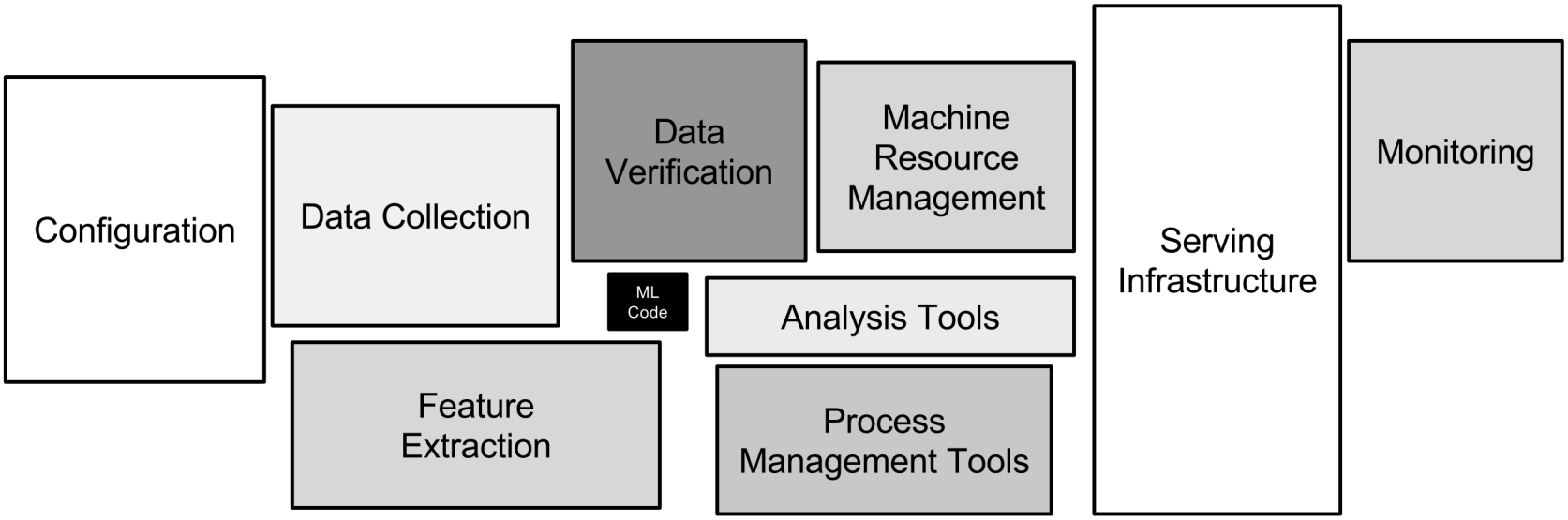
# Supervised Learning - I

**Practical Machine Learning >> Regression Algorithms**

**Kannan Singaravelu, CQF**

# Machine Learning Systems

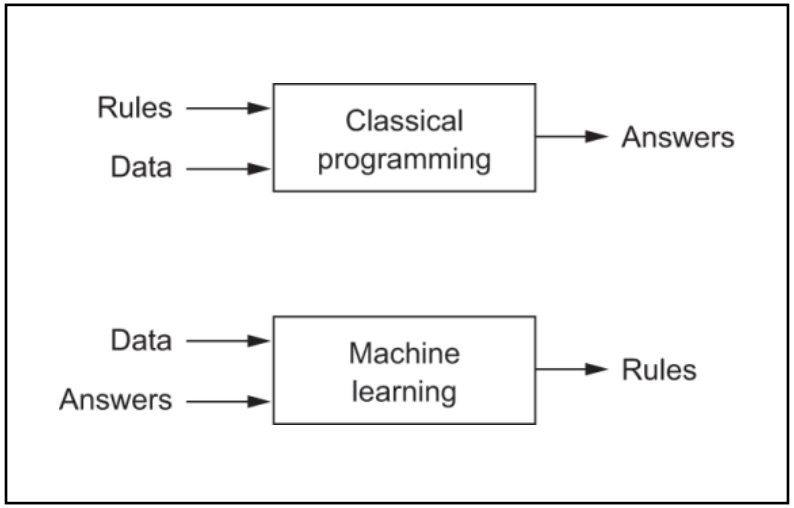



Source: Hidden Technical Debt in Machine Learning Systems, Google Inc. Refer Advanced Machine Learning workshop for detail discussion on ML systems and pipeline.

# In this lecture..

- What is Machine Learning (ML)?
- Understanding learning and data representations
- How learning algorithm works?
- Understanding ML workflow
- Bias - Variance Tradeoff
- Linear Regression
- Regression loss functions
- Understanding Gradient Descent
- Resampling Methods
- Shrikage methods

# Data Driven Finance



# What Is Not Machine Learning?

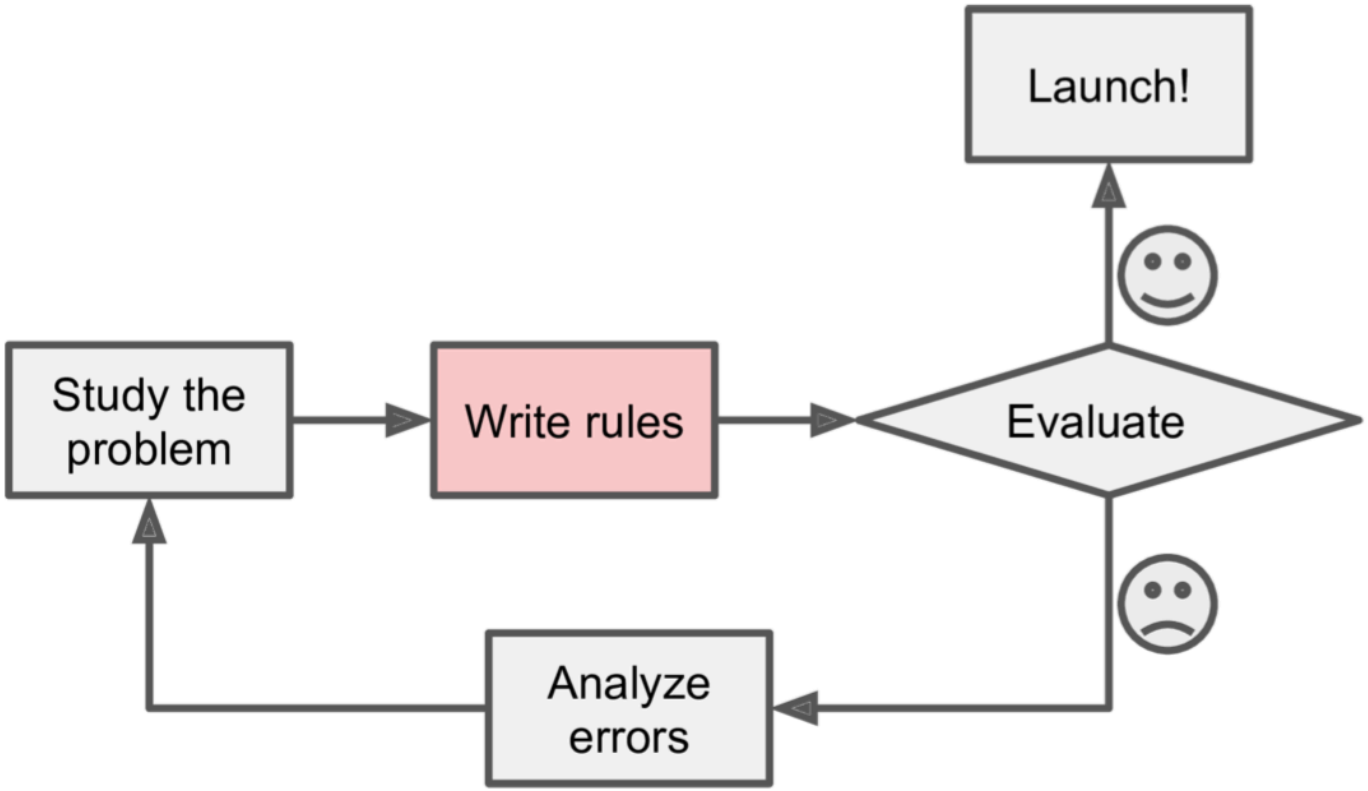



Image source: Aurelien Geron (2019), Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.

# Machine Learning Approach

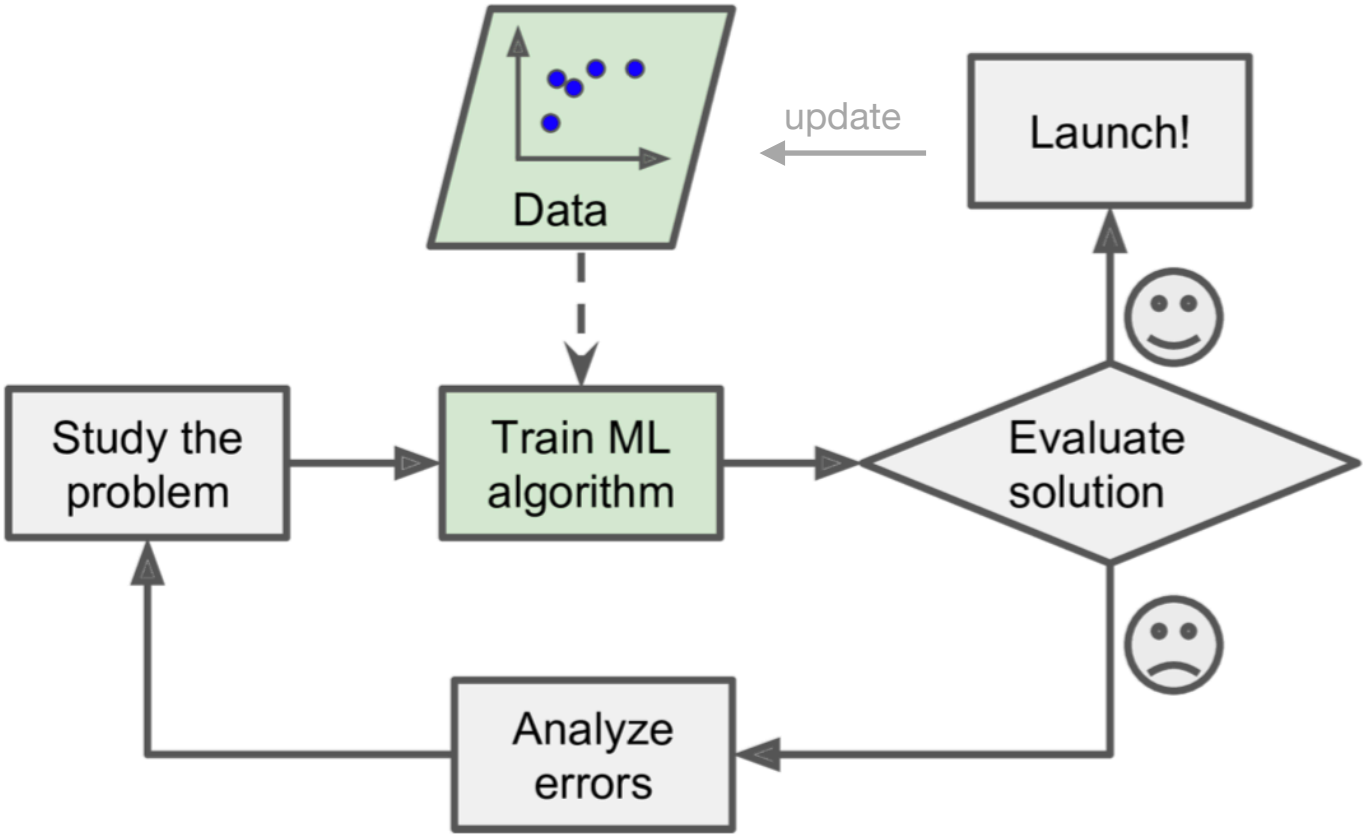



Image source: Aurelien Geron (2019), Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.

