INTRODUCTION TO DATA ANALYTICS

Linear Regression

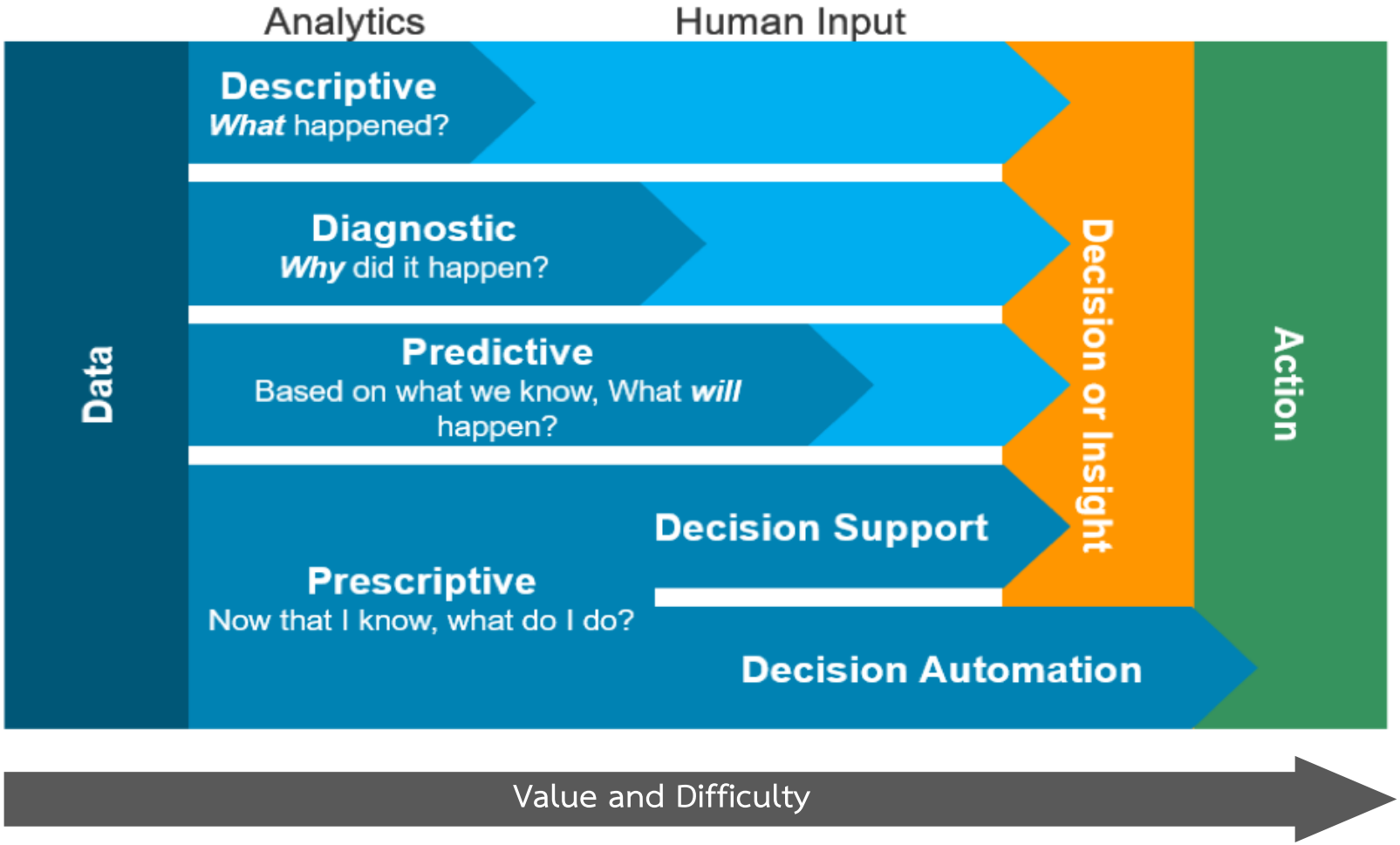
Dr. Rathachai Chawuthai

Department of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang

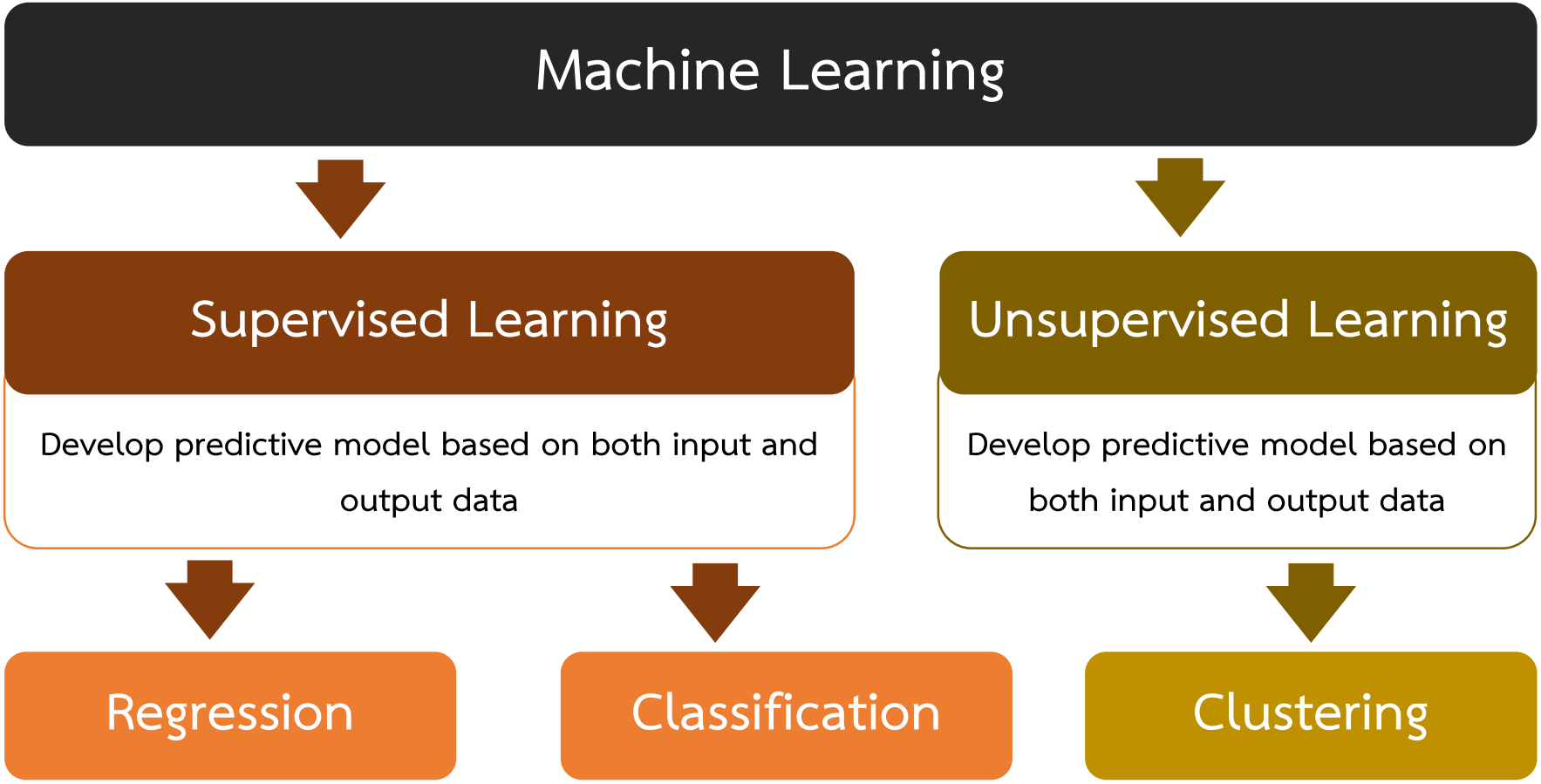
Agenda

- Linear Regression
- Evaluation Methods
- Feature Extraction
- Cross Validation
- Feature Scaling

Data Analytics



Ref:
- Four types of analytics capability (Gartner, 2014)
- (image) https://www.healthcatalyst.com/closed-loop-analytics-method-healthcare-data-insights


Machine Learning
Supervised Learning
Develop predictive model based on both input and output data
Unsupervised Learning
Develop predictive model based on both input and output data
Regression
Classification
Clustering
• Linear Regression
• Polynomial Regression
• Decision Tree
• Logistic Regression
• Neural Network
• etc.
• K-Means
• DB-SCAN
• etc.

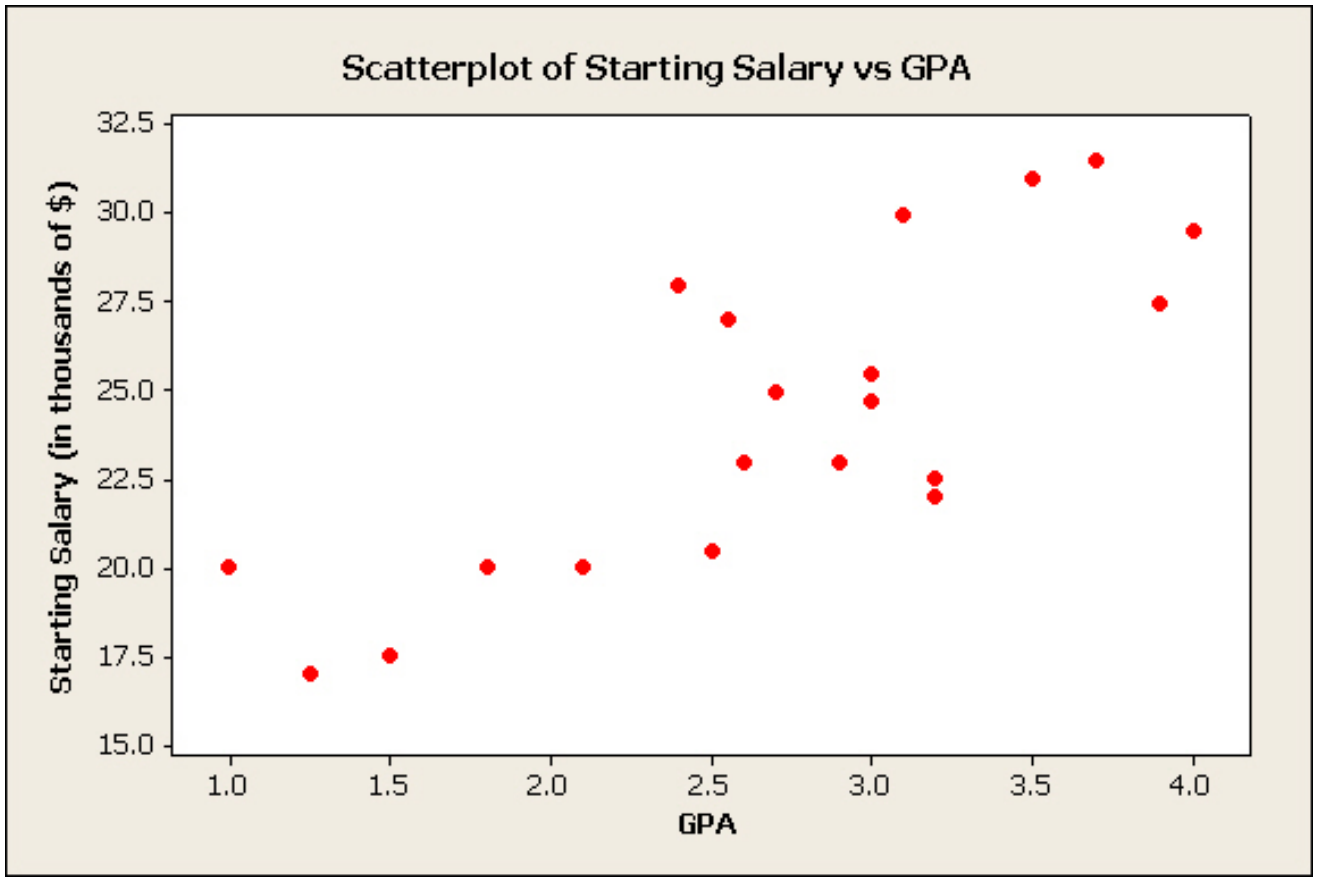
Linear Regression

Your Salary?

GPA	Salary
1.07	15,400
2.32	20,500
2.75	23,300
3.02	28,200
3.25	27,900
3.63	32,100
3.98	30,200

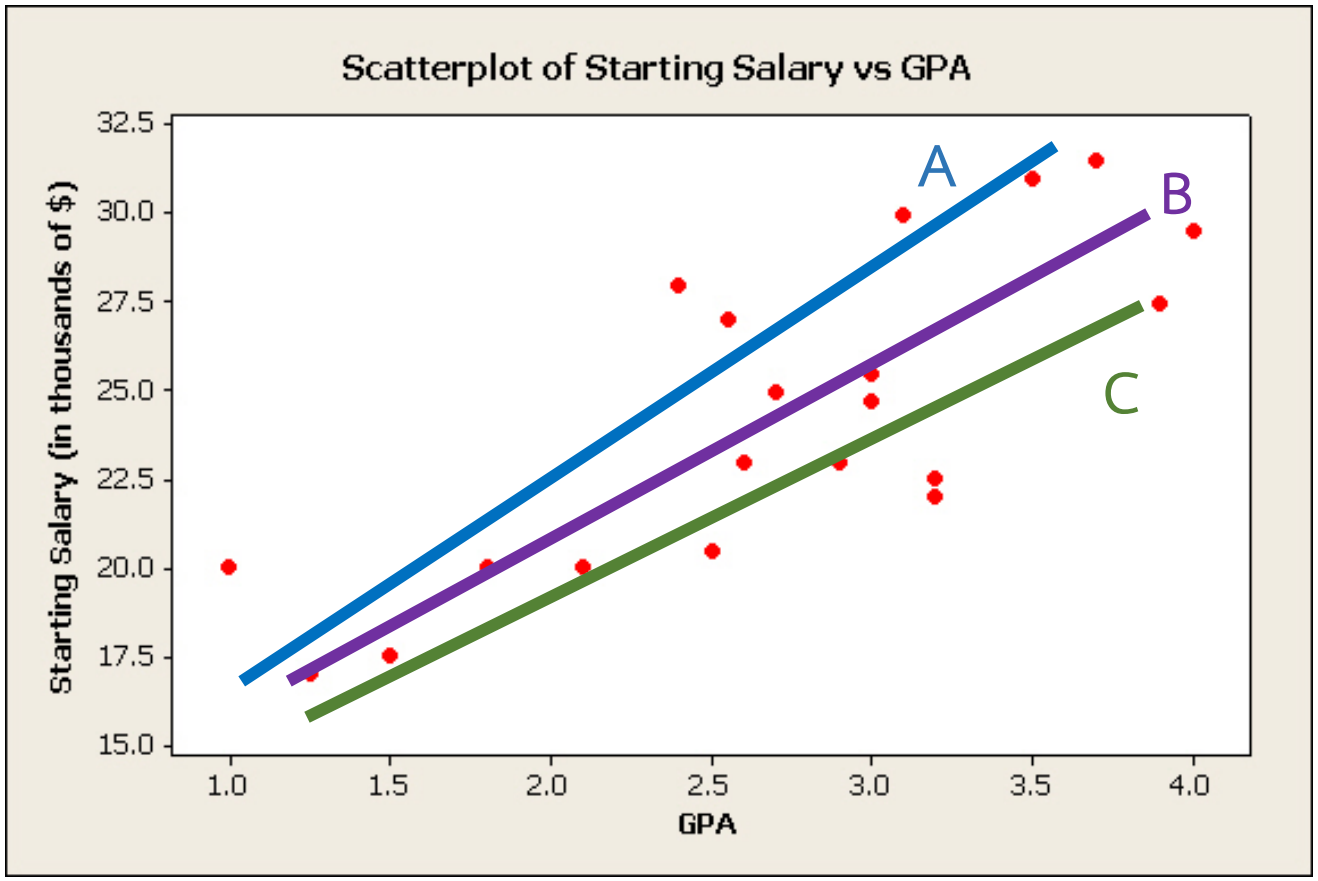
(assumed data)

Prediction !



