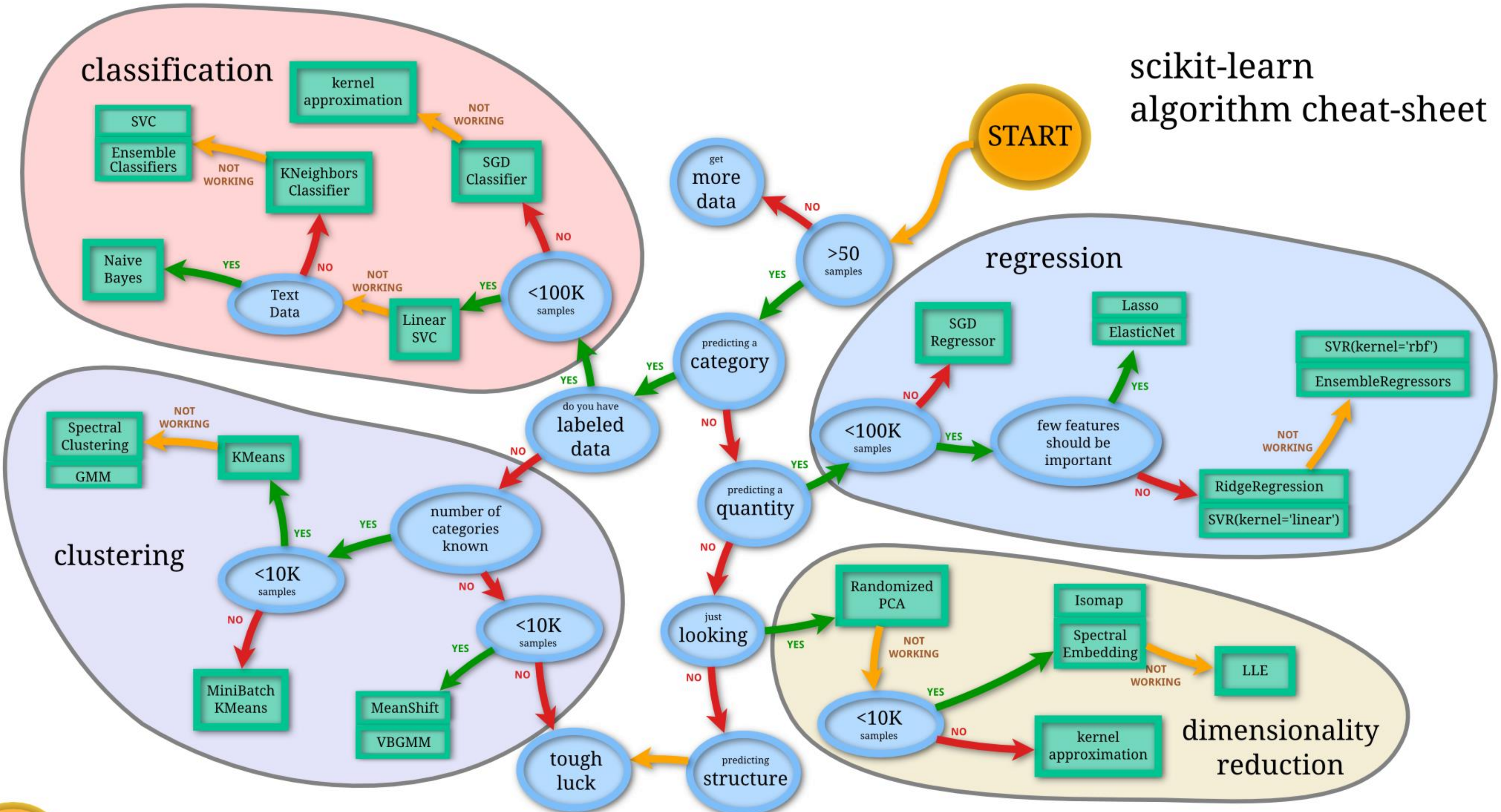
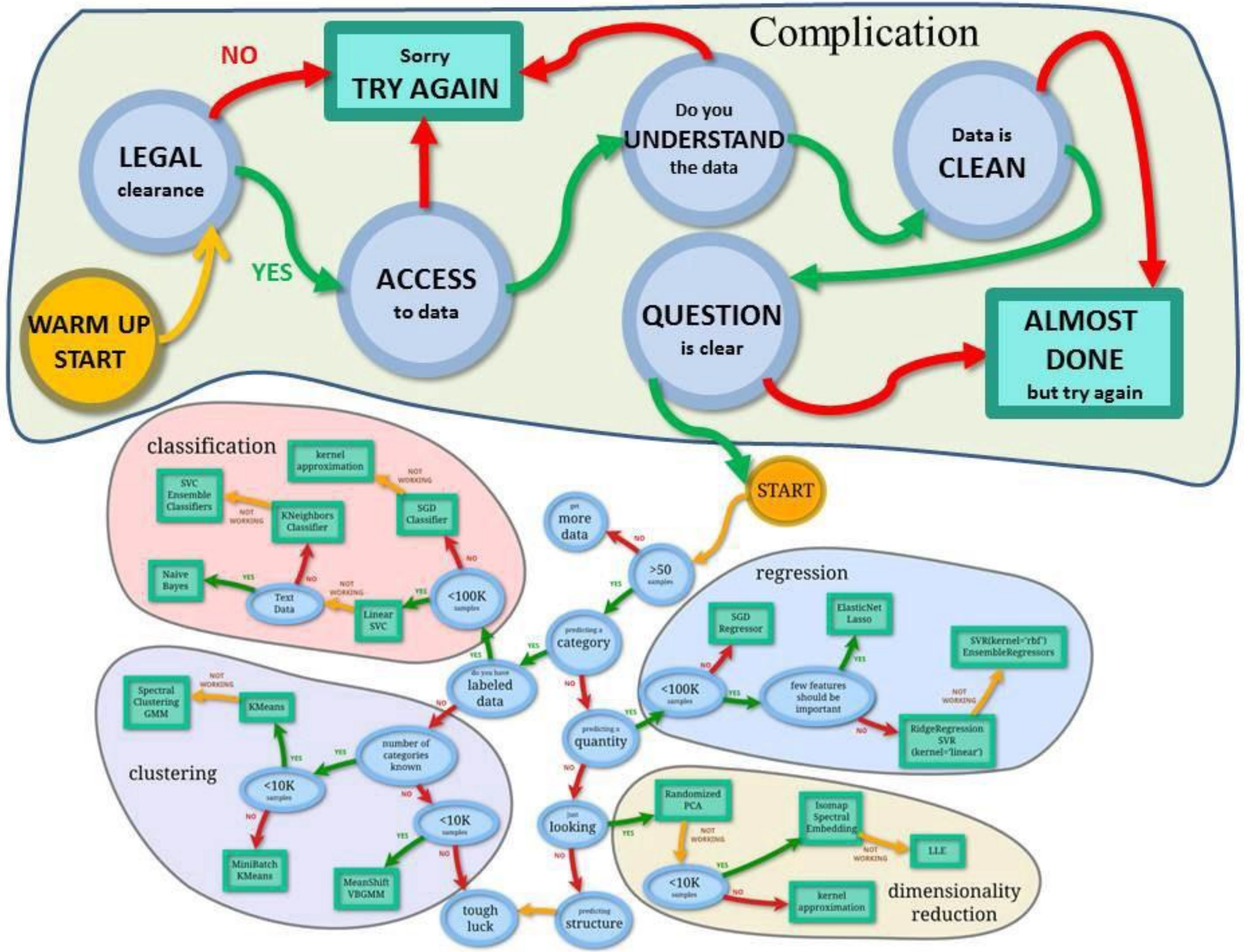
# scikit-learn algorithm cheat-sheet




Back



Complication
WARM UP START
LEGAL clearance
NO
Sorry TRY AGAIN
YES
ACCESS to data
Do you UNDERSTAND the data
Data is CLEAN
QUESTION is clear
ALMOST DONE but try again
START
classification
kernel approximation
SVC Ensemble Classifiers
KNeighbors Classifier
SGD Classifier
Naive Bayes
Text Data
Linear SVC
<100K samples
get more data
>50 samples
predicting a category
do you have labeled data
regression
SGD Regressor
ElasticNet Lasso
SVR(kernel='rbf') EnsembleRegressors
<100K samples
few features should be important
RidgeRegression SVR (kernel='linear')
clustering
Spectral Clustering GMM
KMeans
number of categories known
<10K samples
<10K samples
MiniBatch KMeans
MeanShift VBGMM
tough luck
predicting a quantity
just looking
predicting structure
Randomized PCA
Isomap Spectral Embedding
LLE
<10K samples
kernel approximation
dimensionality reduction
NOT WORKING
YES
NO

