CloudAEye Webinar series

How to become a data-driven enterprise with AI

Episode 3a: Classification models

Recap: Previous episode

Episode 1

Data-driven Enterprise, Introduction to AI concepts, Code examples

Episode 2

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

Agenda for this episode

- Episode 3(a)
    - Introduction to classification
    - Logistic regression
        - Idea, model
- Episode 3(b)
    - A detailed worked out example of logistic regression
    - Code walkthrough and explanation
- Episode 3(c)
    - Practical aspects of logistic regression
    - How to understand a model
    - Evaluation of classification models

Supervised ML algorithms

Recap from last episode

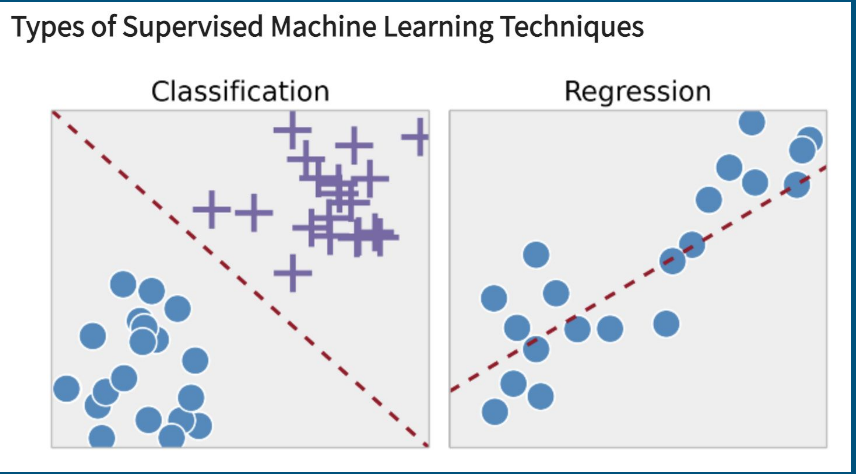
When the data is “labeled” or “tagged” for the correct outcome or value:

Input looks like (X, y)

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

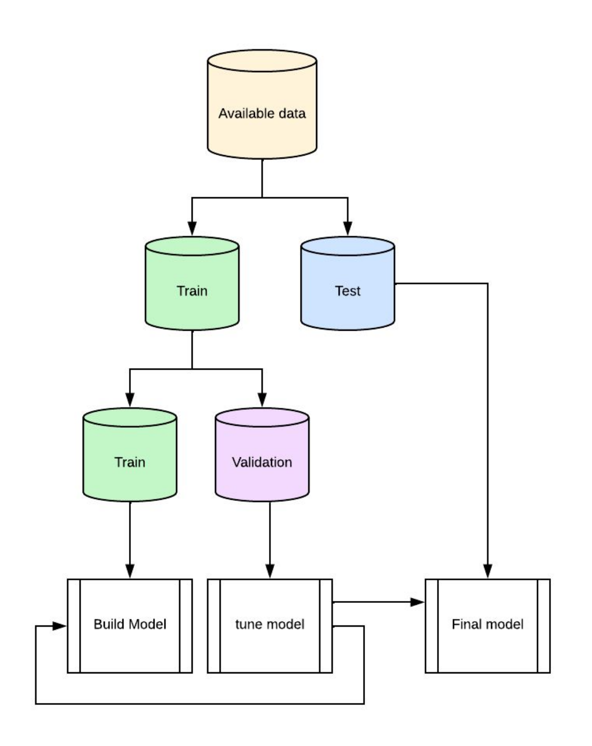
Based on y we have two classes of supervised learning:

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



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

  $\mathbf{Y} = f(X)$ 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.

