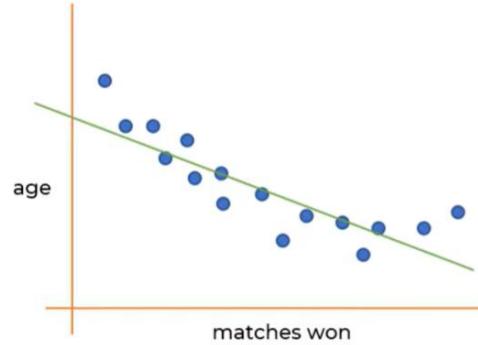


# Ridge and Lasso Regression

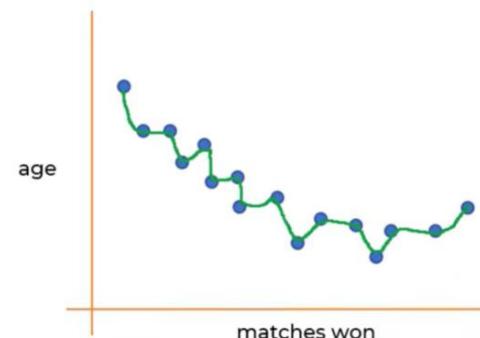
Dr. Dipak Kumar Mohanty  
KIIT University, Bhubaneswr

underfit



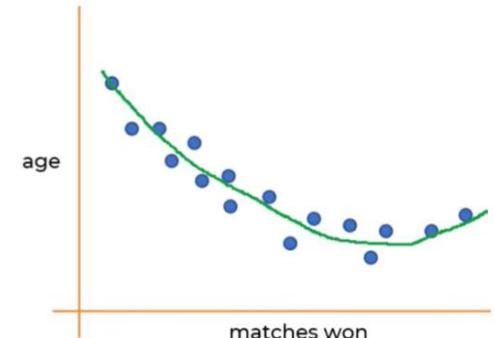
$$\text{match won} = \theta_0 + \theta_1 * \text{age}$$

overfit



$$\begin{aligned}\text{match won} = & \theta_0 + \theta_1 * \text{age} + \theta_2 * \text{age}^2 \\ & + \theta_3 * \text{age}^3 + \theta_4 * \text{age}^4\end{aligned}$$

balanced fit



$$\text{match won} = \theta_0 + \theta_1 * \text{age} + \theta_2 * \text{age}^2$$

# How to reduce overfitting?



$$\text{match won} = \theta_0 + \theta_1 * \text{age} + \theta_2 * \text{age}^2 + \theta_3 * \text{age}^3 + \theta_4 * \text{age}^4$$





$$\text{match won} = \theta_0 + \theta_1 * \text{age} + \theta_2 * \text{age}^2 + \theta_3 * \text{age}^3 + \theta_4 * \text{age}^4$$



Try to make  $\theta_3$  and  $\theta_4$  almost close to zero



$$\text{match won} = \theta_0 + \theta_1 * \text{age} + \theta_2 * \text{age}^2$$



# From Linear Regression we know, MSE...

Mean Squared Error

$$mse = \frac{1}{n} \sum_{i=1}^n (y_i - y_{predicted})^2$$



## Mean Squared Error

$$mse = \frac{1}{n} \sum_{i=1}^n (y_i - h_\theta(x_i))^2$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$



# Regularization

- Regularizations are of two types
  - 1) Ridge Regression or L2- Regularization
  - 2) Lasso Regression or L1- Regularization