# Data Proxy

**Freight**

Baltic Dry Index

BSI

BHSI

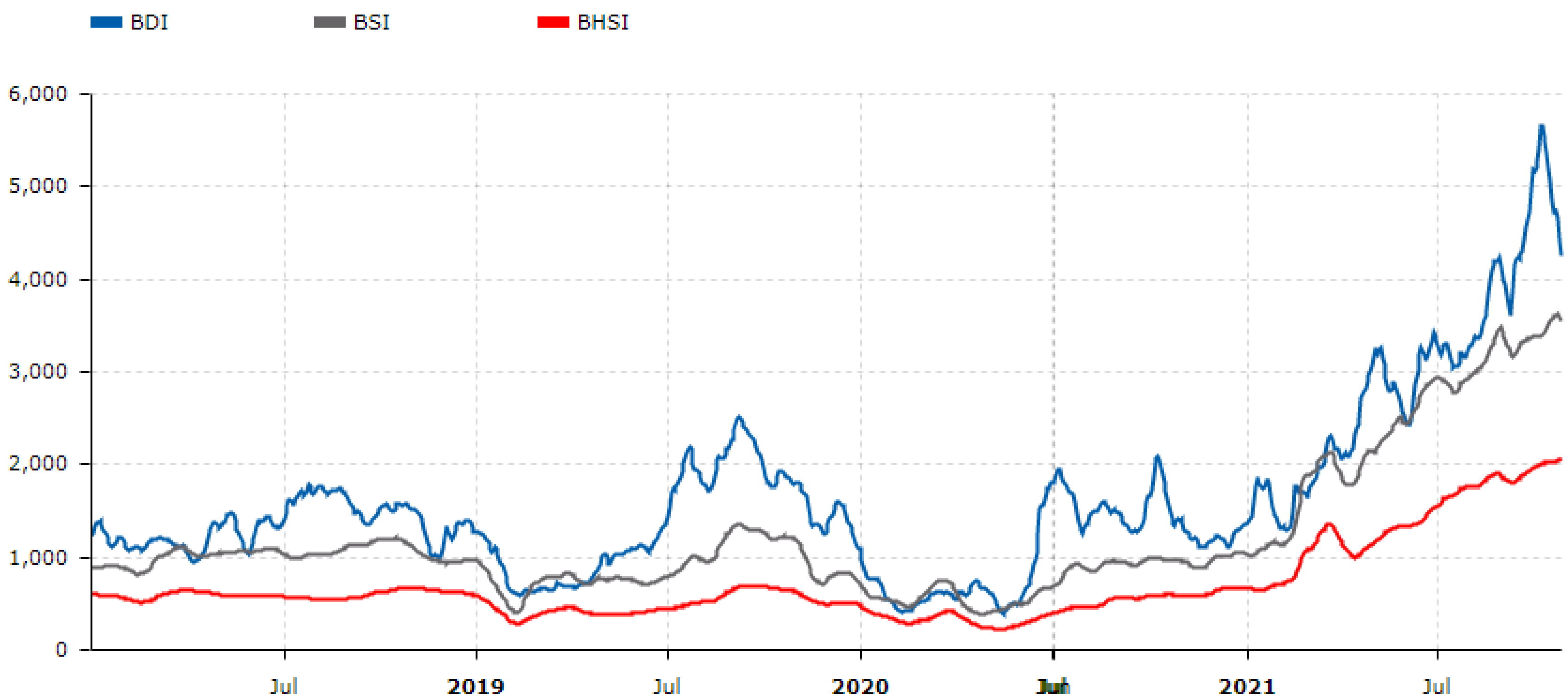
CCFI, SCFI

Global Container Volume

Orderbook-to-fleet-ratio

# Data Proxy

Key: BDI = Baltic Exchange Dry Index
BSI = Baltic Exchange Supramax Index
BHSI = Baltic Exchange Handysize Index



# Data Proxy

## Baltic Indices

| | | BHSI 38,000 | | BHSI 28,000 | | BSI | | | BDI | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | % | Net TC Avg* | Net TC Avg* | | % | Net TC Avg* | | Difference | % |
| 6-mths | 1092 | 88.8 | - | - | 2085 | 70.1 | - | 2788 | 1469 | 52.7 |
| 3-mths | 1736 | 18.8 | - | - | 2871 | 23.5 | - | 3199 | 1058 | 33.1 |
| 1-mth | 1925 | 7.1 | - | - | 3359 | 5.6 | - | 4644 | -387 | -8.3 |
| 1-wk | 2023 | 1.9 | - | - | 3595 | -1.3 | - | 4732 | -475 | -10.0 |
| | | | | | | | | | | |
| 18-Oct-21 | 2023 | - | US$34,587 | US$32,719 | 3595 | - | US$37,570 | 4732 | - | - |
| 19-Oct-21 | 2024 | 0.0 | US$34,617 | US$32,749 | 3610 | 0.4 | US$37,720 | 4714 | -18 | -0.4 |
| 20-Oct-21 | 2033 | 0.4 | US$34,770 | US$32,902 | 3618 | 0.2 | US$37,809 | 4751 | 37 | 0.8 |
| 21-Oct-21 | 2045 | 0.6 | US$34,964 | US$33,096 | 3624 | 0.2 | US$37,867 | 4653 | -98 | -2.1 |
| 22-Oct-21 | 2057 | 0.6 | US$35,181 | US$33,314 | 3584 | -1.1 | US$37,450 | 4410 | -243 | -5.2 |
| 25-Oct-21 | 2062 | 0.2 | US$35,254 | US$33,386 | 3547 | -1.0 | US$37,070 | 4257 | -153 | -3.5 |

Source: The Baltic Exchange
* Net TC Average excludes 5% commission

# Data Proxy

→ ราคาซื้อ in, ราคาขาย out

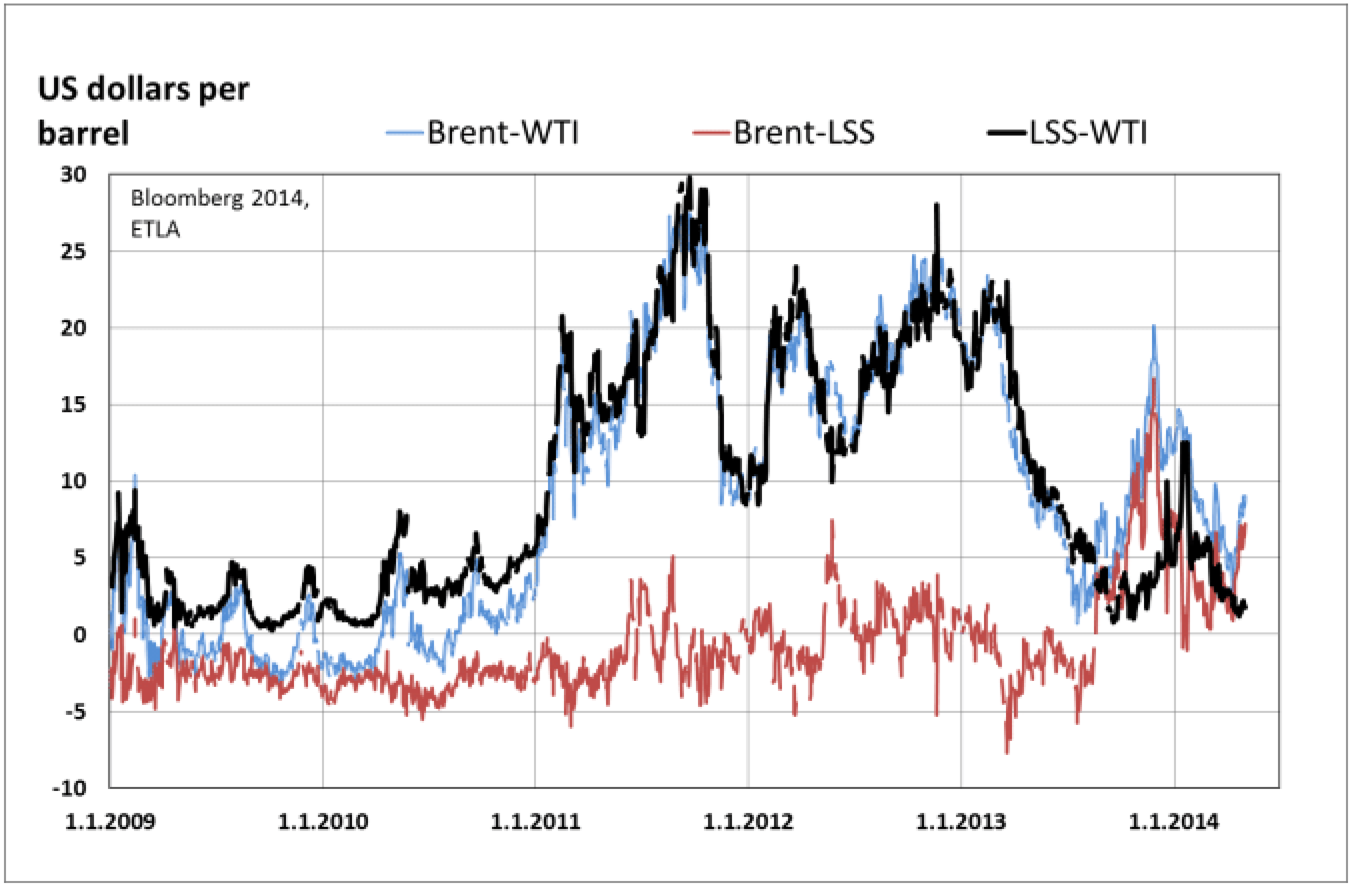
| OIL | Rubber | Coal |
|---|---|---|
| US - WTI | Tokom | New Castle Coal |
| World - Brent | **Iron Ore** | |
| Thai - Dubai | Tianjin Port | |