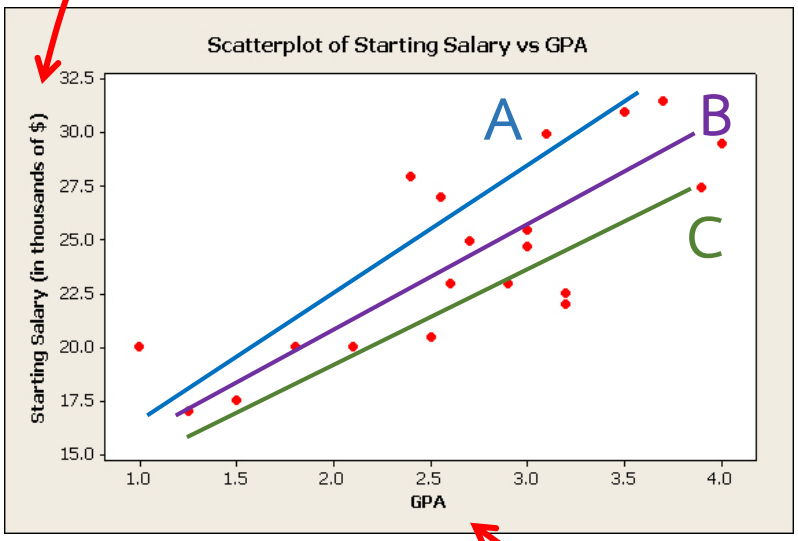
Ref: • (image) http://www.chegg.com/homework-help/questions-and-answers/orientation-meeting-new-students-college-president-wants-emphasize-importance-getting-good-q12615281

Prediction !



Ref: • (image) http://www.chegg.com/homework-help/questions-and-answers/orientation-meeting-new-students-college-president-wants-emphasize-importance-getting-good-q12615281

Linear Equation



$$y = ax + c$$

Simple Linear Regression

$$\hat{Y}_i = b_0 + b_1 X_i$$

Ref:

Simple Linear Regression

$$\hat{Y}_i = b_0 + b_1 X_i$$

Salary
15,400
20,500
23,300
26,200
27,900
32,100
30,200

GPA
1.07
2.32
2.75
3.02
3.25
3.63
3.98

Ref:

Your Salary?

GPA	U. Ranking	Salary
1.07	6	15,400
2.32	7	20,500
2.75	8	23,300
3.02	9	28,200
3.25	7	27,900
3.63	10	32,100
3.98	5	30,200

(assumed data)

Multiple Linear Regression

$$\hat{Y}_i = b_0 + b_1 X_{1i} + b_2 X_{2i} + \cdots$$

Ref:

Multiple Linear Regression

$$\hat{Y}_i = b_0 + b_1 X_{1i} + b_2 X_{2i} + \cdots$$

Salary
15,400
20,500
23,300
28,200
27,900
32,100
30,200

GPA
1.07
2.32
2.75
3.02
3.25
3.63
3.98

U. Ranking
6
7
8
9
7
10
5

Ref:

Coefficients

$$\hat{Y}_i = 4006 + 5655 * GPA + 698 * URank$$

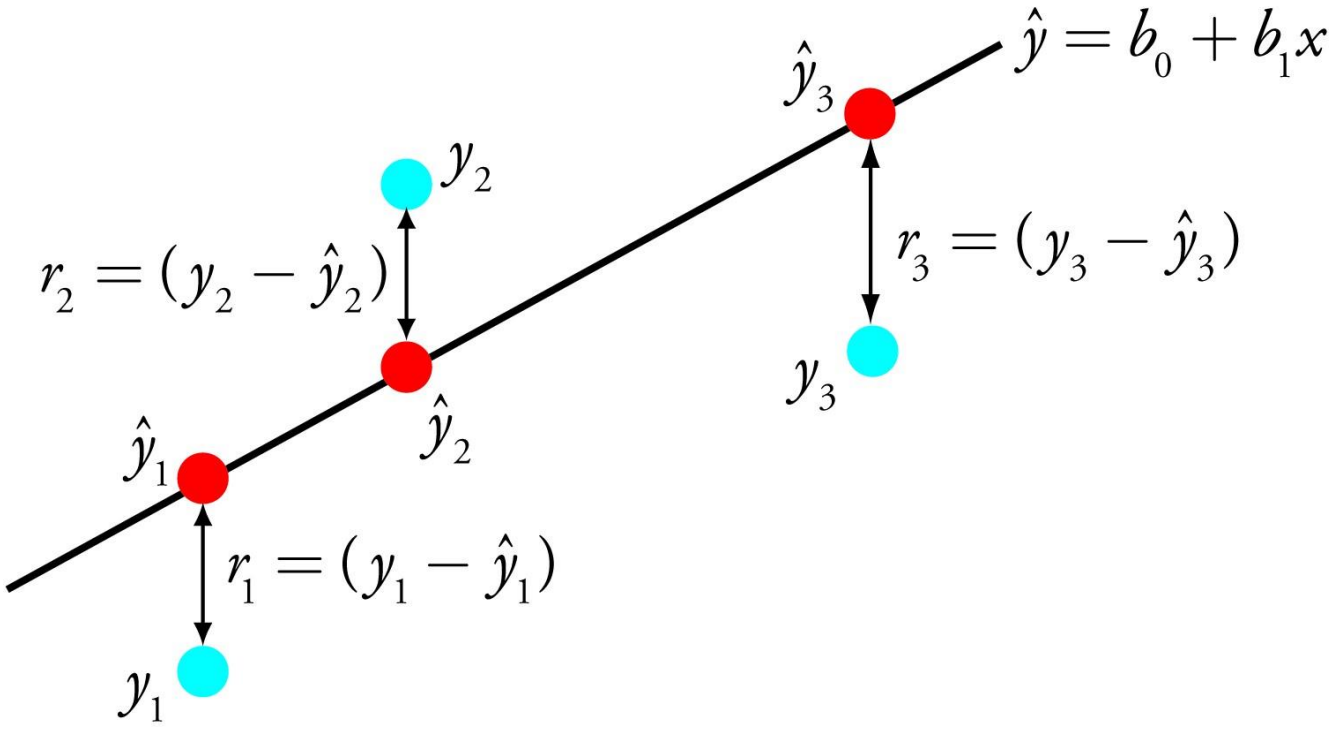
Python

1. from sklearn import linear_model
2. X = df[“GPA”, “Urank”]
3. y = df[“Salary”]
4. lm = linear_model.LinearRegression()
5. lm.fit (X,y)
6. lm.coef_

array([406, 565, 698])

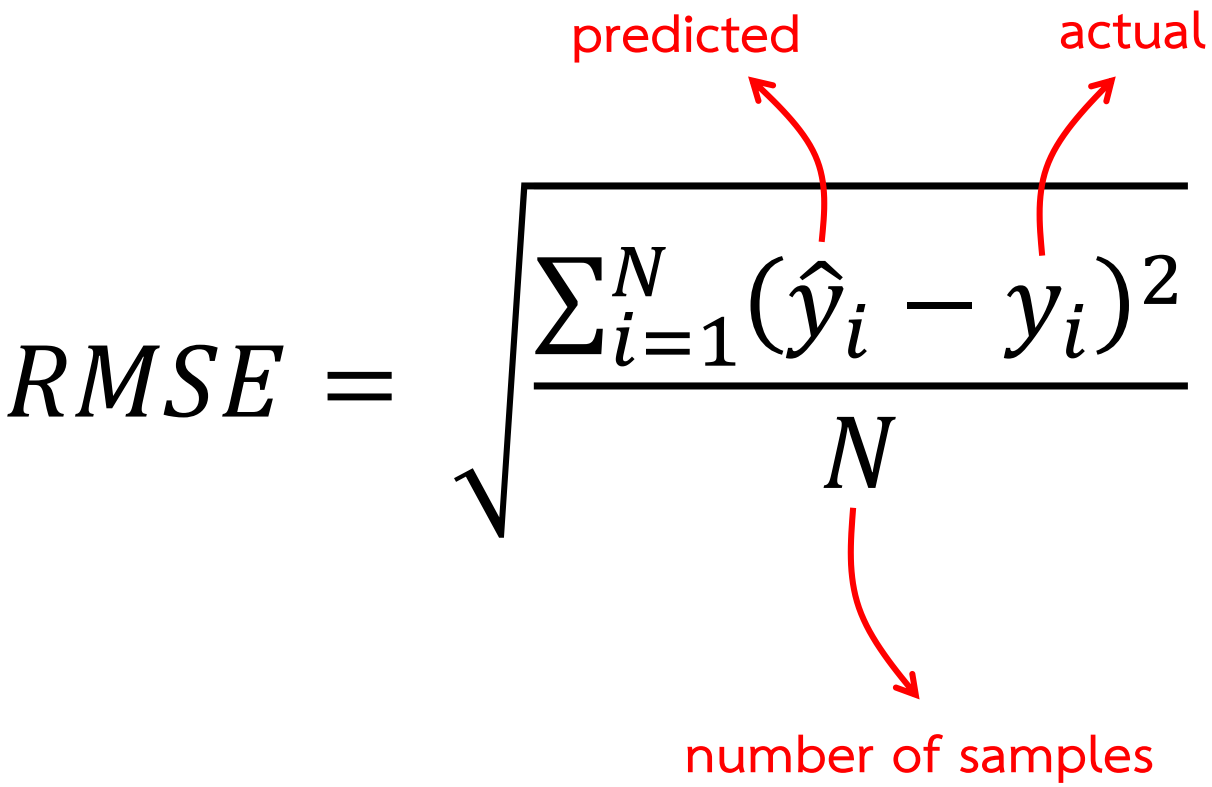
Evaluation Methods

Residual



Ref: • (image) https://towardsdatascience.com/simple-linear-regression-2421076a5892

Root Mean Squared Error (RMSE)



RMSE

$$\hat{Y}_i = ? + ??* GPA + ???* URank$$

GPA	U. Ranking	Actual Salary
1.07	6	15,400
2.32	7	20,500
2.75	8	23,300
3.02	9	28,200
3.25	7	27,900
3.63	10	32,100
3.98	5	30,200

RMSE

$$\hat{Y}_i = 4006 + 5655 * GPA + 698 * URank$$

GPA	U. Ranking	Actual Salary
1.07	6	15,400
2.32	7	20,500
2.75	8	23,300
3.02	9	28,200
3.25	7	27,900
3.63	10	32,100
3.98	5	30,200

RMSE

$$\hat{Y}_i = 4006 + 5655 * GPA + 698 * URank$$

GPA	U. Ranking	Actual Salary	Predicted Salary
1.07	6	15,400	16,350
2.32	7	20,500	23,100
2.75	8	23,300	25,750
3.02	9	28,200	27,600
3.25	7	27,900	27,750
3.63	10	32,100	31,150
3.98	5	30,200	30,400

RMSE

$$\hat{Y}_i = 4006 + 5655 * GPA + 698 * URank$$

GPA	U. Ranking	Actual Salary	Predicted Salary	$\hat{y}_i - y_i$	$(\hat{y}_i - y_i)^2$
1.07	6	15,400	16,350	1,155.15	1,334,371.52
2.32	7	20,500	23,100	-1,511.60	2,284,934.56
2.75	8	23,300	25,750	-1,841.25	3,390,201.56
3.02	9	28,200	27,600	833.90	695,389.21
3.25	7	27,900	27,750	629.25	395,955.56
3.63	10	32,100	31,150	586.35	343,806.32
3.98	5	30,200	30,400	197.10	38,848.41
				SUM	8,483,507.15
				MEAN	1,211,929.59
				RMSE	1,100.88

Other Evaluation Methods

Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\mathrm{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

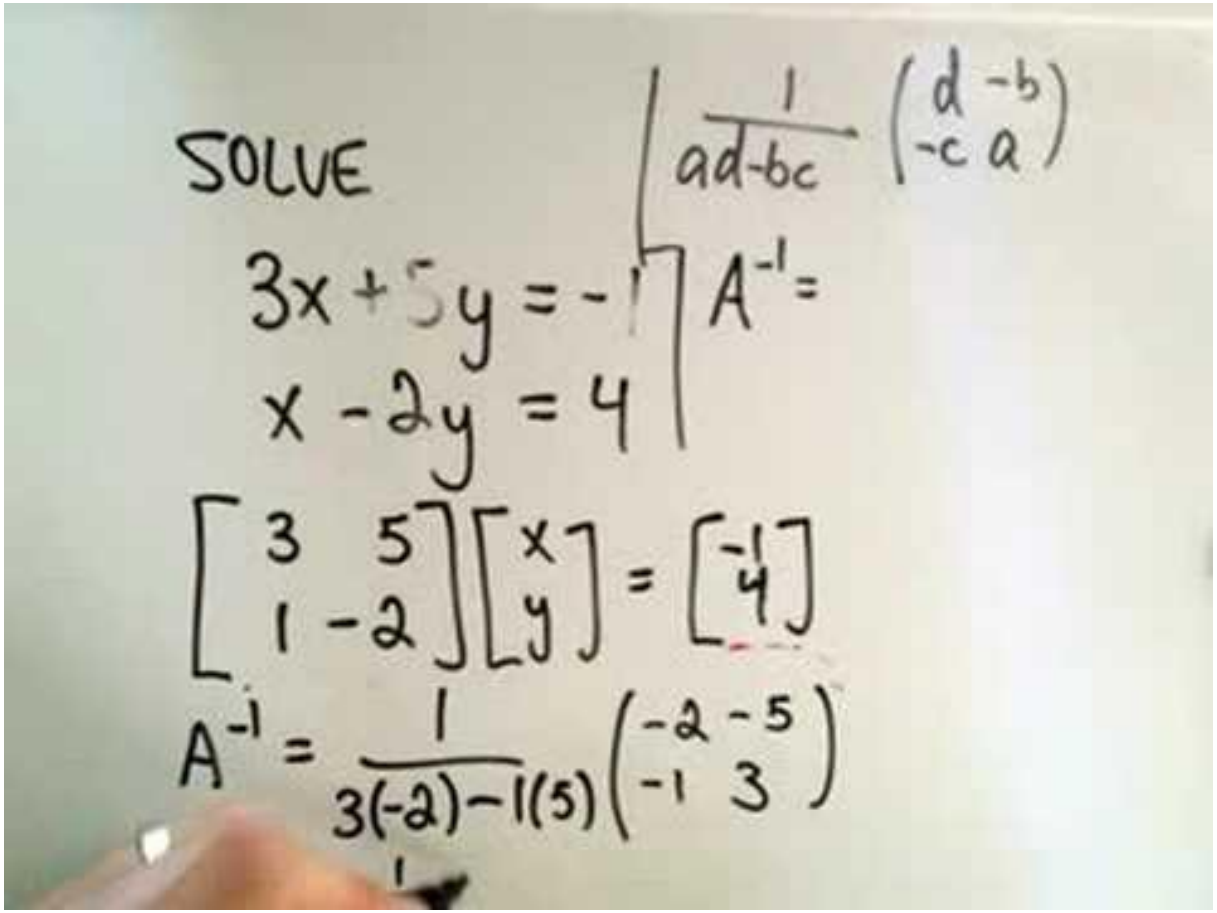
Ref: • (image) https://tex.stackexchange.com/questions/170372/increase-vertical-space-in-table

Python

- from sklearn import linear_model
- from sklearn.metrics import mean_squared_error
- from math import sqrt
- X = df[“GPA”, “Urank”]
- y = df[“Salary”]
- lm = linear_model.LinearRegression()
- lm.fit (X,y)
- y_predicted = lm.predict(X)
- rmse = sqrt(mean_squared_error(y, y_predicted))

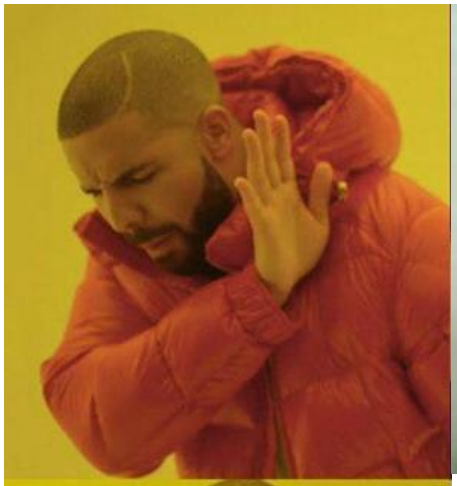
Learning

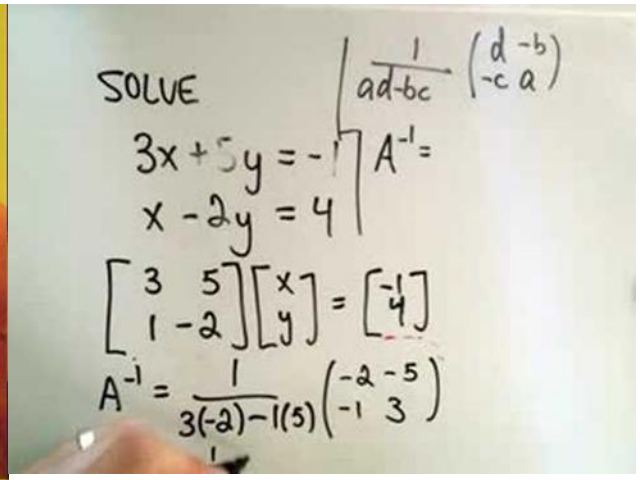
Solve by Matrix



Ref: • https://www.youtube.com/watch?v=PTN8rganLWo

Solve by .....





