Linear Regression

Dr. Ram Prasad K
VisionCog R&D

ram.krish@visioncog.com
https://www.visioncog.com

Machine Learning

Machine Learning:

An algorithmic way of *making sense (learning) from* ***data***.

Applications:

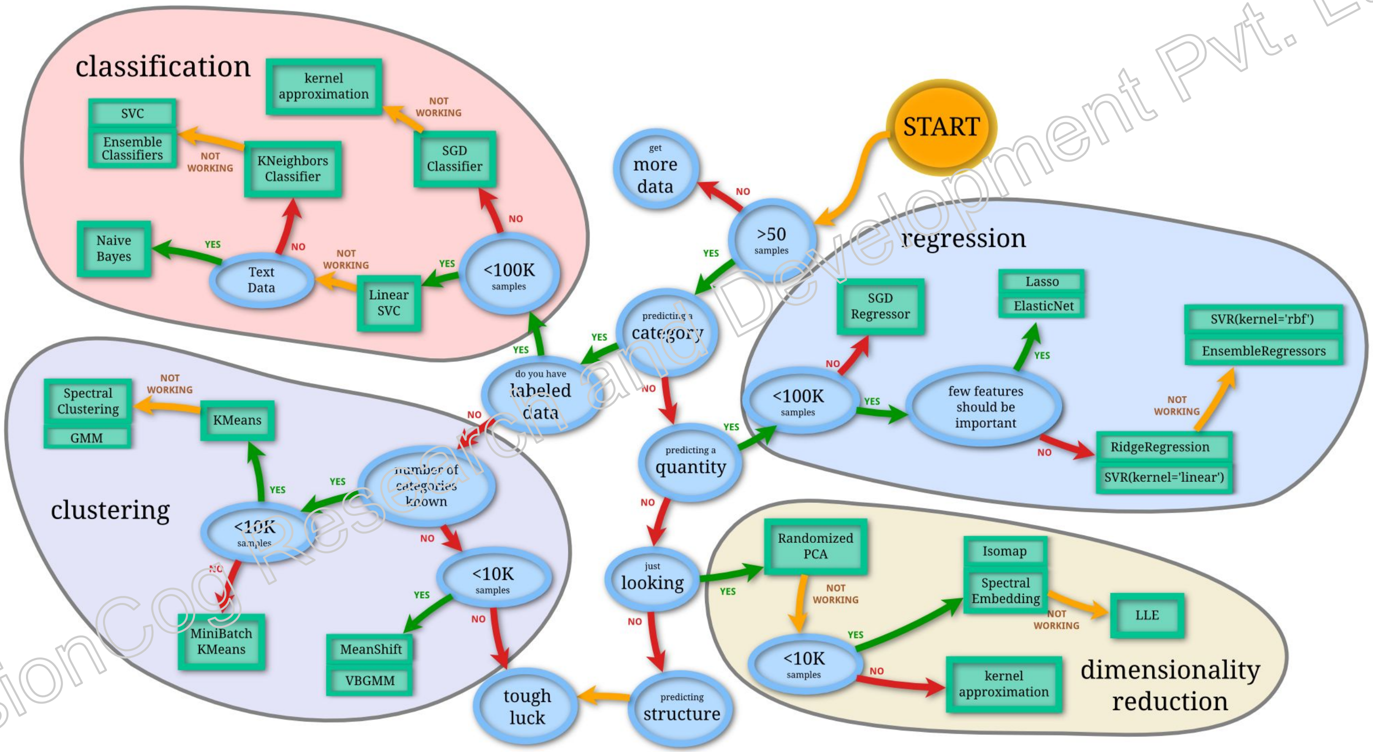
- Spam filters (***Classification***)
- Predict height based on weight and age (***Regression***)
- Online recommendation systems (***Clustering***)
- Visualizing multidimensional data (***Dimensionality reduction***)