# 1) Ridge Regression or L2- Regularization

Ridge Regression

$$R = \text{Loss} + \lambda \|\theta\|_2^2$$

where  $\lambda$  = penalty,

$$\text{where } \|\theta\|_2^2 = \theta_1^2 + \theta_2^2 + \dots + \theta_n^2$$

Here, When  $\theta$  is bigger, Error become bigger, so model will not converge

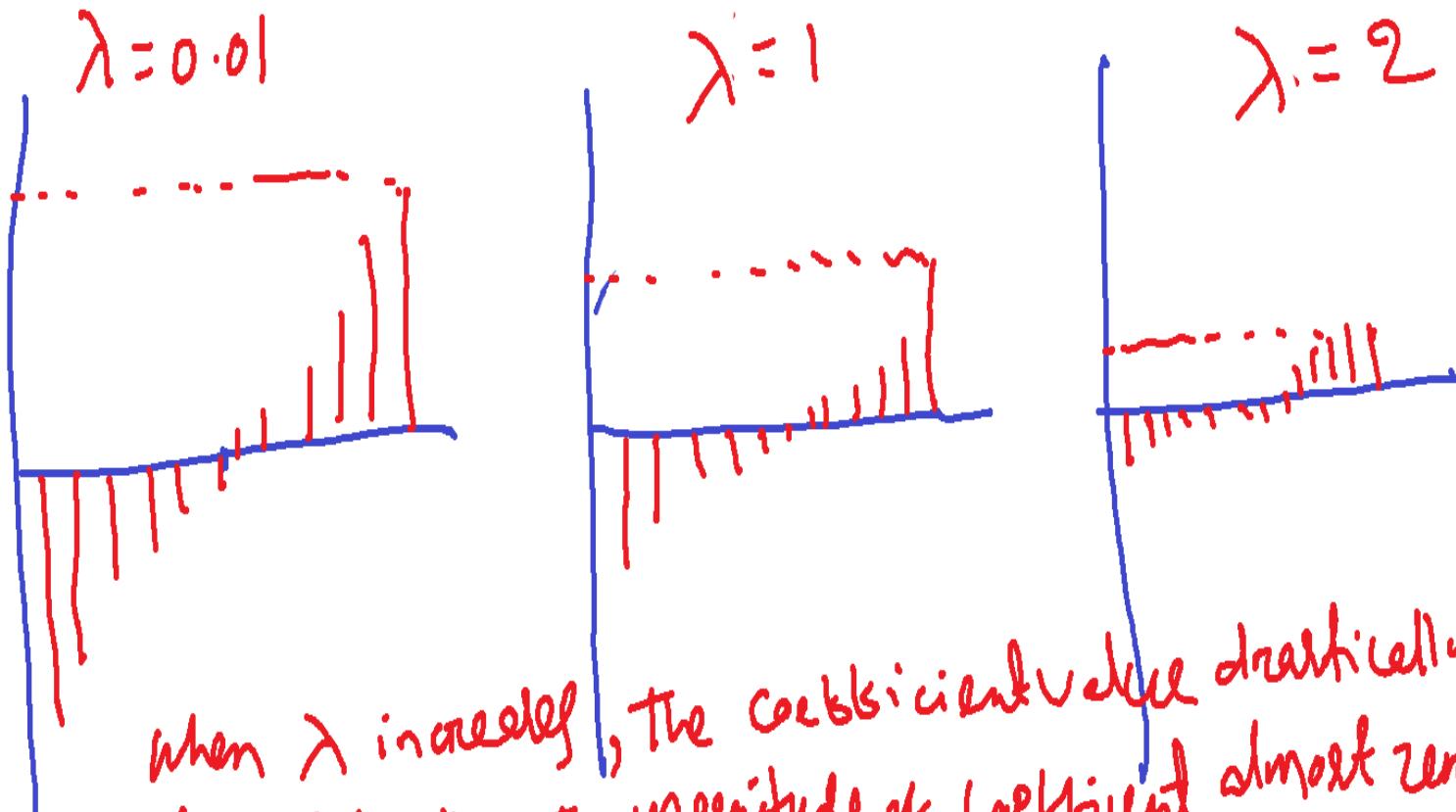
## L2 Regularization

$$mse = \frac{1}{n} \sum_{i=1}^n (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^n \theta_i^2$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$

- Here, Penalty Lambda is used to ensure that  $\theta$  value doesn't go too high.
- When Lambda is bigger,  $\theta$  is smaller and vice versa.
- This approach is called **L2-Regularization**

# Ridge: Example



when  $\lambda$  increases, the coefficient value drastically decreases, here the magnitude of coefficient almost zero.