New algorithm $(n \geq 1)$:

Repeat $\{$

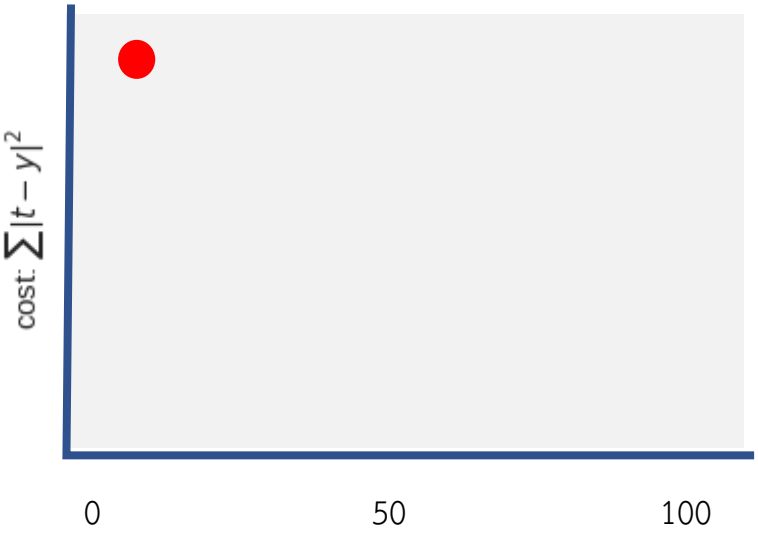
$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

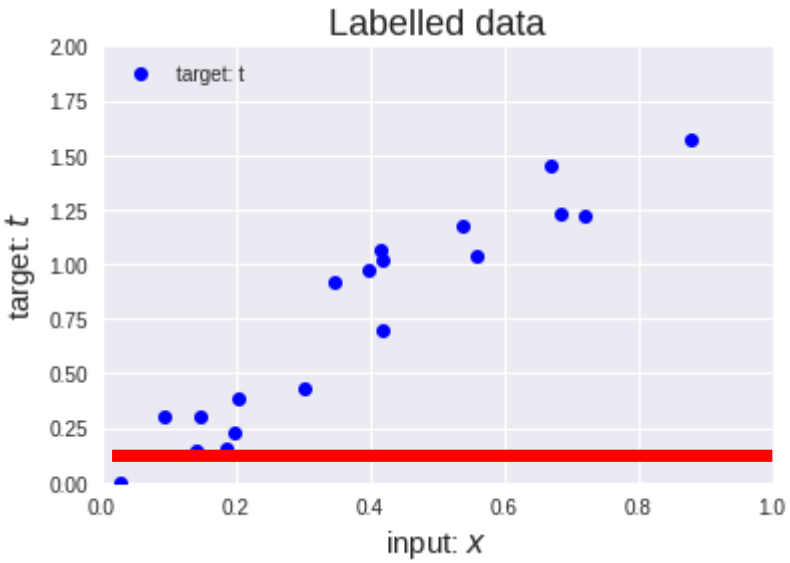
(simultaneously update θ_j for $j = 0, \ldots, n$) $\}$

Ref: • https://www.youtube.com/watch?v=PTN8rganLWo

Learning based on Gradient Descent

$y = ax + c$

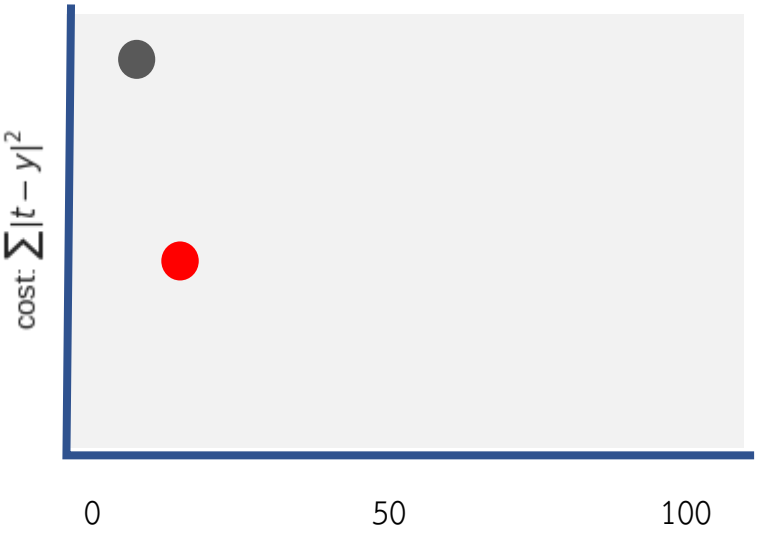


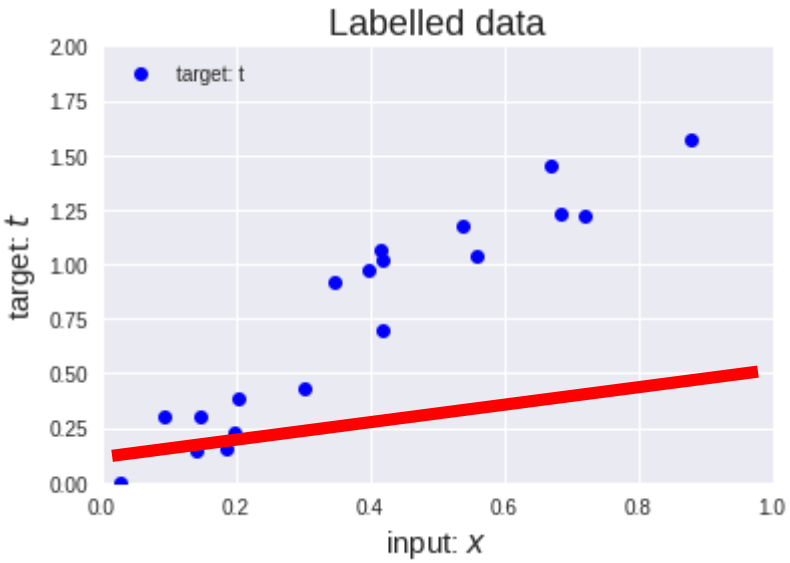


Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

Learning

$y = ax + c$

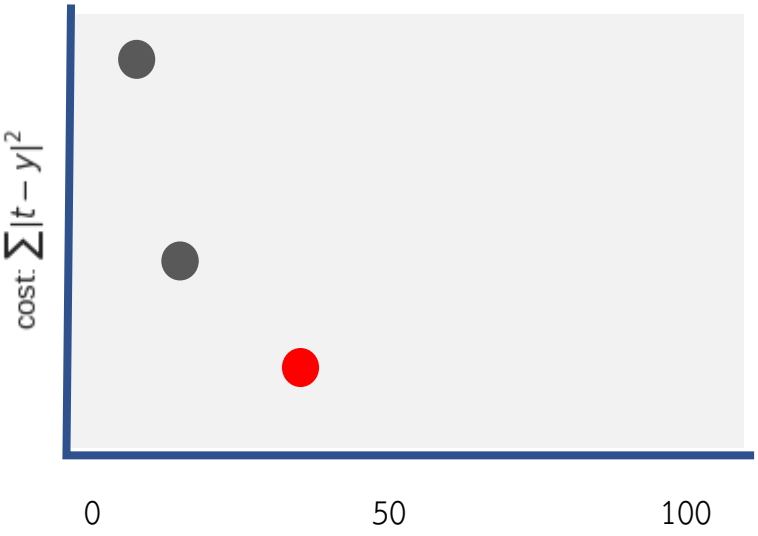


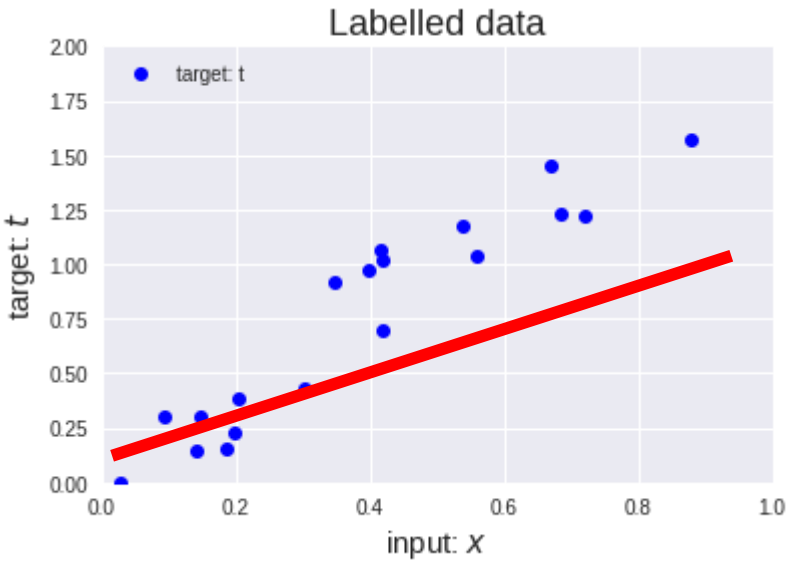


Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

Learning

$y = ax + c$

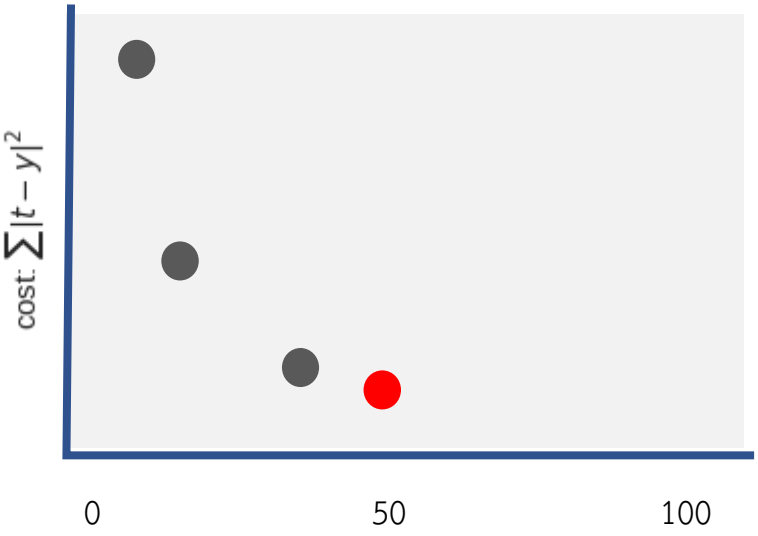


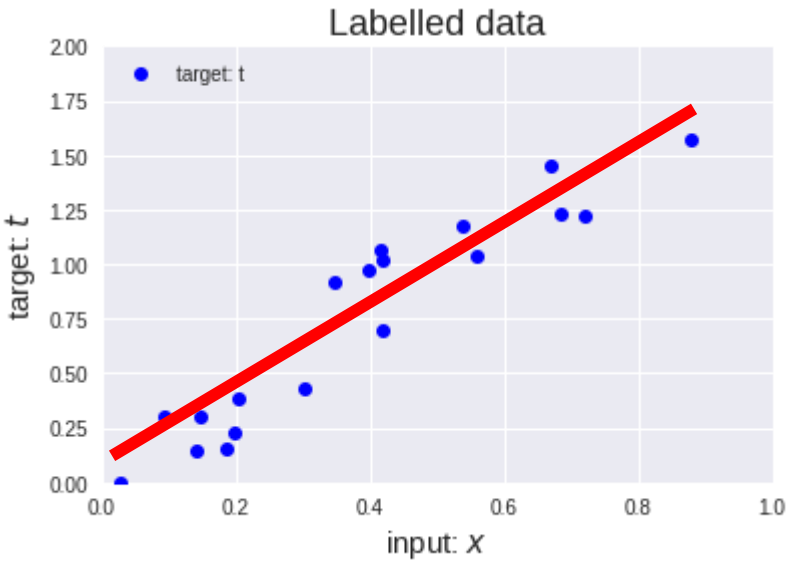


Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

Learning

$y = ax + c$

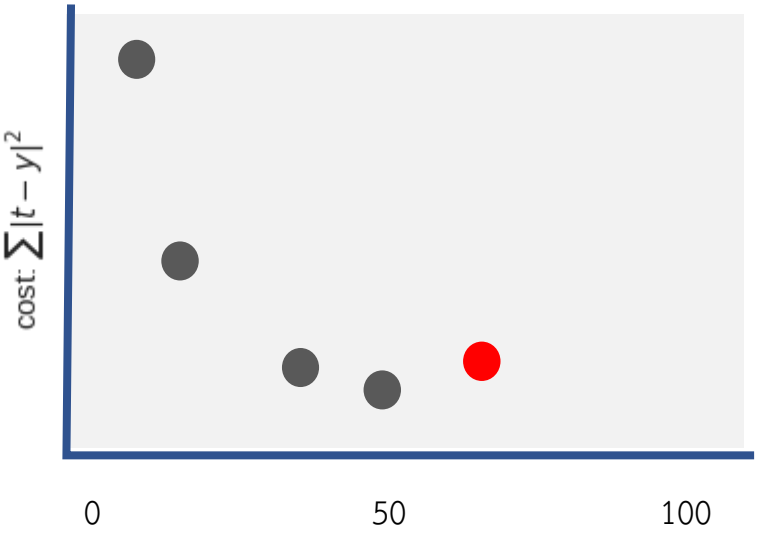




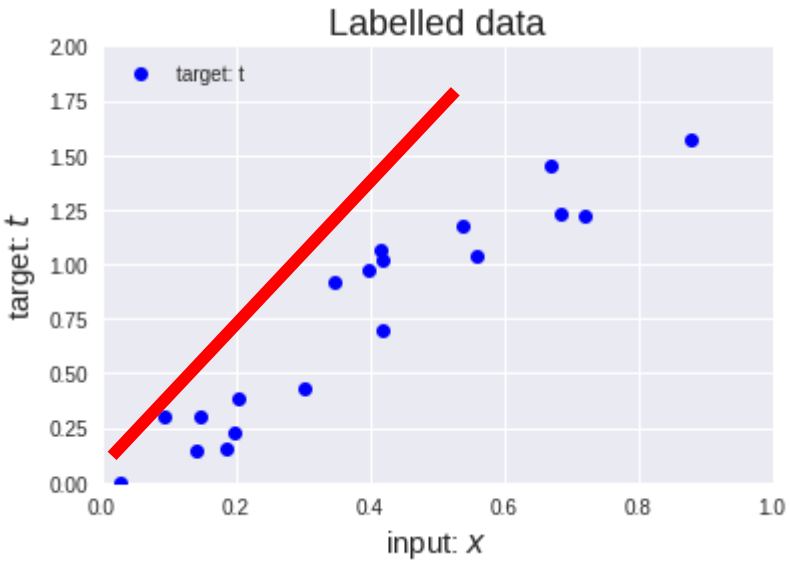
Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

