DEFENCE INSTITUTE OF ADVANCED TECHNOLOGY

DEEMED UNIVERSITY

शास्त्रेण रक्षाम्

शस्त्रं प्रकरोति

Machine Learning for Cyber Security (CS-602)

L#05

Linear Regression Analysis – Part1

By

Dr Sunita Dhavale

Syllabus

- Data Analytics Foundations: R programming, Python Basics -Expressions and Variables, String Operations, Lists and Tuples, Sets, Dictionaries Conditions and Branching, Loops, Functions, Objects and Classes, Reading/Writing files, Hand ling data with Pandas, Scikit Library, Numpy Library, Matplotlib, scikit programming for data analysis, setting up lab environment, study of standard datasets. Introduction to Machine Learning- Applications of Machine Learning, Supervised, unsupervised classification and regression analysis
- Python libraries suitable for Machine Learning Feature Extraction. Data pre-processing, feature analysis etc., Dimensionality Reduction & Feature Selection Methods, Linear Discriminant Analysis and Principal Component Analysis, tackle data class imbalance problem

Syllabus

- Supervised and regression analysis, Regression, Linear Regression, Non-linear Regression, Model evaluation methods, Classification, K-Nearest Neighbor, Naïve Bayes, Decision Trees, Logistic Regression, Support Vector Machines, Artificial Neural Networks, Model Evaluation. Ensemble Learning, Convolutional Neural Networks, Spectral Embedding, Manifold detection and Anomaly Detection
- Unsupervised classification K-Means Clustering, Hierarchical Clustering, Density-Based Clustering, Recommender Systems-Content-based recommender systems, Collaborative Filtering, machine learning techniques for standard dataset, ML applications, Case studies on Cyber Security problems that can be solved using Machine learning like Malware Analysis, Intrusion Detection, Spam detection, Phishing detection, Financial Fraud detection, Denial of Service Detection.

Text/Reference Books

1. Building Machine Learning Systems with Python – Willi Richert, Luis Pedro Coelho
2. Alessandro Parisi, Hands-On Artificial Intelligence for Cybersecurity: Implement smart AI systems for preventing cyber attacks and detecting threats and network anomalies Publication date :Aug 2, 2019, Packt, ISBN-13, 9781789804027
3. Machine Learning: An Algorithmic Perspective – Stephen Marsland
4. Sunita Vikrant Dhavale, "Advanced Image-based Spam Detection and Filtering Techniques", IGI Global, 2017
5. Soma Halder , Sinan Ozdemir, Hands-On Machine Learning for Cybersecurity: Safeguard your system by making your machines intelligent using the Python ecosystem, By Publication date : Dec 31, 2018, Packt, ISBN-13 :9781788992282

1. Stuart Russell, Peter Norvig (2009), "Artificial Intelligence – A Modern Approach", Pearson Elaine Rich & Kevin Knight (1999), "Artificial Intelligence", TMH, 2nd Edition
2. NP Padhy (2010), "Artificial Intelligence & Intelligent System", Oxford
3. ZM Zurada (1992), "Introduction to Artificial Neural Systems", West Publishing Company
4. Research paper for study (if any) – White papers on multimedia from IEEE/ACM/Elsevier/Spinger/ Nvidia sources.

Lab assignments

1	**Python Programming part-1**
2	Python Programming part-2
3	Study and Implement Linear Regression Algorithm for any standard dataset like in cyber security domain
4	Study and Implement KMeans Algorithm for any standard dataset in cyber security domain
5	Study and Implement KNN for any standard dataset in cyber security domain
6	Study and Implement ANN for any standard dataset in cyber security domain
7	Study and Implement PCA for any standard dataset in cyber security domain
8	Case Study: Use of ML along with Fuzzy Logic/GA to solve real world Problem in cyber security domain
9	Mini assignment: Apply ML along with PSO/ACO to solve any real world problem in cyber security domain
10	ML Practice Test – 1 Quiz

Defence Institute of Advanced Technology

School of Computer Engineering & Mathematical Sciences

SEMESTER-I TIME TABLE (AUTUMN 2024)$

PROGRAMMES: (I) CS [M.TECH IN CYBER SECURITY] (II) AI [M.TECH CSE (ARTIFICIAL INTELLIGENCE)] BATCH: 2024-2026

Lecture / Day	L1 0900-1000	L2 1000-1100	L3 1100-1200	L4 1200-1300		L4 1400-1500	L4 1500-1600	L4 1600-1700	L4 1700-1800
Monday	CE-602 (AI) CS-602 (CS)	CE-604 (AI) CS-603 (CS)	CE-601 (AI) CS-604 (CS)	CE-601 (AI) LAB CS-603 (CS)	Lunch Break 1300-1400	LAB CE-601 (AI) LAB CS-602 (CS)		AM607	
Tuesday	CE-603 (AI) LAB CS-603 (CS)	CE-602 (AI) CS-602 (CS)	CE-601 (AI) CS-605 (CS)	CE-604 (AI) CS-604 (CS)		PGC 601		AM607	
Wednesday	CS-605 (CS)	CE-603 (AI) CS-602 (CS)	CE-602 (AI) CS-603 (CS)	CE-604 (AI) CS-604 (CS)		CE-605(AI) LAB CS-605 (CS)	LAB CS-605 (CS)	AM607	
Thursday	LAB CE-604 (AI) CS-603 (CS)	LAB CE-604 (AI) CS-605 (CS)	LAB CE-602 (AI) CS-601 (CS)	CE-603 (AI) CS-601 (CS)		PGC 601		AM607	
Friday	LAB CE-603 (AI) LAB CS-601 (CS)		LAB CE-602 (AI) CS-601 (CS)	LAB CS-604 (CS)		CE-605(AI) LAB CS-604 (CS)	CE-605(AI)	LAB CE-605(AI)	

COURSE CODE & COURSE NAME		FACULTY
Programme: CS [M.Tech in Cyber Security] Classroom: Arjun	Programme: AI [M.Tech CSE (Artificial Intelligence)] Classroom: Kaveri	
CS-601 Data Security & Privacy	CE-601 Responsible Artificial Intelligence;	MJN: Dr. Manisha J. Nene
CS-602 ML for Cyber Security	CE-604 Practical Machine Learning;	SVD: Dr. Sunita V. Dhavale
CS-605 Network and Cloud Security	CE-602 Intelligent Algorithms	CRS: Prof. CRS Kumar
CS-604 Advanced System Security	---------	DVV: Dr. Deepti V. Vidyarthi
CS-603 Applied Cryptography	---------	AM: Dr. Arun Mishra
---------	CE-603 Deep Neural Network;	US: Dr. Upasna Singh
---------	CE-605 Mathematics for ML;	Unit-2: Dr Upasna, Unit 4: Dr Sunita, Unit3:MJN, Unit 1: Faculty To be Nominated
AM-607 Mathematics for Engineers	AM-607 Mathematics for Engineers	OO/DS/DP: Dr Odellu O., Dr Dasari S., Dr. Debasis P.
PGC-601 Research Methodology	PGC-601 Research Methodology	Common Subject for All

$ TENTATIVE T.T. SUBJECT TO CHANGE

Program Coordinator,
M.Tech (CS & AI), Batch 2024-26

Director, SoCE&MS

Linear Regression Analysis

Motivation

- Purpose is to build a functional relationship (model) between dependent variable(s) and independent variable(s)
- Dependent variables also known as Response variable, Regressand, Predicted variable, output variable - denoted as y
- Independent variable also known as Predictor variable, Regressor, Exploratory variable, input variable, features- denoted as x
- Example Business :
  - What is the effect of price on sales? (Can be used to fix the selling price of an item)

Introduction

- Regression techniques are a category of supervised machine learning algorithms used for investigating relationships between dependent variable **(Y)** and one/more independent variables **(X)**.
- It is widely used for prediction and forecasting.
- dependent variable is continuous, e.g. price of wheat in the world market, the number of deaths from lung cancer etc.
- independent variable(s) can be continuous or discrete
- nature of regression line is linear.