# L1-Regularization or Lasso Regression

$$\text{Lasso } R = \text{Loss} + \lambda \|\theta\|_1$$

$$\|\theta\|_1 = |\theta_1| + |\theta_2| + \dots + |\theta_n|$$

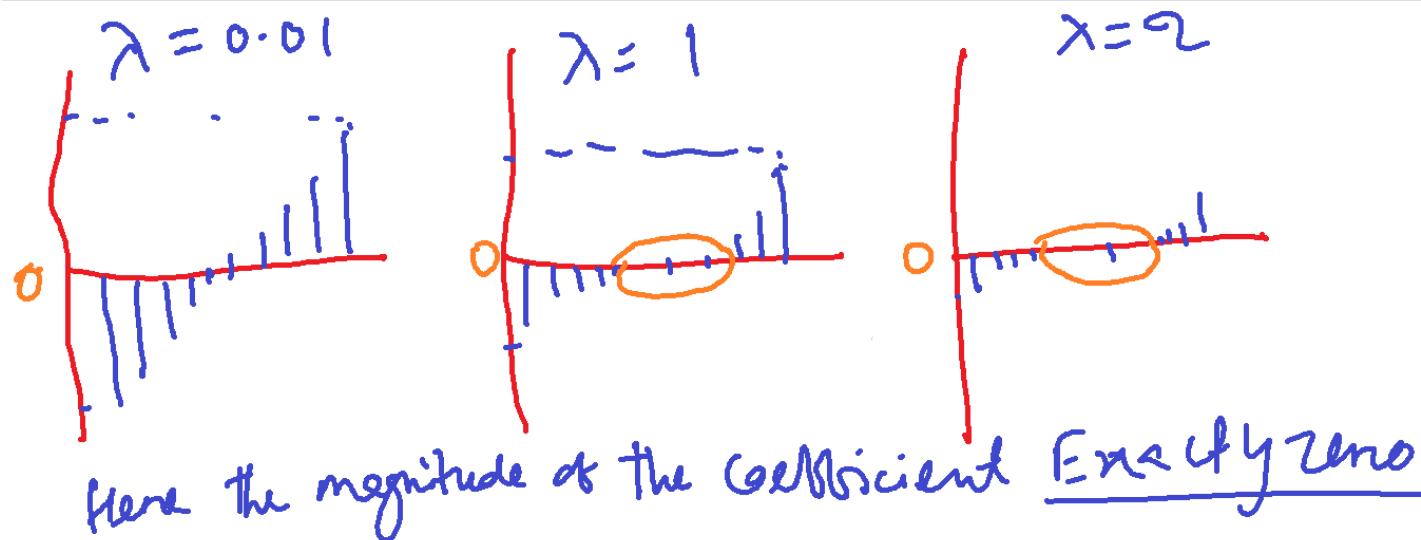
# L1-Regularization

## L1 Regularization

$$mse = \frac{1}{n} \sum_{i=1}^n (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^n |\theta_i|$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$

# Lasso Regression: Example



# Example

Let  $y = \theta_0 + \theta_1 n_1 + \theta_2 n_2 + \theta_3 n_3$

$$y = 0.8 + 1.2n_1 + 30n_2 + 49n_3$$

A New Ridge ↑

$$y = 0.8 + 1.2n_1 + 0.2n_2 + 0.3n_3$$

A Later Lasso

$$y = 0.8 + 1.2n_1 + 0.2n_2 + 0.1n_3$$

ML/16\_regularization/ Untitled - Jupyter Notebook

localhost:8888/notebooks/ML/16\_regularization/Untitled.ipynb?kernel\_name=python3

jupyter Untitled Last Checkpoint: 18 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Rooms Propertycount Distance Bedroom2 Bathroom Car Landsize BuildingArea Price Suburb\_Aberfeldie ... CouncilArea\_Moorabool Shire Council CouncilArea\_Council Area

	Rooms	Propertycount	Distance	Bedroom2	Bathroom	Car	Landsize	BuildingArea	Price	Suburb_Aberfeldie	...	CouncilArea_Moorabool	Shire Council	CouncilArea_Council Area
1	2	4019.0	2.5	2.0	1.0	1.0	202.0	160.2564	1480000.0	0	...	0	0	0
2	2	4019.0	2.5	2.0	1.0	0.0	156.0	79.0000	1035000.0	0	...	0	0	0
4	3	4019.0	2.5	3.0	2.0	0.0	134.0	150.0000	1465000.0	0	...	0	0	0
5	3	4019.0	2.5	3.0	2.0	1.0	94.0	160.2564	850000.0	0	...	0	0	0
6	4	4019.0	2.5	3.0	1.0	2.0	120.0	142.0000	1600000.0	0	...	0	0	0