Learning

$y = ax + c$



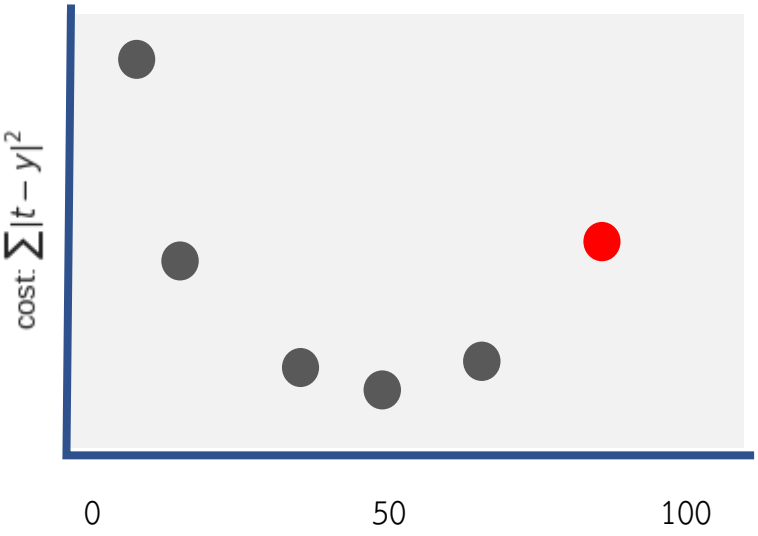
a



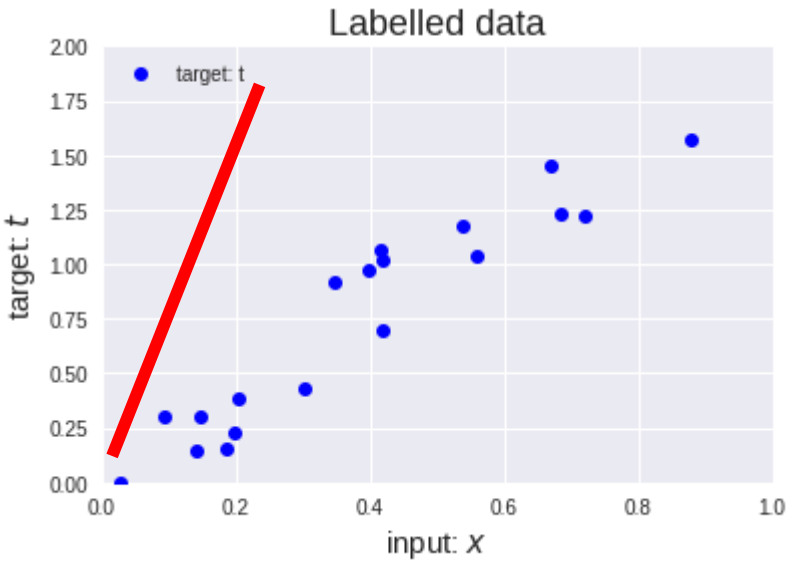
Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

Learning

$y = ax + c$



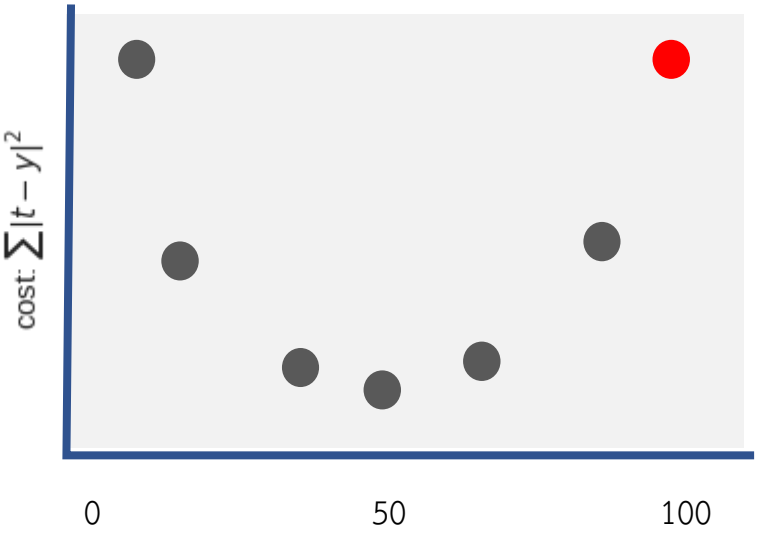
a



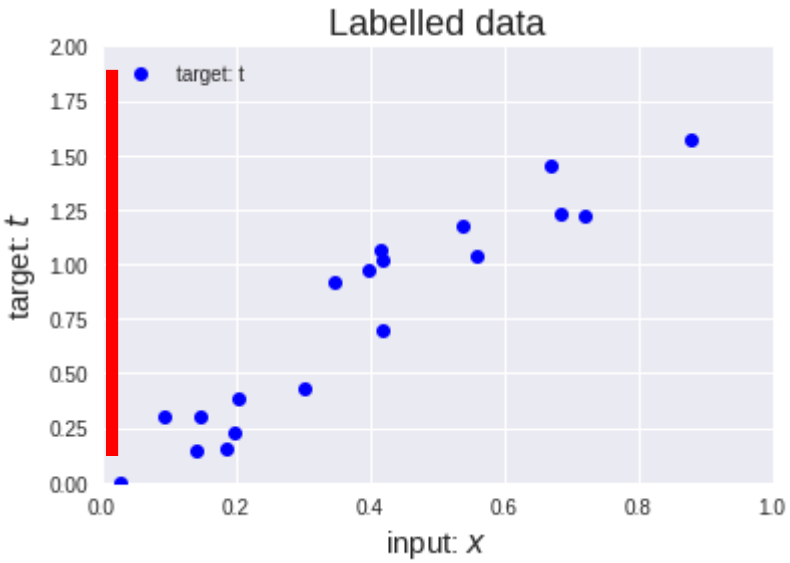
Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

Learning

$y = ax + c$

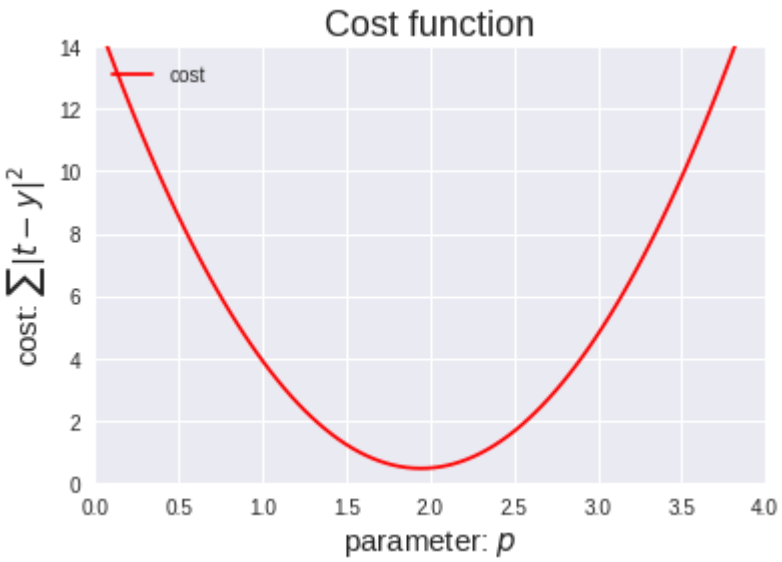


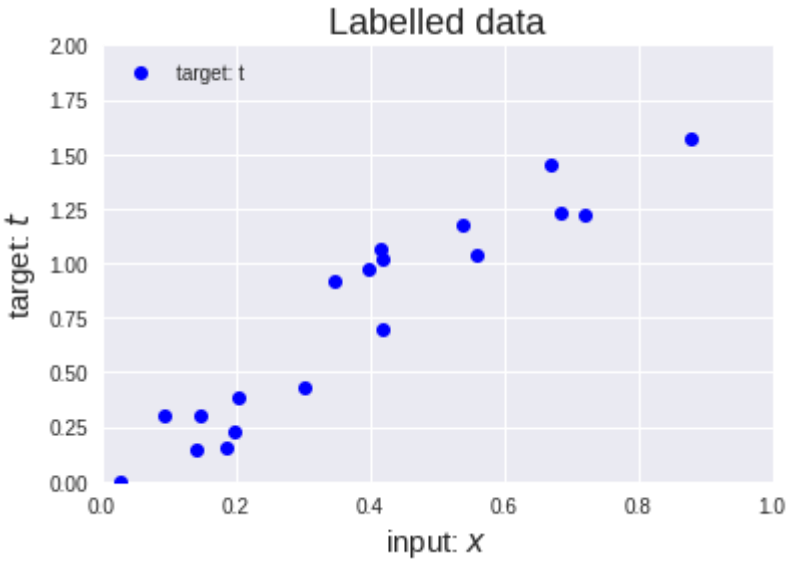
a



Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

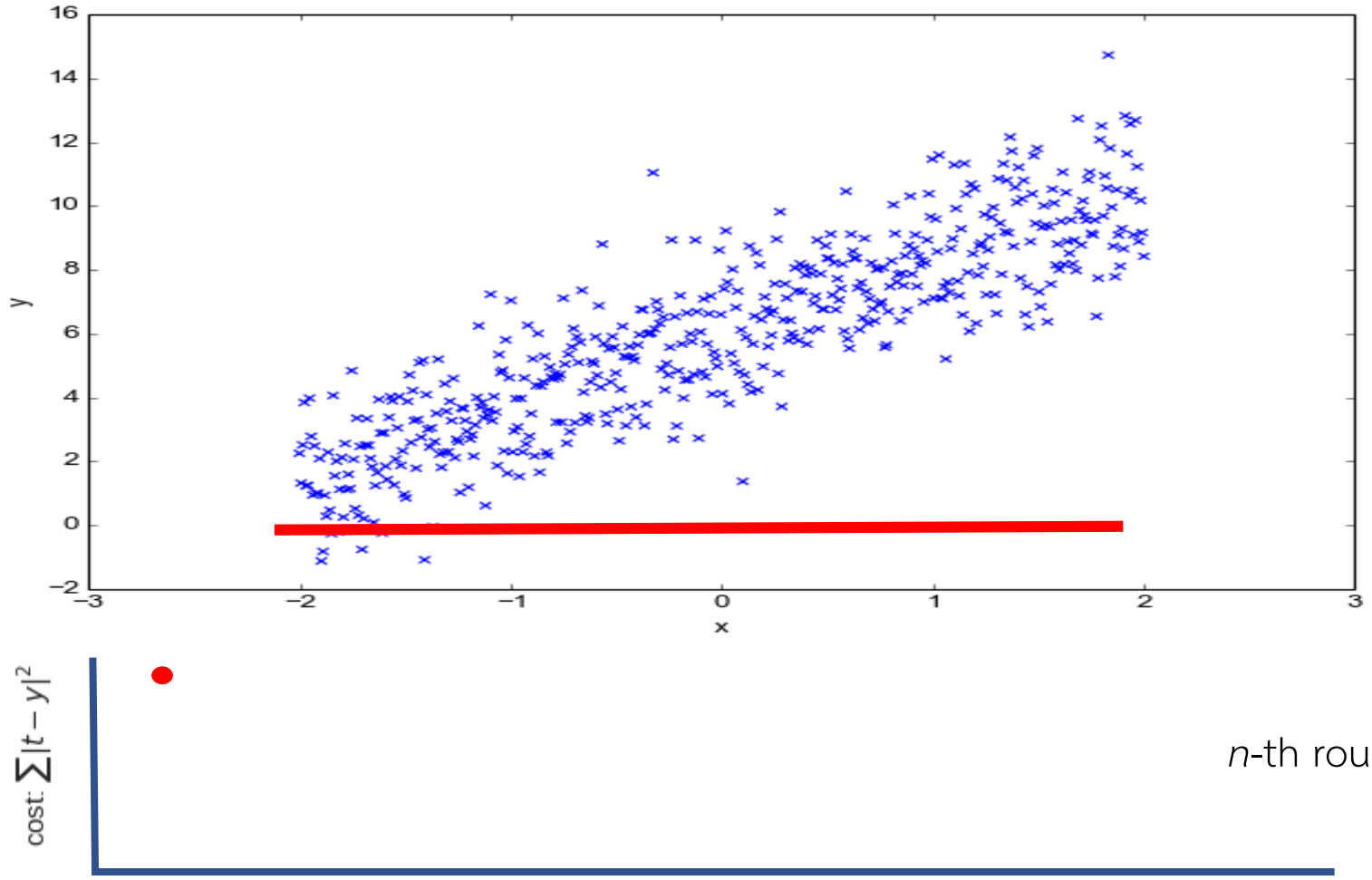
Learning





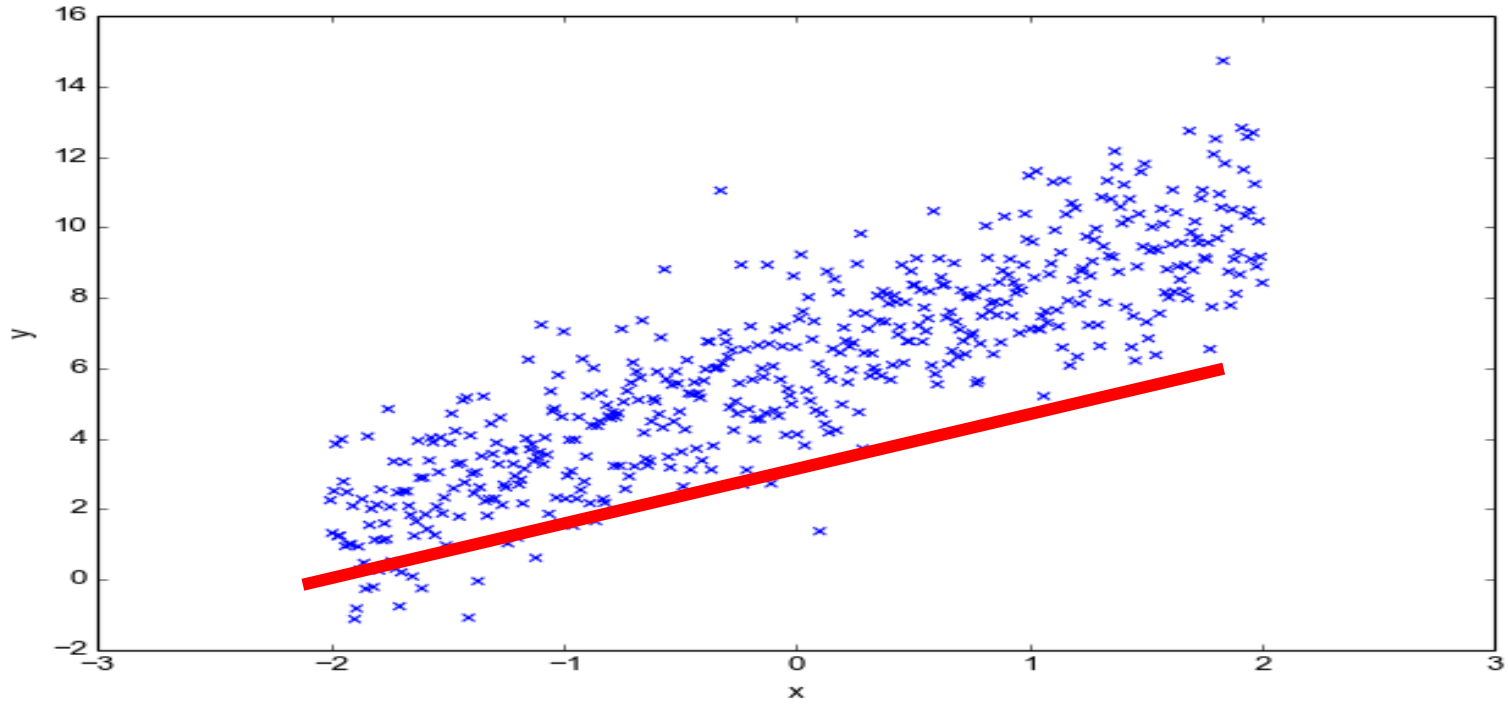
Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