Machine Learning

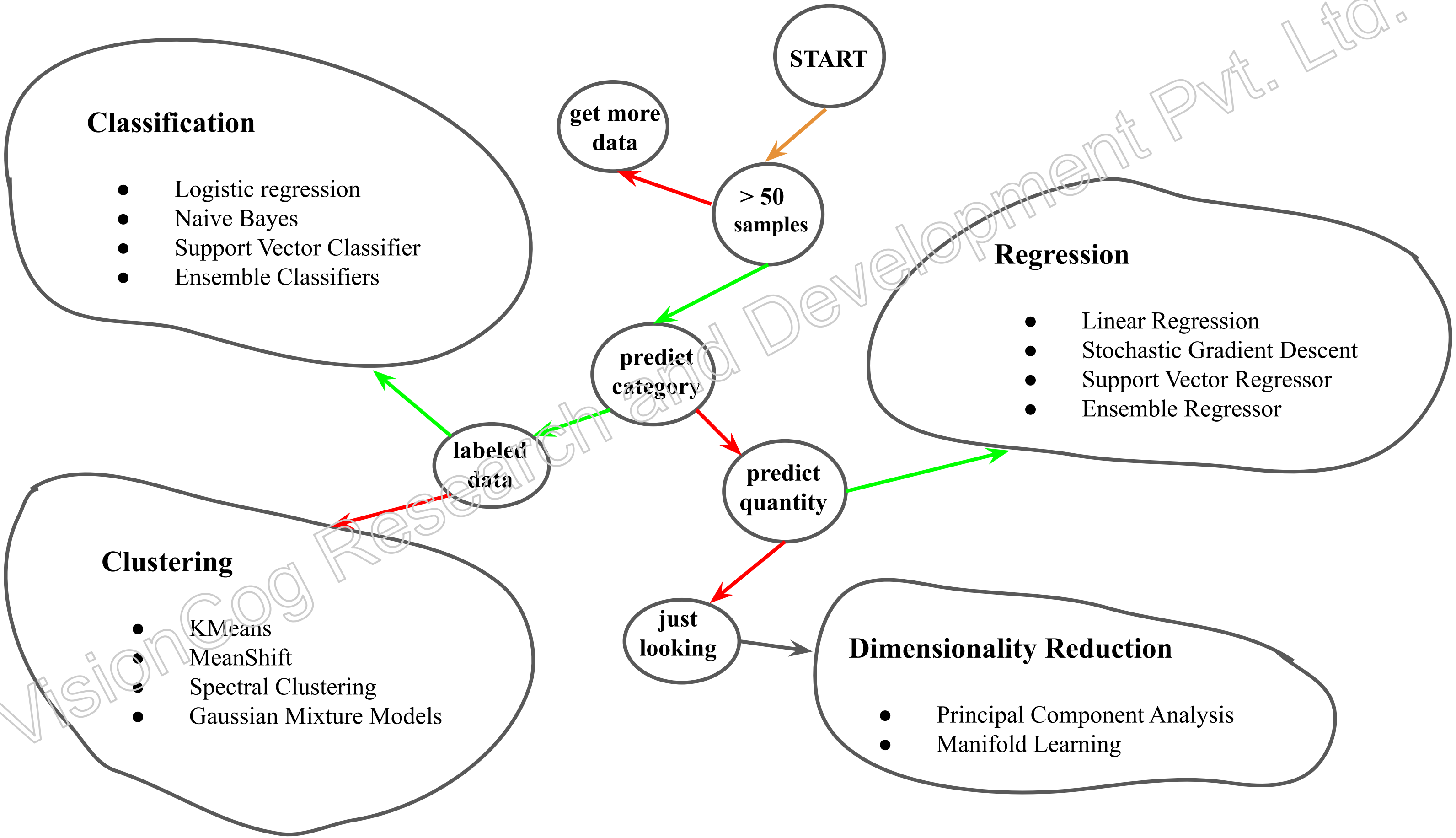
Scikit Learn

- Machine Learning library - in **Python**
- Simple and efficient tools for data analysis
- Built on NumPy, SciPy, and matplotlib
- API is remarkably well designed

Machine Learning



MACHINE LEARNING



Linear Regression

Dependent and Independent variable

Expression	Independent	Dependent
$y = 3 + 2x$	x	y
$y = x^2 - 2x$	x	y
$z = 5x^2 + 8y^3$	x, y	z

Regression:

Modeling a relationship between ***dependent*** and ***independent*** variables for ***prediction***.

Linear Regression

Simple Linear Regression or ***Univariate Linear Regression***.

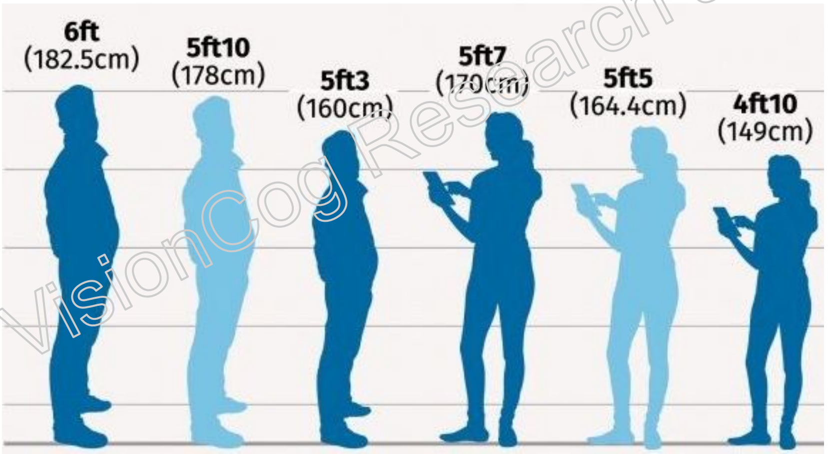
Only one independent variable

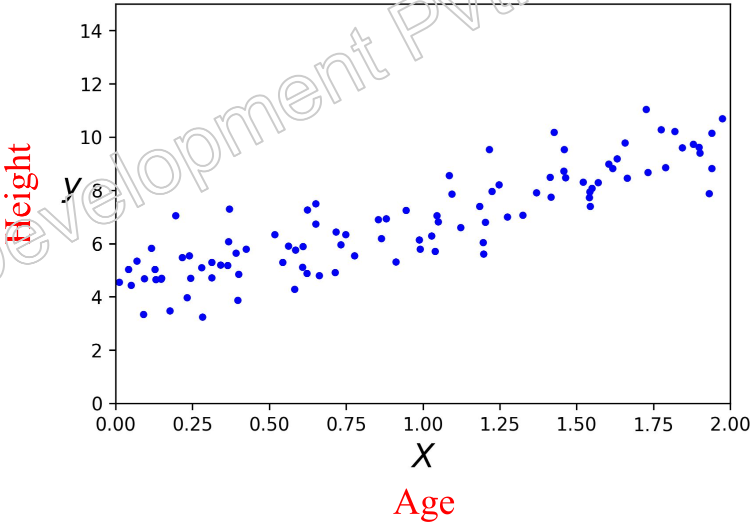
Multiple Linear Regression or ***Multivariate Linear Regression***.