5 rows × 745 columns

In [14]: X = dataset.drop('Price', axis=1)  
y = dataset['Price']

In [15]: from sklearn.model\_selection import train\_test\_split  
train\_X, test\_X, train\_y, test\_y = train\_test\_split(X, y, test\_size=0.3, random\_state=2)

In [ ]:

Type here to search

CODE BASICS

# reg: Regular Linear Regression

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** "Untitled - Jupyter Notebook" and "localhost:8888/notebooks/ML/16\_regularization/Untitled.ipynb?kernel\_name=python3".
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Help, Run, Code, Logout.
- Data Preview:** A preview of a dataset with 5 rows and 745 columns. The first two rows are shown:

	5	3	4019.0	2.5	3.0	2.0	1.0	94.0	160.2564	850000.0	0	...	0
6	4	4019.0	2.5	3.0	1.0	2.0	120.0	142.0000	1600000.0	0	...	0	

5 rows × 745 columns

- Code Cells:**
  - In [14]: `X = dataset.drop('Price', axis=1)`  
`y = dataset['Price']`
  - In [15]: `from sklearn.model_selection import train_test_split`  
`train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3, random_state=2)`
  - In [16]: `from sklearn.linear_model import LinearRegression`  
`reg = LinearRegression().fit(train_X, train_y)`
  - In [17]: `reg.score(test_X, test_y)`
- Output:** Out[17]: 0.13853683161500907
- Input Cell:** In [ ]: (empty)

# Overfitting case: train accu=0.68, test accu= 0.13,

The screenshot shows a Jupyter Notebook interface running on a Windows operating system. The notebook has two tabs open: 'ML/16\_regularization/' and 'Untitled - Jupyter Notebook'. The 'Untitled' tab is active and displays the following code and output:

```
In [14]: X = dataset.drop('Price', axis=1)
y = dataset['Price']

In [15]: from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3, random_state=2)

In [16]: from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(train_X, train_y)

In [17]: reg.score(test_X, test_y)
Out[17]: 0.13853683161500907

In [18]: reg.score(train_X, train_y)
Out[18]: 0.6827792395792723
```

The notebook interface includes a toolbar with various icons for file operations, a code editor with a dropdown menu, and a status bar at the bottom.

# Lasso: L1 regularization

scikit learn Install User Guide API Examples More ▾

Prev Up Next

scikit-learn 0.23.2 Other versions

Please [cite us](#) if you use the software.

[sklearn.linear\\_model.Lasso](#)  
Examples using [sklearn.linear\\_model.Lasso](#)

## sklearn.linear\_model.Lasso

```
class sklearn.linear_model.Lasso(alpha=1.0, *, fit_intercept=True, normalize=False, precompute=False, copy_X=True, max_iter=1000, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic')
```

[source]

Linear Model trained with L1 prior as regularizer (aka the Lasso)

The optimization objective for Lasso is:

```
(1 / (2 * n_samples)) * ||y - Xw||^2_2 + alpha * ||w||_1
```