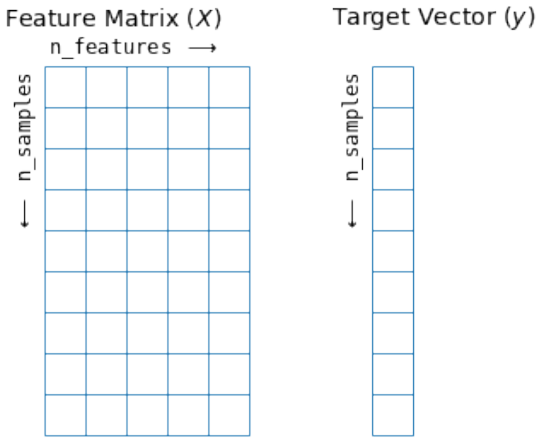
# Learning Representations

- Statistical Learning - aka Machine Learning - emerged as one of the main sub fields in statistics
- The primary focus is on modelling and prediction
- Input data : Features; Expected output : Label
- Learning useful representation of the input data
  - **Learning:** automatic search for better representation
  - **Representation**: a way to look at data = encoding of the input data (Ex: RGB, Hexcodes)
  - Searching on a hypothesis space (a predefined space of possibilities or operations)
  - Not very effective in a high dimensional space

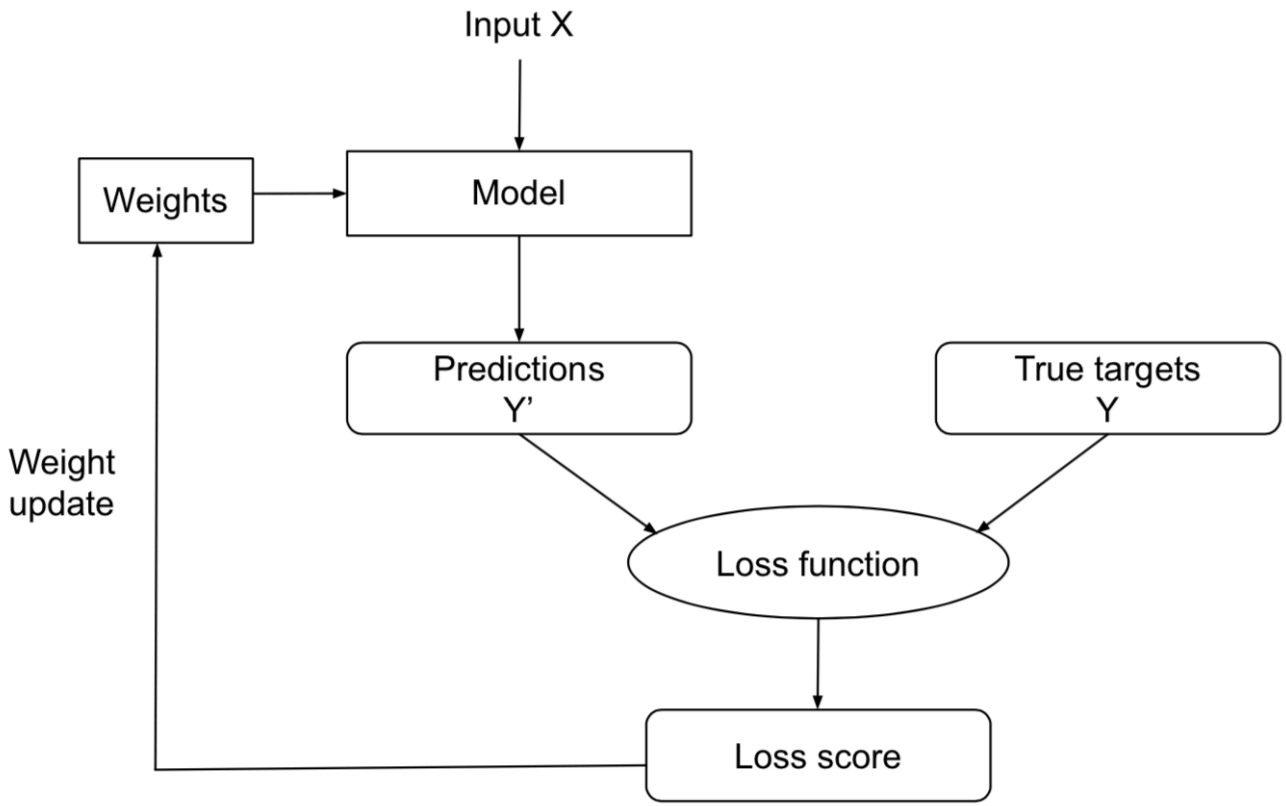
# Data Representations

- **Supervised Learning** problem can be stated as given a matrix $X$, of dimensions $n$ x $p$ , we can create a predictive relationship or function $f(X)$ where $f(X) = y$, where $y$ is a vector of dimension $n$.

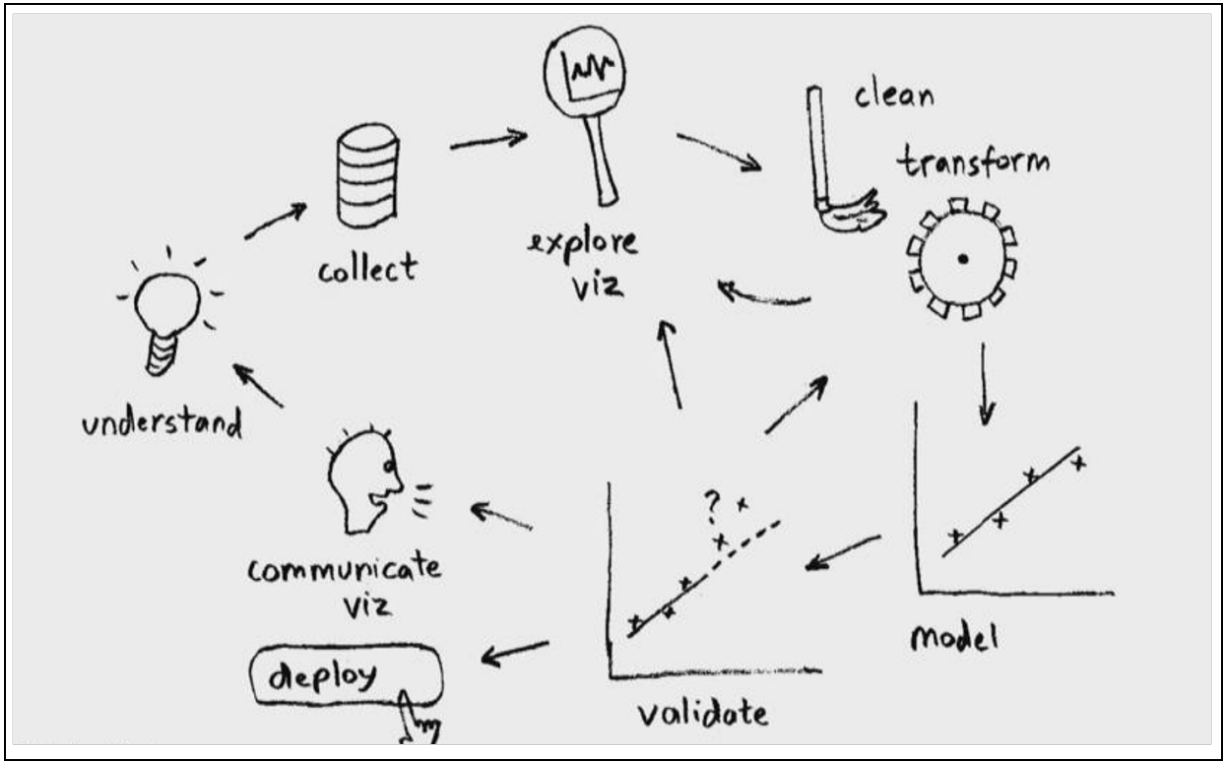
- **Feature Matrix ($X$)**
  - *n_samples $n$: number of samples*
  - *n_features $p$: number of features*
- **Target Vector ($y$)**
  - *n_samples $n$: number of samples*



# How Learning Works?



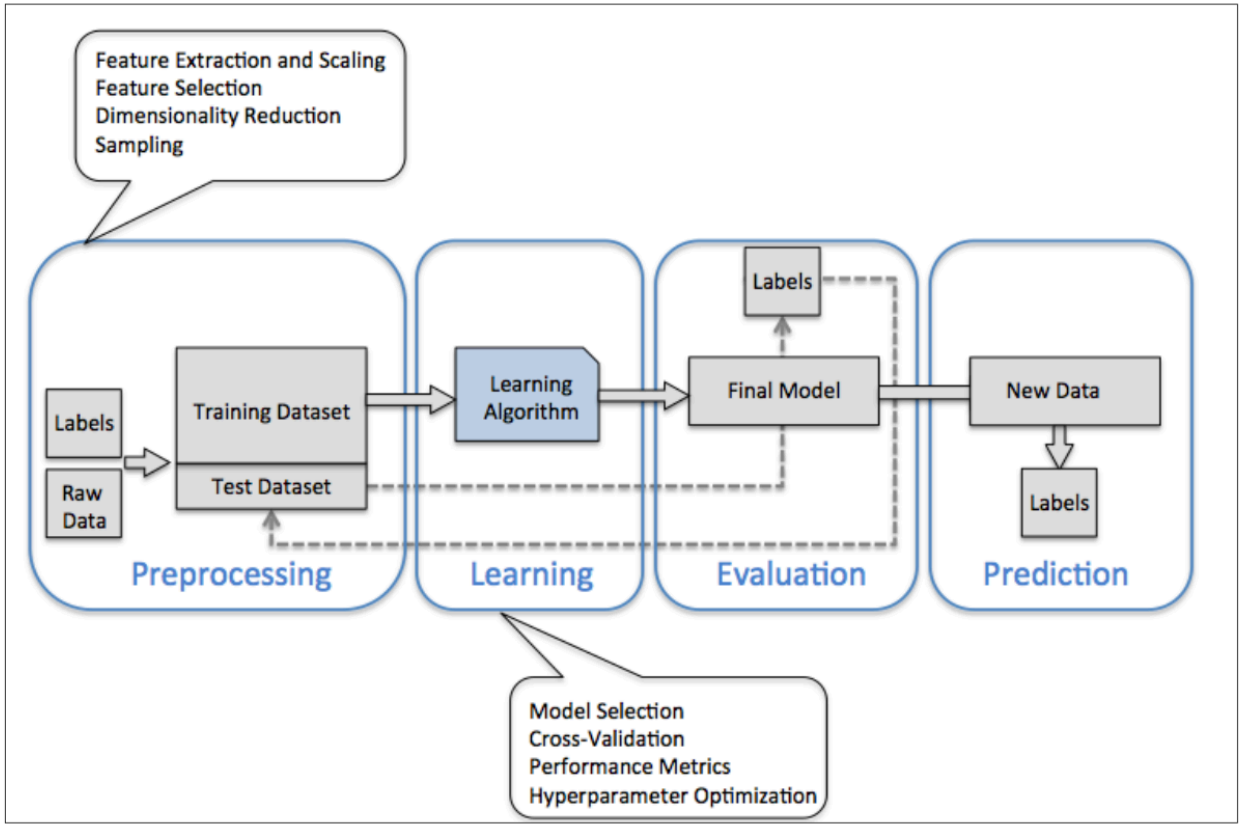
# Model Development Steps




*Representation of Machine Learning Workflow*

Note: Refer Python Labs and Advanced Machine Learning workshop for application of Feature Engineering, Transformation, Selection Methods, Cross-Validation and allied frameworks.