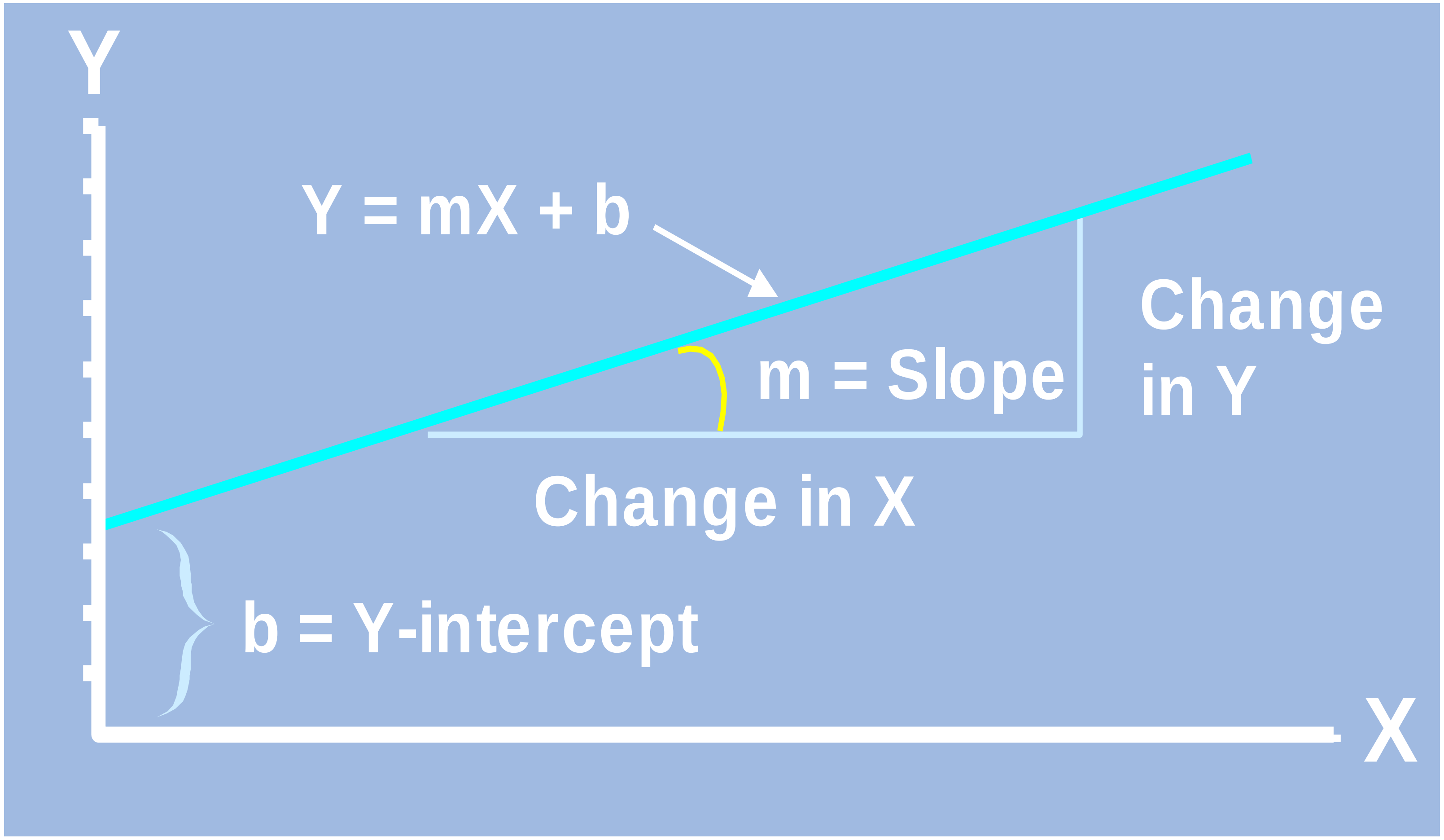
Classification of Regression Analysis

- Univariate vs Multivariate
  - Univariate: single independent variable x
  - Multivariate: Multiple independent variable (x1,x2,…xn)
- Linear vs Nonlinear
  - Linear: Relationship is linear between dependent and independent variables, rate of change of a function is constant
  - Nonlinear: Relationship is nonlinear between dependent and independent variables, rate of change of a logarithmic function is not constant, $f(x) = x^2$, $f(x) = x^3 - 3x$, $f(x) = 2^x$

Introduction – simple Linear regression

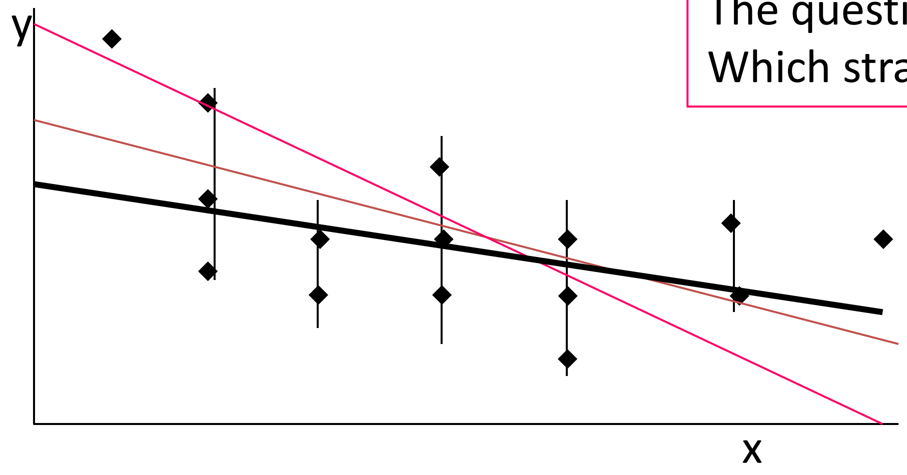
- It establishes a relationship between **(Y)** and **(X)** using a **best fit straight line** (also known as regression line).
- E.g. slope-intercept format equation **Y=a+b*X** + e, where a is intercept, b is slope of the line and e is error term.
- model coefficients/parameters calculated from training data
- This equation can be used to predict the value of target variable based on given predictor variable(s).

Example



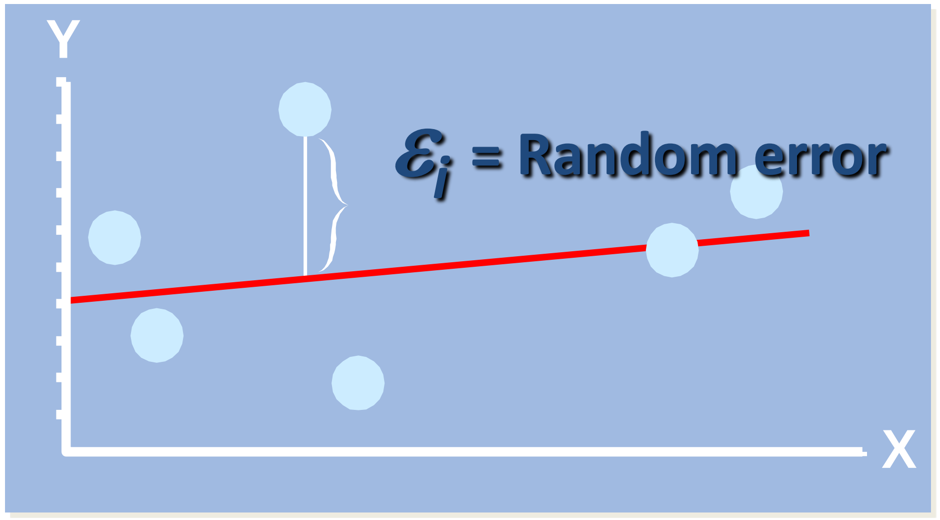
Estimating Coefficients

- Use scatterplot to add points in dataset
- The estimates are determined by
  - drawing a sample from the population of interest,
  - calculating sample statistics.
  - producing a straight line that cuts into the data.