อยากรู้อะไร ดูอะไร ทั้งหมด Google ได้

INVESTIC

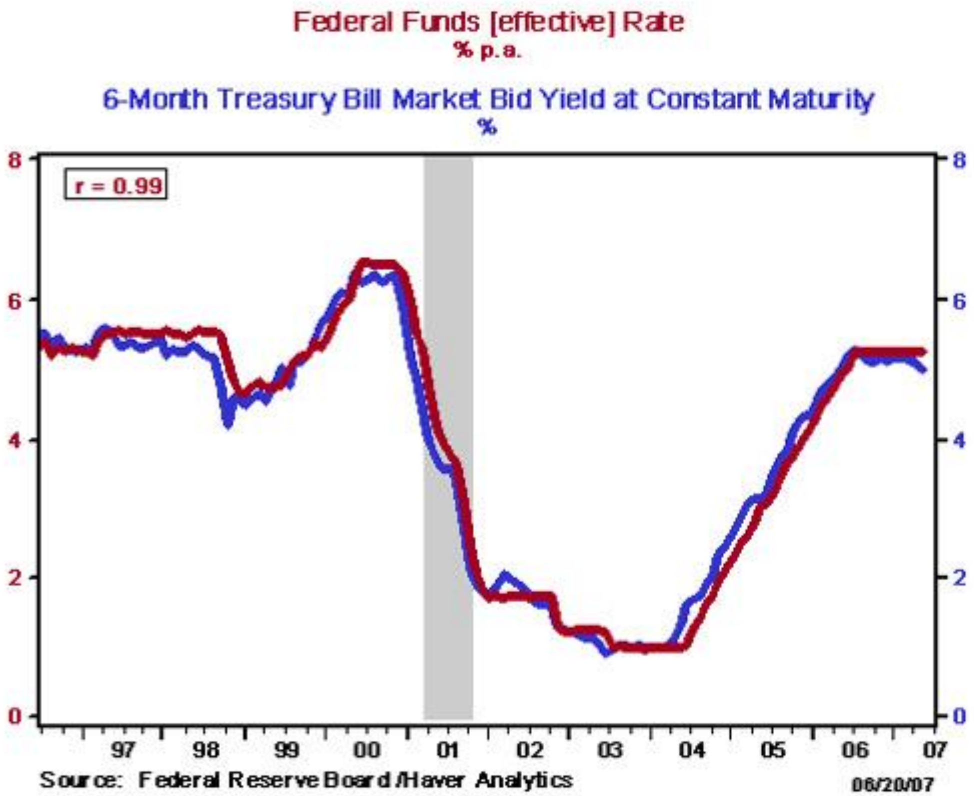
# Data Proxy

สามารถดูว่านำต่อ ก่อน, ไม่ก่อน



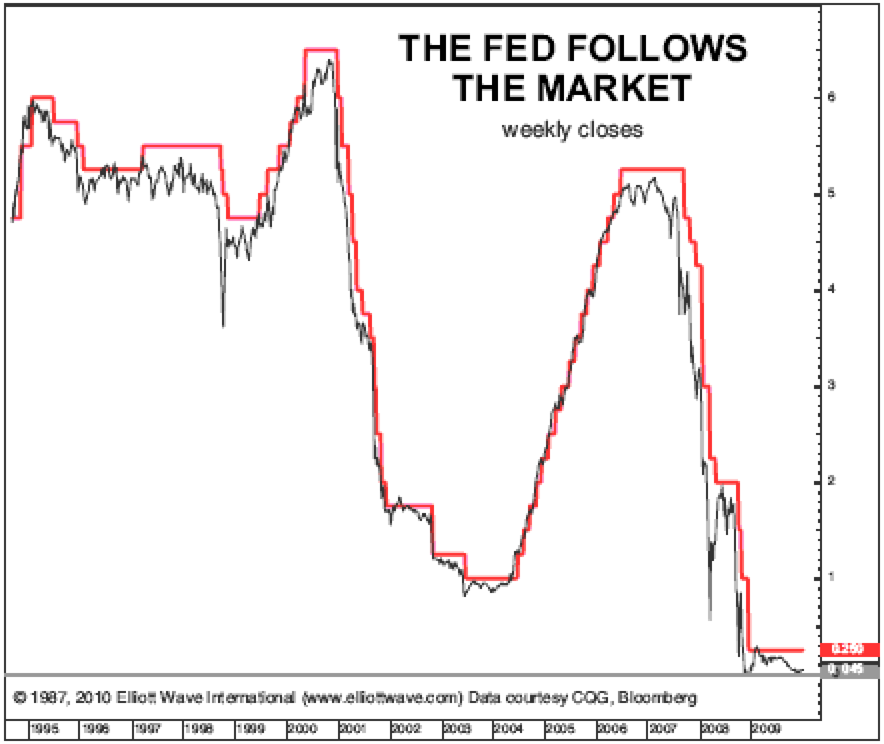

INVESTIC

# Data Proxy




INVESTIC

# Really??



THE FED FOLLOWS
THE MARKET
weekly closes
6
5
4
3
2
1
© 1987, 2010 Elliott Wave International (www.elliottwave.com) Data courtesy CQG, Bloomberg
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009