# Predictive Modeling Process




*Representation of a typical workflow diagram for using machine learning in predictive modeling*

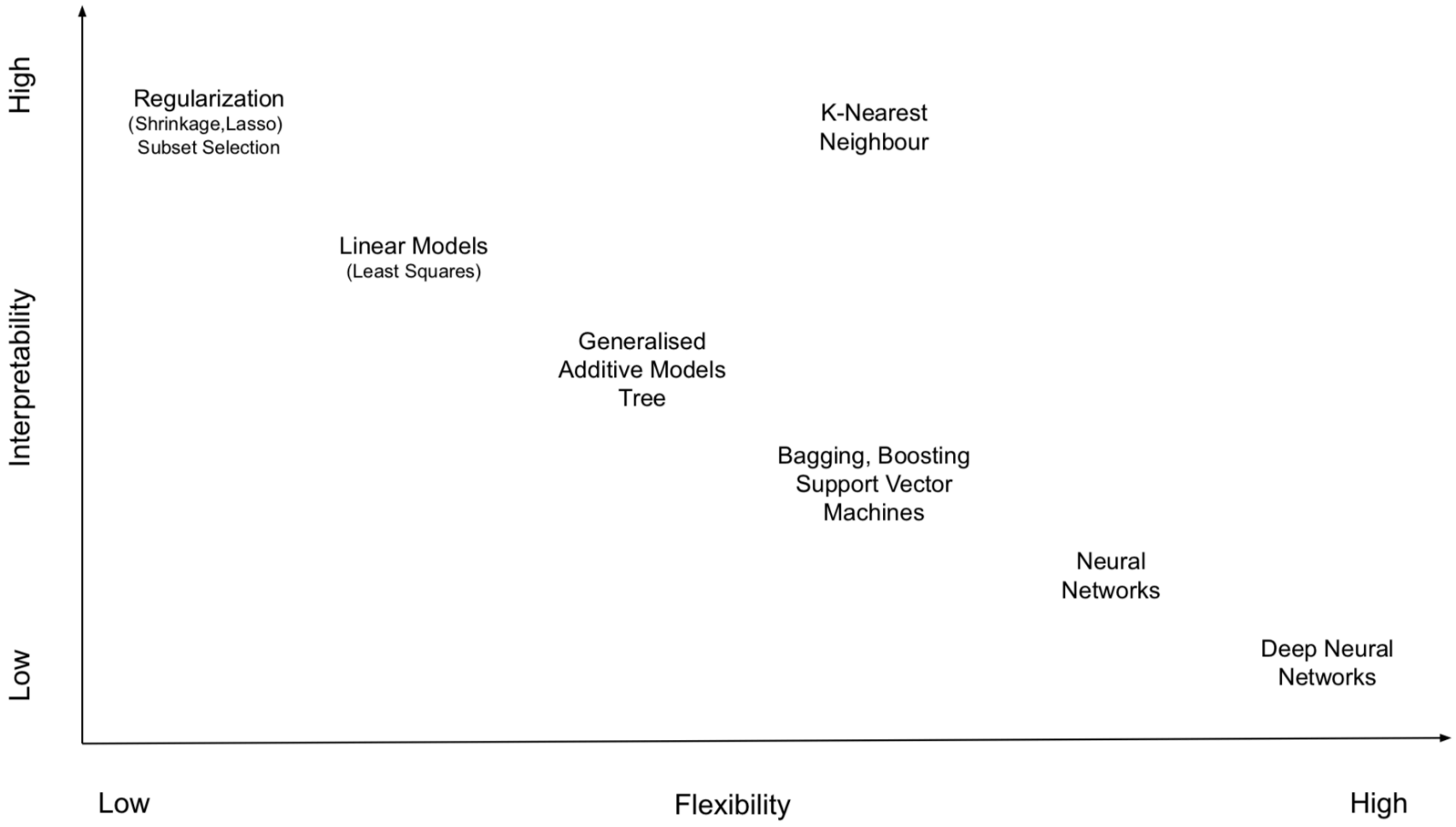
Image source: S. Raschka. Python Machine Learning. Refer Python Labs and Advanced Machine Learning Workshop for application of Feature Engineering, Resampling Methods and allied frameworks.

# Feature Engineering

- The process of using domain knowledge to select and transform the most relevant features or variables from raw data.
- A preprocessing step including data preparation and exploratory analysis with a goal of improving the performance of learning algorithms. Those are
  - Feature Creation
  - Feature Transformation or Scaling
  - Feature Selection

Refer Python Labs, Machine Learning Exercises and Advanced Machine Learning workshop for application.

# Interpretability - Flexibility Tradeoff




Adapted from Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani (2013), An Introduction to Statistical Learning.

# Bias - Variance Tradeoff

- Understanding how different sources of error lead to bias and variance helps improve the data fitting process resulting in more accurate models.
- **Bias :** Arise out of wrong assumptions; a high-bias model is most likely to under-fit the data.
- **Variance :** Excessive sensitivity to small variations in the data; a high degrees of freedom overfit the data.
- **Irreducible Error :** Noisiness of the data itself and can be addressed by clean the data.

# Bias - Variance Tradeoff

- Irreducible error is the noise term that cannot be reduced by any model.
- In a true model with infinite data to calibrate, we should be able to reduce both bias and variance to zero.
- However, in real world with imperfect models and finite data, there is a tradeoff between minimising the bias and the variance.

# Bias - Variance Tradeoff

- If $Y = f(X) + \epsilon$, where the error term $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$, we may estimate a model $\hat{f}(X)$ of $f(X)$ using linear regressions or such other modelling technique.
- In this case, the expected squared prediction error at a point $x$ is

$$Err(x) = E\left[(Y - \hat{f}(x))^2\right]$$

- This error can then be decomposed into bias and variance components and model generalisation error can be expressed as the sum of three different errors

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = Bias^2 + Variance + Irreducible\,Error$$

Adapted from Scott Fortmann Roe (2012), Understanding the Bias-Variance Tradeoff.

# Bias - Variance of the Squared Error

$$\hat{y} = f(x)$$
$$\hat{y} = f(\hat{x}) = h(x)$$
$$\begin{aligned} S &= (y - \hat{y})^2 \\ &= (y - E[\hat{y}] + E[\hat{y}] - \hat{y})^2 \\ &= (y - E[\hat{y}])^2 + (E[\hat{y}] - \hat{y})^2 + 2(y - E[\hat{y}])(E[\hat{y}] - \hat{y}) \end{aligned}$$

*taking expectation,*

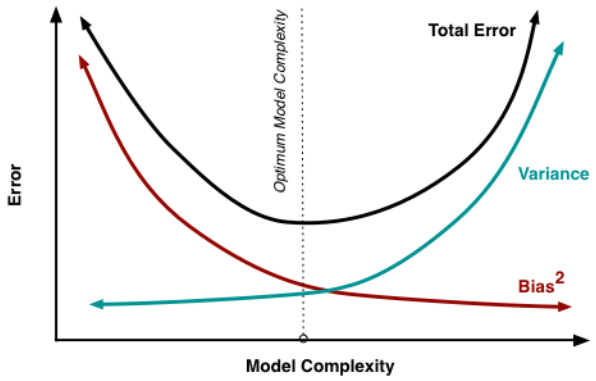
$$\begin{aligned} E[S] &= E[(y - \hat{y})^2] \\ E[(y - \hat{y}^2)] &= (y - E[\hat{y}])^2 + E[(E[\hat{y}] - \hat{y})^2] \\ &= Bias^2 + Variance \end{aligned}$$

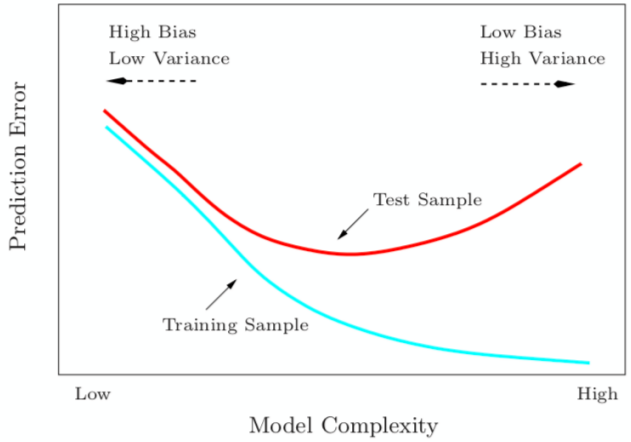
# Bias - Variance Tradeoff

- Gauss - Markov theorem implies OLS coefficients are BLUE.
- Least squares estimator has the smallest mean squared error of all linear estimators with no bias.
- Biased estimator with smaller mean squared error may exist.
- Estimator would trade a little bias for a larger reduction in variance.
- Any method that shrinks or sets to zero some of the least squared coefficients may results in a biased estimate.

# Bias - Variance Tradeoff

- Bias - Variance Tradeoff
  - Increasing model's complexity (flexibility) increases its variance and reduces its bias.
  - Reducing model's complexity (flexibility) reduces its variance and increases its bias.




Adapted from Scott Fortmann Roe (2012), Understanding the Bias-Variance Tradeoff.

# Bias - Variance Intuition

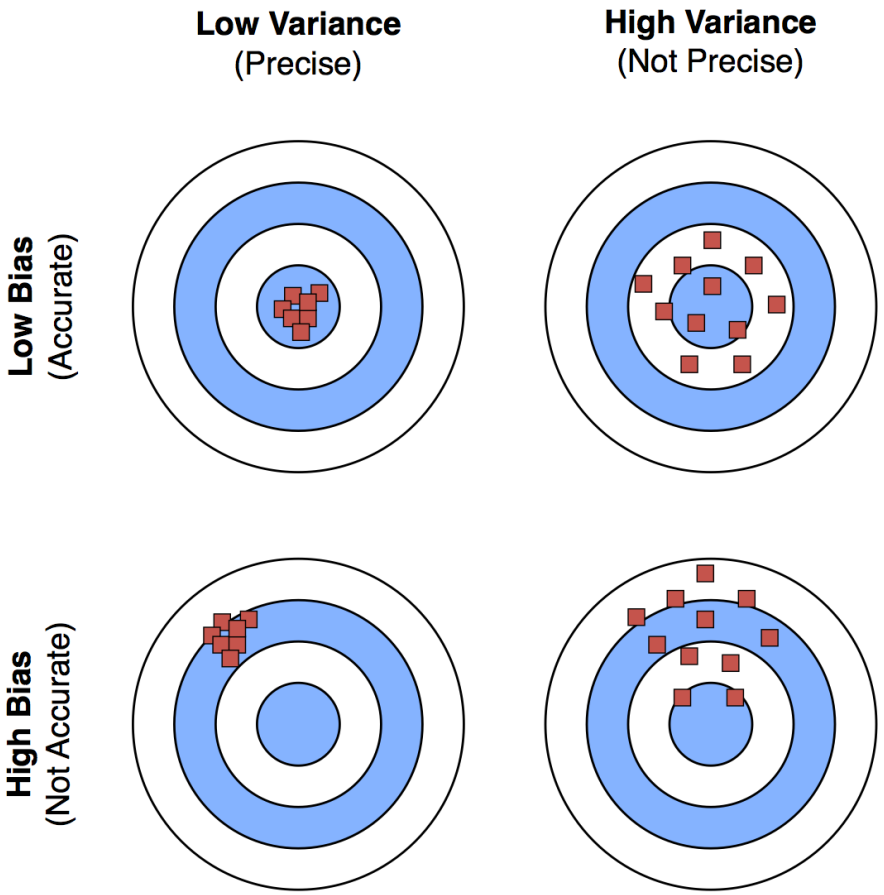



Image source: S. Raschka. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning, November 2018.

# Linear Regression

- Regression models are models which predict a continuous outcome.
- An approach used to model the relationship between
  - a scalar dependent variable $y$ and
  - one or more explanatory variables (aka independent variables) denoted by $X$.
- In machine learning parlance,
  - explanatory variables or predictors are called **features** and
  - target or the dependent variable **labels or targets**.