More than one independent variable

Linear Regression

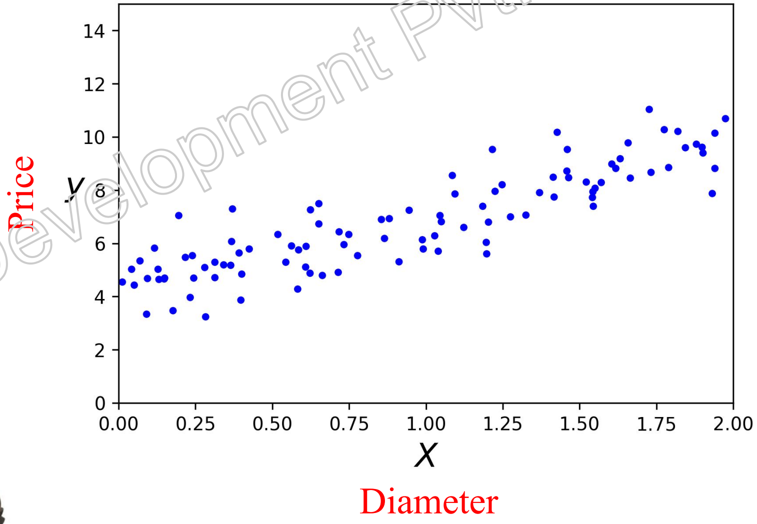
```python
import numpy as np
np.random.seed(42)

X = 2 * np.random.rand(100, 1)

y = 4 + 3 * X + np.random.randn(100, 1)
```





Linear Regression

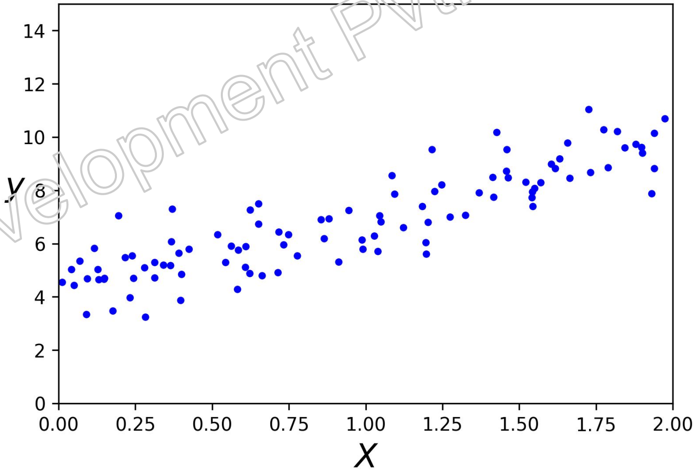
```
import numpy as np
np.random.seed(42)

X = 2 * np.random.rand(100, 1)

y = 4 + 3 * X + np.random.randn(100, 1)
```



Linear Regression

```
import numpy as np
np.random.seed(42)

X = 2 * np.random.rand(100, 1)

y = 4 + 3 * X + np.random.randn(100, 1)
```

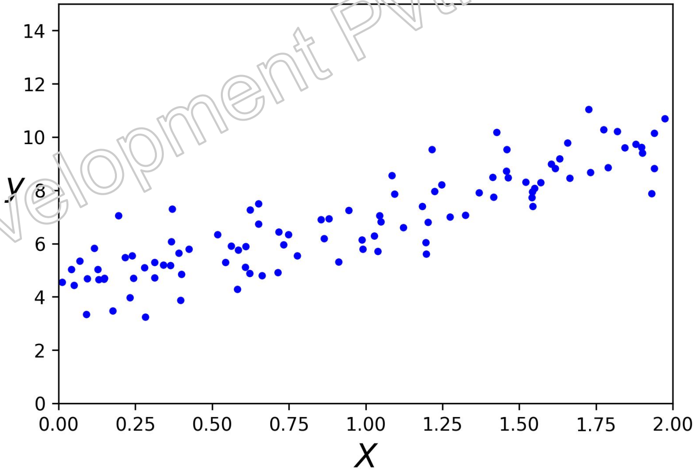
Now assume you don't know how **y** was calculated from **X**



Linear Regression

Assume you now only have **X** and **y**.

Split the dataset into training and test set.