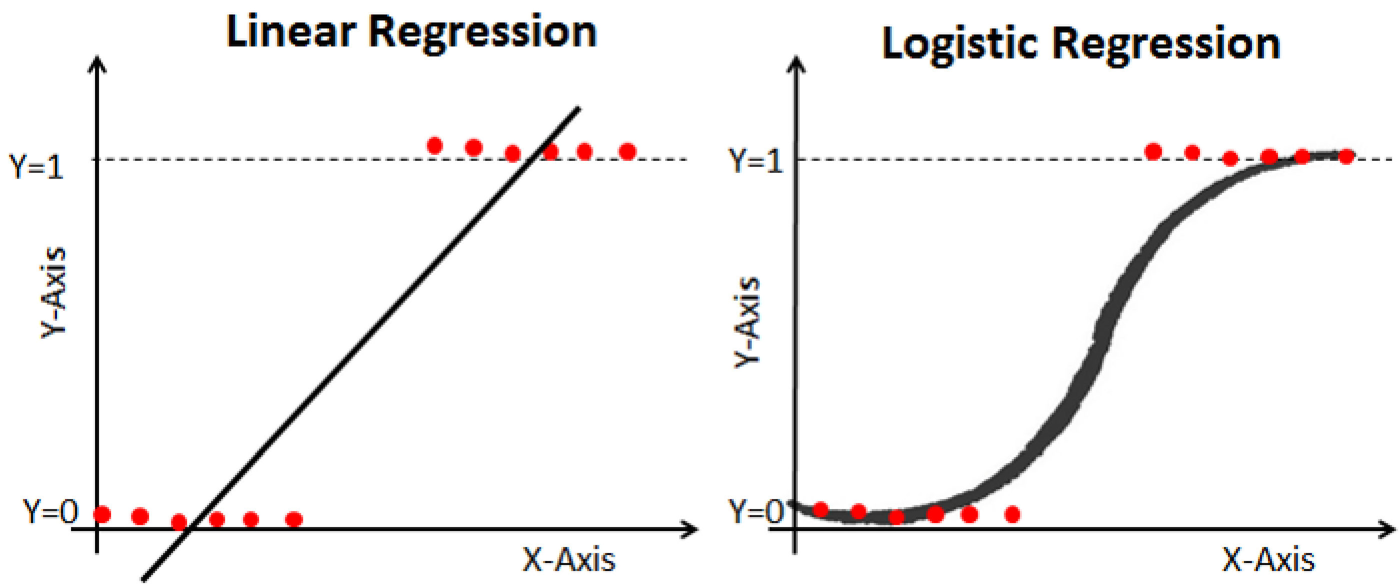
# Generalization

Generalization refers to how well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning.

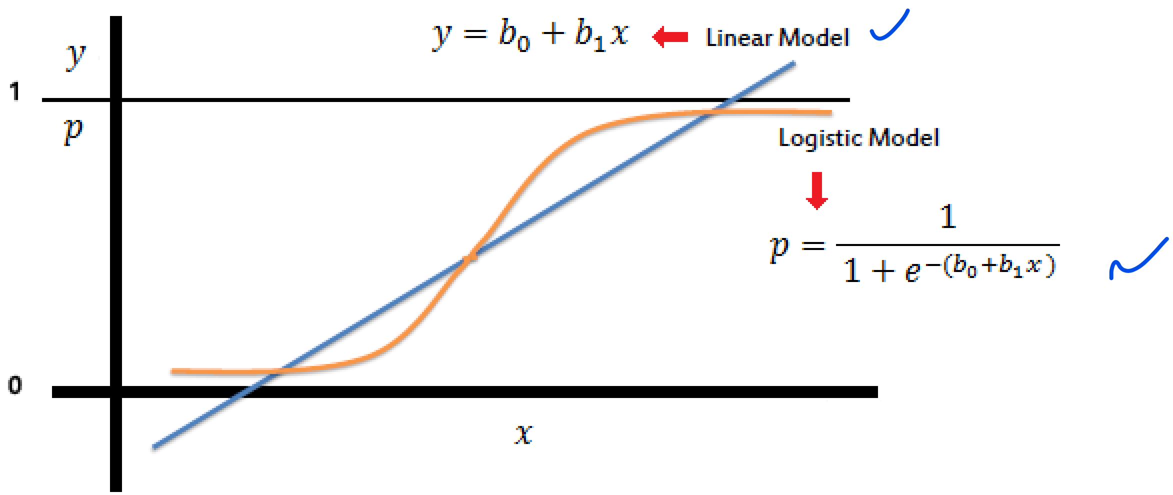
**Overfitting** refers to a model that models the training data too well.

**Underfitting** refers to a model that can neither model the training data nor generalize to new data.

# Logistic Regression

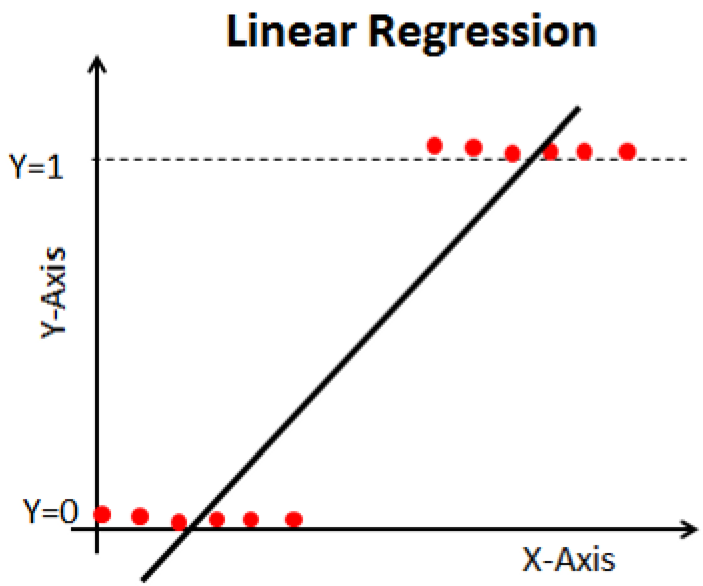


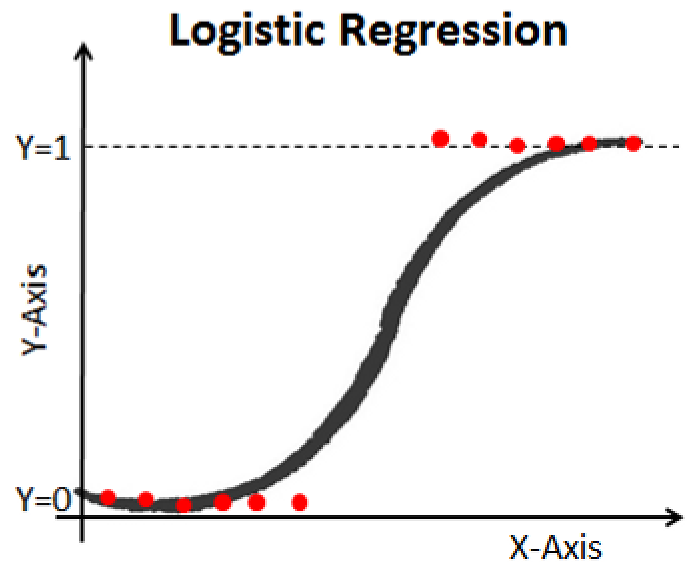
# Logistic Regression



# Logistic Regression

**Input** Return (%) Return (%)



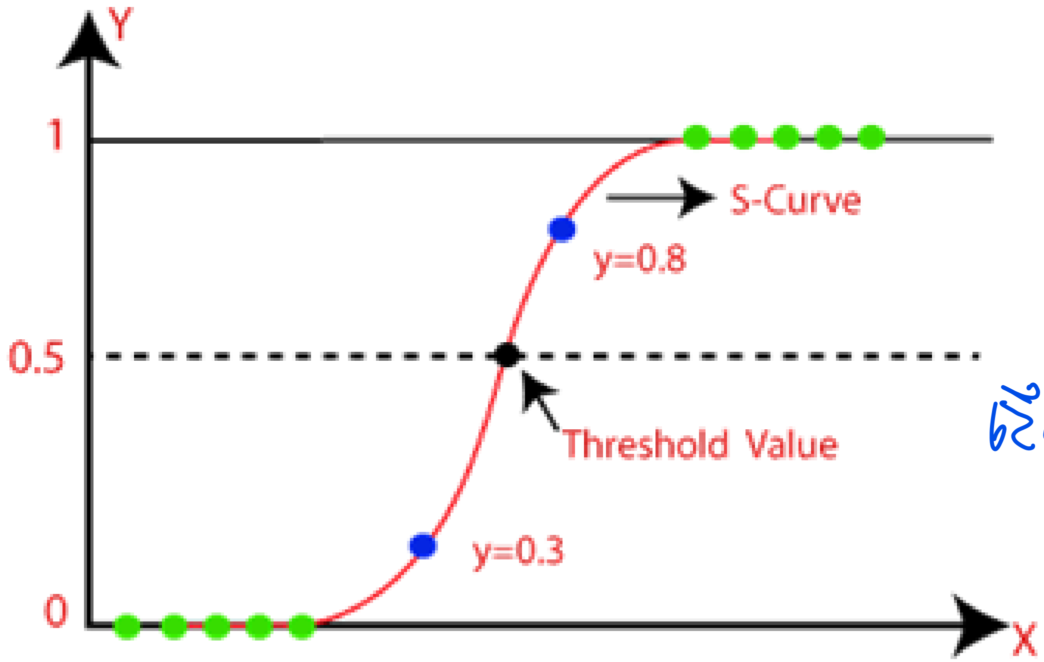




**Output** 0.34 Round to 0,1

[illegible] 0 או 1

INVESTIC

# Logistic Regression




เลือก Cut off
เองได้

# Logistic Regression

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```

**Parameters:**

**penalty : *{'l1', 'l2', 'elasticnet', 'none'}, default='l2'***

Specify the norm of the penalty:

- `'none'`: no penalty is added;
- `'l2'`: add a L2 penalty term and it is the default choice;
- `'l1'`: add a L1 penalty term;
- `'elasticnet'`: both L1 and L2 penalty terms are added.