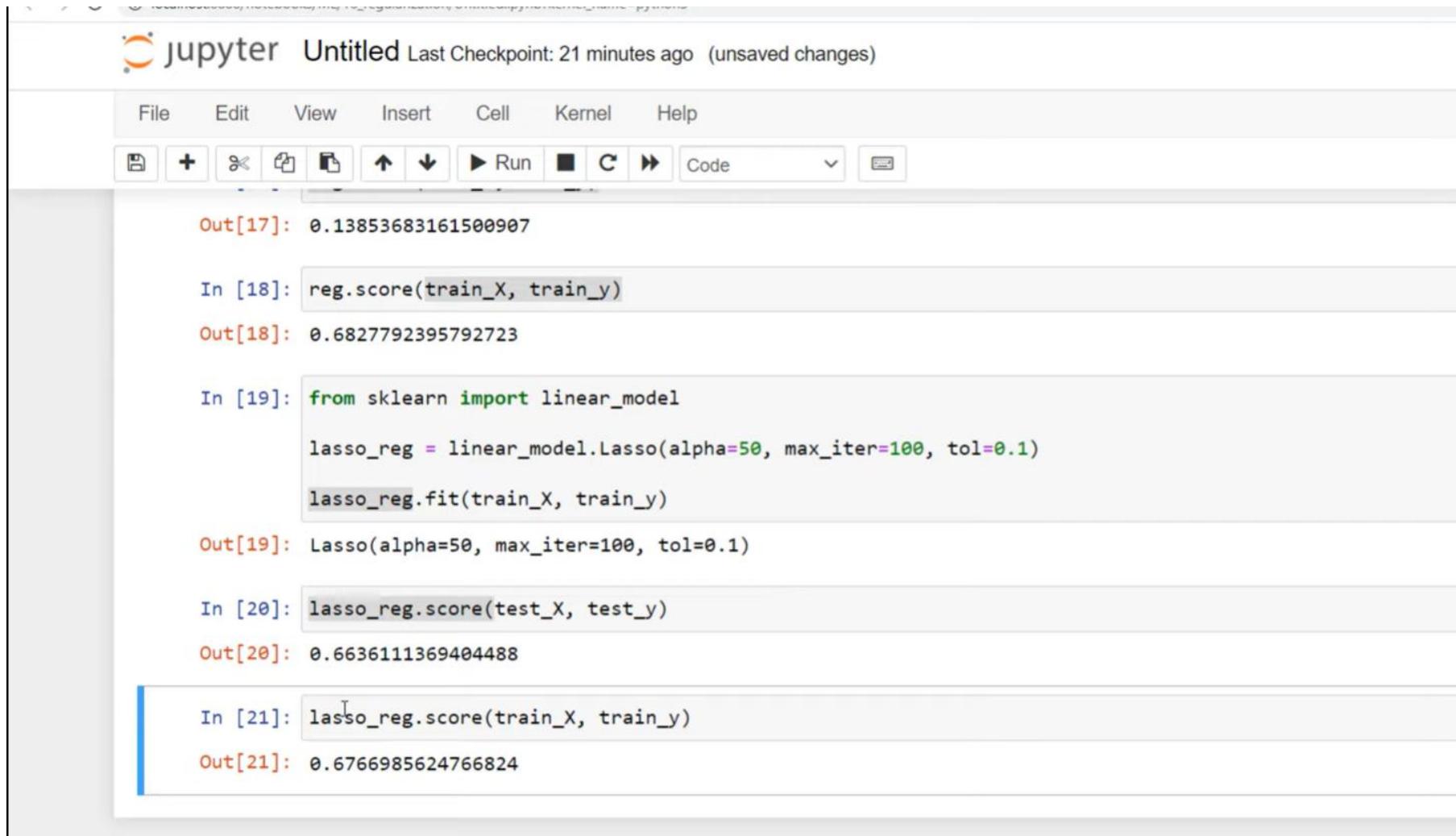
Technically the Lasso model is optimizing the same objective function as the Elastic Net with `l1_ratio=1.0` (no L2 penalty).

Read more in the [User Guide](#).

**Parameters:** `alpha : float, default=1.0`

Constant that multiplies the L1 term. Defaults to 1.0. `alpha = 0` is equivalent to an ordinary least square, solved by the [LinearRegression](#) object. For numerical reasons,

# Result in Lasso: Good with 66%,67% accuracy



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter Untitled Last Checkpoint: 21 minutes ago (unsaved changes)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Help, and various icons for file operations.
- Code Cells:**
  - In [17]: `reg.score(train_X, train_y)`
  - Out[17]: `0.13853683161500907`
  - In [18]: `reg.score(train_X, train_y)`
  - Out[18]: `0.6827792395792723`
  - In [19]:

```
from sklearn import linear_model
lasso_reg = linear_model.Lasso(alpha=50, max_iter=100, tol=0.1)
lasso_reg.fit(train_X, train_y)
```
  - Out[19]: `Lasso(alpha=50, max_iter=100, tol=0.1)`
  - In [20]: `lasso_reg.score(test_X, test_y)`
  - Out[20]: `0.6636111369404488`
  - In [21]: `lasso_reg.score(train_X, train_y)`
  - Out[21]: `0.6766985624766824`