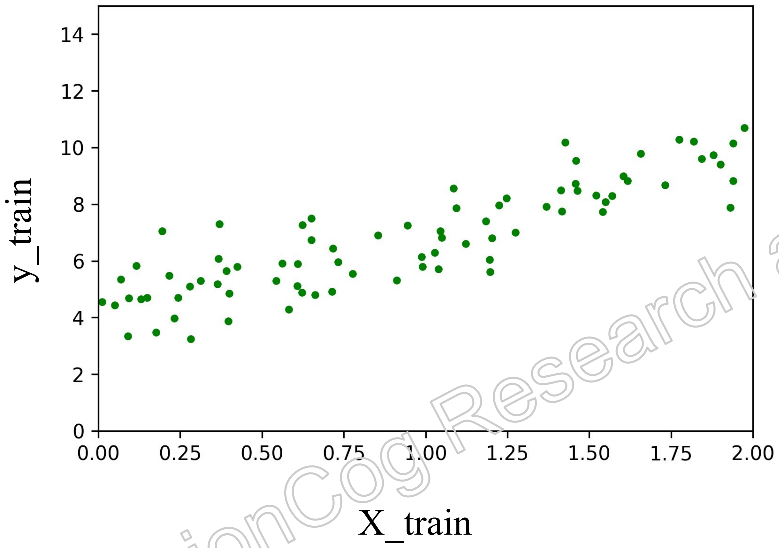


Now assume you don't know how **y** was calculated from **X**

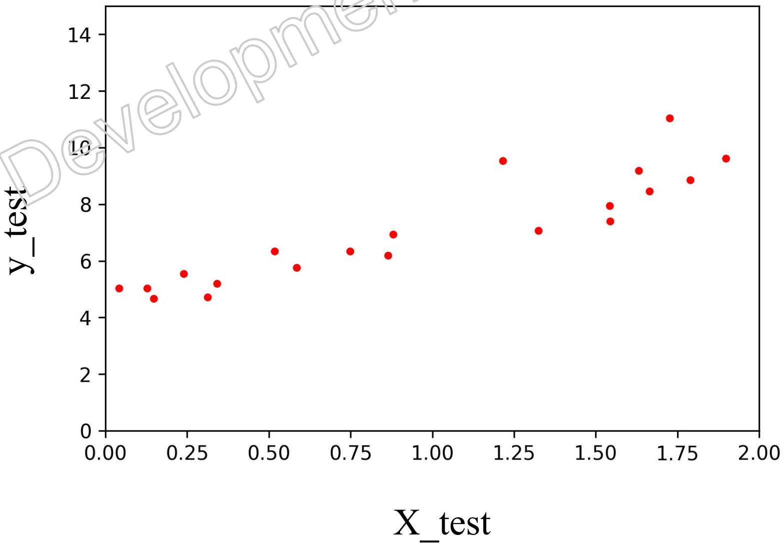
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.20, random_state = 42)
```

Linear Regression

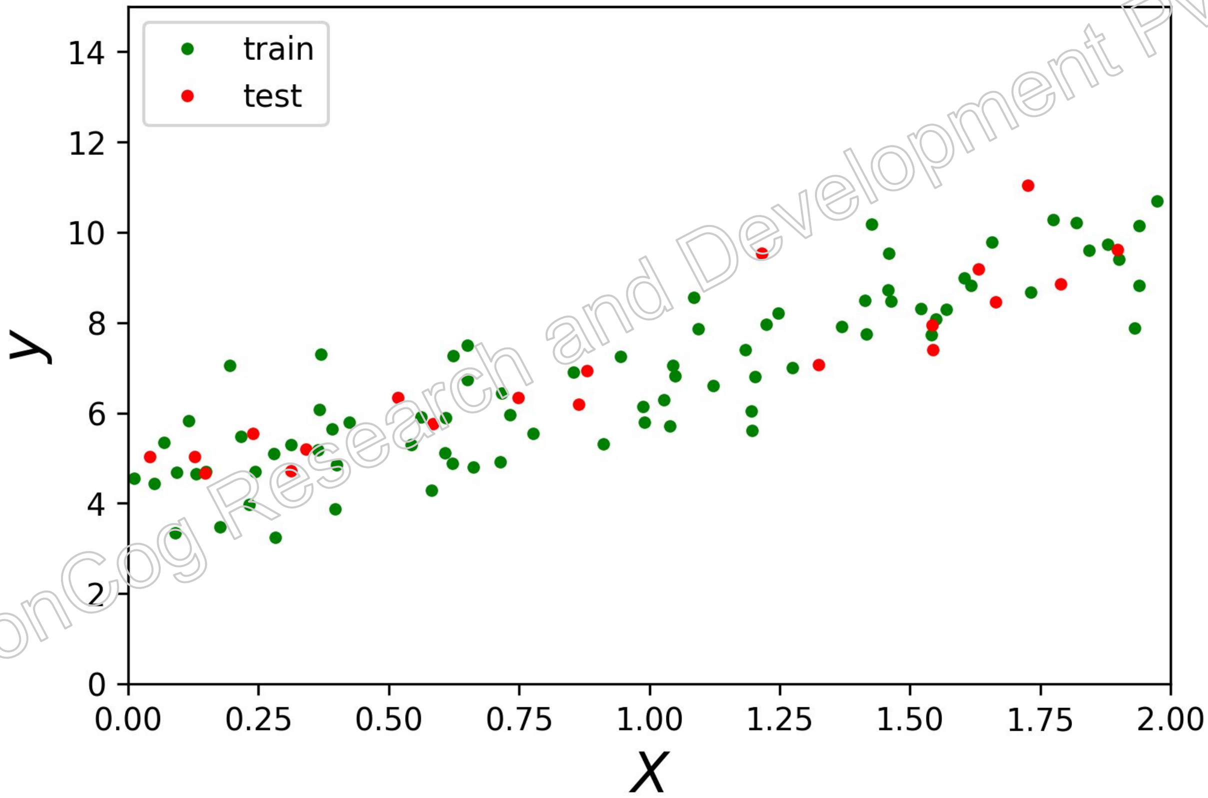
80% training



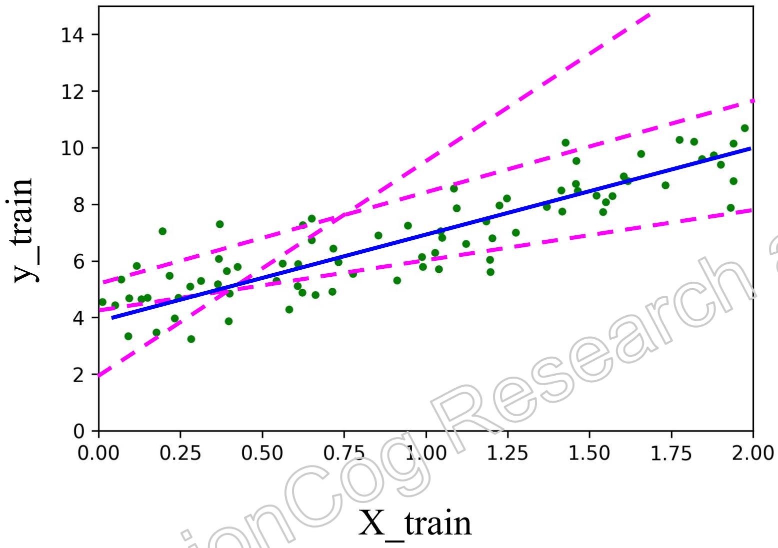
20% testing

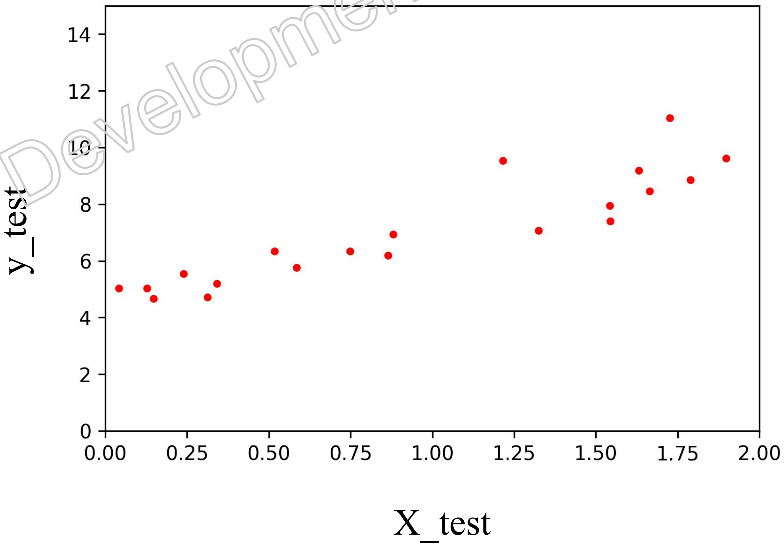


Linear Regression

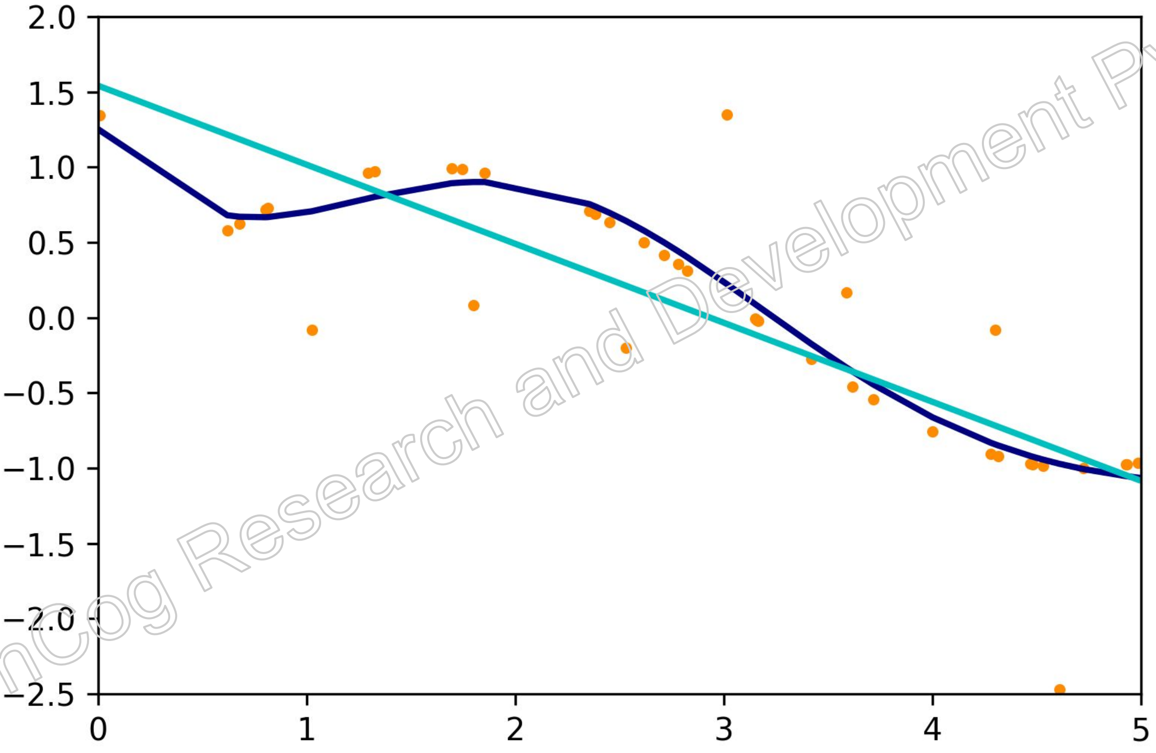


Linear Regression

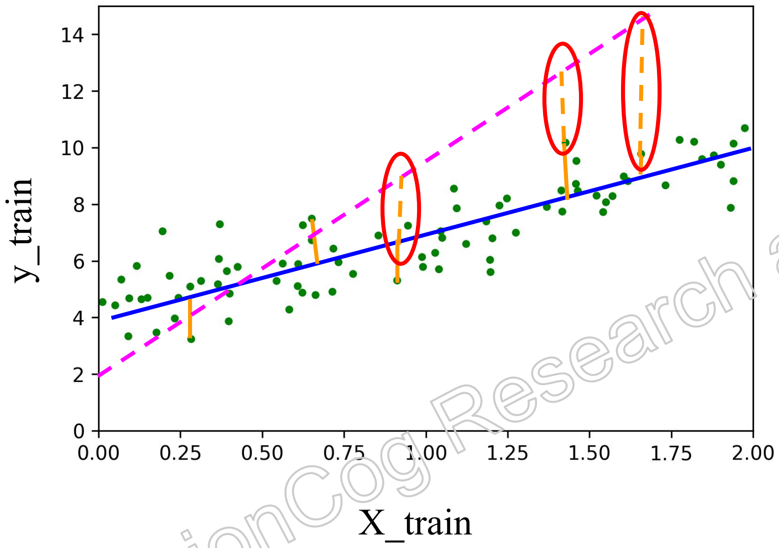


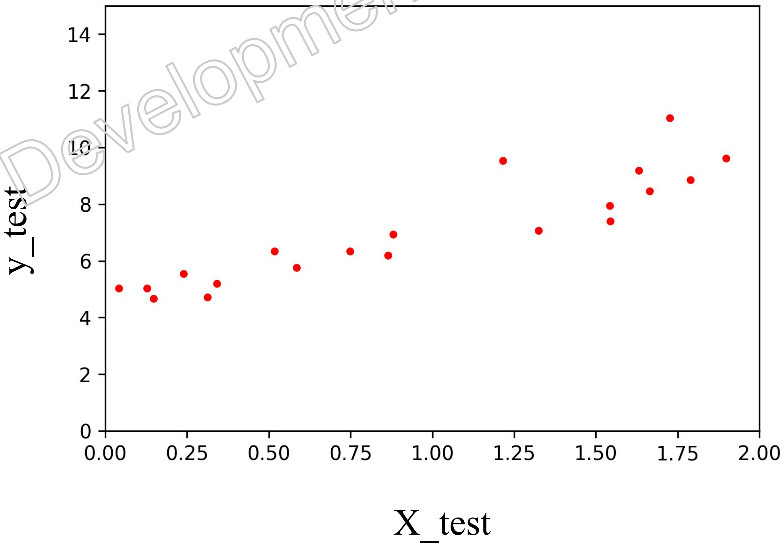


Linear Regression

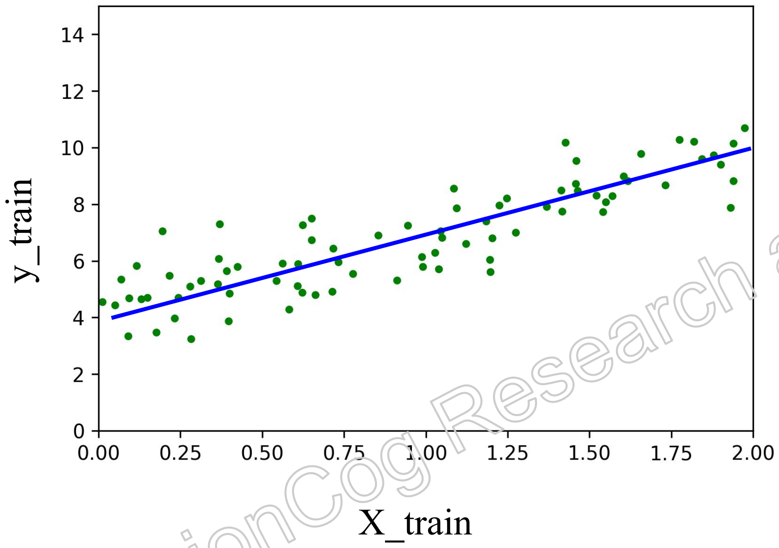


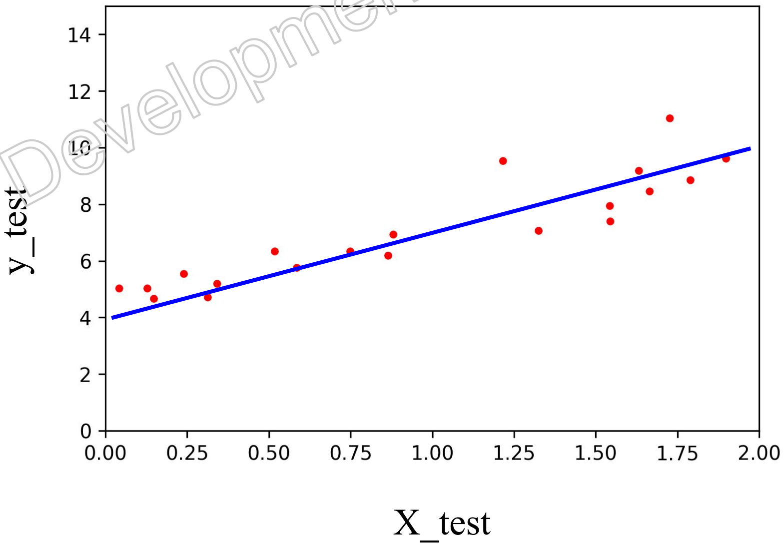
Linear Regression





Linear Regression





Linear Regression