# Logistic Regression

## Types of Logistic Regression

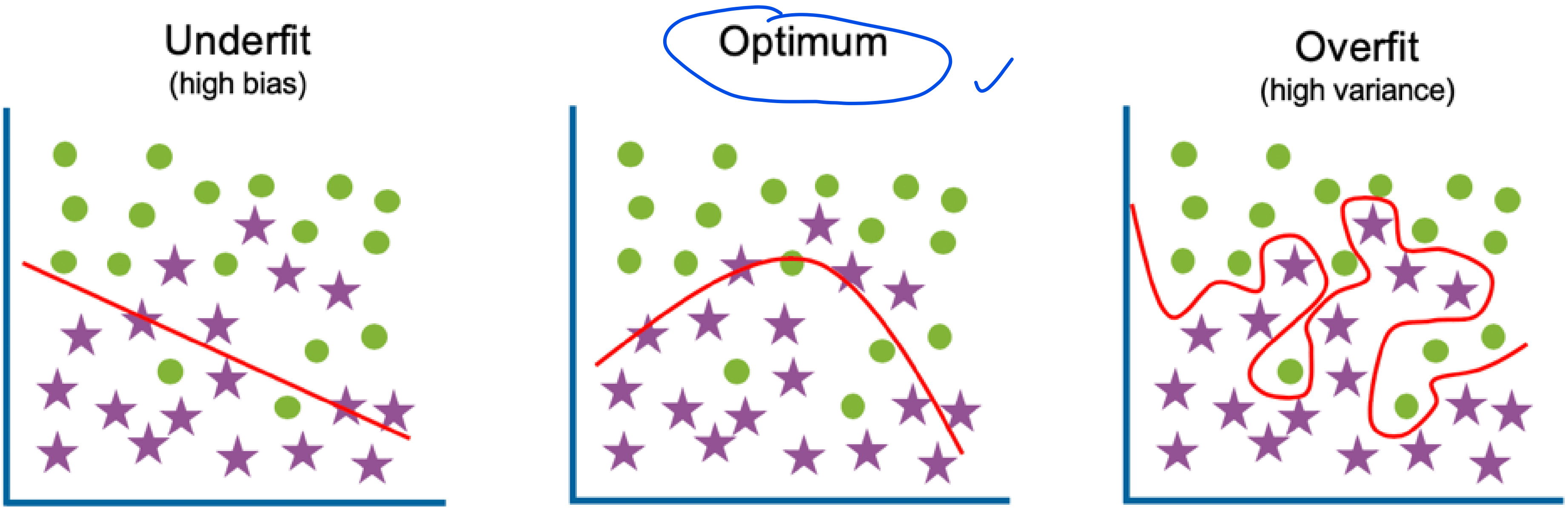
Types of Logistic Regression:

- Binary Logistic Regression: The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer.
- Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.
- Ordinal Logistic Regression: the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

# MODEL Example

https://colab.research.google.com/drive/1Hs5S9wKDNjG9SPPfN2RaSFH8r2QxmgGq?usp=sharing