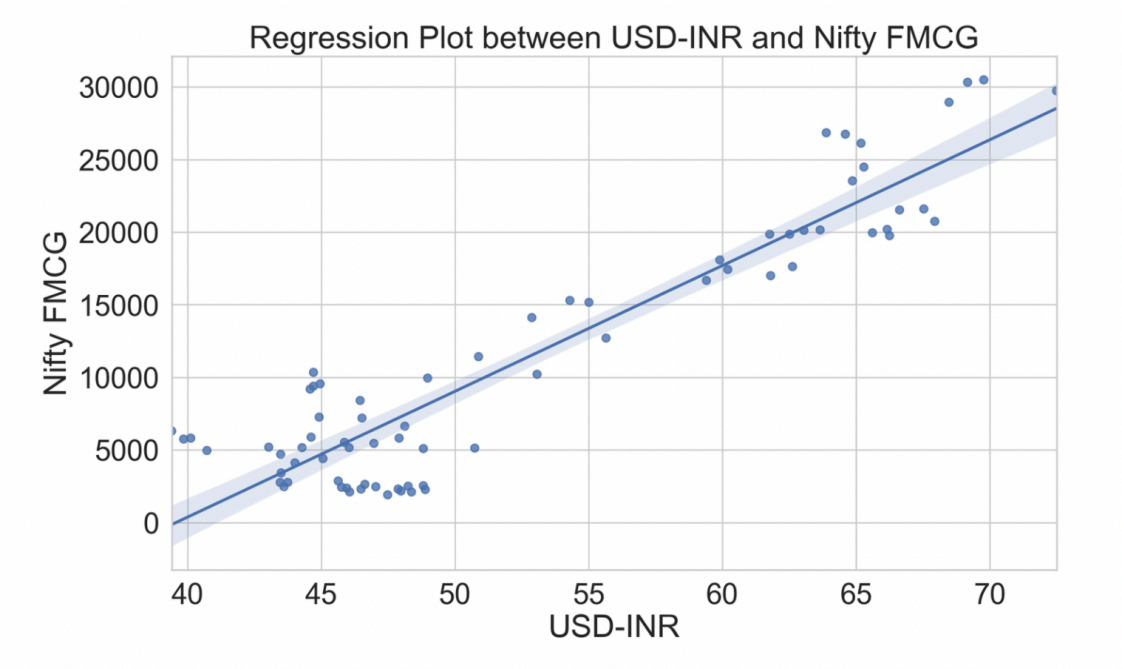
# Simple Linear Regression

- Predicting a quantitative response $y$ on the basis of a single predictor variable $X$
- Univariate regression can be stated as,

$$y_i \approx w_0 + w_1 x_i \qquad \text{here, } i = 1,...,n$$

- $w_0$ and $w_1$ are unknown and response $y$ is predicted using coefficients estimated from the data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Goal is to achieve coefficient estimates such that the linear model fits the available data well.

# Regression Plot



# Estimating Coefficients

- One of the common approaches to measure the closeness of fit is the least squares criterion.
- Let $\hat{y}_i = \hat{w}_0 + \hat{w}_1 x_i$ be the prediction of $y$ based on the $i$ th value of $X$
- Then $e_i = y_i - \hat{y}_i$ represents the $i$ th residual which is the difference between the $i$ th observed and actual response value predicted by the linear model.
- Residual Sum of Squares (RSS) is then defined as $RSS = e_1^2 + e_2^2 + \ldots + e_n^2 = \sum_i^n (y_i - \hat{y}_i)^2$
- Least squares approaches chooses $w_0$ and $w_1$ to minimise RSS where the minimisers are

$$\hat{w}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})\ (y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \quad \text{and} \quad \hat{w}_0 = \hat{y} - \hat{w}_1 \hat{x}\ ,$$

where $\bar{y}$ and $\bar{x}$ are sample means

# Accuracy of the Model

- $R^2$ statistic explains the proportion of variance and always takes on a value between 0 and 1, and is defined as

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

where Total Sum of squares (TSS) is defined as $TSS = \sum_{i}^{n} (y_i - \bar{y}_i)^2$

- TSS is the amount of the variability inherent in the response $y$ before the regression is performed, while RSS measure the amount of variability that is left unexplained after performing regression.
- Hence, TSS - RSS measures the amount of variability in the response that is explained by performing the regression.
- Thus $R^2$ measures the proportion of variability in $y$ that can be explained using $X$

# Evaluation Metrics for Regression

- **RSS - Residual Sum of Square -** used with regression models that output continuous values

$$J(W) = \sum_{i=1}^{n} \left( y_i - w_0 - \sum_{j=1}^{p} x_{ij} w_j \right)^2$$

- **Mean Squared Error -** used with regression models that output continuous values

$$J(W) = \frac{1}{n} \sum_{n=1}^{n} \left( y^{[i]} - f(x^{[i]}; W) \right)^2 = \frac{1}{n} \sum_{i=1}^{n} \left( y_i - w_0 - \sum_{j=1}^{p} x_{ij} w_j \right)^2 \quad (1)$$

# Multi Linear Regression

- Multivariate regression can be stated as,

$$y_i \approx w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_p x_p$$

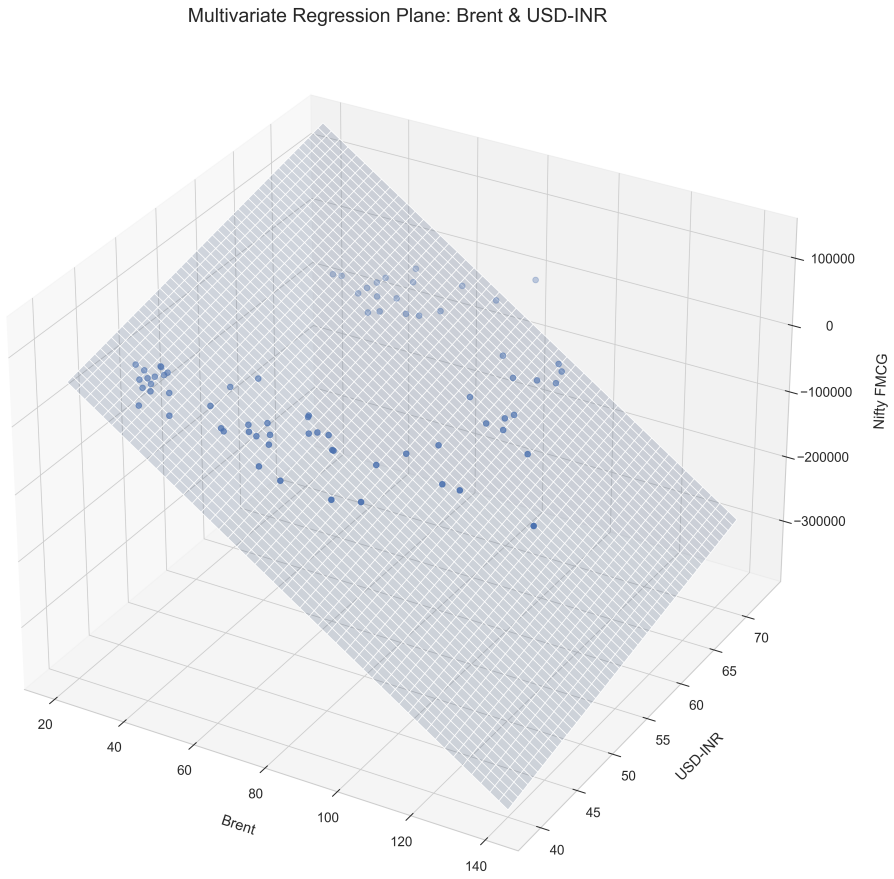
where, $w_0$ is called intercept and $w_1, w_2 \ldots w_p$ are the coefficients of the regression.

- More generally,

$$y_i = w_0 + \sum_{j=1}^{p} w_j X_{ij} + \epsilon_i \,, \qquad i = 1,...,n$$

where, $n$ is the number of samples and $p$ is the number of features.

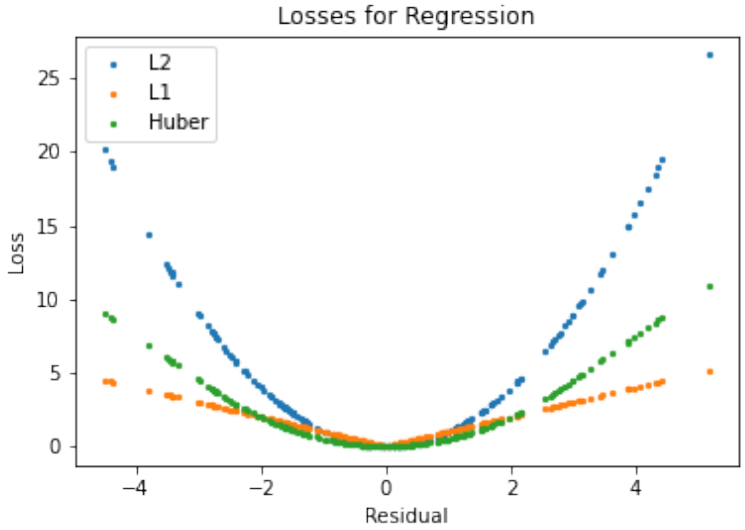
# Regression Plane



# Loss Functions for Regression

- In general, loss function take the form $(y, \hat{y}) \rightarrow l(y, \hat{y}) \in R$
- **Residual**
  - Regression losses usually depend on the residual $r = y - \hat{y}$
  - Loss $l(y, \hat{y})$ is called **distance-based** if it
    - only depends on the residual
    - Loss is zero when residual is 0
    - These losses are translation-invariant, i.e., $l(y + a, \hat{y} + a) = l(y, \hat{y})$
    - Does not capture relative error.

# Loss Functions for Regression



# Loss Functions for Regression

- **Square or $l_2$ Loss**
  - $l(r) = r^2$
  - Quadratic, Differentiable
  - Only local minimum of the loss function is the global minimum
  - Penalises the model for making large errors
  - Outlier dominate objective function
  - Not robust

# Loss Functions for Regression

- **Absolute or Laplace or $l_1$ Loss**
  - $l(r) = |r|$
  - Gives median regression
  - Robust to outliers when compared to Square loss
  - Not differentiable

# Loss Functions for Regression

- **Huber Loss**
  - Combination of Square and Absolute loss
  - Quadratic for $|r| \leq \delta$ and linear for $|r| > \delta$

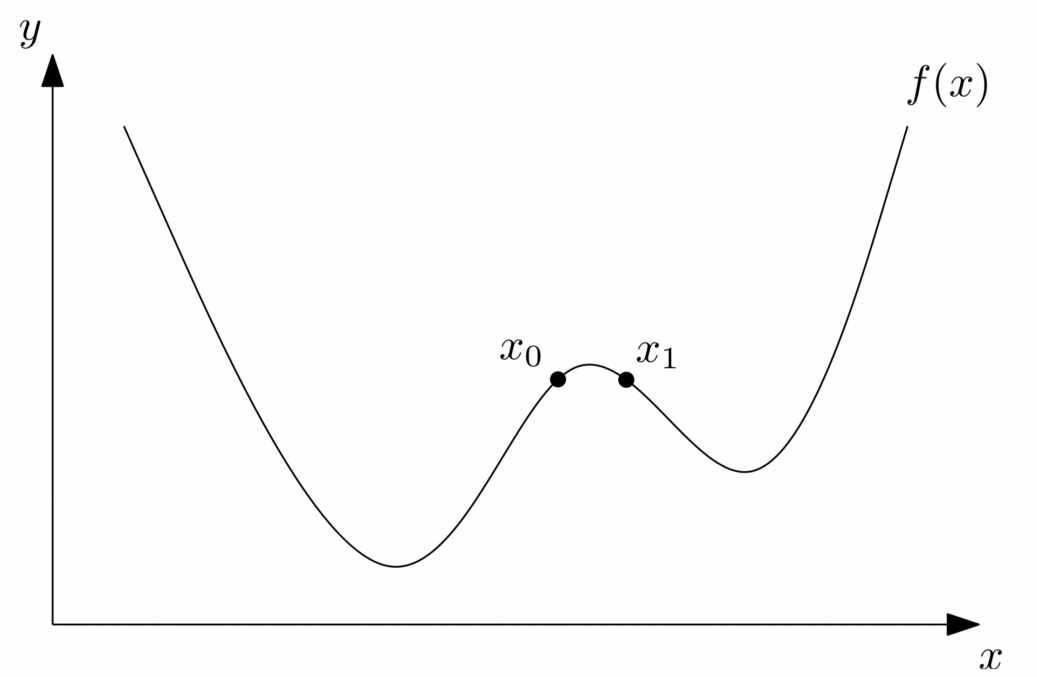
$$L_\delta(r) = \begin{cases} \frac{1}{2} r^2 & for \ \ |r| \leq \delta \\ \delta(|r| - \frac{1}{2}\delta) & otherwise \end{cases}$$

  - Less sensitive to outliers than Square loss
  - Robust and differentiable

# Gradient Descent

- Optimisation algorithm used to find optimal solutions.
- The goal of gradient descent is to minimise a function via greedy local search.
- Gradient descent scales well to large data sets (especially with some tweaks and if an approximately optimal solution is good enough).
- Used to tweak parameters iteratively in a model to minimise a cost function.
- Measures local gradient of the error functions with regards to parameter vector $w$
- Once the gradient is zero, we have reached a minimum.