The question is:
Which straight line fits best?

Functional relationship between the Yi and Xi is the equation of a straight line



$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Observed value

β0 is the intercept, the value of Yi when X = 0,
i= 1, 2,...,n where, total n observations
and β1 is the slope of the line, the rate of change in Yi per unit change in X
B0 and β1 =>regression parameters.

Estimating Coefficients

- simple linear model has two parameters β0 and β1, which are to be estimated from the data
- If there were no random error in Yi, any two data points could be used to solve explicitly for the values of the parameters.
- The random variation in Y , however, causes each pair of observed data points to give different results.
- All estimates would be identical only if the observed data fell exactly on the straight line.
- A method is needed that will combine all the information to give one solution which is “best” by some criterion

Ordinary Least Squares Estimation

- uses the criterion that the solution must give the smallest possible sum of squared deviations of the observed Yi from the estimates of their true means provided by the solution
- 'Best Fit' Means Difference Between Actual Y Values & Predicted Y Values is a Minimum. *But* Positive Differences Off-Set Negative ones. So square errors!
- objective of linear regression to be J(θ1, θ0) to be minimized.
- LS Minimizes the Sum of the Squared Differences (errors) (SSE)

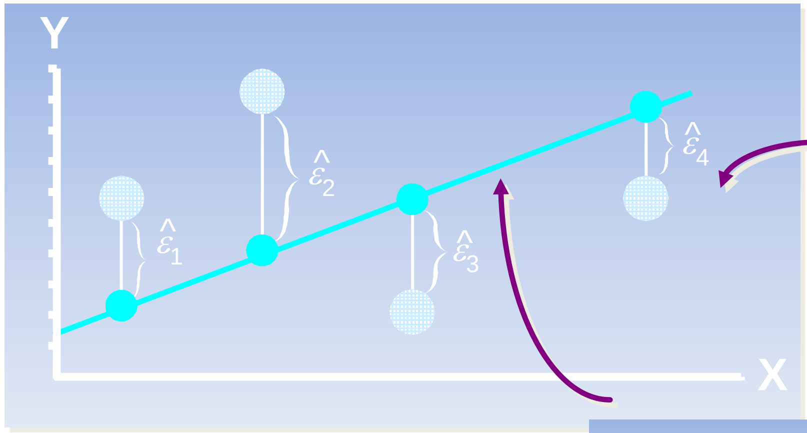
$$y_i = \theta_1 x_i + \theta_0 + \varepsilon_i$$

$$J(\theta_1, \theta_0) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \left(y_i - \theta_1 x_i - \theta_0\right)^2$$

The least squares principle chooses β0 and β1 that minimize the sum of squares of the residuals, SS(Res)

$$\begin{aligned} SS(Res) &= \sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2 \\ &= \sum e_i^2, \end{aligned}$$

LS minimizes $\sum_{i=1}^{n} \hat{\varepsilon}_i^2 = \hat{\varepsilon}_1^2 + \hat{\varepsilon}_2^2 + \hat{\varepsilon}_3^2 + \hat{\varepsilon}_4^2$



$$Y_2 = \hat{\beta}_0 + \hat{\beta}_1 X_2 + \hat{\varepsilon}_2$$

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$$

$e_i = (Y_i - \widehat{Y}_i)$ is the observed residual for the ith observation.

Derivation of Parameters (1)

- Least Squares (L-S):

  Minimize squared error

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

$$\sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

$$\sum_{i=1}^{n} \left(\mathbf{Y}_i - \hat{\mathbf{Y}}_i\right)^2 = \sum_{i=1}^{n} \hat{\varepsilon}_i^2$$

$$0 = \frac{\partial \sum \varepsilon_i^2}{\partial \beta_0} = \frac{\partial \sum (y_i - \beta_0 - \beta_1 x_i)^2}{\partial \beta_0}$$

$$= -2\left(n\bar{y} - n\beta_0 - n\beta_1 \bar{x}\right)$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Derivative is just the rate of change of the function, i.e. slope.
The maximum or minimum of a differentiable function can be attained only at points where the derivative is zero.

Derivation of Parameters (1)

- Least Squares (L-S):
  Minimize squared error

$$0 = \frac{\partial \sum \varepsilon_i^2}{\partial \beta_1} = \frac{\partial \sum \left(y_i - \beta_0 - \beta_1 x_i\right)^2}{\partial \beta_1}$$

$$= -2 \sum x_i \left(y_i - \beta_0 - \beta_1 x_i\right)$$

$$= -2 \sum x_i \left(y_i - \bar{y} + \beta_1 \bar{x} - \beta_1 x_i\right)$$

$$\hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

Coefficient Equations

- Prediction equation

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

- Sample slope

$$\hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

- Sample Y - intercept

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

more convenient forms for hand computation of sums of squares and sums of products

$$x_i = (X_i - \overline{X}) \text{ and } y_i = (Y_i - \overline{Y})$$

$$\begin{aligned}\widehat{\beta}_1 &= \frac{\sum(X_i - \overline{X})(Y_i - \overline{Y})}{\sum(X_i - \overline{X})^2} = \frac{\sum x_i y_i}{\sum x_i^2} \\ \widehat{\beta}_0 &= \overline{Y} - \widehat{\beta}_1 \overline{X}.\end{aligned}$$

$$\begin{aligned}\sum x_i^2 &= \sum X_i^2 - \frac{(\sum X_i)^2}{n} \\ \sum x_i y_i &= \sum X_i Y_i - \frac{(\sum X_i)(\sum Y_i)}{n}\end{aligned}$$

$$\widehat{\beta}_1 = \frac{\sum X_i Y_i - \frac{(\sum X_i)(\sum Y_i)}{n}}{\sum X_i^2 - \frac{(\sum X_i)^2}{n}}$$

$$\widehat{Y}_i = \widehat{\beta}_0 + \widehat{\beta}_1 X_i.$$

Assumptions behind Linear Regression

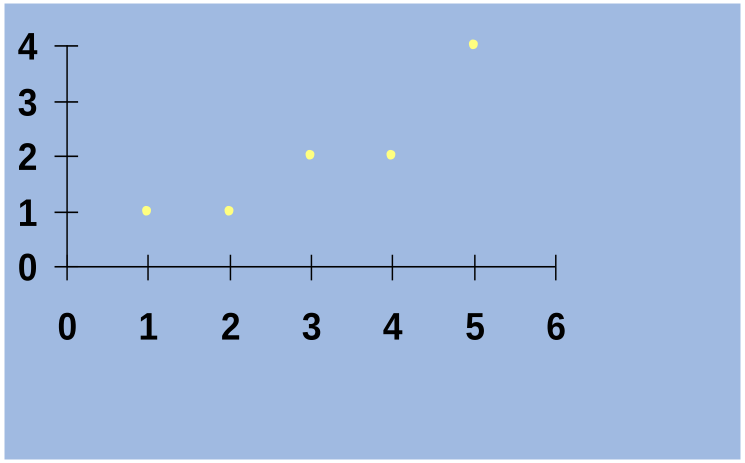
- The data used in fitting the model are representative of the population.
- Linearity: The true underlying relationship between X and Y i.e. independent and dependent variables is linear.
- Homoscedasticity: The variance of residual is the same for any value of X. The variance of the residuals is constant.
- Independence: Observations are independent of each other. No Autocorrelation in residuals. The residuals must be independent.
- Normality: For any fixed value of X, the residuals are normally distributed.
- Number of observations Greater than the number of predictors/number of independent variables
- No Multi-colinearlity in the data – i.e. avoid highly correlated features, As high collinearity -> two variables vary very similarly and contain the same kind of information->redundancy in the dataset -> only the complexity of the model increase -> no new information or pattern is learned by the model

Parameter Estimation Example

- Obstetrics: What is the **relationship** between Mother's Estriol level & Birthweight using the following data?