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

Linear Regression: Least Square Solution

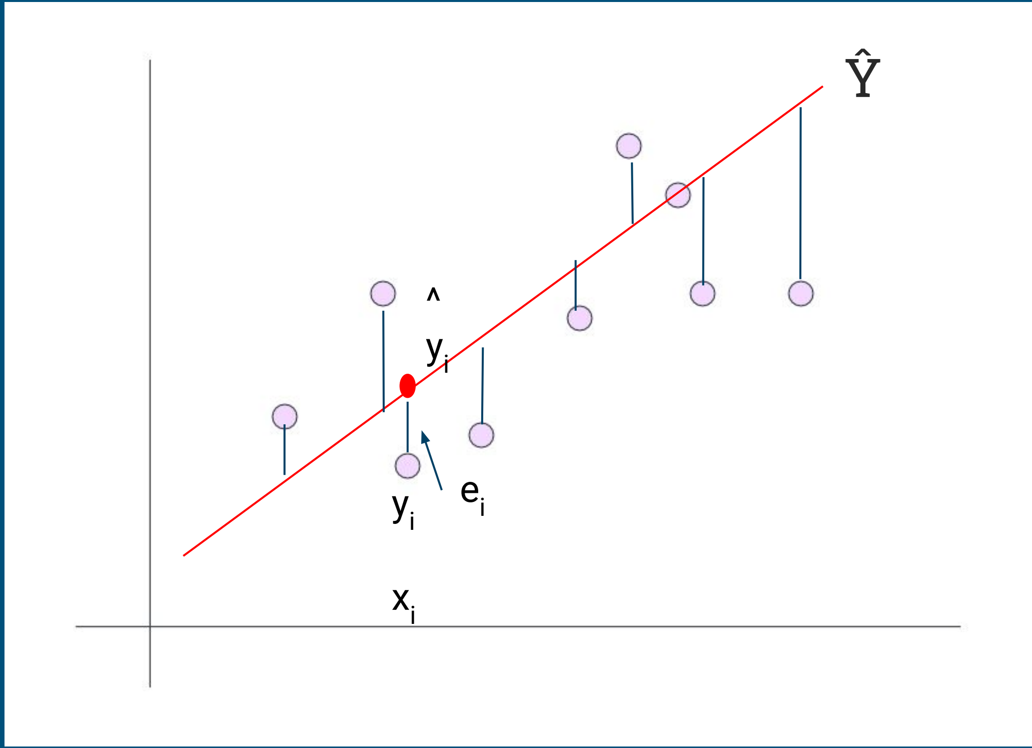
At each training point x_i, the actual output is y_i but the model predicts $\hat{y}_i$
So at each point the error ei = ŷi - y_i

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?

From Linear regression to Logistic Regression

- Classification has discrete valued targets
  - Binary – {YES, NO}, {0, 1}, {Normal, abnormal}
  - Multi-valued – images classified into {Cats, Dogs, Birds, Horses, ..}
- Linear regression models that we have seen before are not suitable for this type of applications
- The output values are not guaranteed to

  Be bounded between [0,1]

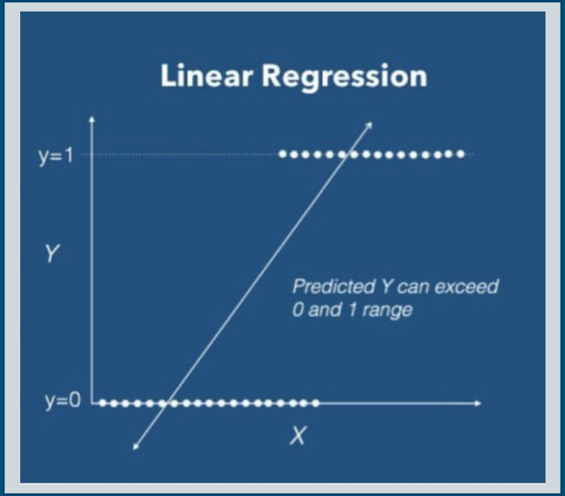


Image source:
https://medium.com/analytics-vidhya/a-comprehensive-guide-to-logistic-regression-e0cf04fe738c

From Linear regression to Logistic Regression

So we use a different type of model that takes the linear regression output and passes it through a squashing function

Called the “Sigmoid” or Logistic function

σ(z) = 1/(1 + e ^ (-z)) so, the model equation

becomes

$$y = \frac{1}{1+e^{-(w_0 + w_1 x)}}$$

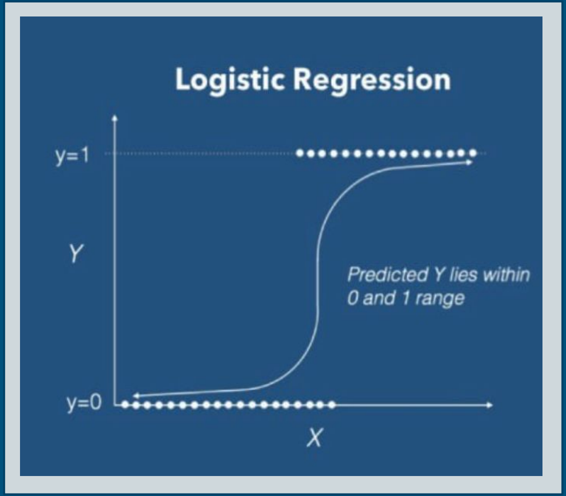


Image source:
https://medium.com/analytics-vidhya/a-comprehensive-guide-to-logistic-regression-e0cf04fe738c

Converting the logistic function into classifier

The logistic or sigmoid function $\sigma(z)$ provides a score between 0 and 1

We can use the score to define a **threshold** value that corresponds to a “decision boundary” to perform the classification task

For instance if **threshold** = 0.5 then

We can assign

$y = 0$ if $\sigma(.) <= 0.5$ and

$y = 1$ if $\sigma(.) > 0.5$

Thus a threshold converts the output of the logistic function $\sigma()$ to a classifier

White board

The derivation of the equation

Log odds ratio

Thank You

Do 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

CloudAEye

www.cloudaeye.com