# Gradient Descent




- If $f'(x) > 0$, f is increasing : move $x$ a little to the left
- If $f'(x) < 0$, move $x$ a little to the right

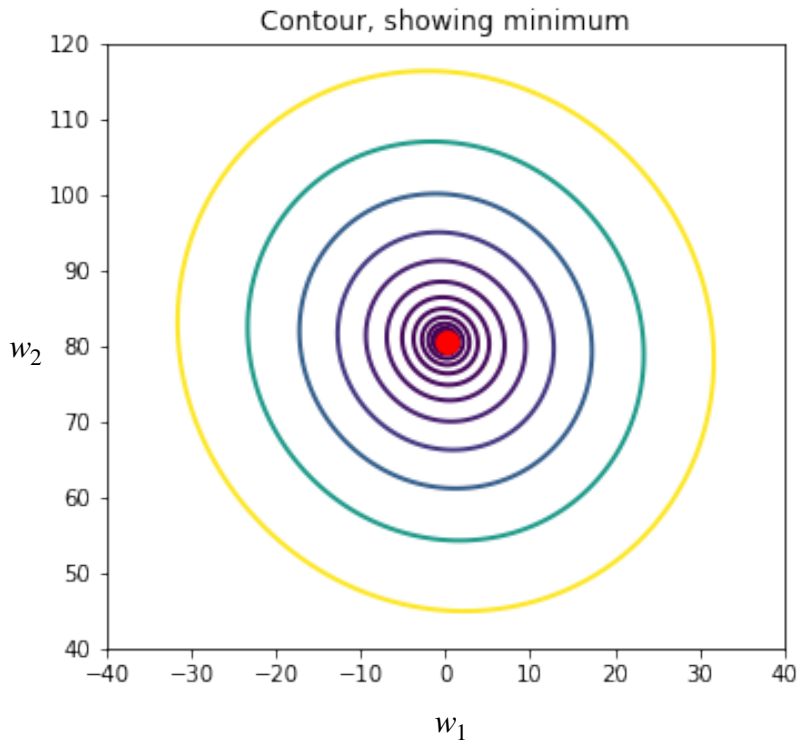
Adapted from The Modern Algorithmic Toolbox. Gradient Descent Basics

# Gradient Descent

- Descent step is given as

$$W_{new} \leftarrow W_{old} - \eta \frac{\partial J(W)}{\partial W}$$

where $\eta$ is the learning rate



# Batch Gradient Descent

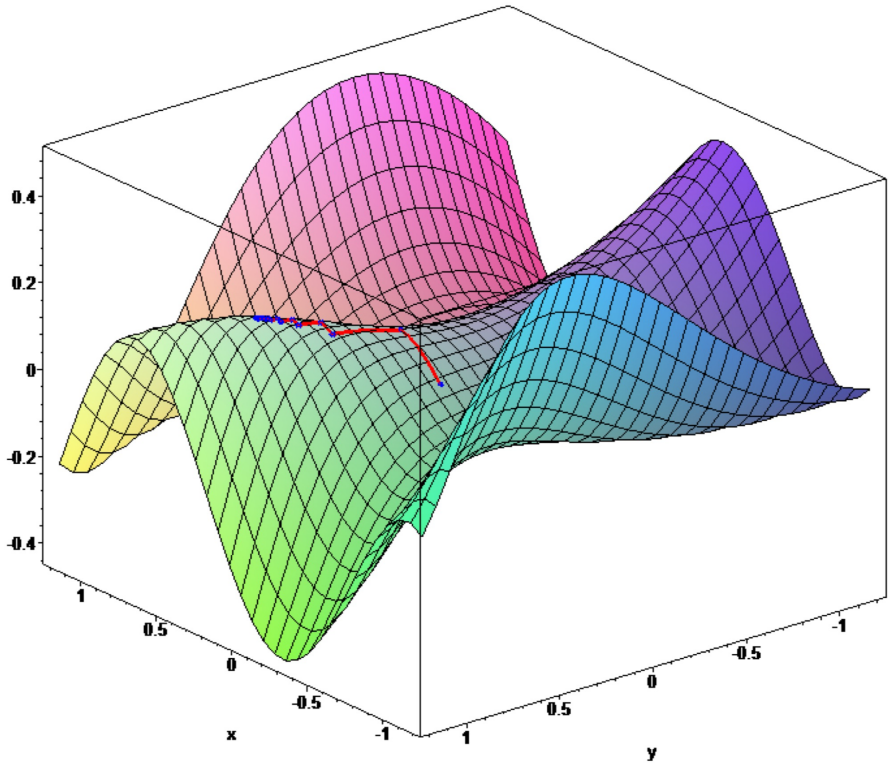



Image source : Wikimedia Commons

# Batch Gradient Descent

- Gradient descent algorithms are computationally expensive
- One idea is to compute gradient using single data point → $\frac{\partial J_i(W)}{\partial W}$
- Single data point (SGD) computation can be very noisy
- Computing gradient by taking batch of points is a good practice
- Mini-batch ensure more accurate estimation of gradient and lead to fast training

$$\frac{\partial J(W)}{\partial W} = \frac{1}{B}\sum_{k=1}^{B}\frac{\partial J_k(W)}{\partial W}$$

- The true gradient is then the average of the gradient from each of those batches

# Batch Gradient Descent

- Descent step is given as

$W_{new} \leftarrow W_{old} - \eta \frac{\partial J(W)}{\partial W}$, where $\eta$ is the learning rate

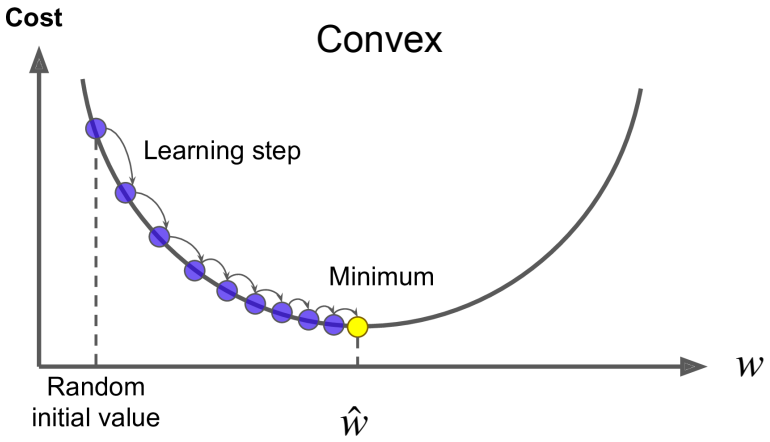
- Batch Size = Size of Training Set → Batch Gradient Descent
- Batch Size = 1 → Stochastic Gradient Descent
- 1 < Batch Size < Size of Training Set → Mini-batch Gradient Descent

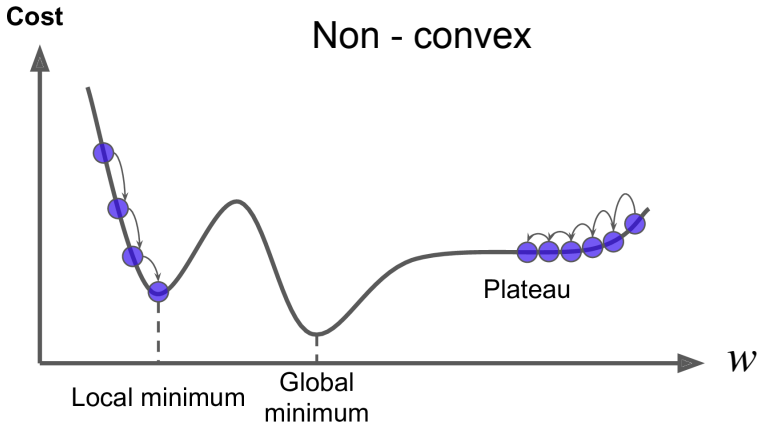
# Learning Rate

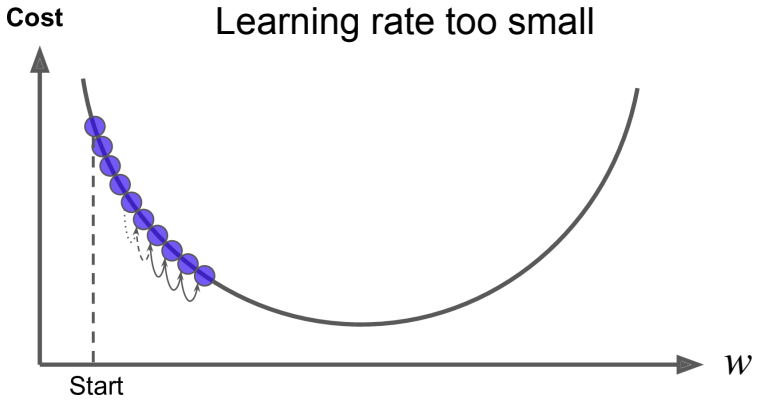
- Tuning parameter in an optimization algorithm that determines the step size at each iteration
- There can be multiple local extremum
- Loss functions can be difficult to optimise
- Small learning rate converges slowly and gets stuck in false local minima
- Large learning rate overshoot and become unstable and diverge

Image source : Aurelien Geron (2019), Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.

# Learning Rate



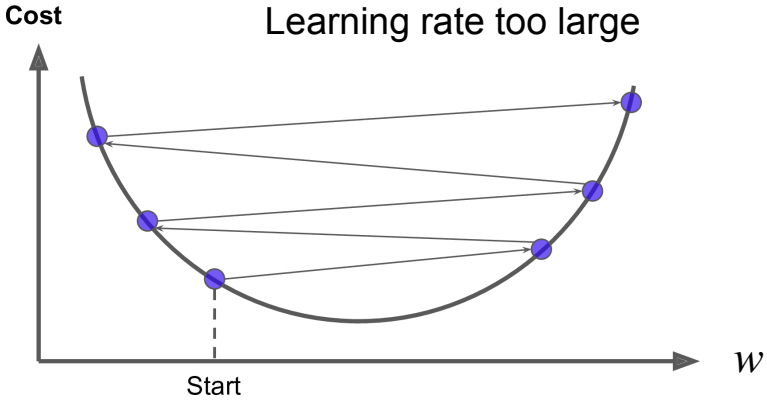
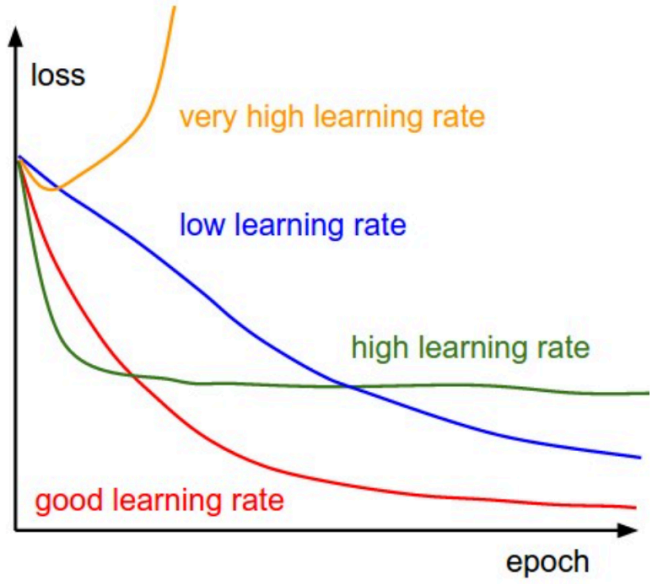




Image source : Aurelien Geron (2019), Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.