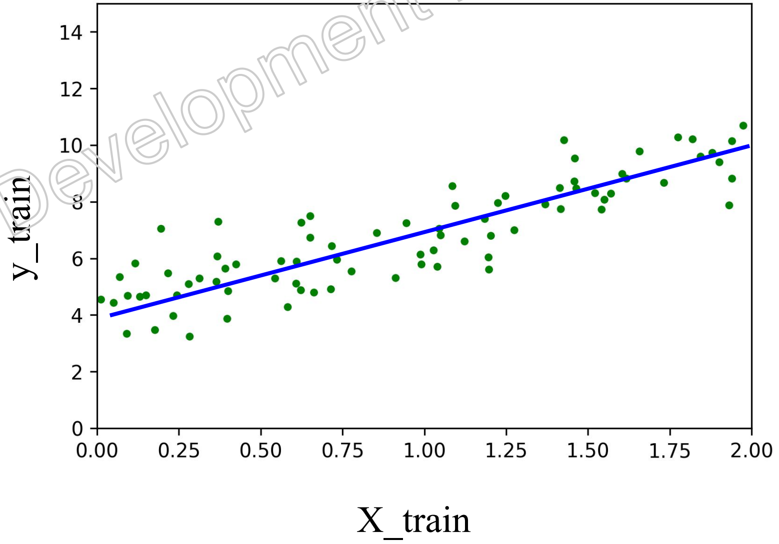
We can obtain here a **straight line** which passes close to as many points as possible.

What parameters are required to represent a straight line?

y-intercept
slope

Equation of a straight line:

$$y = c + mx$$



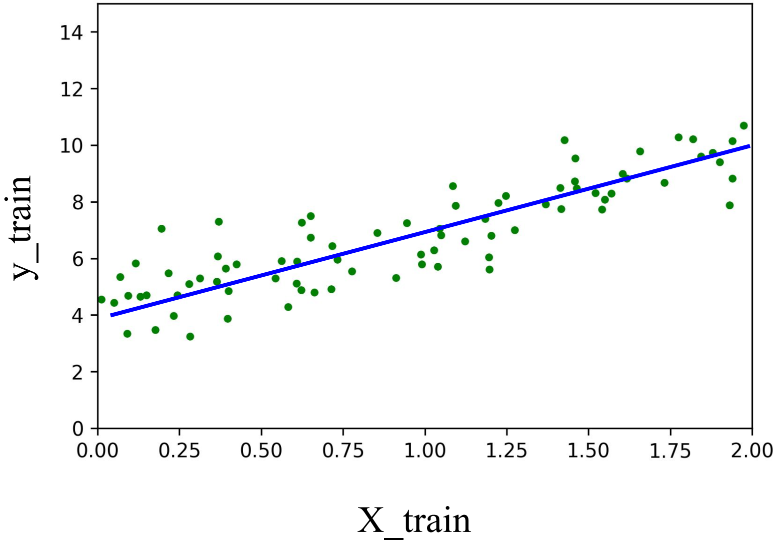
Linear Regression

Mathematical model for Simple Linear Regress

model parameters

$$\hat{y} = \theta_0 + \theta_1 x$$

intercept coefficient



The line models the relationship between cake independent and dependent variable.

Linear Regression

General/Multiple Linear Regression

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

$\hat{y}$ is the predicted value

n is the number of features

x_i is the i^{th} feature value