Estriol (mg/24h)	**Birthweight** (g/1000)
1	1
2	1
3	2
4	2
5	4

Scatterplot



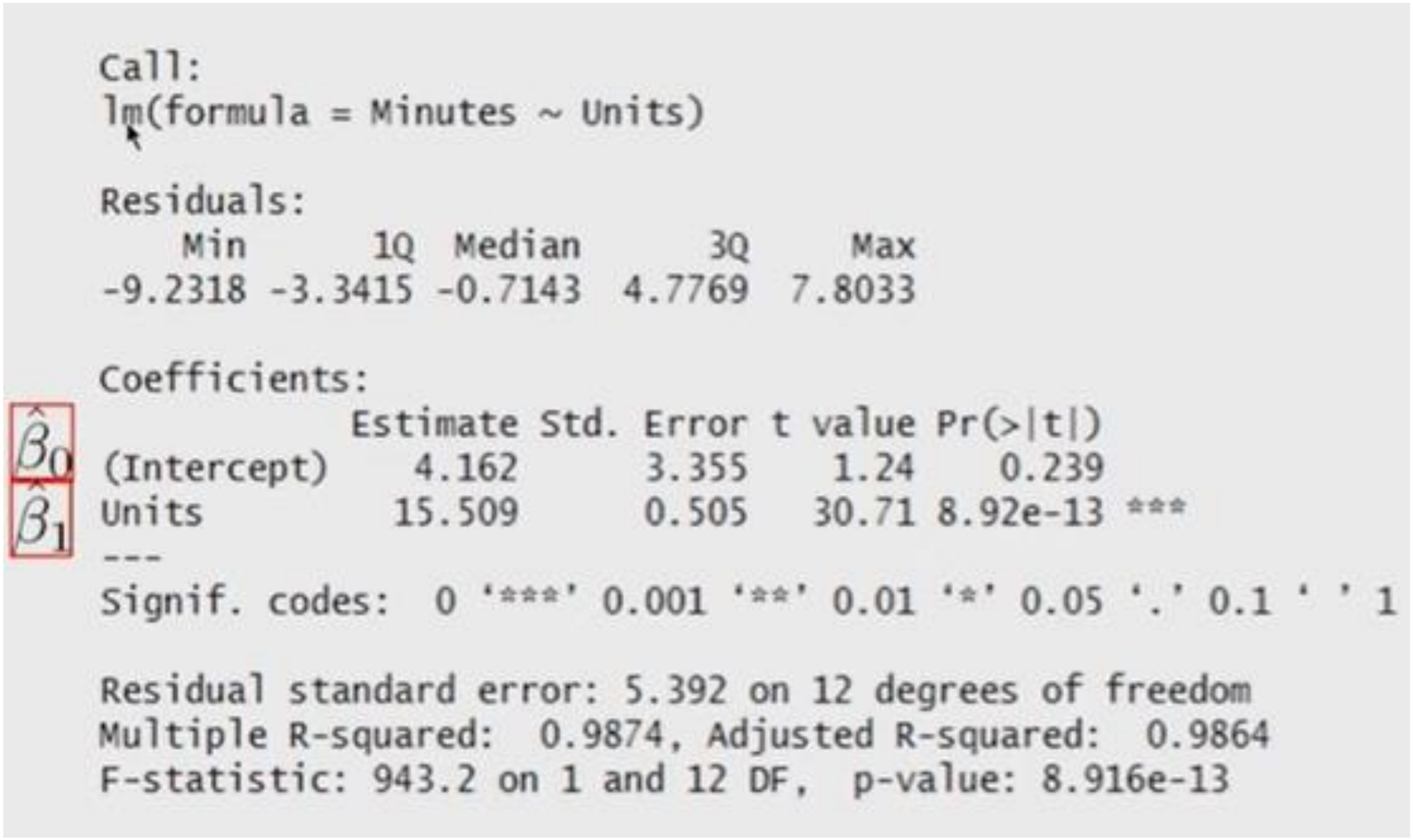
X_i	Y_i	X_i^2	Y_i^2	X_iY_i
1	1	1	1	1
2	1	4	1	2
3	2	9	4	6
4	2	16	4	8
5	4	25	16	20
15	10	55	26	37

Parameter Estimation Solution

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} X_i Y_i - \frac{\left(\sum_{i=1}^{n} X_i\right)\left(\sum_{i=1}^{n} Y_i\right)}{n}}{\sum_{i=1}^{n} X_i^2 - \frac{\left(\sum_{i=1}^{n} X_i\right)^2}{n}} = \frac{37 - \frac{(15)(10)}{5}}{55 - \frac{(15)^2}{5}} = 0.70$$

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X} = 2 - (0.70)(3) = -0.10$$

R command



```
Call:
lm(formula = Minutes ~ Units)

Residuals:
    Min      1Q  Median      3Q     Max 
-9.2318 -3.3415 -0.7143  4.7769  7.8033 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
```

$\hat{\beta}_0$

```
(Intercept)    4.162      3.355    1.24    0.239    
```

$\hat{\beta}_1$

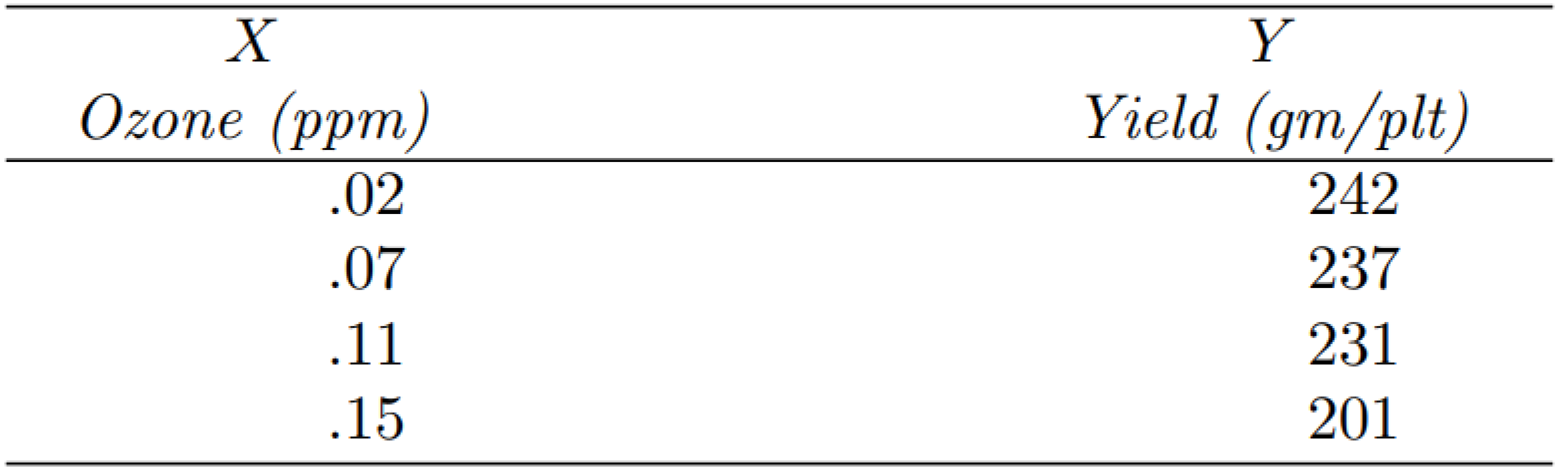
```
Units         15.509      0.505   30.71 8.92e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.392 on 12 degrees of freedom
Multiple R-squared:  0.9874,	Adjusted R-squared:  0.9864 
F-statistic: 943.2 on 1 and 12 DF,  p-value: 8.916e-13
```

Solve same problem using R Code

- x=c(1,2,3,4,5)
- y=c(1,1,2,2,4)
- lr=lm(y~x)
- lr
- lr$fitted.values
- sum((lr$fitted.values-y)^2)
- summary(lr)
- #residual is the observed value minus the predicted value, or the error in our prediction,