# Setting the Learning Rate



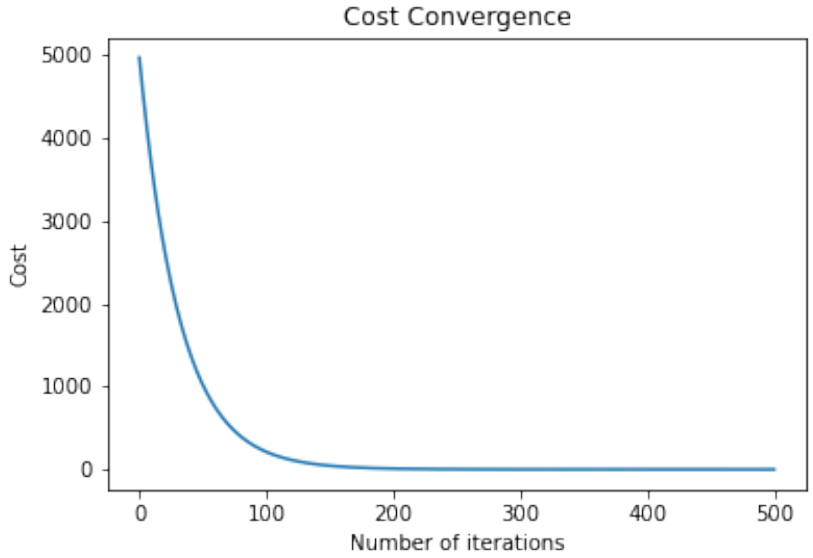




Image source : cs231n

# Setting the Learning Rate

- Gradient descent relies on choosing step-size and convergence criteria
- Selecting adaptive learning rates can address these issues
- Fixed vs Adaptive learning rates

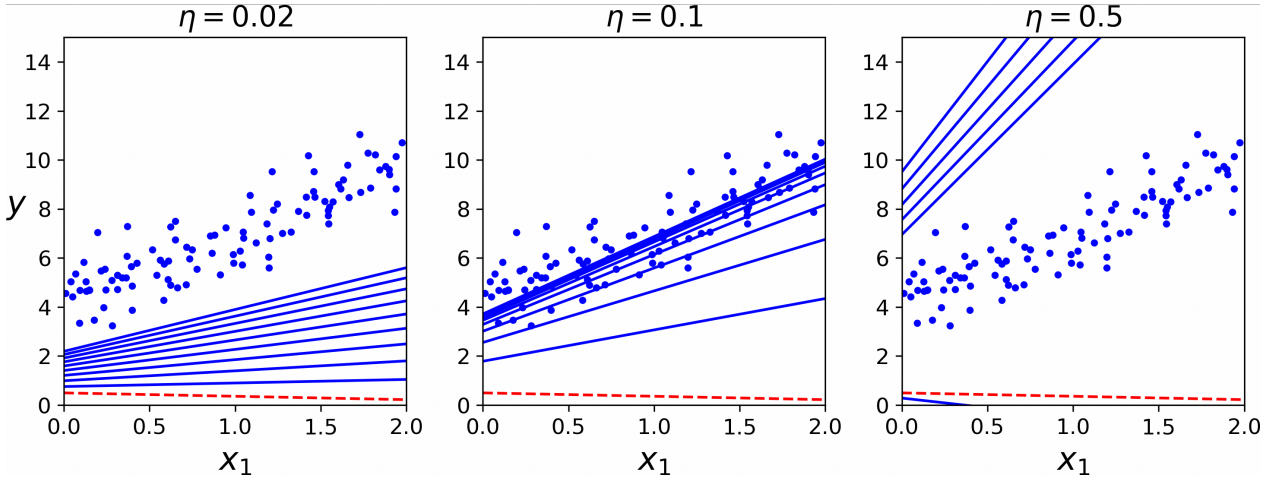



Image source : Aurelien Geron (2019), Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.

# Problem of Overfitting

- Under-fitting is the model that doest not have the capacity to fully learn from the data.
- Overfitting is too complex and does not generalise well with the data as it starts to memorise the training data.

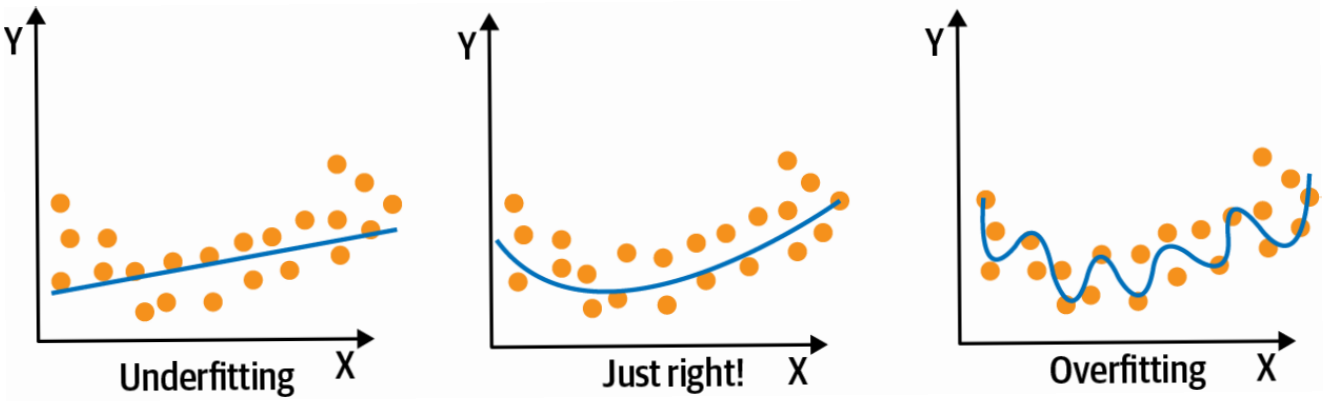



Image source : Hariom Tatsat, Sahil Puri, and Brad Lookabaugh (2020), Machine Learning & Data Science Blueprints for Finance.

# Problem of Overfitting

- The process of fighting overfit is regularisation.
- **Regularisation I: Dropout**
  - Used in Neural Networks where we drop neurons (more on this later)
- **Regularisation II: Early Stopping**
  - Stop training as soon as the validation error reaches a minimum
  - Simple and efficient regularisation technique, the “Beautiful Free Lunch”

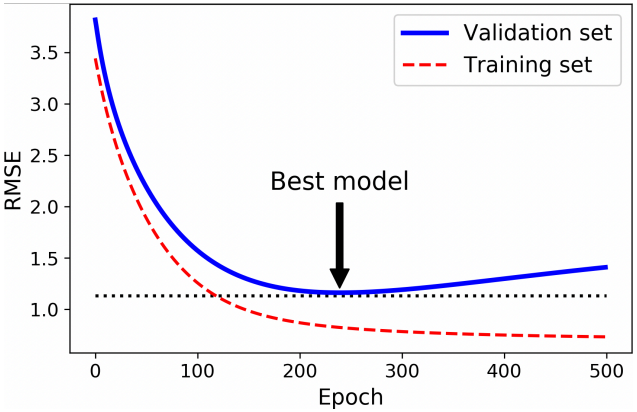



Image source : Aurelien Geron (2019), Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.

# Addressing Overfitting

- Reduce number of features
  - Manual selection of features
  - Model selection algorithms
- Regularisation
  - Keep most (all) features, but reduce the magnitude of the coefficients
  - Penalise large coefficients
  - Better suited for predicting responses with large set of features.

# Resampling Methods

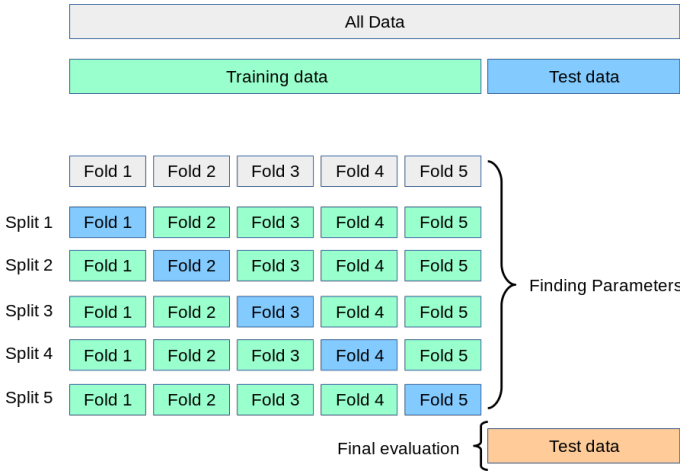
- Draw samples repeatedly from a training set and refit a model on each samples.
- Allow us to obtain information that would not be available from fitting the model only once using the original training set.
- Resampling approaches can be computationally expensive.
- The most commonly used resampling method is “Cross-validation”.
- Cross-validation is used to estimate the test error to
  - Evaluate model performance [model assessment]
  - Select proper level of flexibility [model selection]

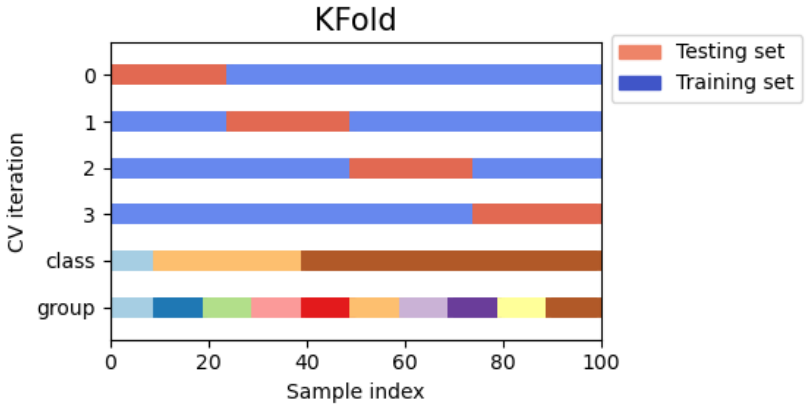
# Validation Set Approach

- A validation dataset is a sample of data held back from training the model that is used in tuning the model's hyperparameters.
- Validation set approach
  - data is divided into three parts : training set, validation set, and the test set.
  - drastically reduce the number of samples which can be used for learning the model,
  - results can depend on a particular random choice for the pair of (train, validation) sets.
  - Overestimate the test error rate.

# Cross Validation

- Cross Validation is a resampling procedure used to evaluate estimator (model) performance where that dataset is split into training set and the test set.




- It is also used to to determine best parameters by GridSearch.
- Strictly do not use $k$-fold CV for time series.

Image source : Scikit-learn User Guide.

