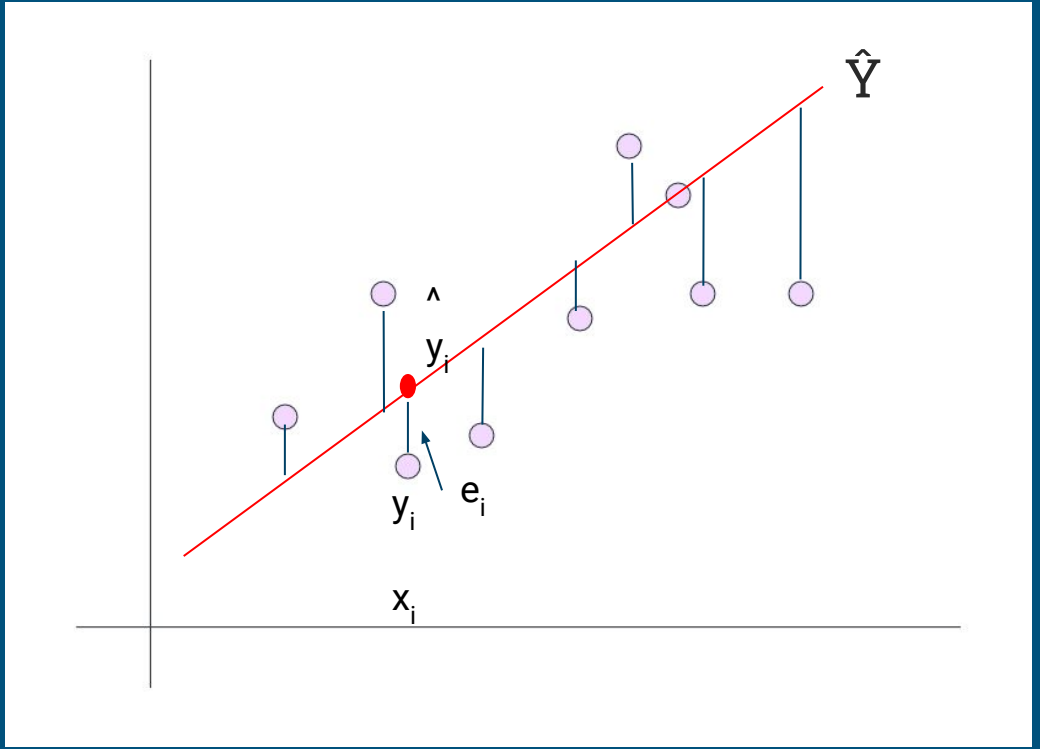
At each training point $x_i$, the actual output is $y_i$ but the model predicts $\hat{y}_i$
So at each point the error ei = ŷi - $y_i$

We define a Loss function, called the Sum of Squares Error as

$$L(Y, \beta) = \sum e_i^2 = \sum (\hat{y}_i - y_i)^2$$

Recall that $\hat{y}_i = \beta_0 + \beta_1 x_i$ So the loss function becomes

$$L(Y, \beta) = \sum (\beta_0 + \beta_1 x_i - y_i)^2$$



# Fitting the model

In our Notebook we see that we can use scikit-learn package for fitting a linear regression model by calling:

```
regr = linear_model.LinearRegression()
# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)
```

Model.fit is a very important concept in scikit-learn as all models follow the same interface

What exactly is a model? How is it fitted?