Learning



Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

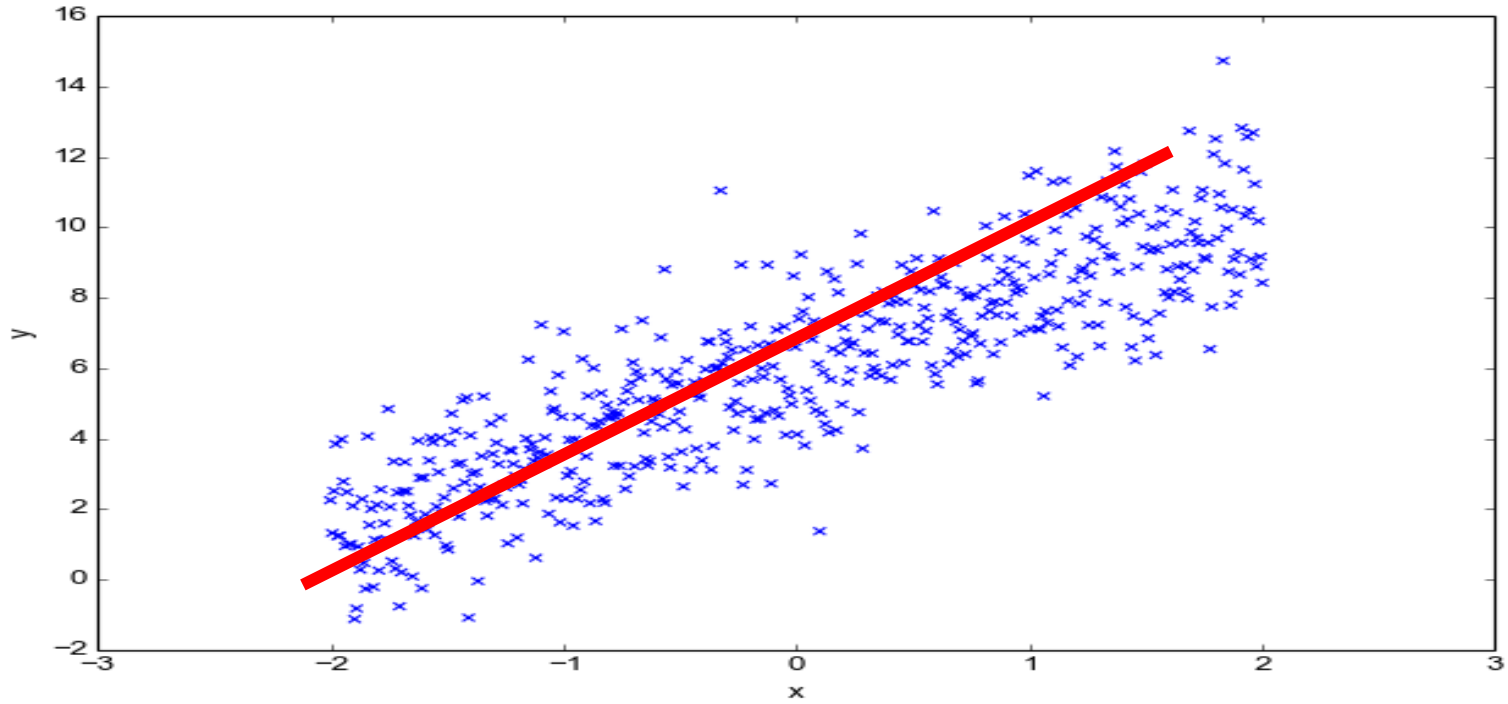
Learning



n-th round

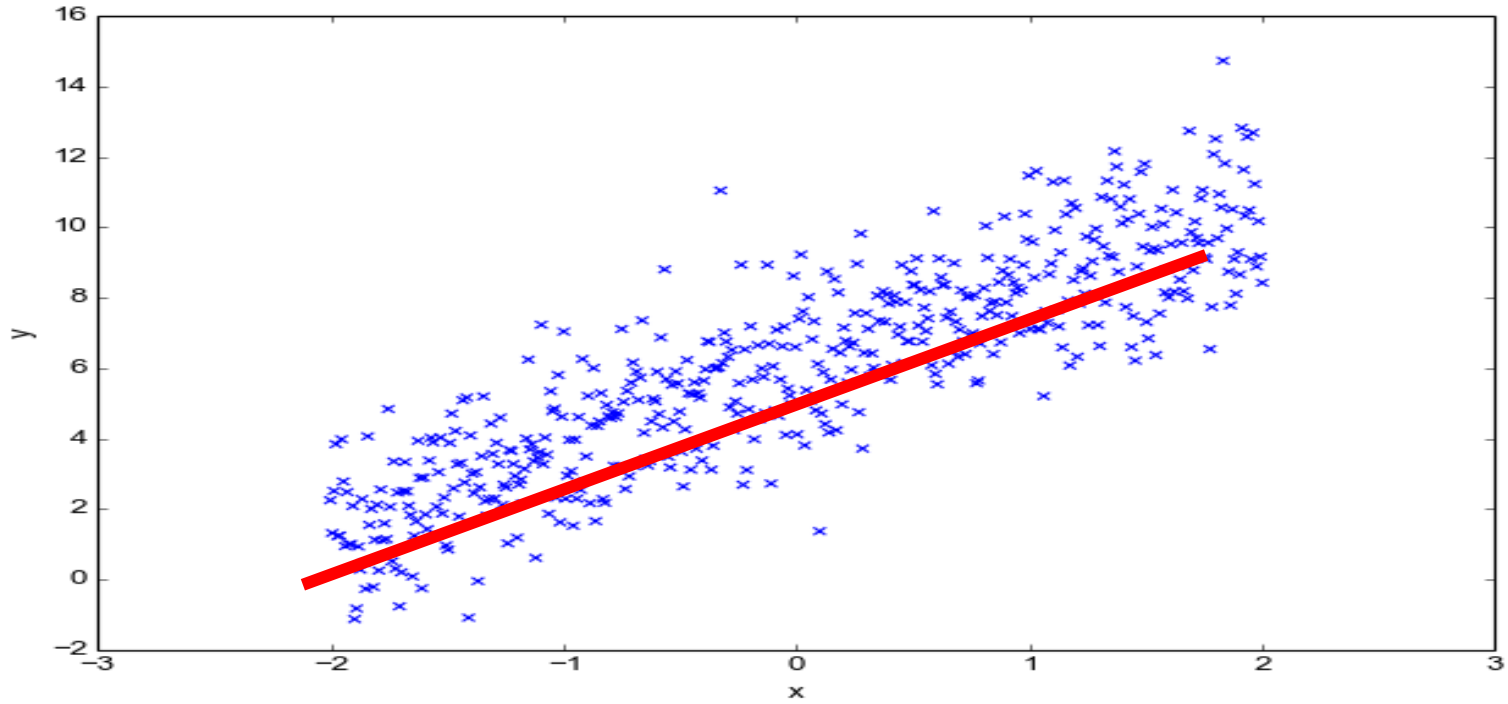
Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

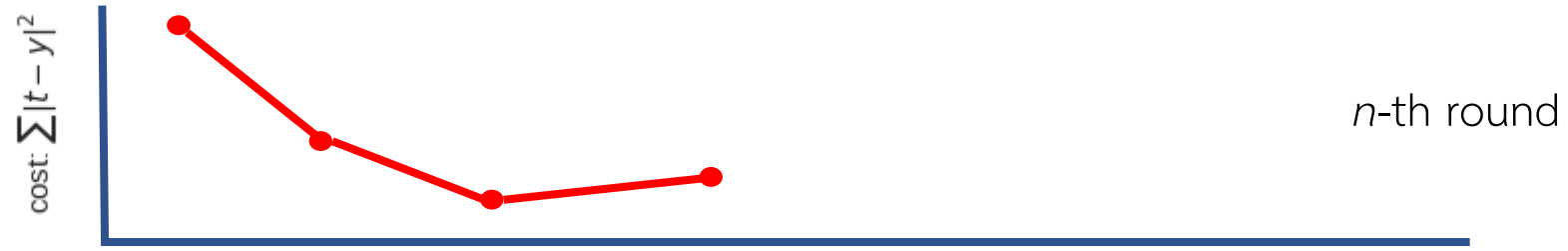
Learning



Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

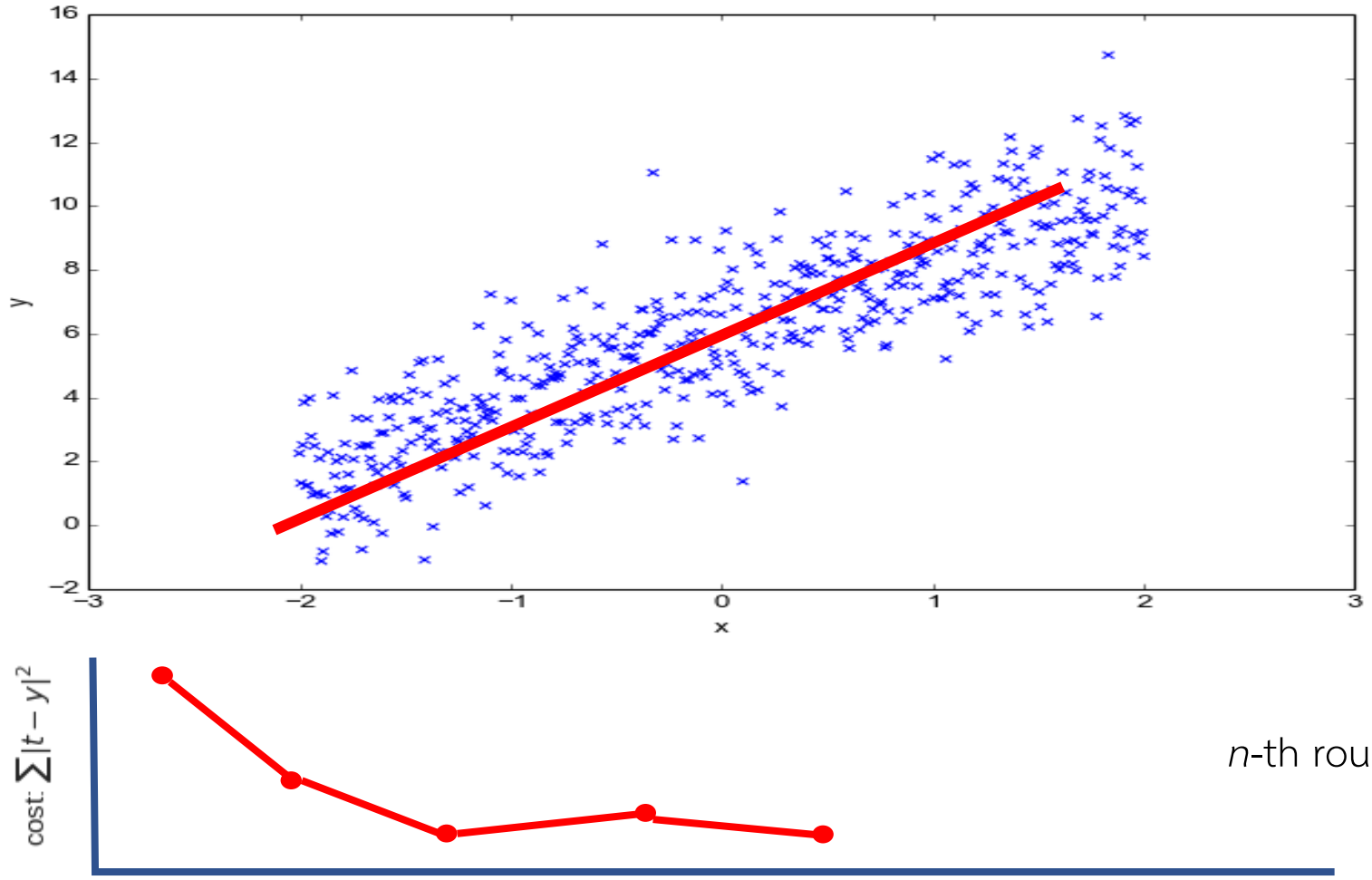
Learning





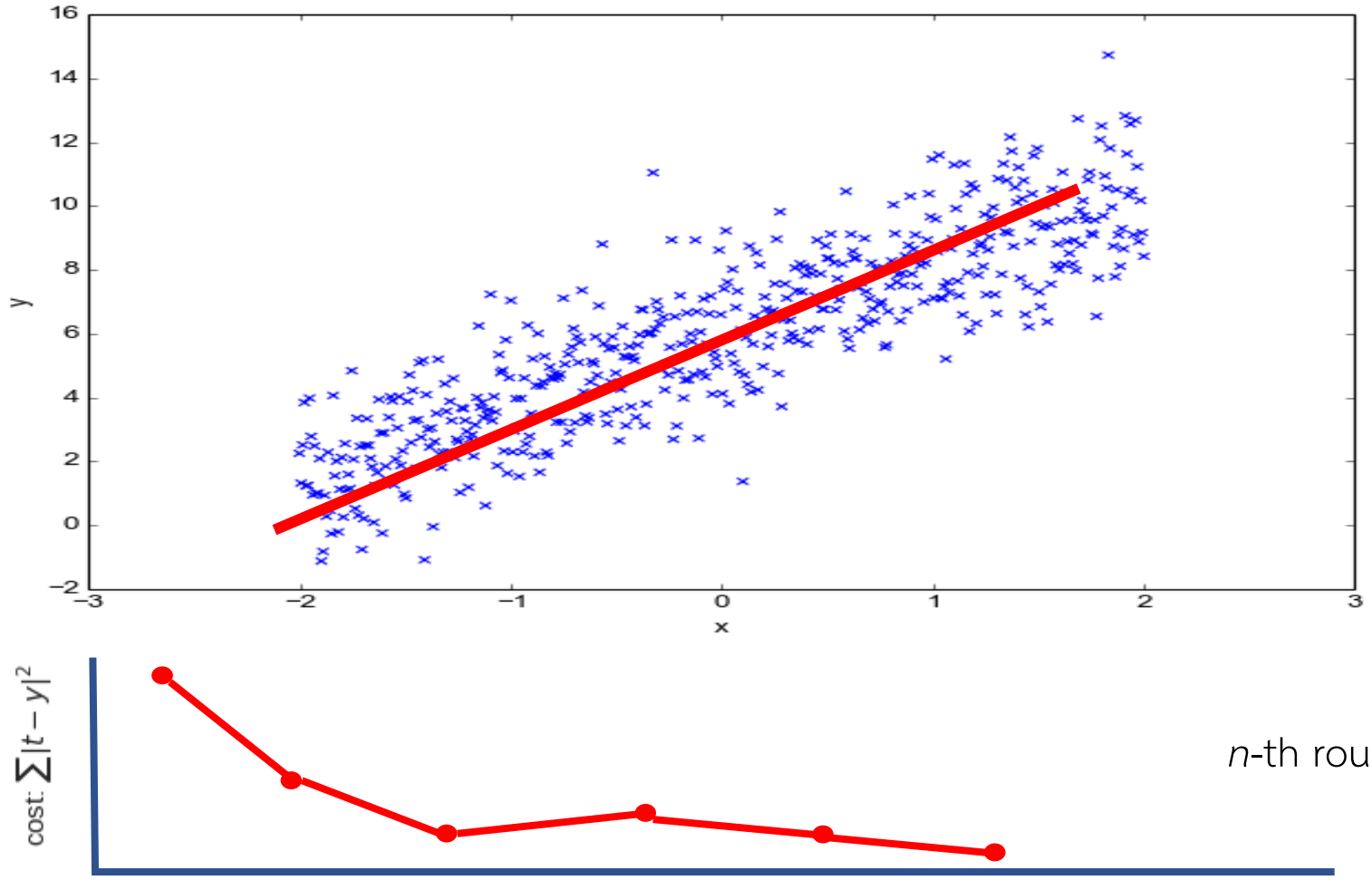
Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