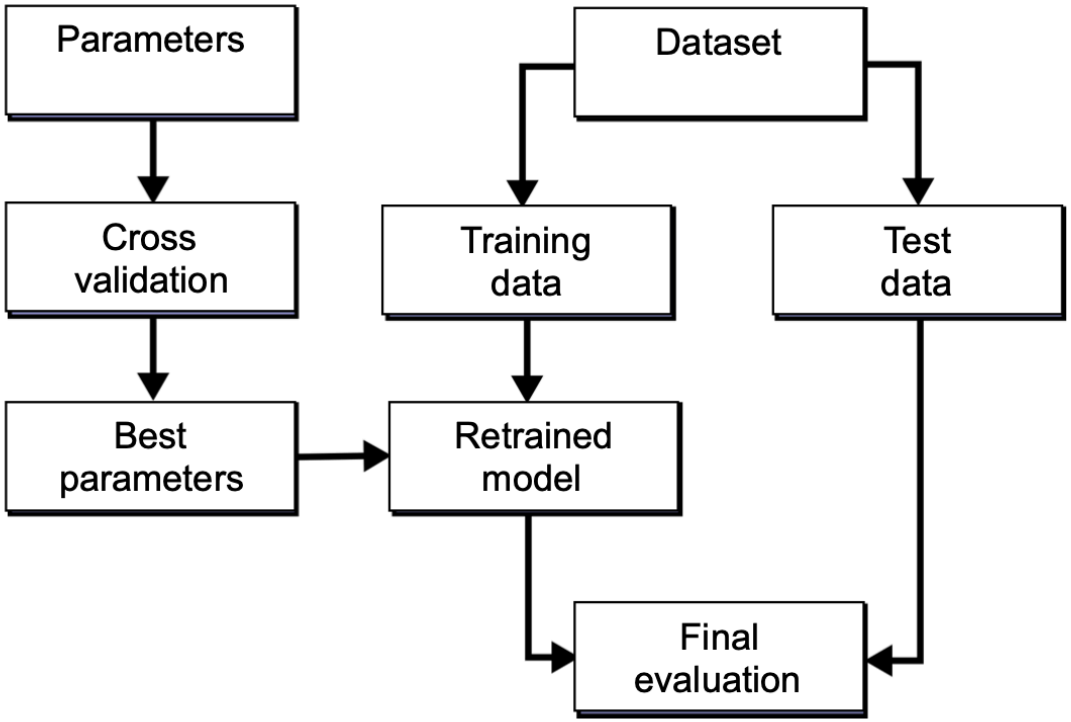
# Cross Validation

- The basic approach of cross validation is $k$-fold CV
  - training set is split into $k$ smaller sets or folds, of ~ equal size.
  - model is trained using $k - 1$ of the folds as training data
  - validation on the remaining part of the data (test data) that is used to compute a performance measure such as accuracy.
  - performance measure by $k$-fold CV is the average values computed in the loop

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i$$

  - computationally expensive, but does not waste too much data

# Cross Validation

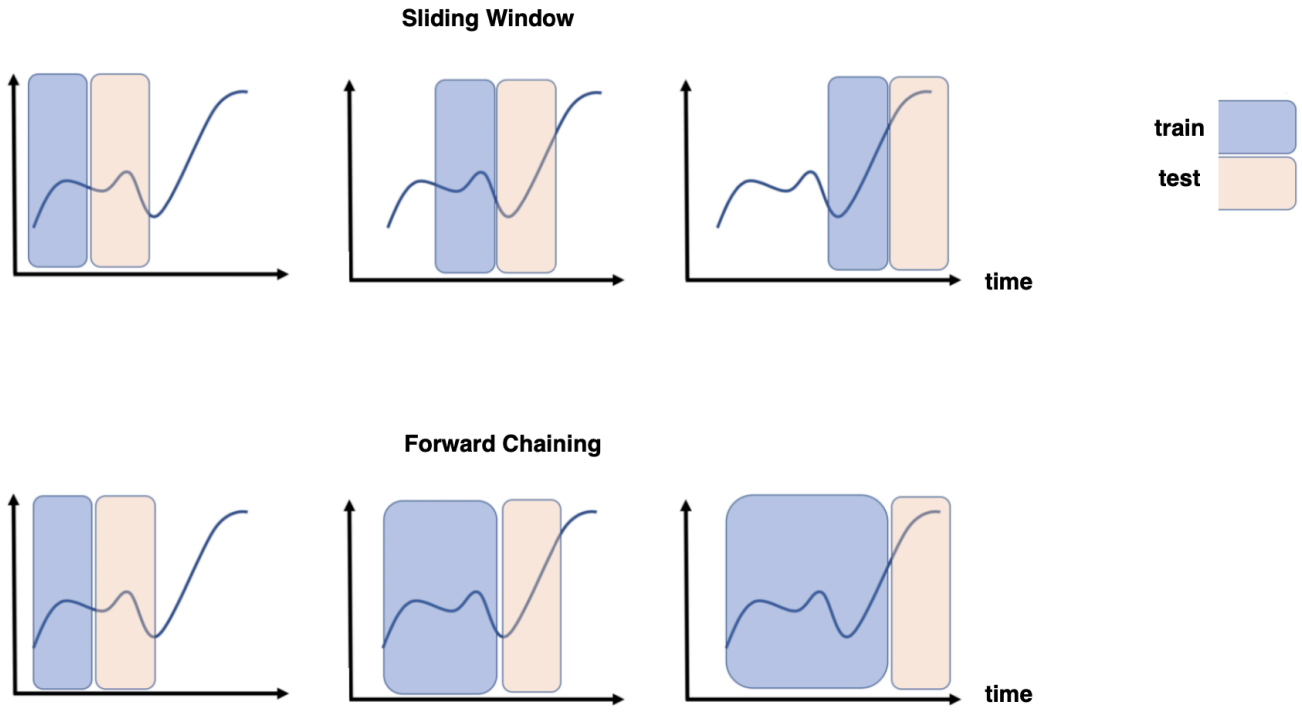



Adapted from Scikit-learn User Guide.

# Cross Validation of Time Series

- Financial time series data are sequential in nature and are characterised by the correlation between observations.
- Classical cross-validation techniques such as $k$-fold assume the samples are independent and identically distributed, and would result in poor estimates when applied on time series data.
- To preserve the order and have training set occur prior to the test set, we use **Time Series Cross Validation** method such as
  - Sliding Window
  - Forward Chaining

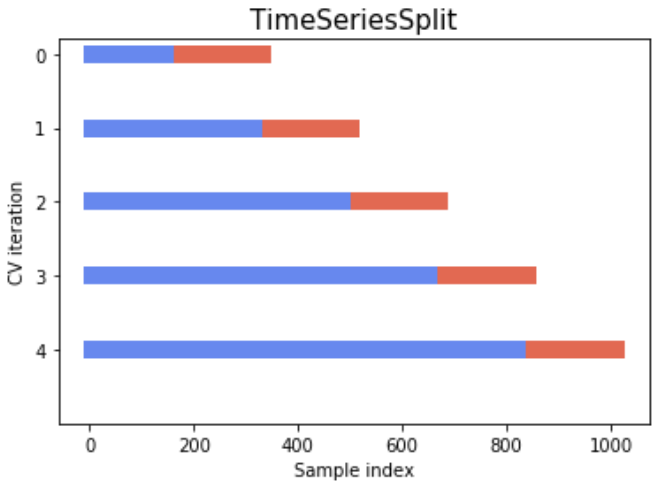
# Cross Validation of Time Series




Refer Python Labs for detailed implementation.

# Cross Validation of Time Series

- Forward chaining cross-validation is a variation from the $k$-fold.
- In the $k^{th}$ split, it returns first $k$ folds as training set and the $(k+1)^{th}$ fold as test set.
- Unlike standard cross-validation method, successive training sets are supersets of those that come before them.



# Shrinkage Methods

- Penalised regression methods are used to reduce the variance of a model through regularisation.
- It can also be used to eliminate redundant variables.
- If the coefficients are too large in linear regression, it can lead to over-fitting.
- Regularisation penalises large coefficients.
- **Lasso** and **Ridge** regression are penalty regression that prevent over-fitting.

# Subset Selection Methods

- Approaches to reduce the number of input variables that are believed to be most useful to a model.
- Divided into three categories depending on how they interact with the classifier.

| Methods | Filter | Wrapper | Embedded |
|---|---|---|---|
| **Approach** | Generic set of methods which do not incorporate a specific machine learning algorithm | Evaluates on a specific machine learning algorithm to find optimal features | Embed features during model building process. Feature selection is done by observing each iteration of model training phase |
| **Time Complexity** | Much faster compared to wrapper methods | High computation time for dataset with many features | Between filter and wrapper methods |
| **Fit** | Less prone to over-fitting | High chances of over-fitting as it involves training of machine learning models with different combination of features | Used to reduce over-fitting by penalizing the coefficients of a model being too large |
| **Examples** | Chi-Square, Anova, Information Gain, Correlation, Variance Threshold | Forward Selection, Backward Elimination, Shap, Boruta | Lasso, Ridge, ElasticNet, Embedded Random Forest / LightGBM |

Refer Python Labs, Machine Learning Exercises and Advanced Machine Learning workshop for application.

# LASSO Regression

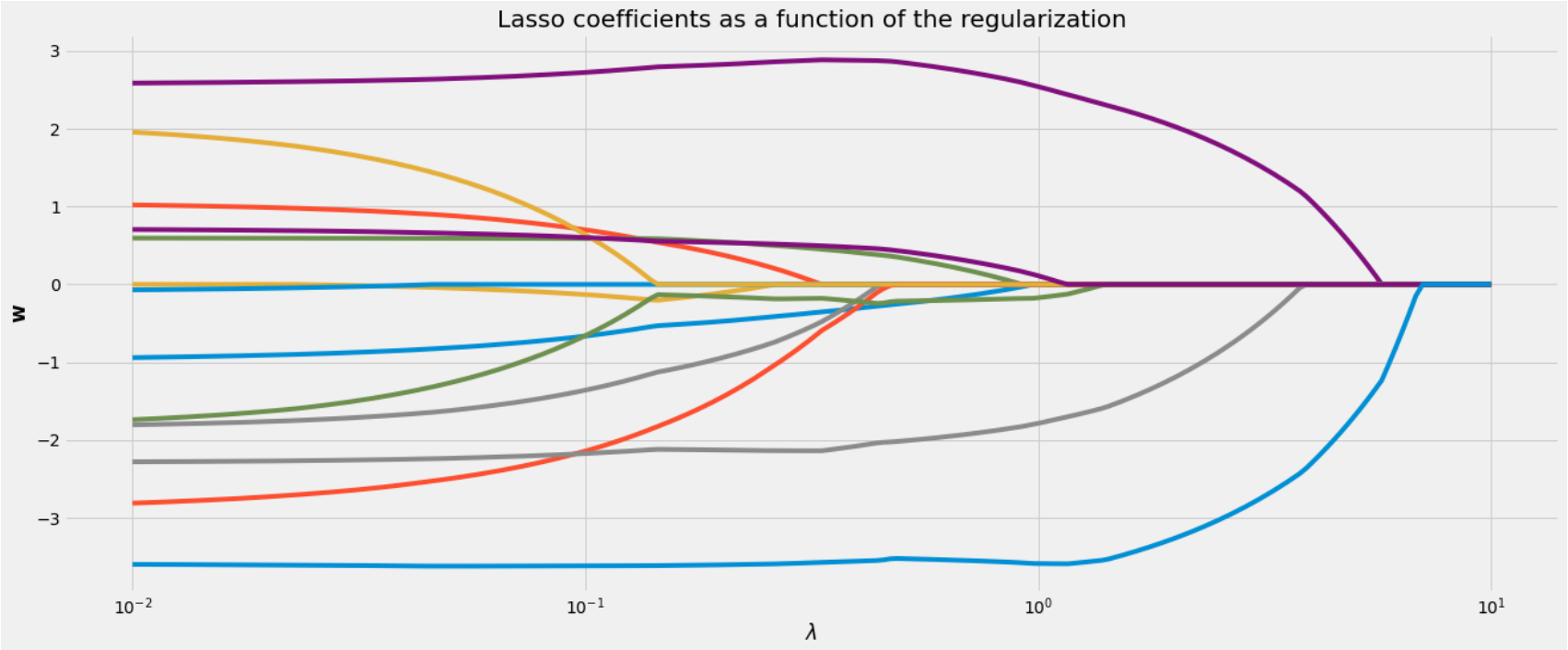
- The Least Absolute Shrinkage and Selection Operator (**LASSO**) is a variation of linear regression.
- In Lasso, the loss function is minimised by limiting the sum of absolute values of the model coefficients where the L1 penalty term is added to the cost function.
- The cost function is thus given as,

$$L = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - w_0 + \sum_{j=1}^{p} x_{ij}w_j\right)^2 + \lambda\sum_{j=1}^{p} \mid w_j \mid \qquad (2)$$

where $\lambda$ is the regularisation penalty.

- The L1 penalty term not only shrinks the coefficients, but shrinks some of them to zero and that is very useful for feature selection.