θ_j is the j^{th} model parameter

θ_0 is the intercept (also called ***bias*** term)

Linear Regression

Vectorized general form

$$\hat{y} = h_\theta(\mathbf{x}) = \theta^T \cdot \mathbf{x}$$

θ is the models ***parameter*** vector

θ_0 is the *bias/intercept*

$\theta_1, \theta_2, \ldots, \theta_n$ are ***coefficients*** or feature weights.

$\mathbf{x}$ is the ***feature*** vector x_0 to x_n with x_0 always 1

$\theta^T \cdot \mathbf{x}$ is the dot product of θ^T and $\mathbf{x}$

h_θ is the ***hypothesis*** function using model parameters θ

Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta^T \cdot \mathbf{x}$$

$$\text{MSE}(\mathbf{X}, h_{\boldsymbol{\theta}}) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2$$

cost function

Normal Equation

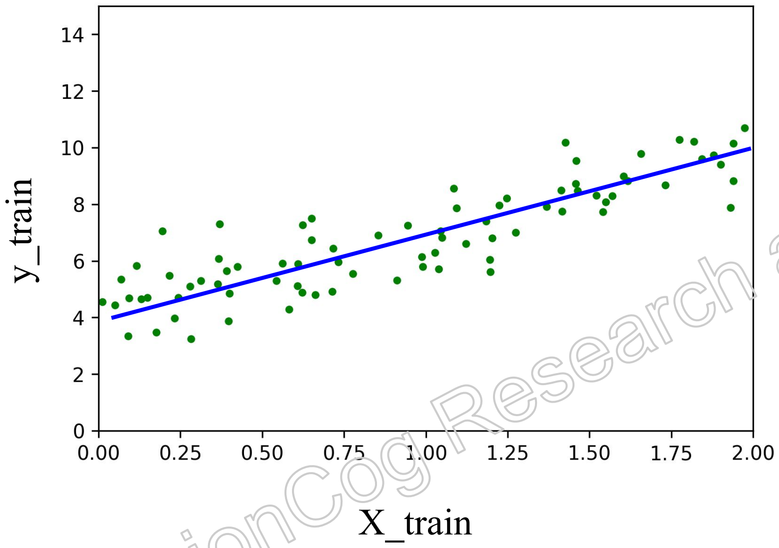
To find the parameters, we have a closed-form solution:

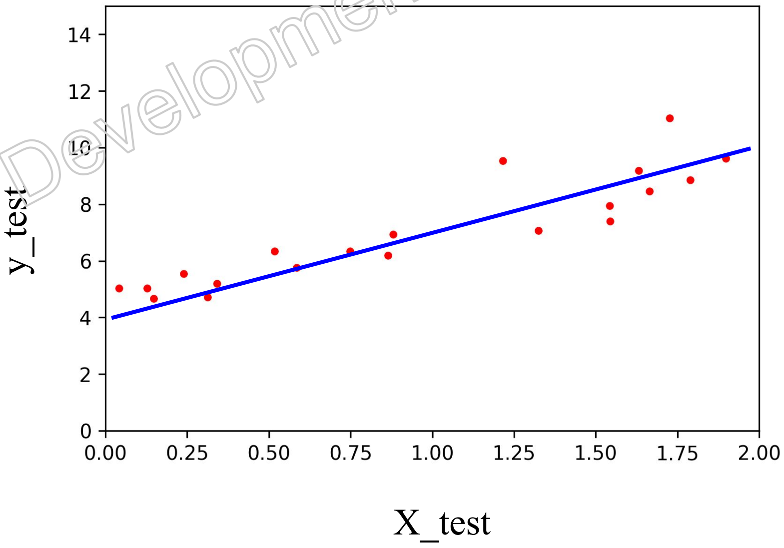
$$\hat{\theta} = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

$\hat{\theta}$ is the value of θ that minimizes the cost function (least squares)

$\mathbf{y}$ is the vector of target values

Linear Regression





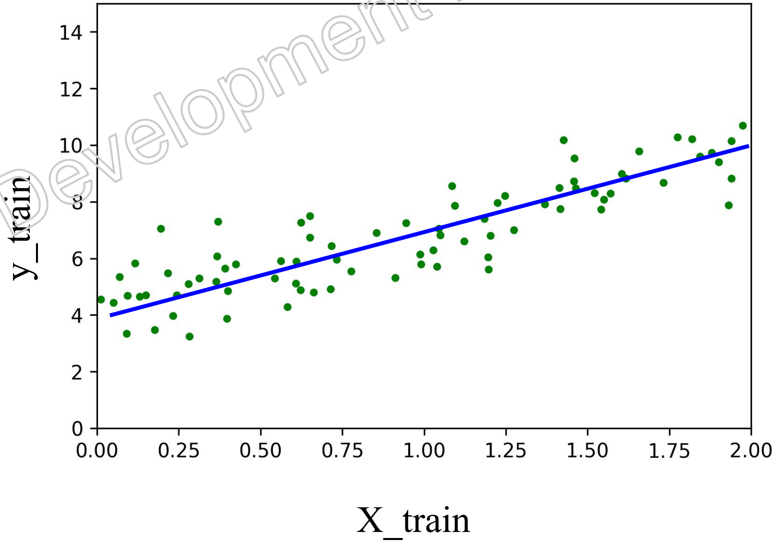
LINEAR REGRESSION

Mathematical model for Simple Linear Regress

model parameters

$$\hat{y} = \theta_0 + \theta_1 x$$

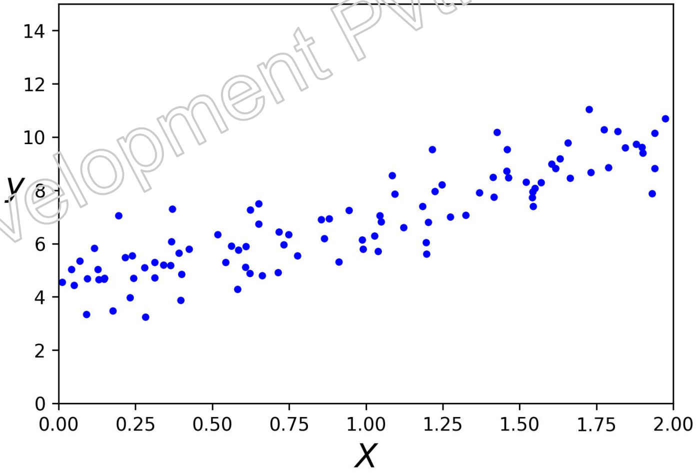
intercept coefficient



The line models the relationship between cake independent and dependent variable.

Linear Regression