# Model Fitting: overfitting and underfitting

The most desired property of a model is generalization

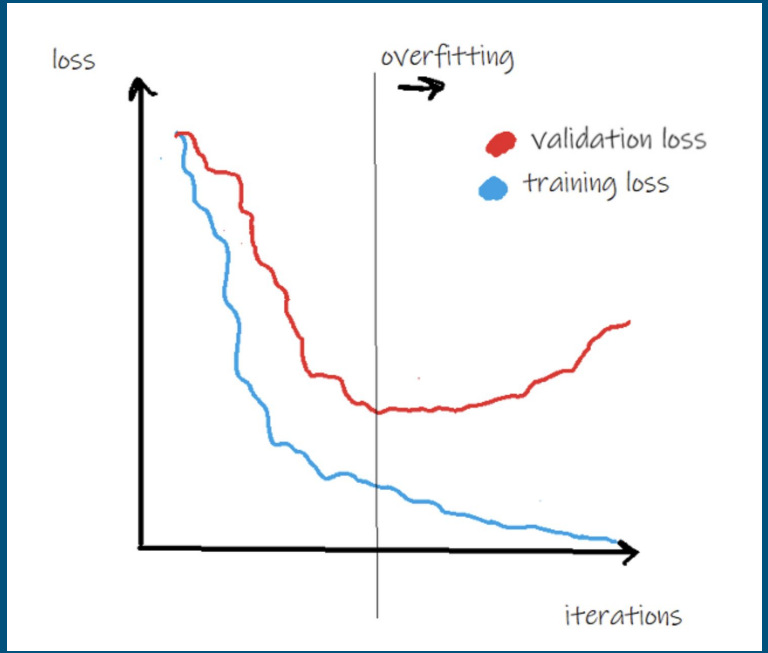
- So that the fitted model works fine with unseen data
- For that reason we use train/test split to check the model performance

  But most often models fail to achieve a good fit and exhibit lack of generalization in terms of underfitting and overfitting