# Overfit

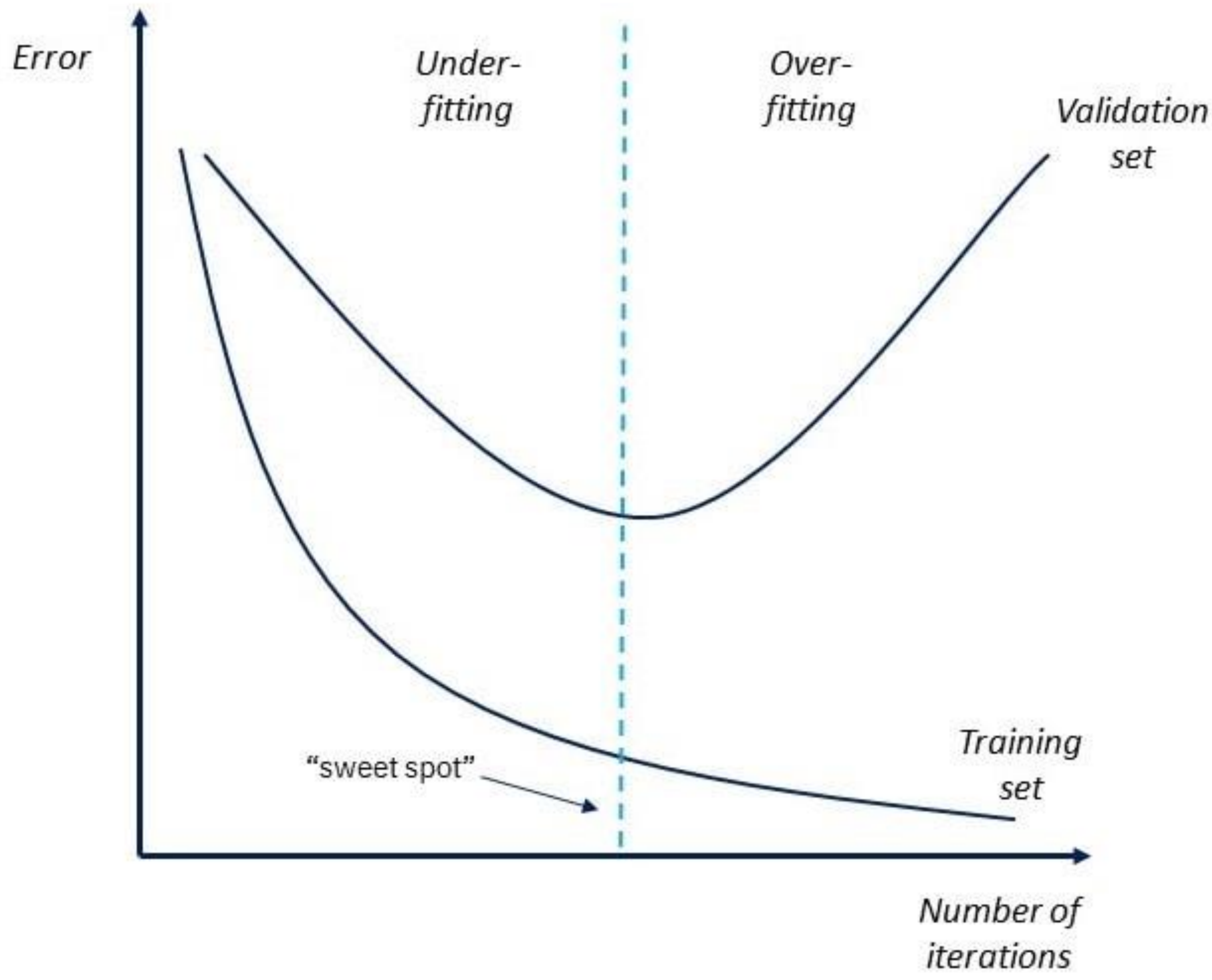



High training error
High test error

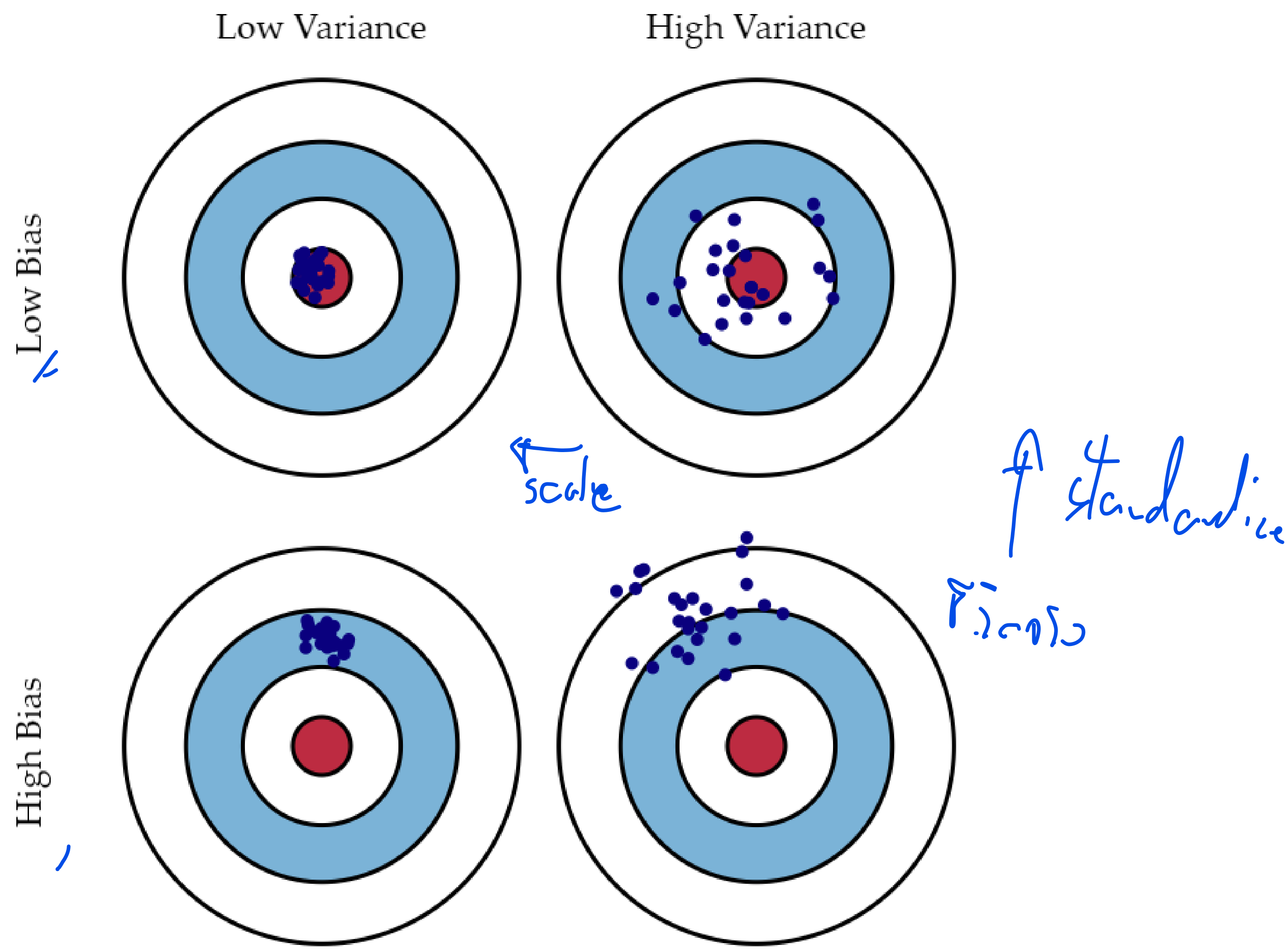
Low training error
Low test error

Low training error
High test error

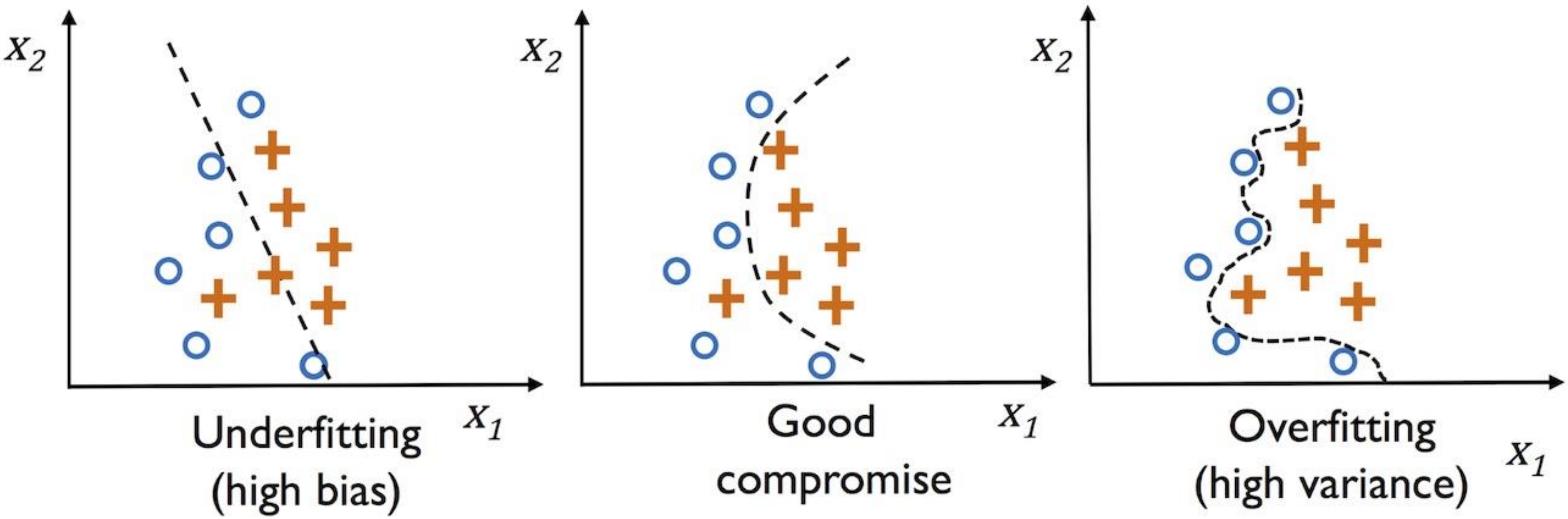
# Overfit


Error
Under-
fitting
Over-
fitting
Validation
set
Training
set
"sweet spot"
Number of
iterations