Example 2: Mean yields of soybean plants (gms per plant) obtained in response to the indicated levels of ozone exposure over the growing season



X *Ozone (ppm)*	Y *Yield (gm/plt)*
.02	242
.07	237
.11	231
.15	201

Example 2 solved

$$\begin{array}{rlrl}
\sum X_i = & .35 & \sum Y_i = & 911 \\
\overline{X} = & .0875 & \overline{Y} = & 227.75 \\
\sum X_i^2 = & .0399 & \sum Y_i^2 = & 208,495 \\
\end{array}$$

$$\sum X_i Y_i = 76.99$$

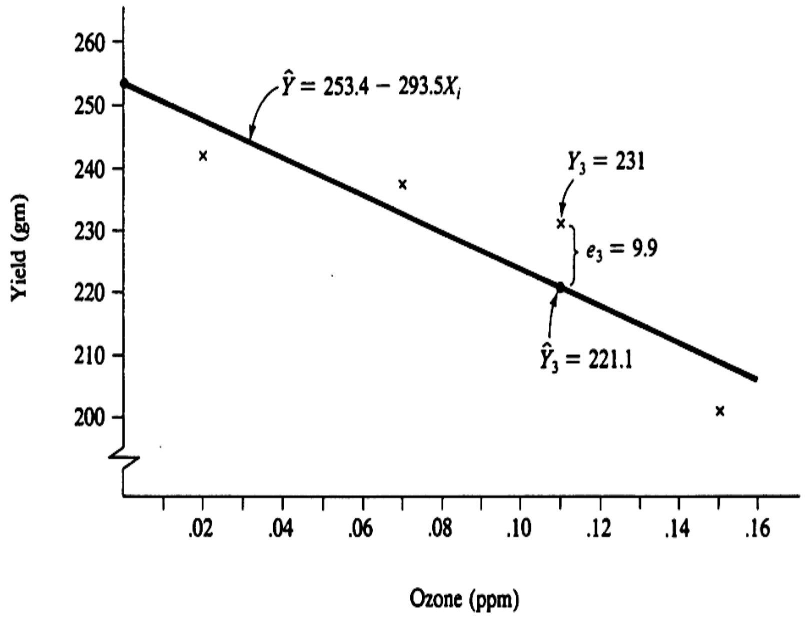
$$\widehat{\beta}_1 = \frac{76.99 - \frac{(.35)(911)}{4}}{.0399 - \frac{(.35)^2}{4}} = -293.531$$

$$\widehat{\beta}_0 = 227.75 - (-293.531)(.0875) = 253.434$$

$$\widehat{Y}_i = 253.434 - 293.531 X_i$$

Observed values, estimated values, and residuals for the linear regression of soybean yield on ozone dosage

Y_i	$\widehat{Y}_i$	e_i	e_i^2
242	247.563	−5.563	30.947
237	232.887	4.113	16.917
231	221.146	9.854	97.101
201	209.404	−8.404	70.627
		$\sum e_i = 0.0$	$\sum e_i^2 = 215.592$



$$e_i = Y_i - \widehat{Y}_i$$

The residuals measure the discrepancy between the data and the fitted model.

The least squares estimation procedure has minimized the sum of squares of the e_i. That is, there is no other choice of values for the two parameters β_0 and β_1 that will provide a smaller $\sum e_i^2$.

Problems with OLS Method

- Least squares (special case of an optimization problem) -> single solution can be found analytically and in single iteration.
- OLS just finds the minima of equation using partial differentiation.
- can give unreliable results if the data is not normally distributed i.e. many outliers exists/mixed distributions present.
- computationally heavy with an order of complexity of $O(n^3)$ for an n*n matrix, where 'n' is the number of features.
- Suitable for small and less sparse data.

Gradient descent

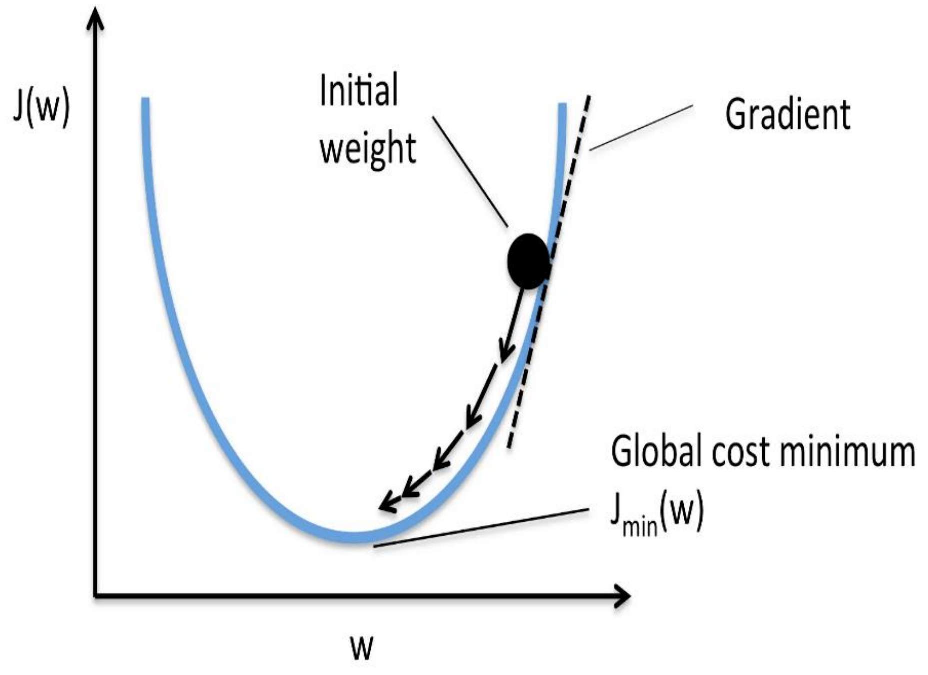
- Alternative: Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient.
- In GD approach, we take the same partial derivatives of ‘**m**’ and ‘**b**’, but instead of equating them to zero, we use a predictor method based on learning rate to come to the best possible value of ‘**m**’ and ‘**b**’.
- Cost function of Gradient Descent

$$J_{m,b} = \frac{1}{N}\sum_{i=1}^{N}(Error_i)^2$$

GD is an algorithm to construct the solution of an optimization problem approximately for multivariate dataset. Less affected by Outliers.
GD can be applied to any objective function/ optimization problem, not just squared distances using an iterative brute force process.

Gradient descent

- The global aim is to minimize objective function J(θ0, θ1)
- use gradient descent to update the parameters of our model.
- Need to select step size and initial estimates
- Good for large dataset/large no. of features



GD offers flexibility and scalability, allowing for optimization of non-linear cost functions and handling large datasets efficiently, it requires careful tuning of hyperparameters such as learning rate.

Initialize your parameters with random values calculate the gradient of Error w.r.t to both '***m***' and '***b***'

Consider objective function J

$$\min_{\theta} \quad J = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

In order to seek optimal parameters θ_0 and θ_1,

$$\frac{\partial J}{\partial \theta_0} = -2 \sum_{i=1}^{n} (y_i - \theta_1 x_i - \theta_0)$$

$$\frac{\partial J}{\partial \theta_1} = -2 \sum_{i=1}^{n} (y_i - \theta_1 x_i - \theta_0) x_i$$

Initialize $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_m]$ vector

Update m weights (or parameters) vector using GD rule

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$$

Repeat the above step until a convergence rule is reached

$$\| \theta_j - \theta_{j+1} \| < \varepsilon$$

'Learning Rate': configurable hyper-parameter often in the range between 0.0 and 1.0

Small value of α leads to slow convergence

Large values of α may fail to converge or might diverge instead

Risk of finding local minima instead of global minima

Summarizing Linear Regression

Supervised learning of linear models can be divided into 2 phases:

- **Training**:

  1. Read training data points with labels $\{\mathbf{x}_{1:n}, y_{1:n}\}$, where $\mathbf{x}_i \in \mathbb{R}^{1\times d}$, $y_i \in \mathbb{R}^{1\times c}$;

  2. Estimate model parameters $\hat{\theta}$ by certain learning Algorithms.

     Note: The parameters are the information the model learned from data.