Learning



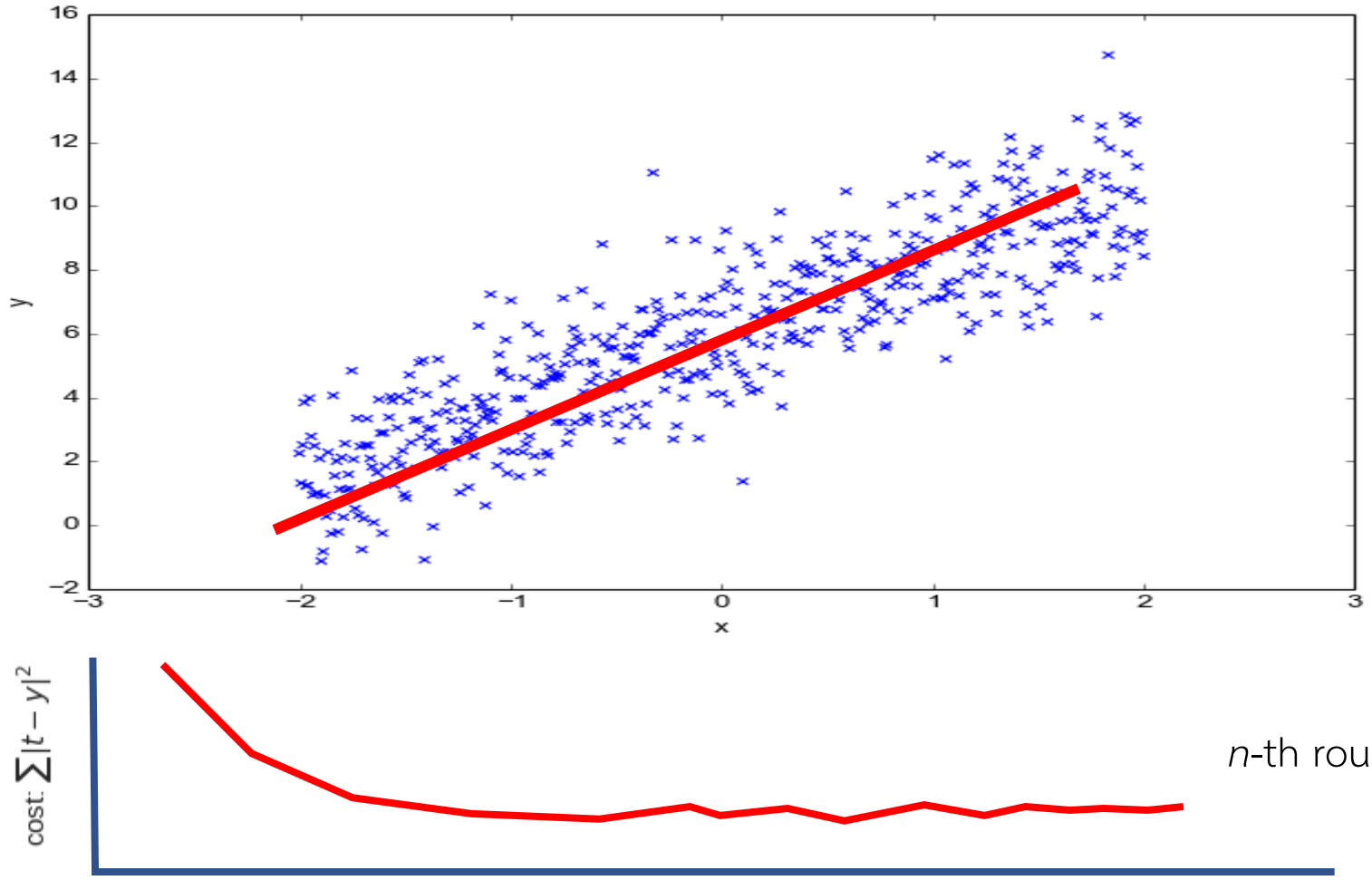
Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

Learning



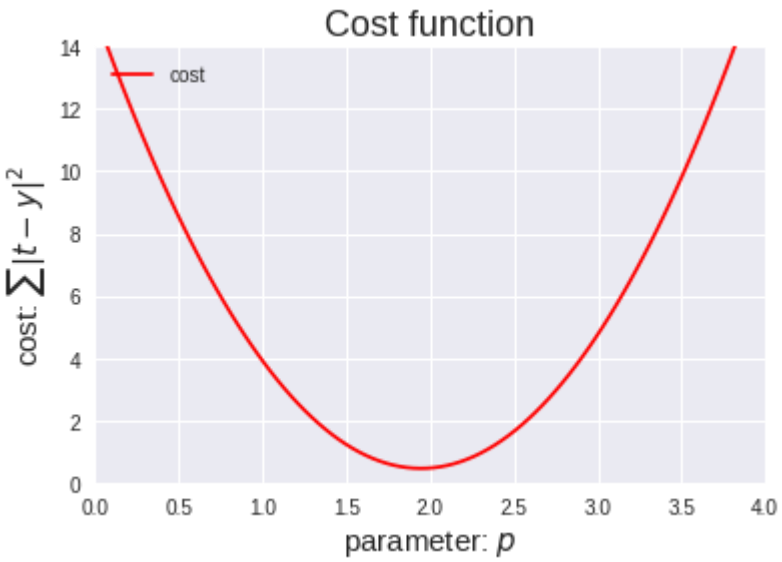
Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