# LASSO Regression



# Ridge Regression

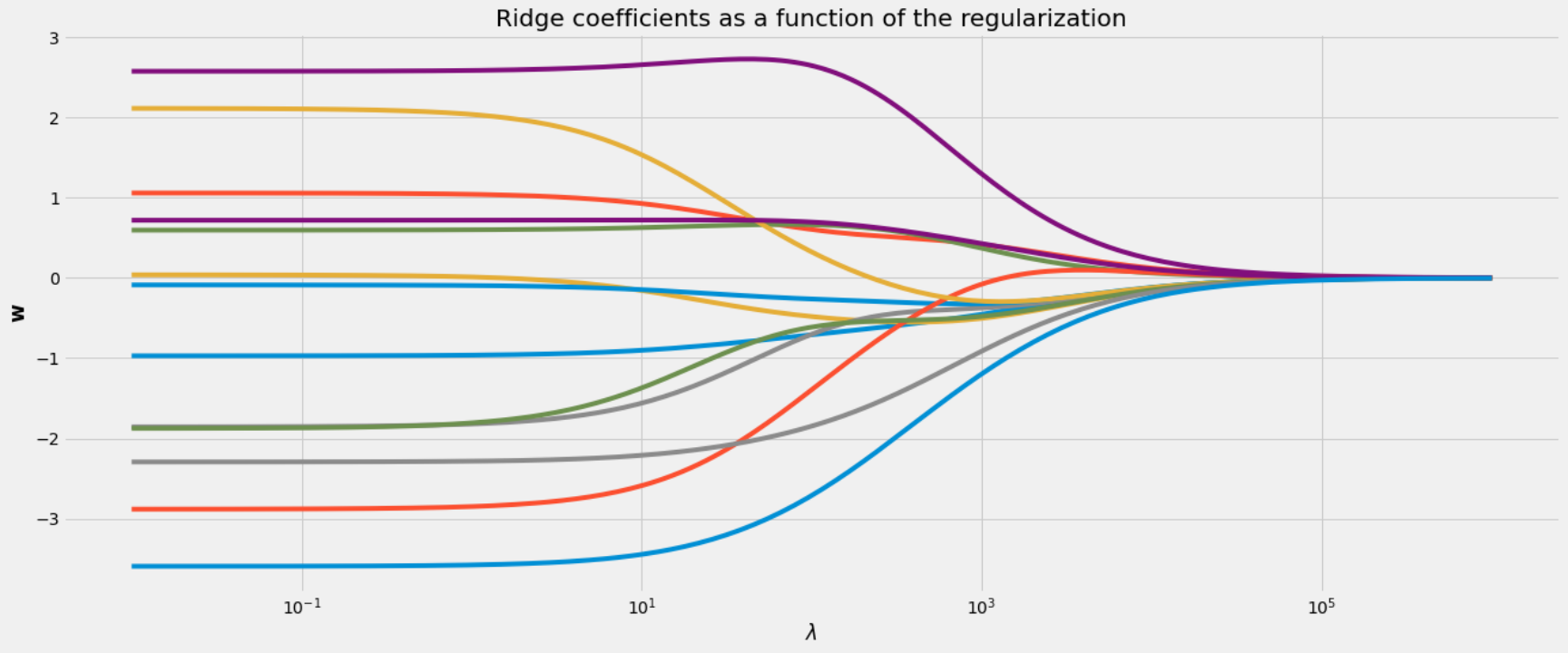
- In ridge regression, the cost function is altered by adding a L2 penalty equivalent to square of the magnitude of the coefficients.
- The cost function is given as,

$$L = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - w_0 - \sum_{j=1}^{p} x_{ij}w_j\right)^2 + \lambda\sum_{j=1}^{p} w_j^2 \qquad (3)$$

where $\lambda$ is the regularisation penalty and when $\lambda \to 0$, the cost function becomes similar to the linear regression cost function.

- The Ridge regression shrinks the coefficients and helps to reduce the multi-collinearity.

# Ridge Regression



# ElasticNet Regression

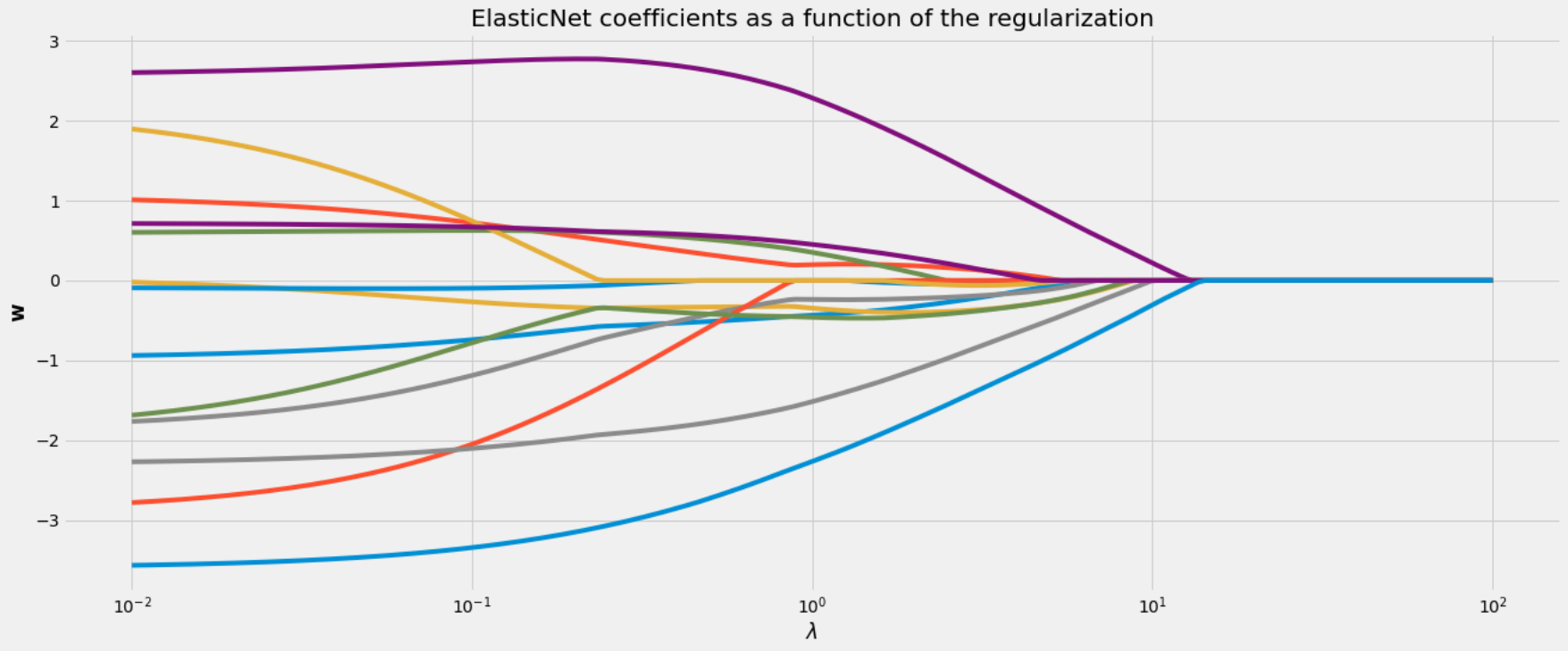
- ElasticNet combines the properties of both Lasso and Ridge regression. It penalises the model using both the L1 and L2 norm.
- Loss function is thus given as,

$$L = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - w_0 - \sum_{j=1}^{p} x_{ij}w_j\right)^2 + \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{p} w_j^2 + \alpha\sum_{j=1}^{p} |\, w_j \,|\right) \quad (4)$$

where $0 \leq \alpha \leq 1$

- ElasticNet is same as lasso when $\alpha = 1$.
- ElasticNet is same as ridge when $\alpha = 0$.
- For other values of $\alpha$, the penalty term interpolate between L1 and L2 norm of the coefficient.

# ElasticNet Regression



# Another Formulation for Regularisation

- An alternative formulation of penalised regression is to view it as constrained optimisation.

$$\underset{W}{minimize} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left( y_i - w_0 + \sum_{j=1}^{p} x_{ij} w_j \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} |w_j| \leq s \tag{5}$$

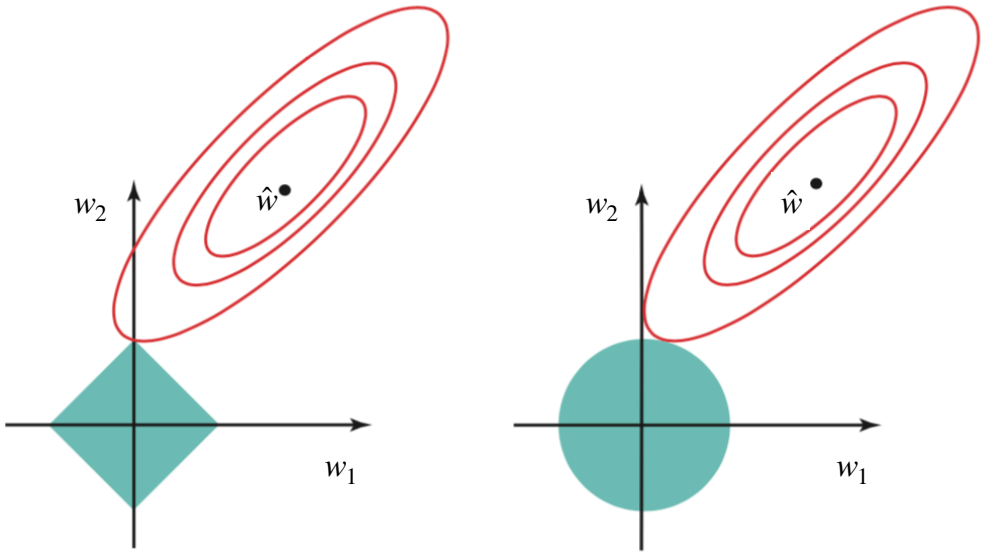
$$\underset{W}{minimize} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left( y_i - w_0 + \sum_{j=1}^{p} x_{ij} w_j \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} w_j^2 \leq s \tag{6}$$

# Another Formulation for Regularisation

- For every value of $\lambda$, there is some $s$ such that the equations (2) and (5) will give the same lasso coefficient estimates.

- Similarly, for every value of $\lambda$, there is some $s$ such that the equations (3) and (6) will give the same ridge coefficient estimates.

# Another Formulation for Regularisation




Adapted from Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani (2013), An Introduction to Statistical Learning.

# Another Formulation for Regularisation

- Equation $(5)$ indicates that the lasso coefficient estimates have the smallest MSE out of all points that lie within the diamond defined by $|w_1| + |w_2| \leq s$
- Equation $(6)$ indicates that the ridge coefficient estimates have the smallest MSE out of all points that lie within the circle defined by $w_1^2 + w_2^2 \leq s$
- Large value of $s$ corresponds to $\lambda = 0$
- If $s$ is sufficiently large, then the constraint regions will contain $w$, and so the ridge regression and lasso estimates will be the same as the least square estimates.
- Least squares estimates lying outside of the diamond and the circle are not the same as the lasso and ridge regression estimates.

Adapted from Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani (2013), An Introduction to Statistical Learning.

# References


- Scott Fortmann-Roe (2012) , Understanding the Bias-Variance Tradeoff
- Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani (2013), An Introduction to Statistical Learning
- Sebastian Raschka (2015), Python Machine Learning
- Francois Chollet (2017), Deep Learning with Python
- Aurelien Geron (2019), Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
- Hariom Tatsat, Sahil Puri, and Brad Lookabaugh (2020), Machine Learning & Data Science Blueprints for Finance
- D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Franc¸ois Crespo, Dan Dennison (2015), Hidden Technical Debt in Machine Learning Systems, Google Inc.
- Motunrayo Olugbenga (2022), Balanced Accuracy: When Should You Use It?
- Scikit-learn User Guide and Documentation
- Google Developers, Introduction to Machine Learning