- **Prediction**:

  1. Read a new data point without label $\mathbf{x}_{n+1}$ (typically has never seen before);

  2. Along with parameter $\hat{\theta}$, estimate unknown label $\hat{y}_{n+1}$.

Matrix form

- In general, each data point xi should have d dimensions, and the corresponding number of parameters should be (d+1).

The mathematical form of linear model is:

$$\hat{y}_i = \sum_{j=0}^{d} \theta_j x_{ij}$$

The matrix form of linear model is:

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1d} \\ 1 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nd} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix}$$

Or in a more compact way:

$$\hat{\mathbf{y}} = \mathbf{X}\theta$$

Matrix form

$$J(\theta) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$
$$= (\mathbf{y} - \mathbf{X}\theta)^{\mathsf{T}} (\mathbf{y} - \mathbf{X}\theta)$$

$$\begin{aligned}
\frac{\partial J(\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta} (\mathbf{y} - \mathbf{X}\theta)^{\mathsf{T}} (\mathbf{y} - \mathbf{X}\theta) \\
&= \frac{\partial}{\partial \theta} \left(\mathbf{y}^{\mathsf{T}}\mathbf{y} + \theta^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{X}\theta - 2\mathbf{y}^{\mathsf{T}}\mathbf{X}\theta\right) \\
&= \mathbf{0} + 2\left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{\mathsf{T}}\theta - 2\left(\mathbf{y}^{\mathsf{T}}\mathbf{X}\right)^{\mathsf{T}} \\
&= 2\left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)\theta - 2\left(\mathbf{X}^{\mathsf{T}}\mathbf{y}\right) \\
&\triangleq \mathbf{0}
\end{aligned}$$

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

prediction function

$$\begin{aligned}
\hat{\mathbf{y}} &= \mathbf{X}\hat{\theta} \\
&= \mathbf{X}\left(\mathbf{X}^{\mathsf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y} \triangleq \mathbf{H}\mathbf{y}
\end{aligned}$$

H refers to hat matrix where X+ is pseudo inverse of X

$$X^{\dagger} = (X^{\top}X)^{-1}X^{\top}$$

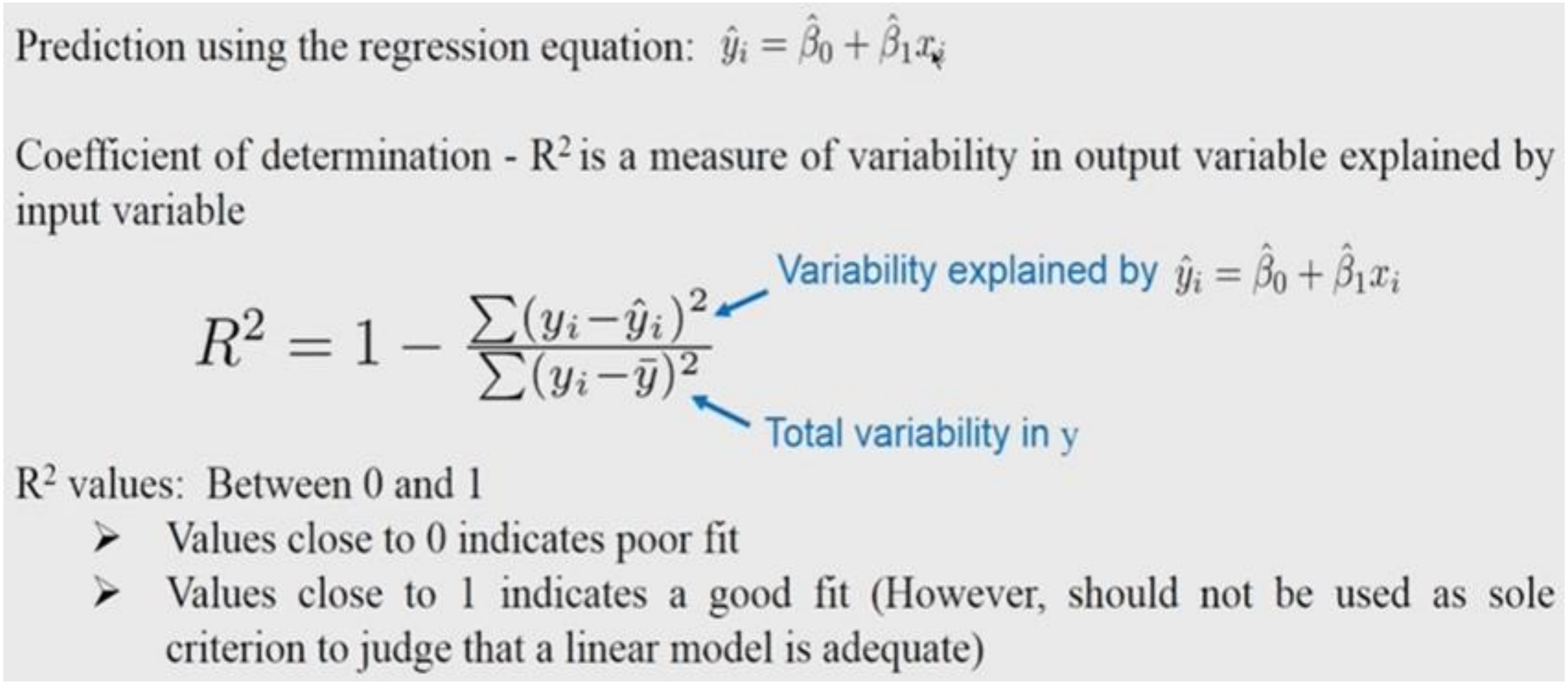
Assessing the Model

- The least squares method will produce a regression line whether or not there is a linear relationship between x and y.
- Consequently, it is important to assess how well the linear model fits the data.
- Sum of squares for errors
  - This is the sum of differences between the points and the regression line.
  - It can serve as a measure of how well the line fits the data.

$$SSE = SS(\mathrm{Re}\,s)$$

$$= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Assessing the Model – testing goodness of fit



Prediction using the regression equation: $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$

Coefficient of determination - R^2 is a measure of variability in output variable explained by input variable

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$$

Variability explained by $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$

Total variability in y

R^2 values: Between 0 and 1

- Values close to 0 indicates poor fit
- Values close to 1 indicates a good fit (However, should not be used as sole criterion to judge that a linear model is adequate)