# Bias & Variances

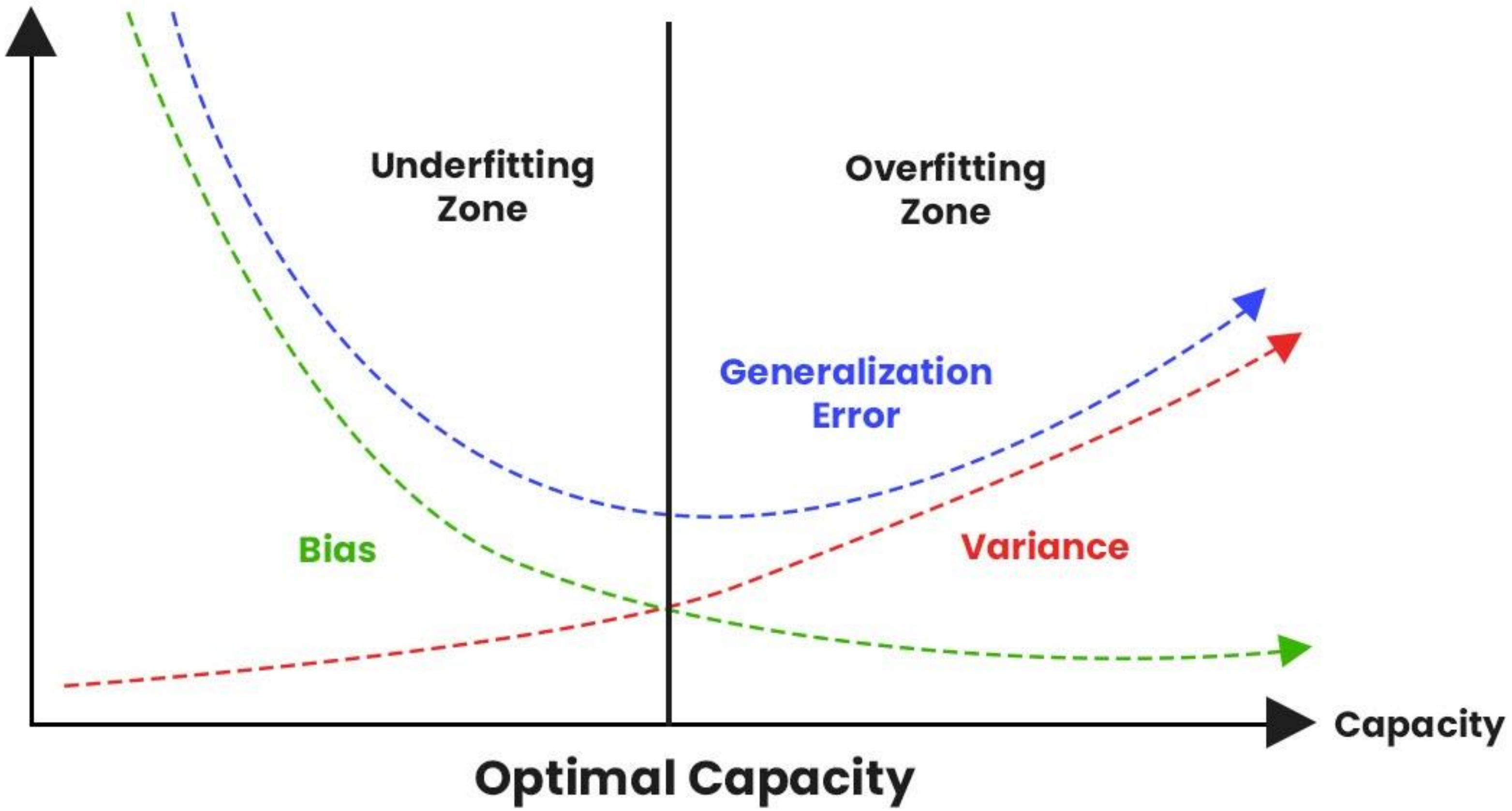


# Altogether



$X_2$
$X_1$
Underfitting
(high bias)
$X_2$
$X_1$
Good
compromise
$X_2$
$X_1$
Overfitting
(high variance)

# Headache?



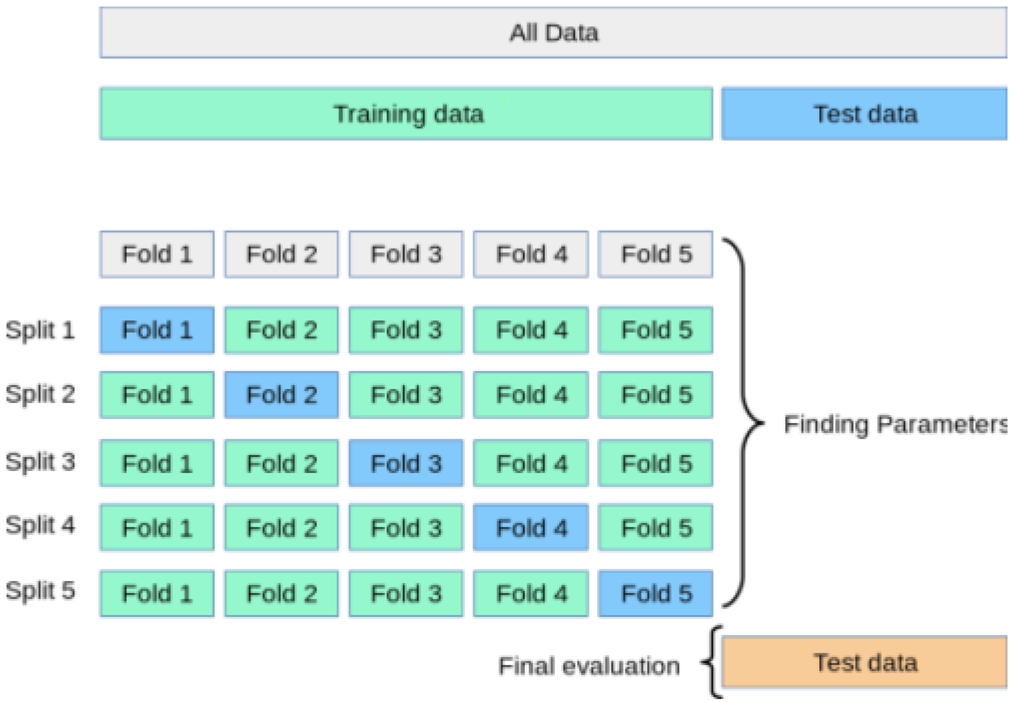

INVESTIC

# How to avoid Overfit

There are two important techniques that you can use when evaluating machine learning algorithms to limit overfitting:

1. Use a resampling technique to estimate model accuracy.
2. Hold back a validation dataset.

# K-Fold Cross Validation




INVESTIC

# Regularization

L1 L2

- L1 Regularization
- L2 Regularization
- Early Stopping
- Dropout Regularization
- Training knowledge Augmentation
- Batch standardization

**Types of Regularization in ML**

01 Ridge Regression

02 Lasso Regression

# Regularization

## L1/L2 Regularization

- L2 adds "***squared magnitude***" of coefficient as penalty term to the loss function.

$$Loss = Loss + \lambda \sum \beta^2$$

- L1 adds "***absolute value of magnitude***" of coefficient as penalty term to the loss function.

$$Loss = Loss + \lambda \sum |\beta|$$

- Weight Penalties → Smaller Weights → Simpler Model → Less Overfit

# Regularization

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_{ij} W_j\right)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

Lasso Regression

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_{ij} W_j\right)^2}_{\text{Loss function}} + \lambda \underbrace{\sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

Ridge Regression

# Regularization – ML201

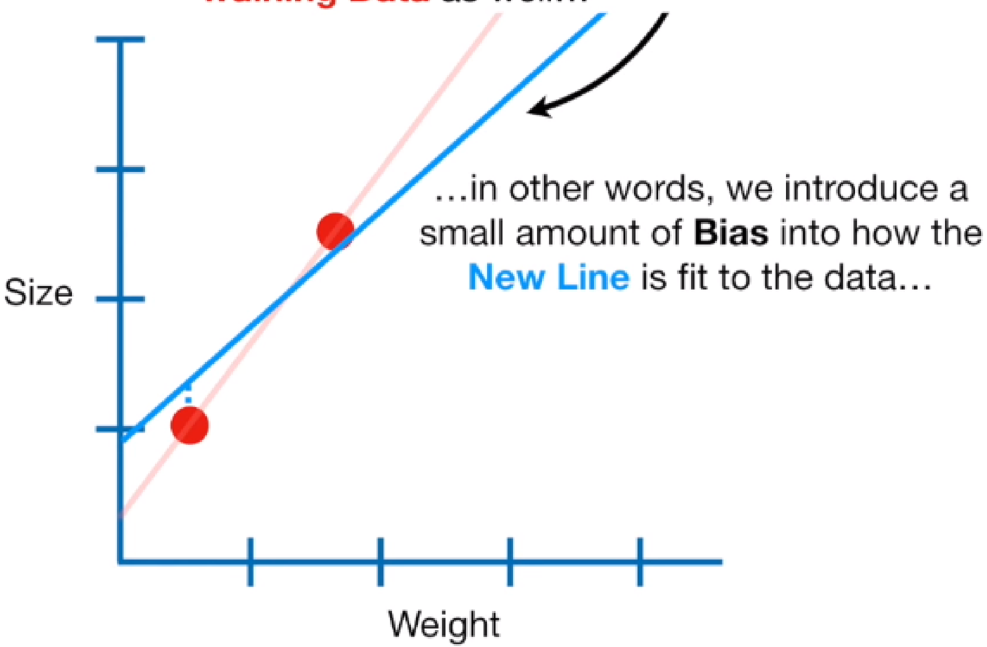
| Comparison of L1 and L2 regularization | |
|---|---|
| ***L1 regularization*** | ***L2 regularization*** |
| Sum of absolute value of weights | Sum of square of weights |
| Sparse solution | Non-sparse solution |
| Multiple solutions | One solution |
| Built-in feature selection | No feature selection |
| Robust to outliers | Not robust to outliers (due to the square term) |

| L1 loss function |
|---|
| Robust |
| Unstable solution |
| Possibly multiple solutions |