# Ridge Regression: L2 Regularization

The screenshot shows a Jupyter Notebook interface with several tabs at the top: 'ML/16\_regularization/' (active), 'Untitled - Jupyter Notebook', and 'sklearn.linear\_model.Ridge — scikit-learn.org' (selected). The main content area displays the scikit-learn documentation for the `sklearn.linear_model.Ridge` class.

**scikit-learn 0.23.2** (Other versions)

Please [cite us](#) if you use the software.

`sklearn.linear_model.Ridge`

Examples using `sklearn.linear_model.Ridge`

**sklearn.linear\_model.Ridge**

Linear least squares with L2 regularization. ↗

Minimizes the objective function:

$$\|y - Xw\|^2_2 + \alpha * \|w\|^2_2$$

This model solves a regression model where the loss function is the linear least squares function and regularization is given by the L2-norm. Also known as Ridge Regression or Tikhonov regularization. This estimator has built-in support for multi-variate regression (i.e., when y is a 2d-array of shape (n\_samples, n\_targets)).

Read more in the [User Guide](#).

**Parameters:** `alpha : {float, ndarray of shape (n_targets,)}, default=1.0`  
Regularization strength; must be a positive float. Regularization improves the conditioning of the problem and reduces the variance of the estimates. Larger values

Toggle Menu

Type here to search

# Result in Ridge: Good with 66%,66% accuracy

The screenshot shows a Jupyter Notebook interface with several code cells and their corresponding outputs. The toolbar at the top includes icons for file operations, run, and code cell selection.

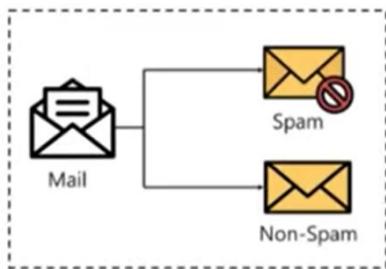
- In [20]:** lasso\_reg.score(test\_X, test\_y)  
**Out[20]:** 0.6636111369404488
- In [21]:** lasso\_reg.score(train\_X, train\_y)  
**Out[21]:** 0.6766985624766824
- In [22]:** from sklearn.linear\_model import Ridge  
ridge\_reg= Ridge(alpha=50, max\_iter=100, tol=0.1)  
ridge\_reg.fit(train\_X, train\_y)  
**Out[22]:** Ridge(alpha=50, max\_iter=100, tol=0.1)
- In [23]:** ridge\_reg.score(test\_X, test\_y)  
**Out[23]:** 0.6670848945194959
- In [24]:** ridge\_reg.score(train\_X, train\_y)  
**Out[24]:** 0.6622376739684328

# Recent Reference Papers

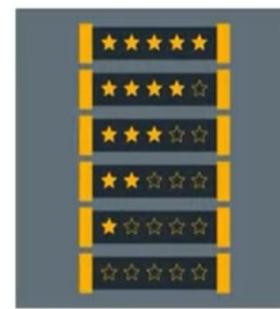
- Looking for interesting machine learning papers to read for the break or the new year? Here is a nicely curated list by Louis-François Bouchard.
- [https://github.com/louisfb01/Best\\_AI\\_papers\\_2020](https://github.com/louisfb01/Best_AI_papers_2020)
-