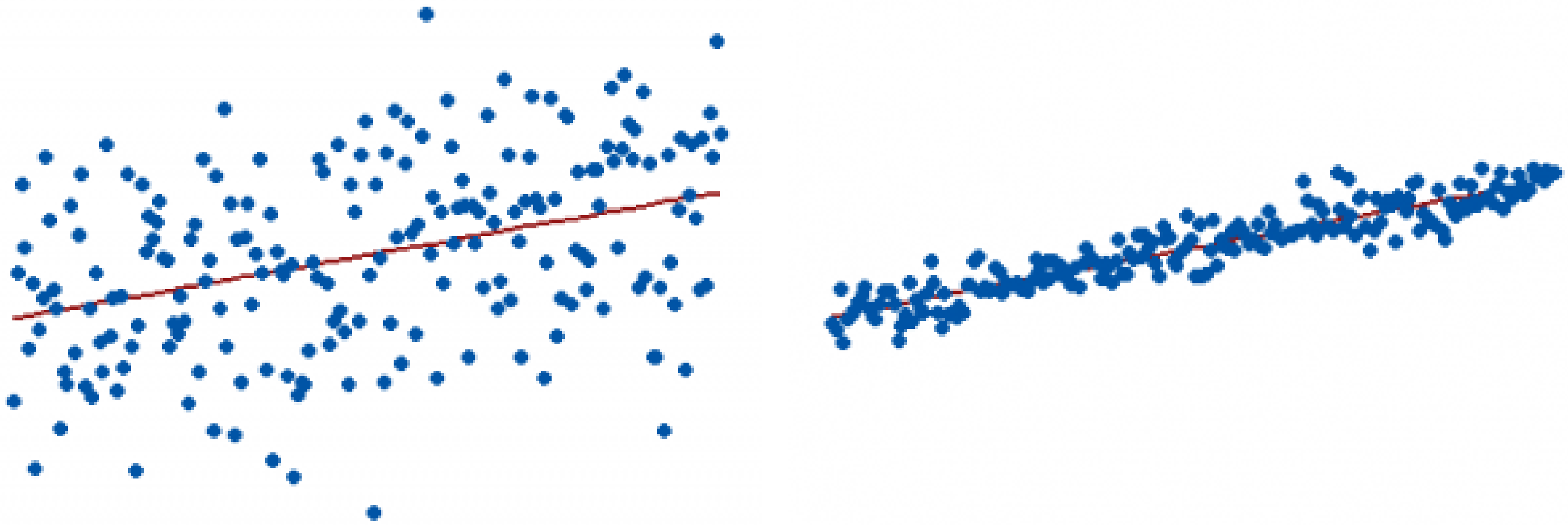
$$R^2 = \frac{\text{Variance explained by the model}}{\text{Total variance}}$$

R-squared is percentage of dependent variable variation that a linear model explains. 100% represents a model that explains all the variation in the response variable around its mean.

R-Squared is also called **coefficient of determination**.

Assessing the Model – testing goodness of fit -> R-squared values

R-squared values represent the scatter around the regression line. goodness of your fit i.e. how well the model fits the data -> R² values

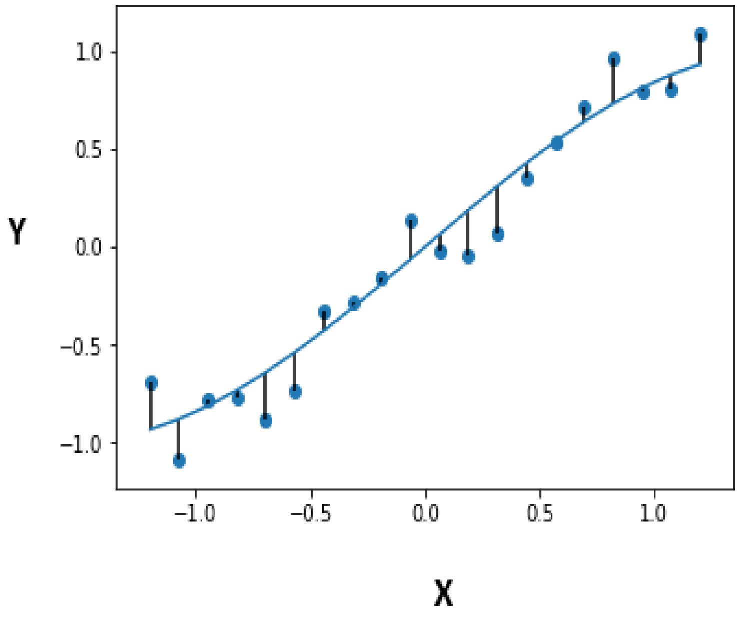


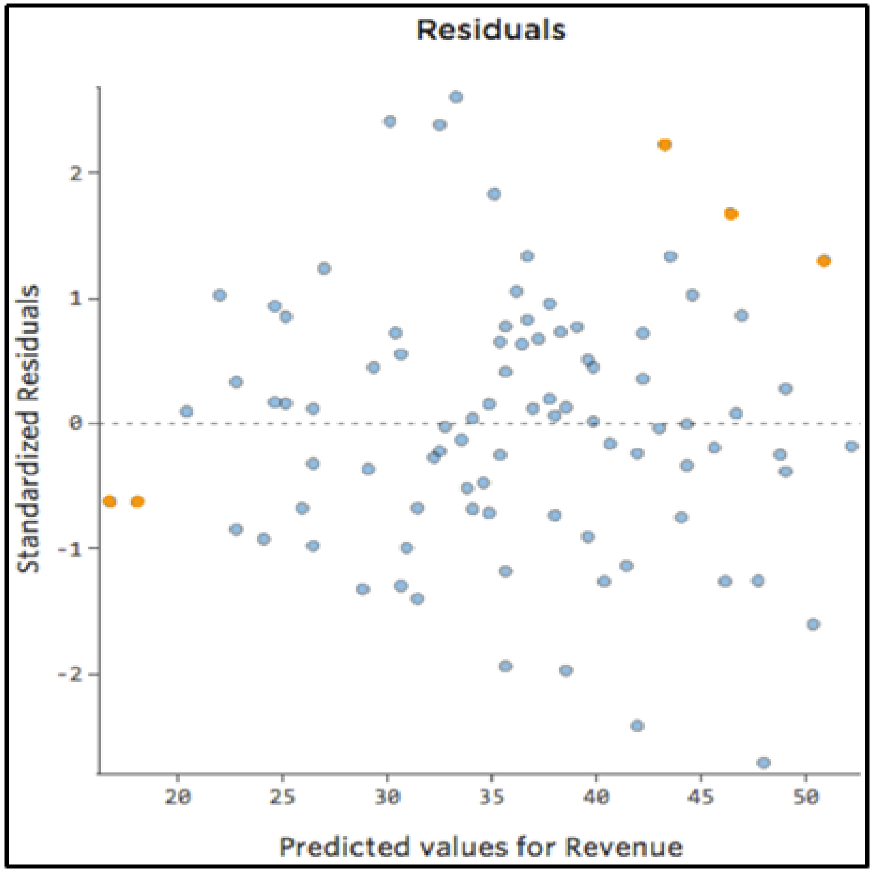
- The R-squared for the regression model on the left is 15%, and for the model on the right it is 85%.
- Linear Regression is very sensitive to Outliers.
- It can terribly affect the regression line and eventually the forecasted values.
- residuals should be randomly scattered around zero
- Non-random residual patterns indicate a bad fit despite a high R2.
- Always check your residual plots!

Regression Process – Residual plots

$$Residual\ (\epsilon) = y - \hat{y}$$

vertical lines are residuals





Residual Plots - To validate your regression models, plot residual values on the Y-axis and the independent variable on the x-axis

Regression Process – Residual plots

- **Residual Plot Analysis -** assumption of a linear regression model is that the ***errors are independent and normally distributed***
- Every regression model inherently has some degree of error since you can never predict something 100% accurately
- randomness and unpredictability are always a part of the regression model.
- if we capture all of the predictive/deterministic information, all that is left behind (residuals) should be completely random & unpredictable i.e. stochastic.
- Hence, we want our residuals to follow a normal distribution.

$$\text{Response} = \text{Deterministic} + \text{Stochastic}$$

Other validation statistics -> adjusted R^2, MAPE (Mean Absolute Percentage Error) scores.