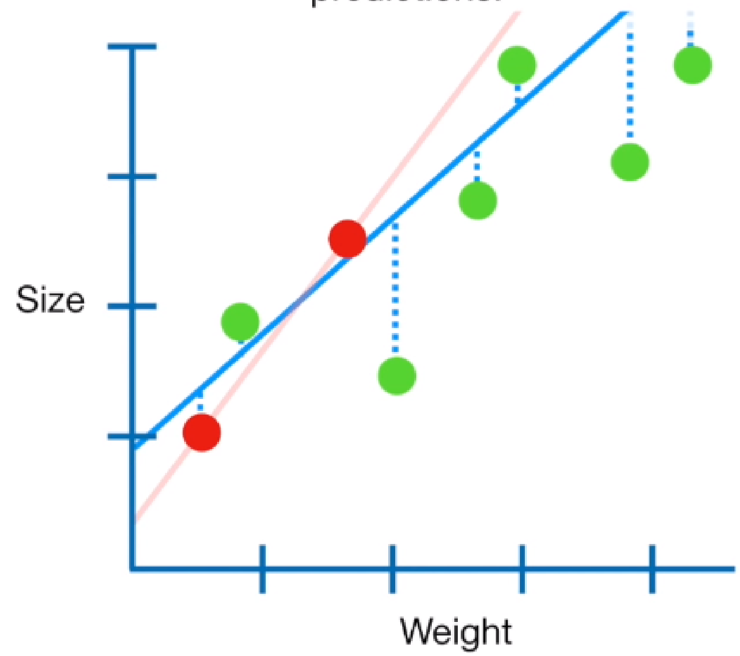
| L2 loss function |
|---|
| Not very robust |
| Stable solution |
| Always one solution |

# Regularization – ML201

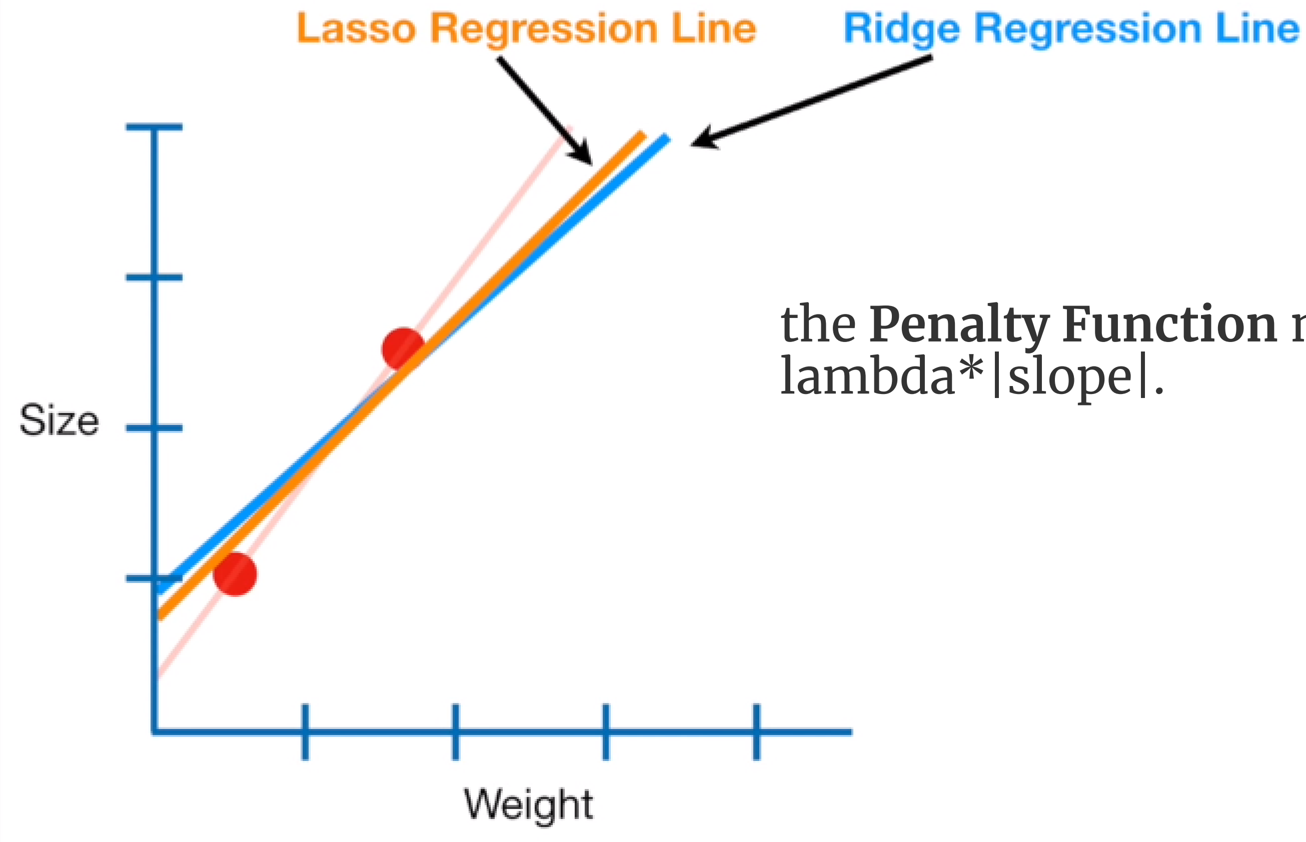
The main idea behind **Ridge Regression** is to find a **New Line** that doesn't fit the **Training Data** as well…




In other words, by starting with a slightly worse fit, **Ridge Regression** can provide better long term predictions.



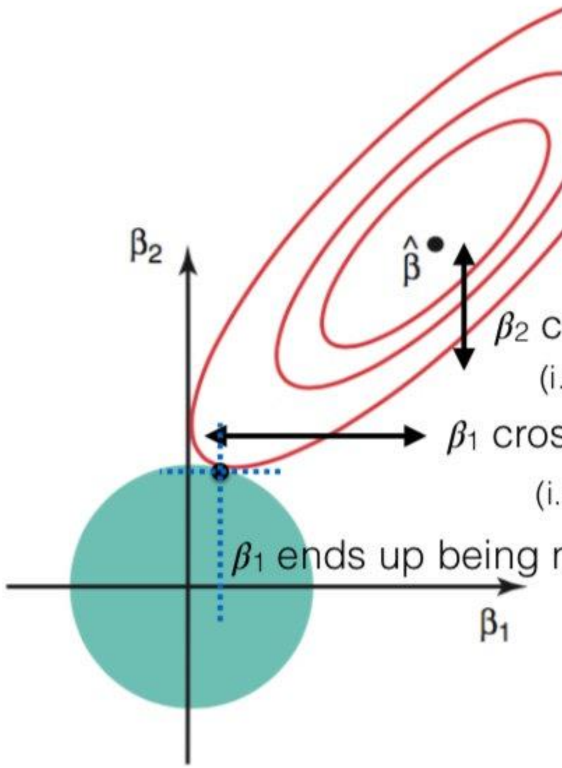
# Regularization – ML201




the **Penalty Function** now is: lambda*|slope|.

# Regularization – ML201

## **Regularization for OLS**: Ridge regression (L2 Norm)




Ridge regression optimization will estimate $\beta$ such that **less** influential features undergo **more** penalization and therefore get shrunk down more

$\beta_2$ crosses gradient rapidly
(i.e., after being standardized, is **more** influential)

$\beta_1$ crosses gradient slowly
(i.e., after being standardized, is **less** influential)

$\beta_1$ ends up being more regularized (attenuated downward)

Ridge regression is comparable to "weight decay" in Neural Nets

# Regularization – ML201

The result of the Lasso Regression is very similar to the Result given by the Ridge Regression. Both can be used in Logistic Regression, Regression with discrete values and Regression with interaction. The big difference between Rdge and Lassp start to be clear when we **Increase the value on Lambda**.

In fact, **Ridge** can only shrink the slope **asynmtotically** close to **zero**, while **Lasso** can shrink the slope **all the way to zero**. The advantage of this is clear when we have lots of parameters in the model.
In **Ridge**, when we increase the value of Lambda, the most important parameters might shrink a little bit and the less important parameter stay at high value.

In contrast, with **Lasso** when we increase the value of Lambda the most important parameters shrink a little bit and the less important parameters goes closed to zero. So, Lasso is able to exclude silly parameters from the model.

.fit_regularized():

# For LogisticRegression

**solver : {*'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default='lbfgs'***

Algorithm to use in the optimization problem. Default is 'lbfgs'. To choose a solver, you might want to consider the following aspects:

- For small datasets, 'liblinear' is a good choice, whereas 'sag' and 'saga' are faster for large ones;
- For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss;
- 'liblinear' is limited to one-versus-rest schemes.

**Warning:** The choice of the algorithm depends on the penalty chosen: Supported penalties by solver:

- 'newton-cg' - ['l2', 'none']
- 'lbfgs' - ['l2', 'none']
- 'liblinear' - ['l1', 'l2']
- 'sag' - ['l2', 'none']
- 'saga' - ['elasticnet', 'l1', 'l2', 'none']

# What to focus now

Explainable “Coincidence” X,y

“Leading Indicator”