```
import numpy as np
np.random.seed(42)

X = 2 * np.random.rand(100, 1)

y = 4 + 3 * X + np.random.randn(100, 1)
```




```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.20, random_state = 42)
```

Linear Regression

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()

model.fit(X_train, y_train)

print(model.intercept_)
print(model.coef_)

# [4.14291332]
# [[2.79932366]]
```

$$y = 4 + 3x + \varepsilon$$

$$y = 4.14 + 2.78x$$

Evaluating the model

Testing error

R^2 / Coefficient of determination / Goodness-of-fit

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

$$SS_{res} = \sum_{i}^{n}(y_i - f(x_i))^2$$ sum of squares explained by model

$$SS_{tot} = \sum_{i=1}^{n}(y_i - \bar{y})^2$$ sum of squares around the mean

Linear Regression

```python
from sklearn.linear_model import LinearRegression

model = LinearRegression()

model.fit(X_train, y_train)

print(model.intercept_)
print(model.coef_)

# [4.14291332]
# [[2.79932366]]

score = model.score(X_test, y_test)
print(score)

# 0.8072059636181392
```

$$y = 4 + 3x + \varepsilon$$

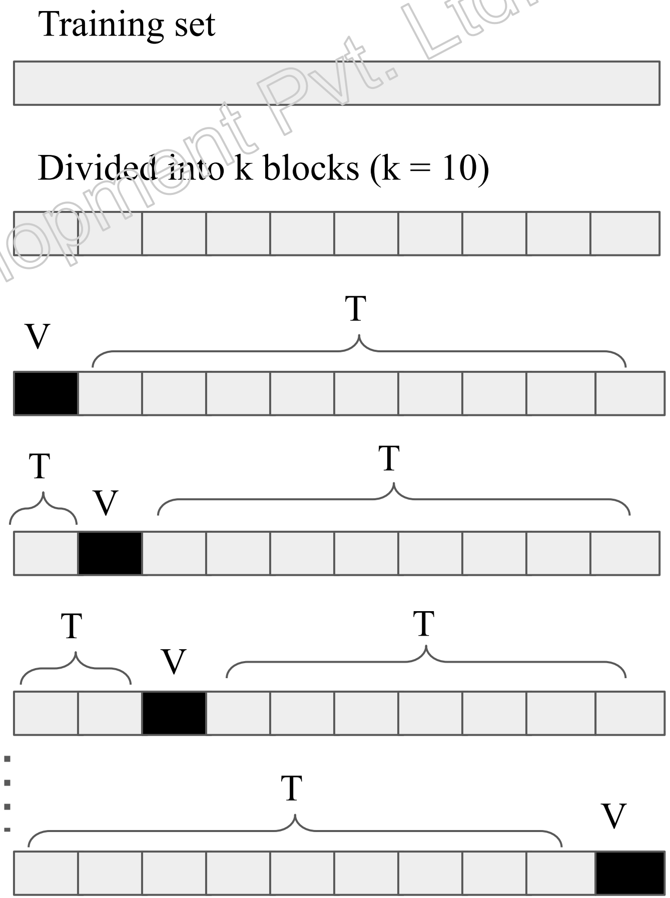
$$y = 4.14 + 2.78x$$

Model Evaluation using Cross-Validation

Linear Regression

Cross-validation

- Training dataset it further divided into training and validation.
- The validation splitting is based on the number of folds.
- k-Fold cross validation divides the training set it k equal blocks.
- In each round:
  - one of the block is used as validation set, and
  - the remaining is used as training set.



Linear Regression

```
from sklearn.model_selection import cross_val_score

scores_CV = cross_val_score(model, X_train, y_train, cv=10)

print(scores_CV)
# [0.88261122 0.83015975 0.5121498  0.57767211 0.21819067 0.80309185
#  0.73759195 0.38341108 0.77308338 0.784341  ]

print(scores_CV.mean())
# 0.6502302819222789

print(scores_CV.std())
# 0.20799865939333312
```

Cross-validated model score: 0.65 +/- 0.21