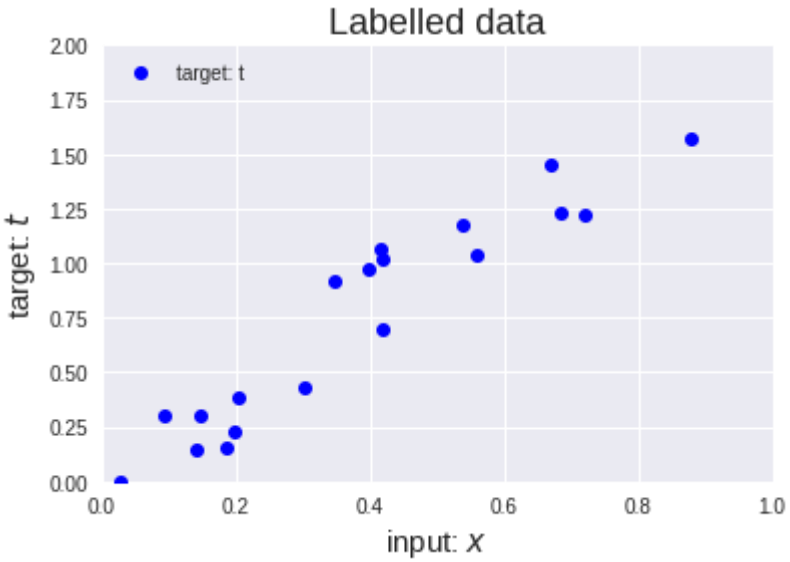
Learning



Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

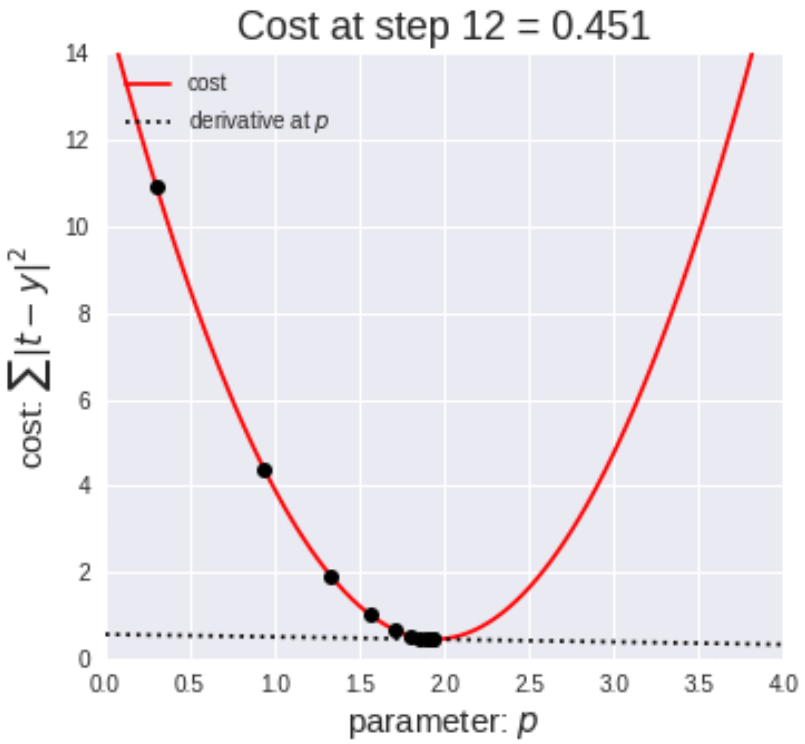
Learning

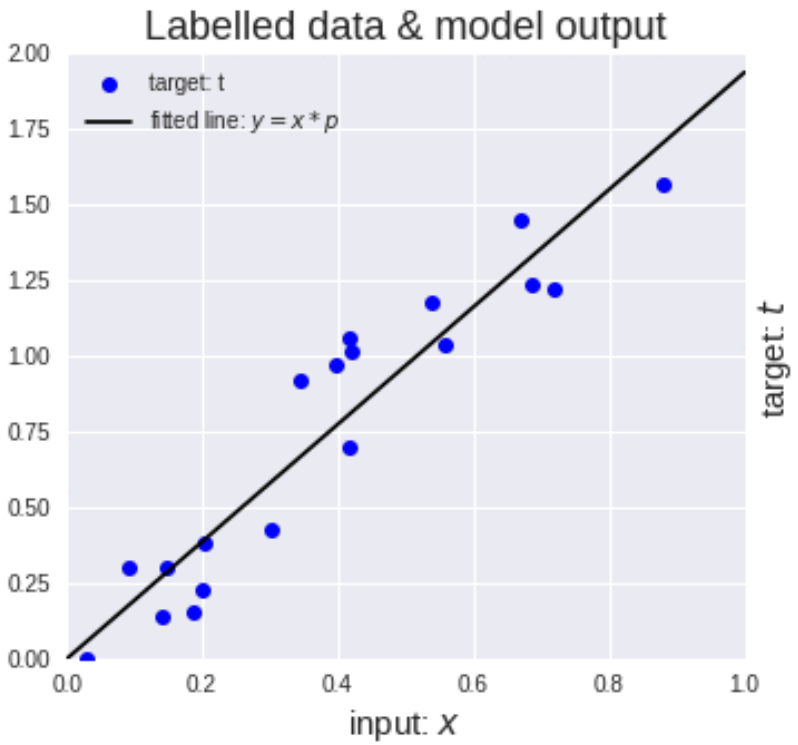




Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

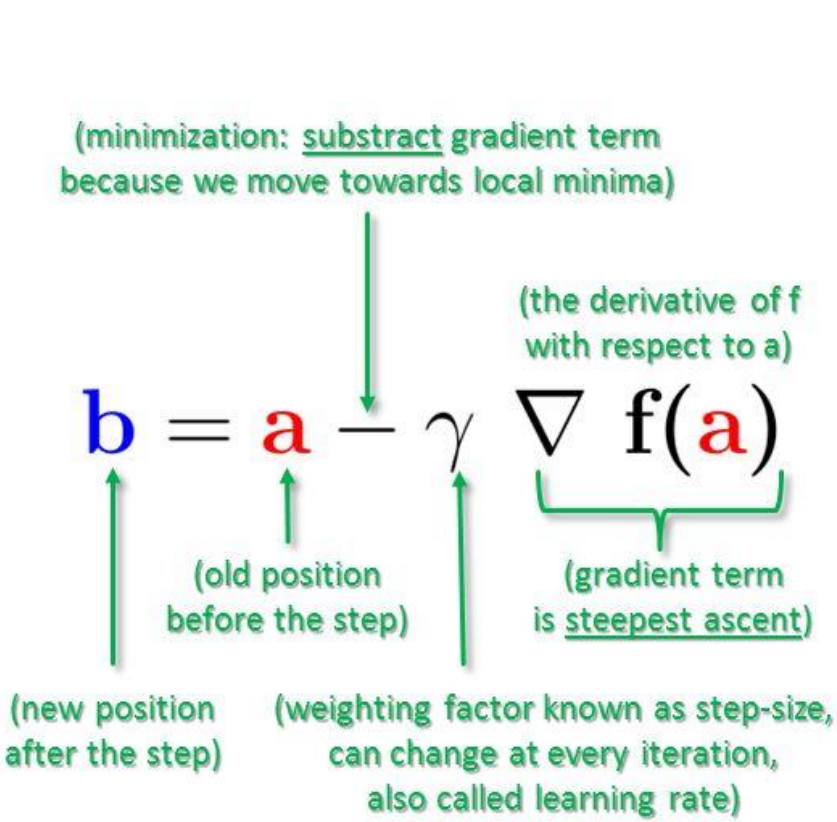
Learning

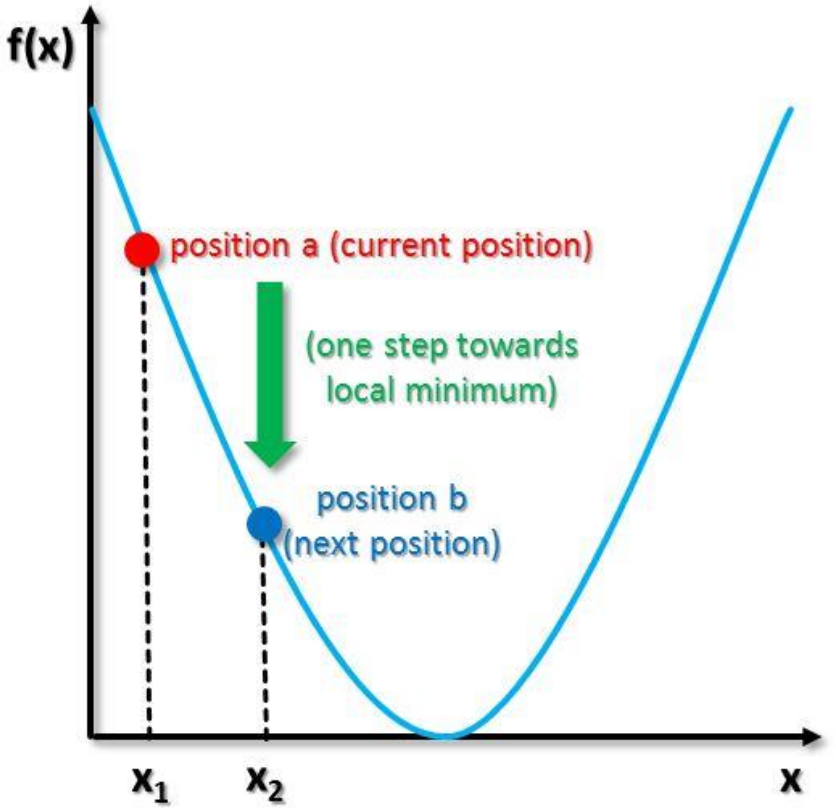




Ref: • (image) https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

Gradient Descent for Linear Regression



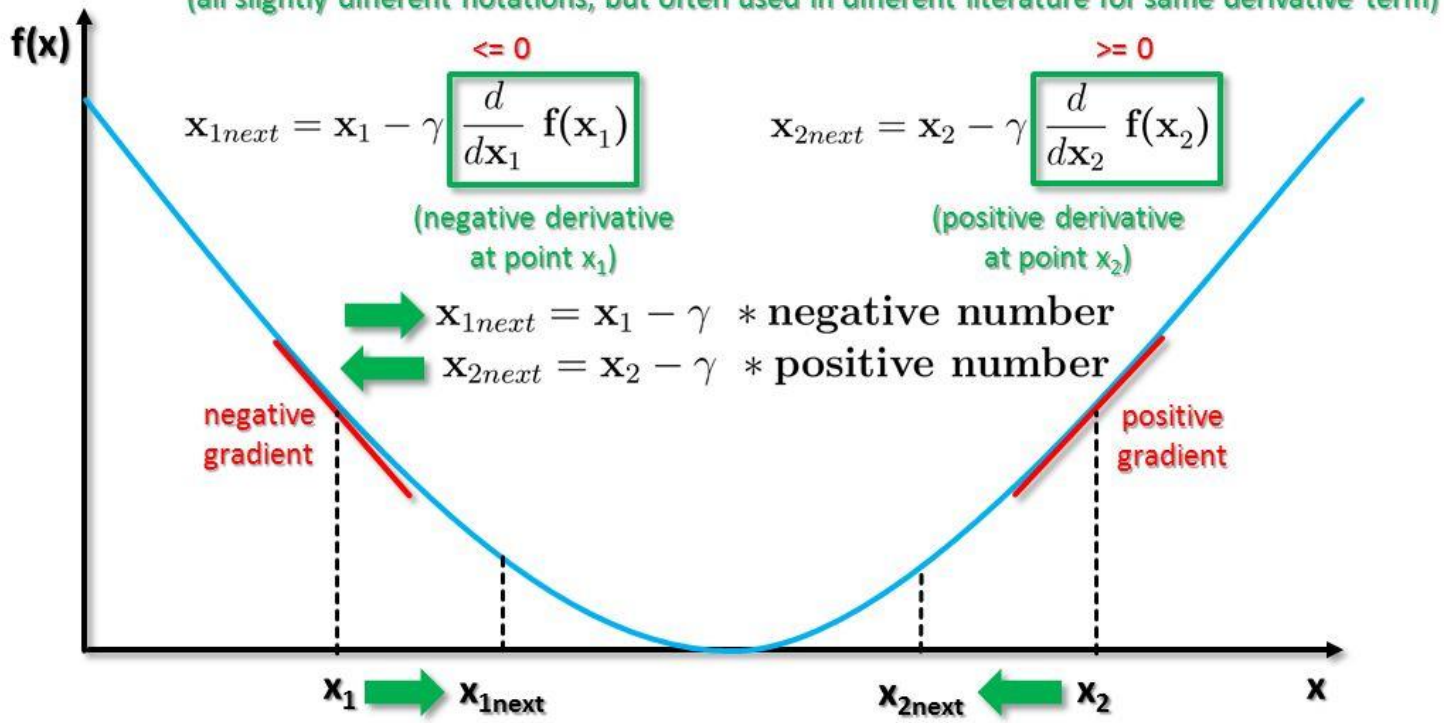


Ref: • http://www.big-data.tips/gradient-descent

Gradient Descent for Linear Regression

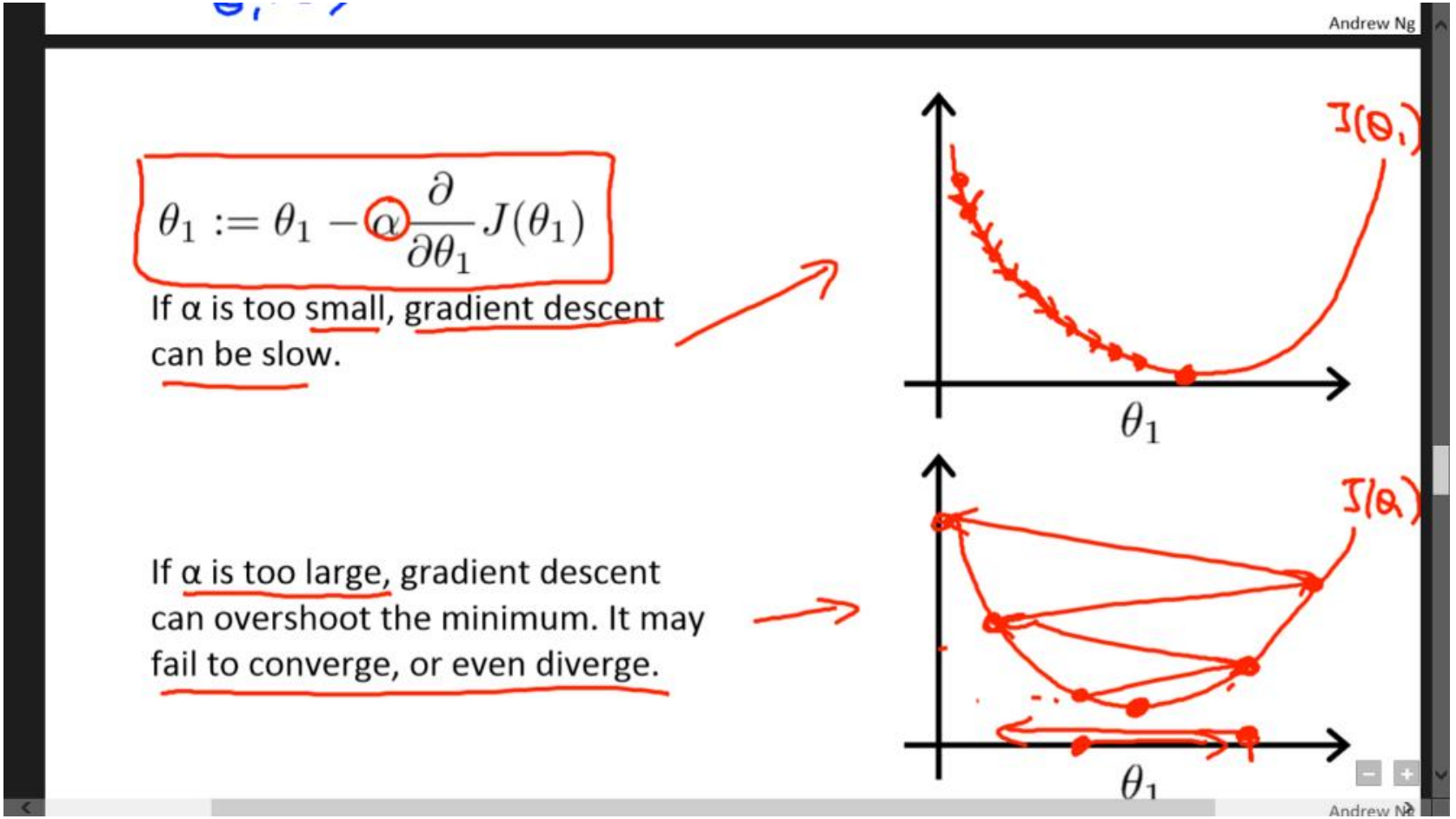
$$\mathbf{b} = \mathbf{a} - \gamma \nabla \mathbf{f}(\mathbf{a}) \quad \mathbf{b} = \mathbf{a} - \gamma \frac{\partial}{\partial \mathbf{a}} \mathbf{f}(\mathbf{a}) \quad \mathbf{b} = \mathbf{a} - \gamma \frac{d}{d\mathbf{a}} \mathbf{f}(\mathbf{a})$$

(all slightly different notations, but often used in different literature for same derivative term)



Ref: • http://www.big-data.tips/gradient-descent

Gradient Descent for Linear Regression



Ref: • screenshot from Andrew Ng video tutor

Learning

$$y = \theta_0 + \theta_1 x_1$$

repeat until convergence: {

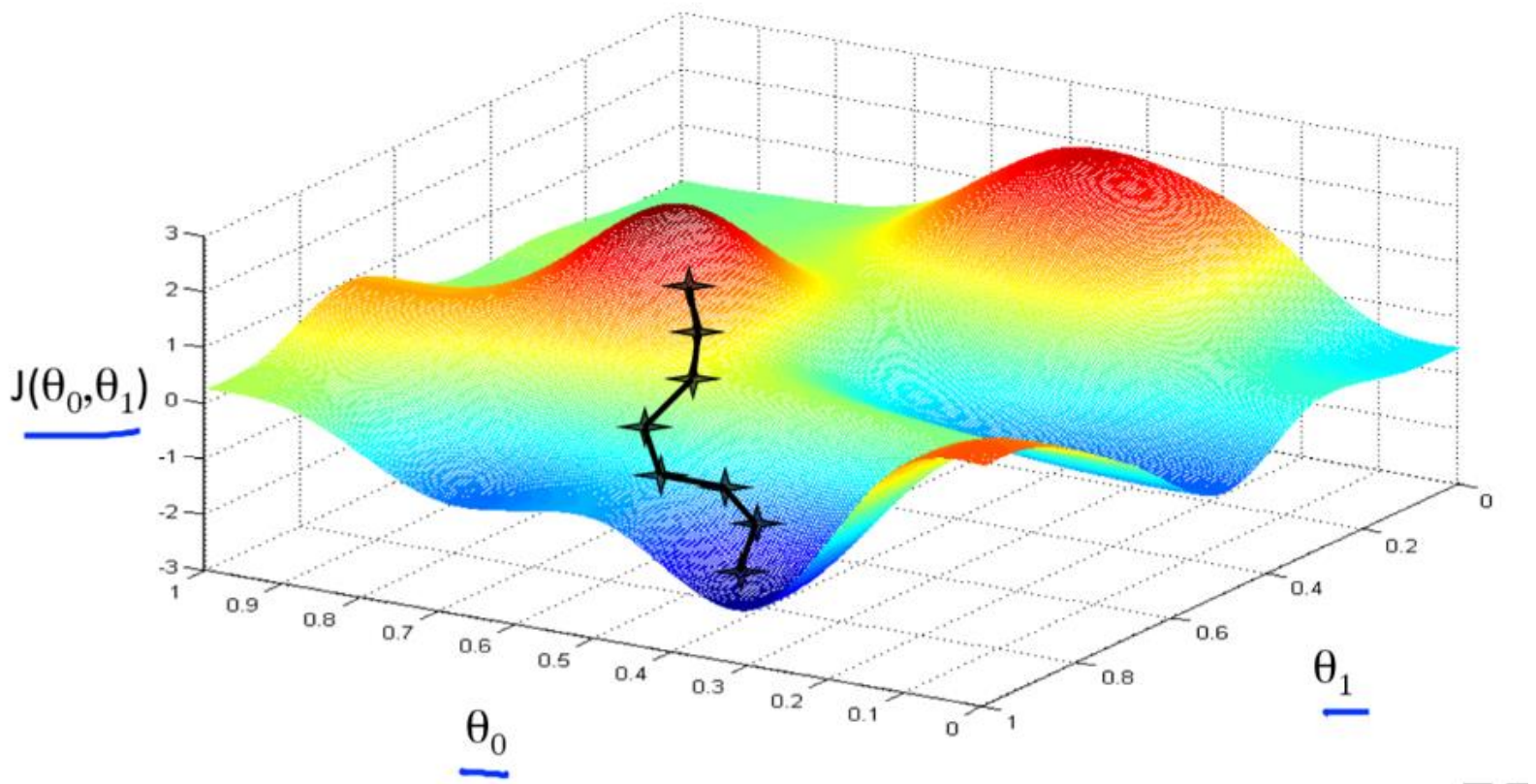
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) x_i)$$

}

Ref: • screenshot from Andrew Ng video tutor

Gradient Descent for Linear Regression



Ref: • screenshot from Andrew Ng video tutor

Feature Extraction

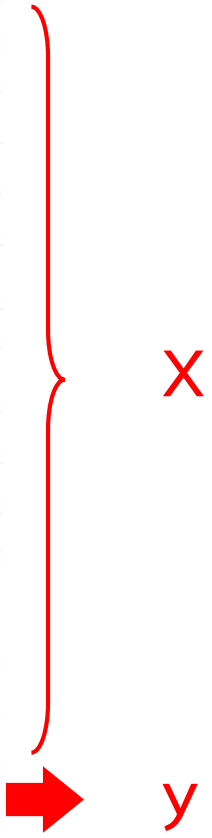
Boston Housing

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	CAT. MEDV
0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24	0
0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6	0
0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7	1
0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4	1
0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2	1
0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7	0
0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9	0
0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1	0
0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5	0
0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9	0
0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15	0
0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9	0
0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71	21.7	0
0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	396.9	8.26	20.4	0
0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	380.02	10.26	18.2	0

Ref: • https://www.solver.com/multiple-linear-regression-example

Boston Housing

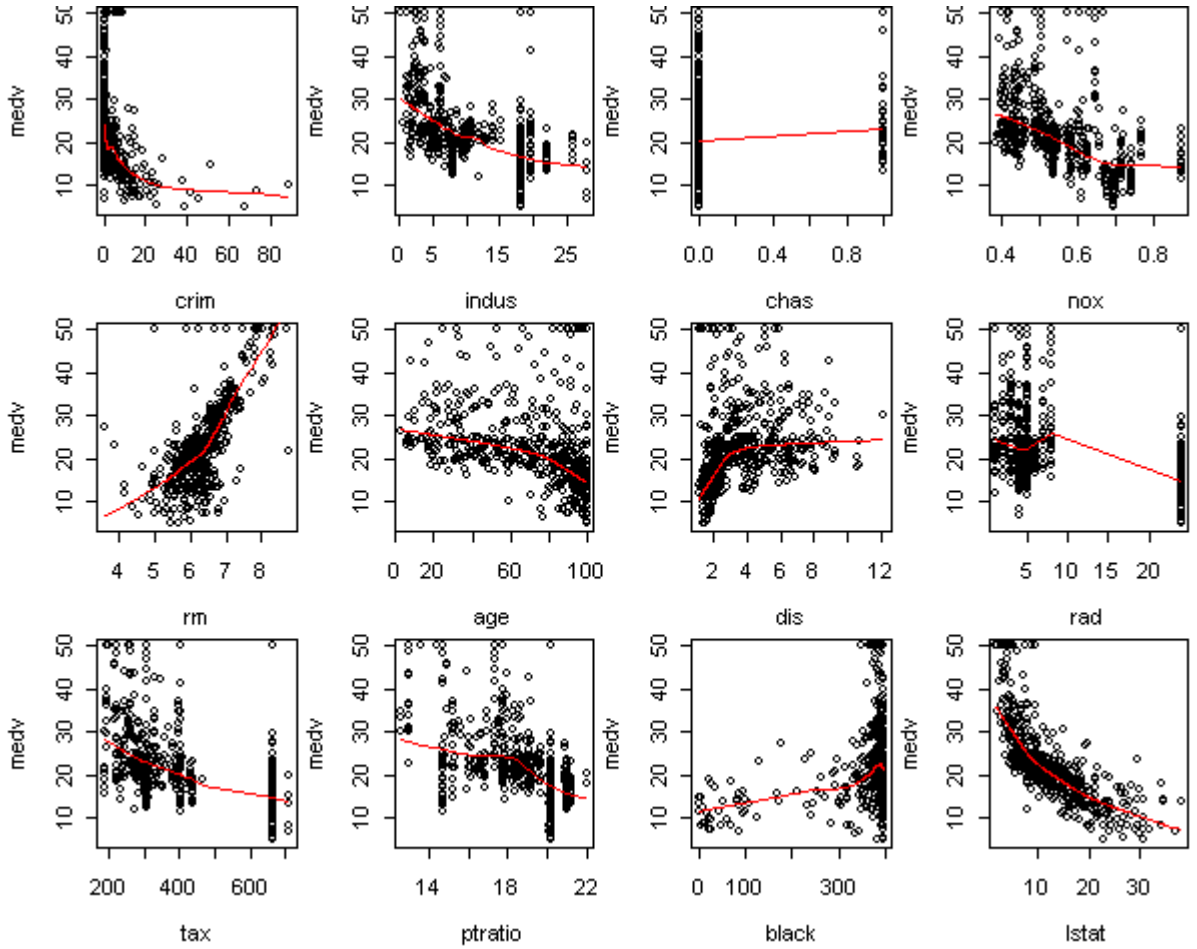
CRIM	Per capita crime rate by town
ZN	Proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS	Proportion of non-retail business acres per town
CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX	Nitric oxides concentration (parts per 10 million)
RM	Average number of rooms per dwelling
AGE	Proportion of owner-occupied units built prior to 1940
DIS	Weighted distances to five Boston employment centers
RAD	Index of accessibility to radial highways
TAX	Full-value property-tax rate per $10,000
PTRATIO	Pupil-teacher ratio by town
B	1000(Bk - 0.63)^2 where Bk is the proportion of African-Americans by town
LSTAT	% Lower status of the population
MEDV	Median value of owner-occupied homes in $1000's
CAT.MEDV	Binary variable that indicates based on the MEDV variable. If MEDV > 30, CAT.MEDV = 1



Ref: • https://www.solver.com/multiple-linear-regression-example

Correlations

- Content



Ref: • (image) http://alumni.media.mit.edu/~tpminka/courses/36-350.2001/lectures/day30/

Correlations

Ref: • (image) http://www.neural.cz/dataset-exploration-boston-house-pricing.html

Cross Validation

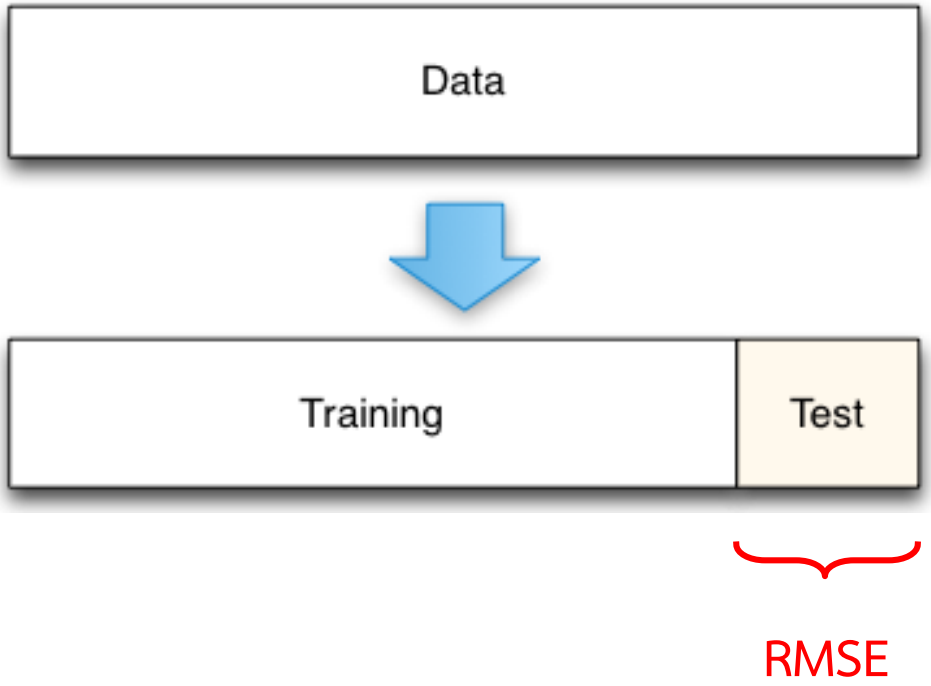
Rotation Estimation

How accurately a predictive model will perform in practice?

- Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.
- In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.

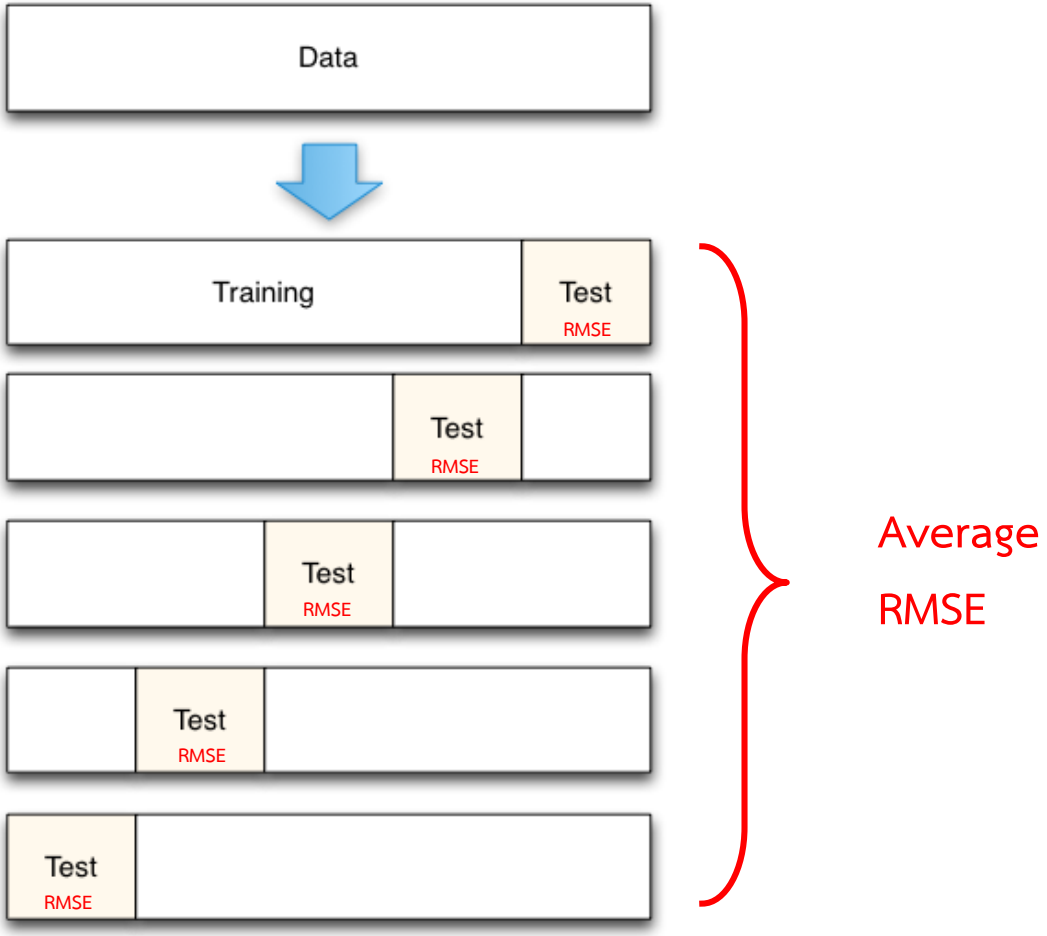
Ref: • (content) https://www.openml.org/a/estimation-procedures/1

Cross-Validation



Ref: • (image) http://blog.kaggle.com/2015/06/29/scikit-learn-video-7-optimizing-your-model-with-cross-validation/

K-Fold Cross-Validation



Ref: • (image) http://blog.kaggle.com/2015/06/29/scikit-learn-video-7-optimizing-your-model-with-cross-validation/

Python Code

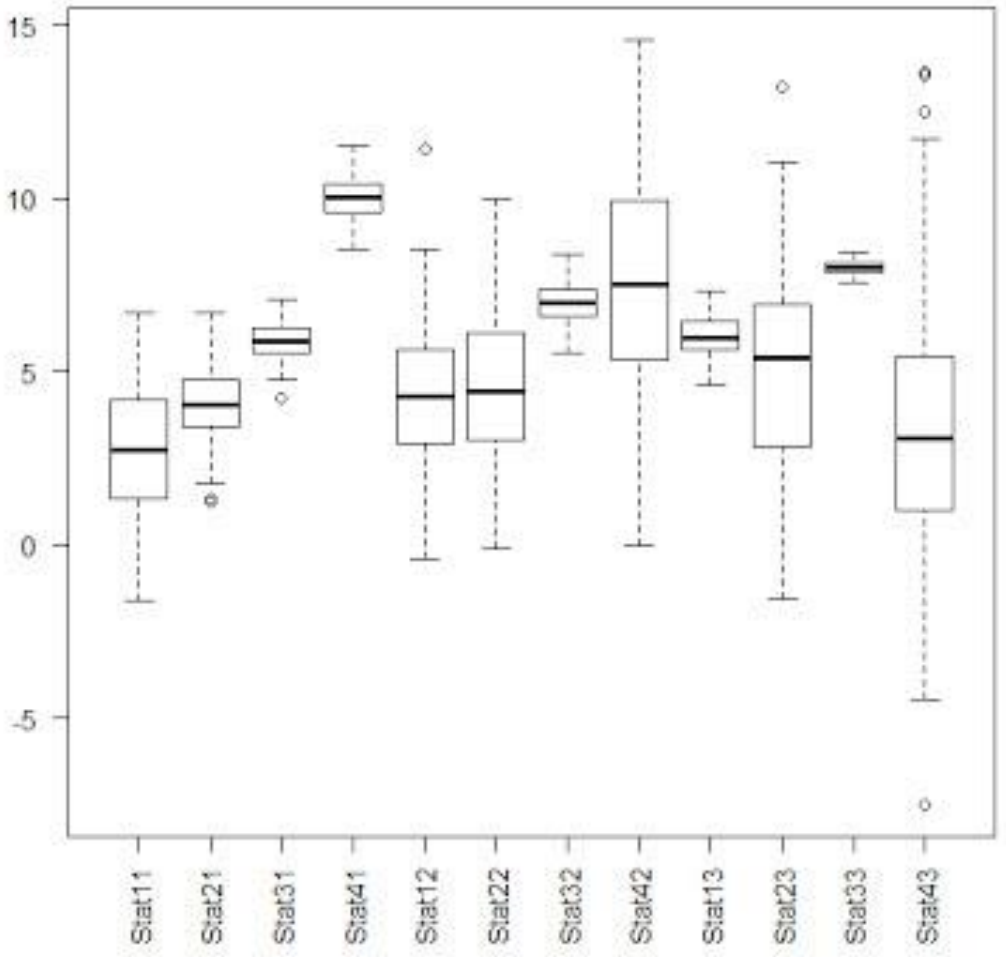
```
1. from sklearn import neighbors, datasets, preprocessing
2. from sklearn.model_selection import train_test_split
3. iris = datasets.load_iris()
4. X, y = iris.data[:, :2], iris.target
5. X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
6. lm = linear_model.LinearRegression()
7. lm.fit (X_train, y_train)
8. y_predicted = lm.predict(X_test)
9. rmse = sqrt(mean_squared_error(y_test, y_predicted))
```

Ref: • https://www.datacamp.com/community/blog/scikit-learn-cheat-sheet

Feature Scaling

Feature Scaling

- Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.



Ref: • (content) https://en.wikipedia.org/wiki/Feature_scaling
• (image) https://www.r-bloggers.com/box-plot-with-r-tutorial/

Rescaling

- The simplest method is rescaling the range of features to scale the range in [0, 1] or [−1, 1]. Selecting the target range depends on the nature of the data. The general formula is given as:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- where x is an original value x' is the normalized value. For example, suppose that we have the students' weight data, and the students' weights span . To rescale this data, we first subtract 160 from each student's weight and divide the result by 40 (the difference between the maximum and minimum weights).

Ref: • https://en.wikipedia.org/wiki/Feature_scaling

Mean Normalization

$$x' = \frac{x - \text{mean}(x)}{\max(x) - \min(x)}$$

- where x is an original value, x' is the normalized value.

Ref: • https://en.wikipedia.org/wiki/Feature_scaling

Standardization

- In machine learning, we can handle various types of data, e.g. audio signals and pixel values for image data, and this data can include multiple dimensions. Feature standardization makes the values of each feature in the data have zero-mean (when subtracting the mean in the numerator) and unit-variance.

$$x' = \frac{x - \bar{x}}{\sigma}$$

- Where x is the original feature vector, x-bar is the mean of that feature vector, and sigma is its standard deviation.

Ref: • https://en.wikipedia.org/wiki/Feature_scaling

Scaling to unit length

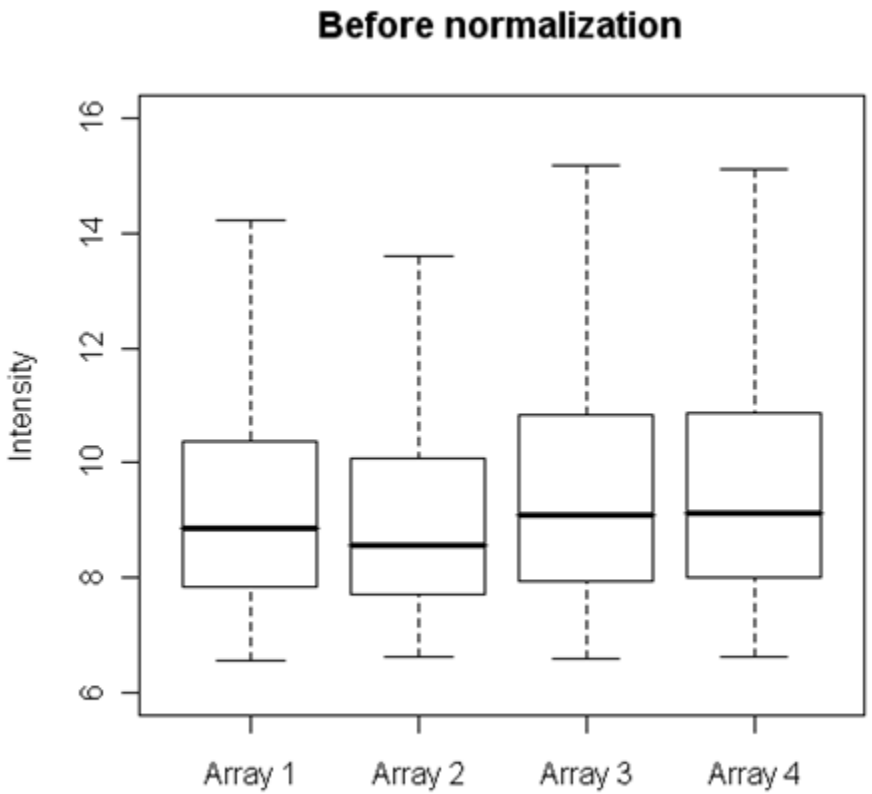
- Another option that is widely used in machine-learning is to scale the components of a feature vector such that the complete vector has length one. This usually means dividing each component by the Euclidean length of the vector:

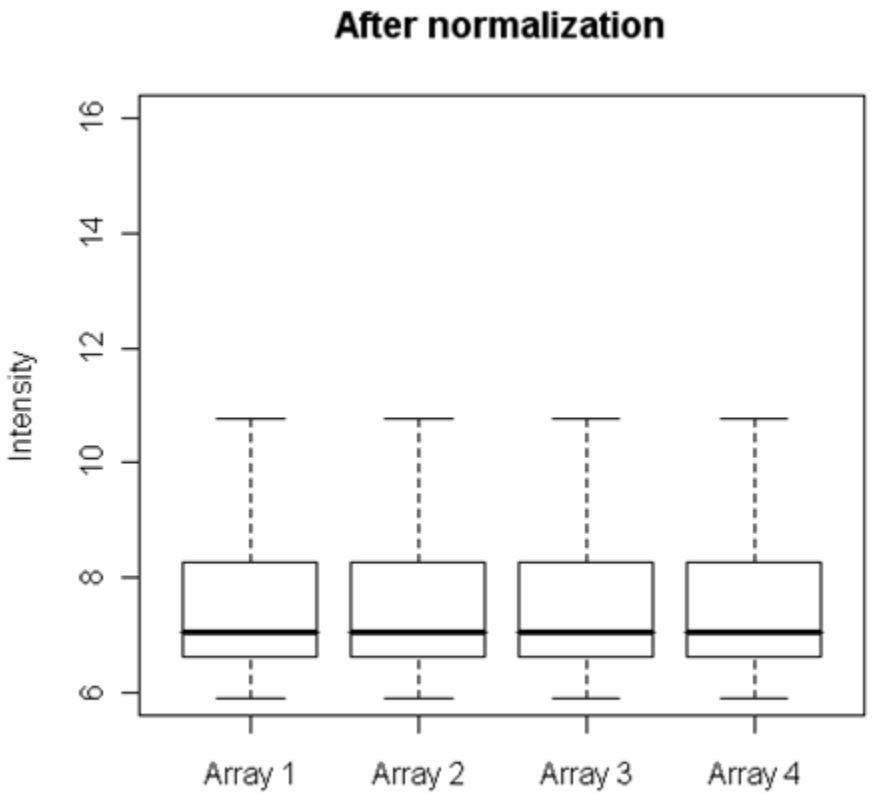
$$x' = \frac{x}{||x||}$$

- In some applications (e.g. Histogram features) it can be more practical to use the L1 norm (i.e. Manhattan Distance, City-Block Length or Taxicab Geometry) of the feature vector. This is especially important if in the following learning steps the Scalar Metric is used as a distance measure.

Ref: • https://en.wikipedia.org/wiki/Feature_scaling

Box Plot





Ref: • (image) http://telemedicina.med.muni.cz/genomic-proteomic-analysis/index-en.php?pg=E-learning--2-data-analysis--2-2-analysis-of-genomic-data--2-2-1-DNA-microarrays-1--2-2-1-6-normalization

Python Code

```
1.  from sklearn import neighbors, datasets, preprocessing
2.  from sklearn.model_selection import train_test_split
3.  from sklearn.metrics import accuracy_score
4.  iris = datasets.load_iris()
5.  X, y = iris.data[:, :2], iris.target
6.  X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
7.  scaler = preprocessing.StandardScaler().fit( X_train )
8.  X_train = scaler.transform(X_train)
9.  X_test = scaler.transform(X_test)
10. # Do Prediction
```

Ref: • https://www.datacamp.com/community/blog/scikit-learn-cheat-sheet

“

Data is a precious thing and will last longer than the systems themselves.

”

Tim Berners-Lee