**Thank You**

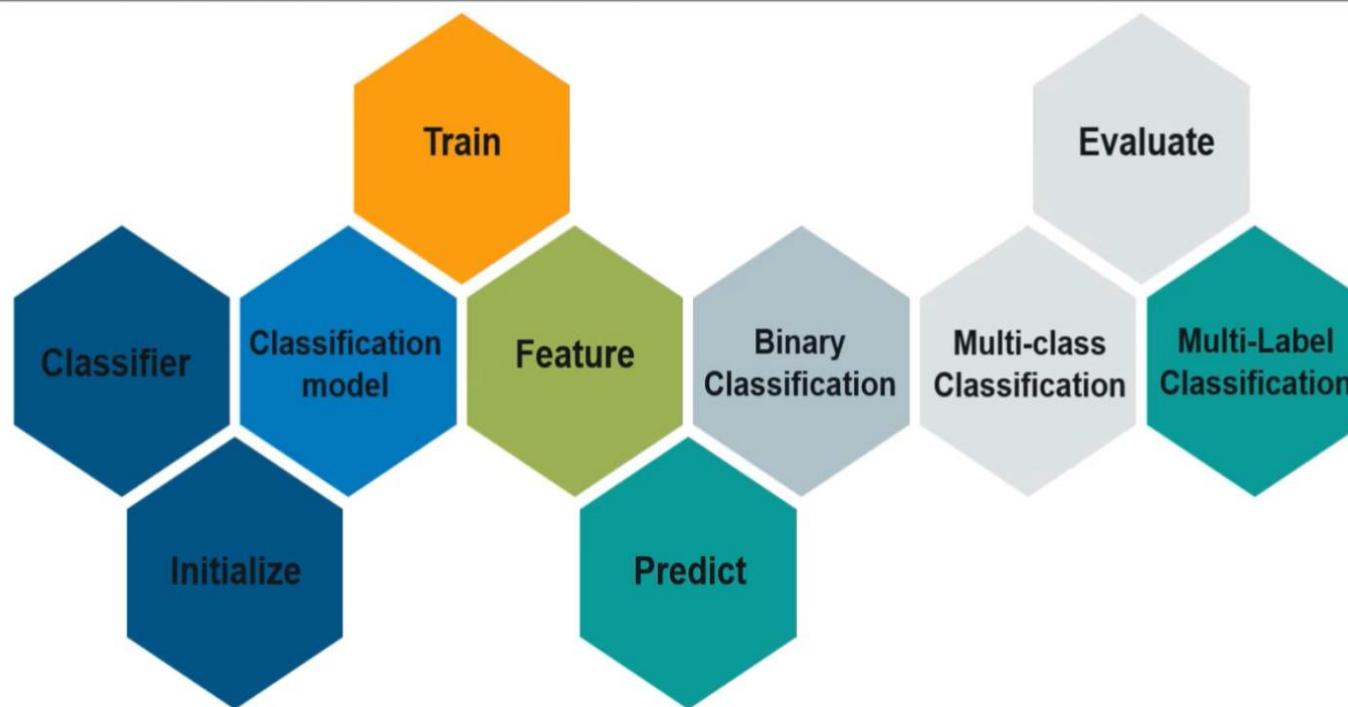
# What is Classification In Machine Learning?



Classification is a process of categorizing a given set of data into classes. It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories.

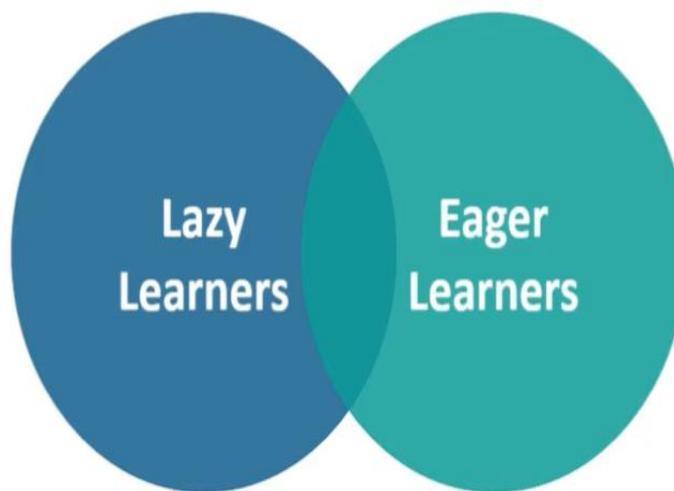


# Classification Terminologies



# Types Of Learners In Classification

Lazy learners simply store the training data and wait until a testing data appears.



Eager learners construct a classification model based on the given training data before getting data for predictions