- Two related terms are bias and variance of the model

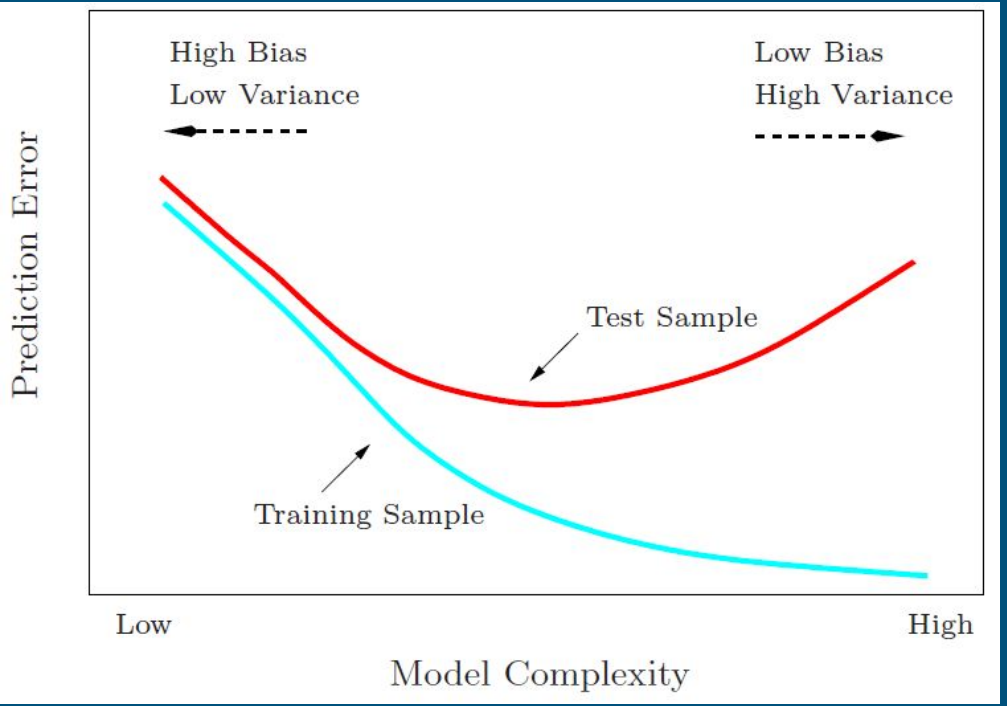
# Bias-variance, overfit and underfit

Let’s use the whiteboard to illustrate the concepts



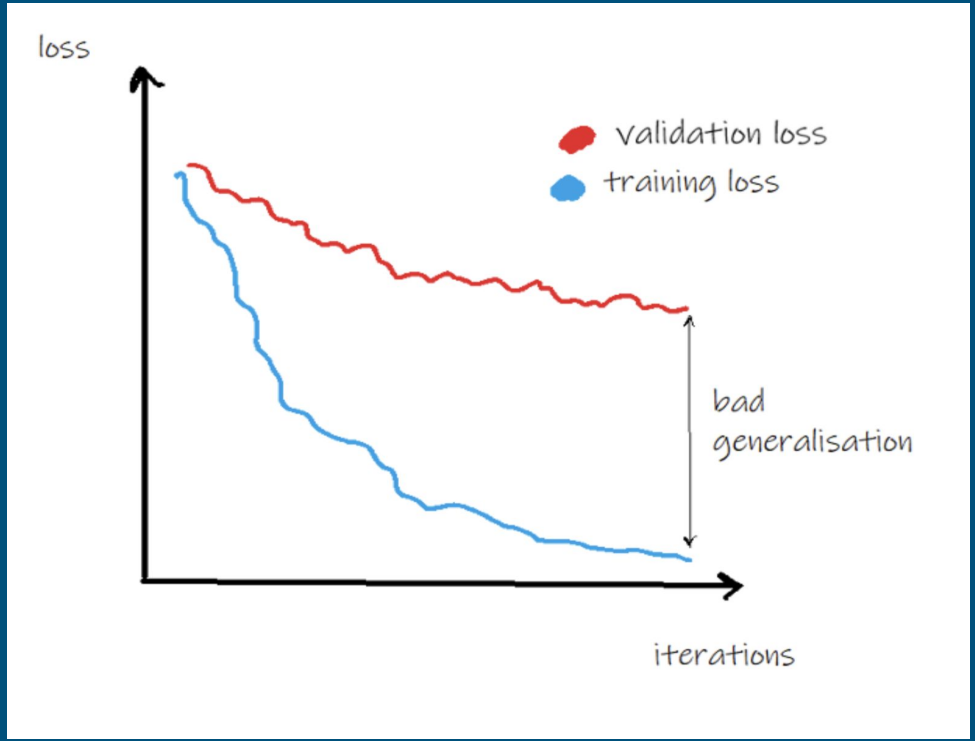
# Bias-variance, overfit and underfit

Let’s use the whiteboard to illustrate the concepts



# Bias-variance, overfit and underfit

Let’s use the whiteboard



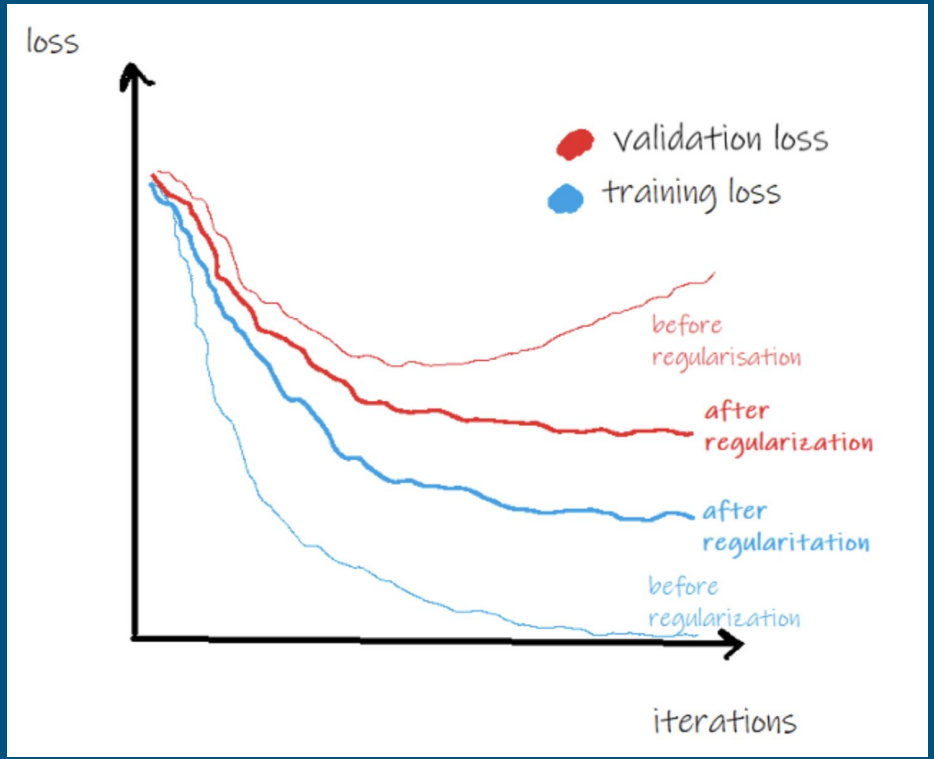
# Regularization as a Solution

Overfitting is one of the most serious problems of model fitting:

You think you have a good model but you don't

Solution:

Regularization, cross-validation



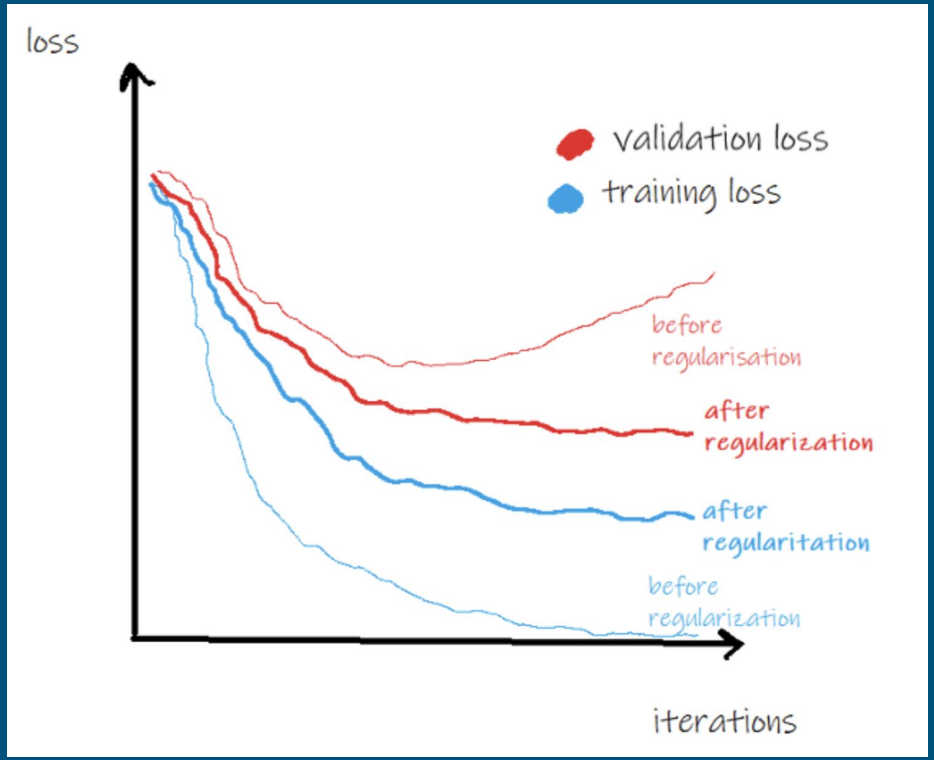

For more info:
https://towardsdatascience.com/generalization-regularization-overfitting-bias-and-variance-in-machine-learning-aa942886b870

# Regularization as a Solution

Regularization:

We have seen that having more complex model leads to overfitting, the goal of regularization is to reduce model complexity so that the model doesn't overfit

L2 and L1 regressions




For more info:
https://towardsdatascience.com/generalization-regularization-overfitting-bias-and-variance-in-machine-learning-aa942886b870

# Coding example

Link to the colab Notebook

https://colab.research.google.com/drive/1jYC-MGtBK32_l9HT8dEQDzcdqMh8iTOu#scrollTo=yhUNUJngfsNy

# Thank You

You have DevOps pain points?

We like to hear from you

Looking for exciting opportunities?

We are hiring

AI/ML engineers and applied scientists

Cloud Engineers

UI/UX designers