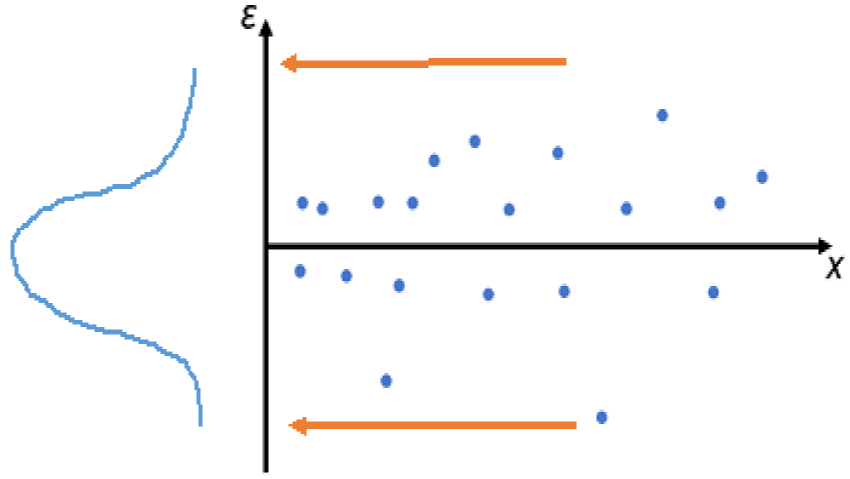
Regression Process - Good Residual Plots

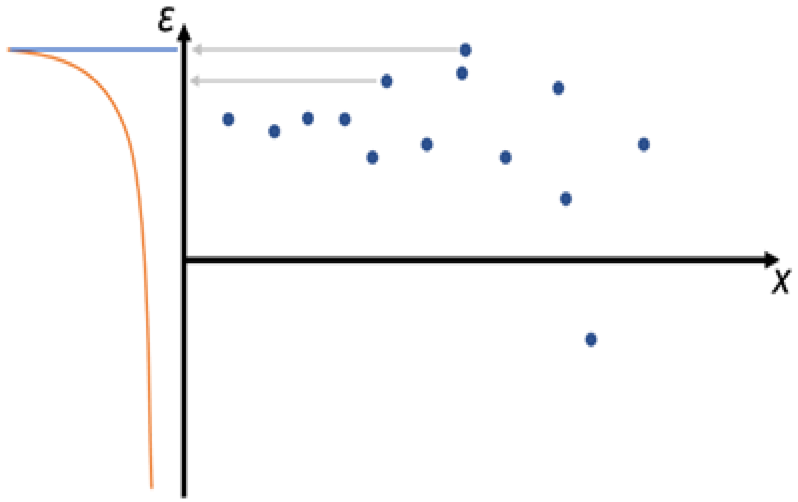
characteristics of a good residual plot are as follows:

1.It has a high density of points close to the origin and a low density of points away from the origin

2.It is symmetric about the origin

Project errors onto the y-axis





- good residual plot -> **residuals are independent and normally distributed**
- Bad residual plots -> signify that we have not completely captured the predictive information of the data in our model

Adjusted R-Squared

- It measures the proportion of variation explained by only those independent variables that really help in explaining the dependent variable.
- It penalizes you for adding independent variables that do not help in predicting the dependent variable in regression analysis.

$$R^2_{\text{adjusted}} = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

where

R^2 = sample R-square

p = Number of predictors

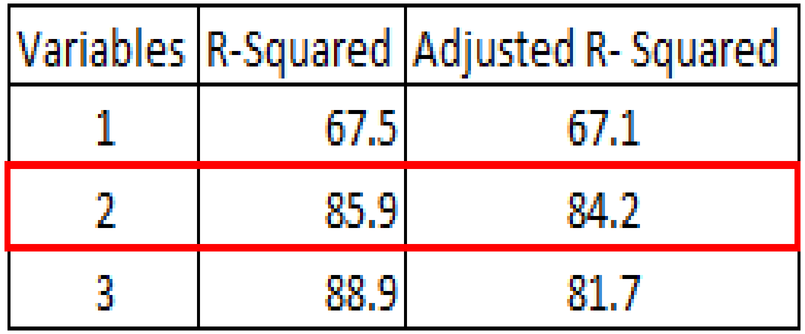
N = Total sample size.

Every time we add independent variable to a model, the **R-squared increases**, even if it is insignificant. Whereas **Adjusted R-squared** increases only when independent variable is significant and affects dependent variable.

Adjusted r-squared can be negative when r-squared is close to zero.

Adjusted r-squared value always be less than or equal to r-squared value.

Adjusted R-square should be used to compare models with different numbers of independent variables. Adjusted R-square should be used while selecting important predictors (independent variables) for the regression model.

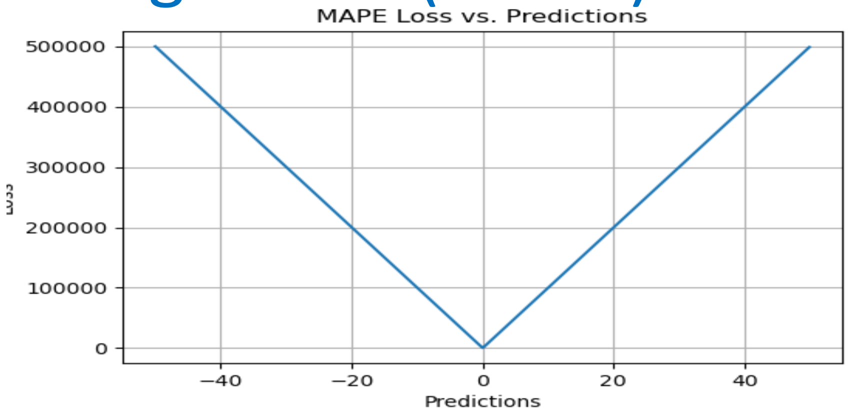


Variables	R-Squared	Adjusted R- Squared
1	67.5	67.1
2	85.9	84.2
3	88.9	81.7

Example: adjusted r-squared is maximum when we included two variables. It declines when third variable is added. Whereas r-squared increases when we included third variable. It means third variable is insignificant to the model.

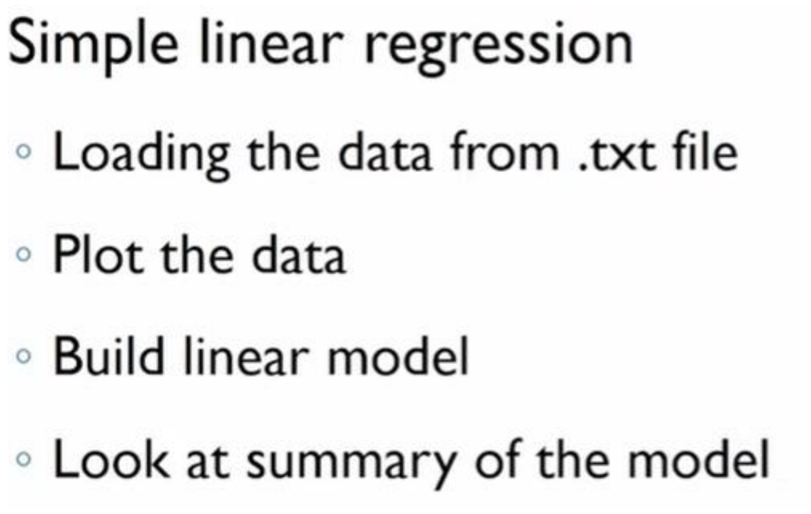
Mean absolute percentage error (MAPE)

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \cdot 100\%$$



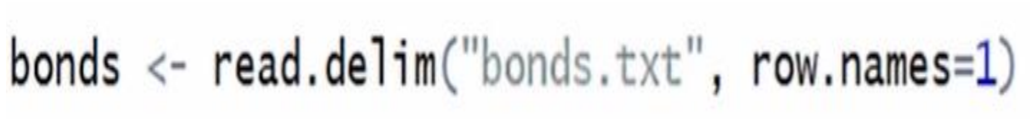
- divide the absolute difference between the actual and predicted values by the actual value. also known as Mean Absolute Percentage Deviation (MAPD), increases linearly with error. Lower MAPE values indicate better model performance.
- MAPE is independent of the scale of the variables since its error estimates are in terms of percentage. All errors are normalized on a common scale and it is easy to understand.
- when the denominator becomes zero, resulting in a "division by zero" challenge.
- MAPE exhibits bias by penalizing negative errors more than positive errors, potentially favoring methods with lower values.
- Due to the division operation, MAPE's sensitivity to alterations in actual values leads to varying losses for the same error. For example, an actual value of 100 and a predicted value of 75 results in a 25% loss, while an actual value of 50 and a predicted value of 75 yields a higher 50% loss, despite the identical error of 25.

Steps

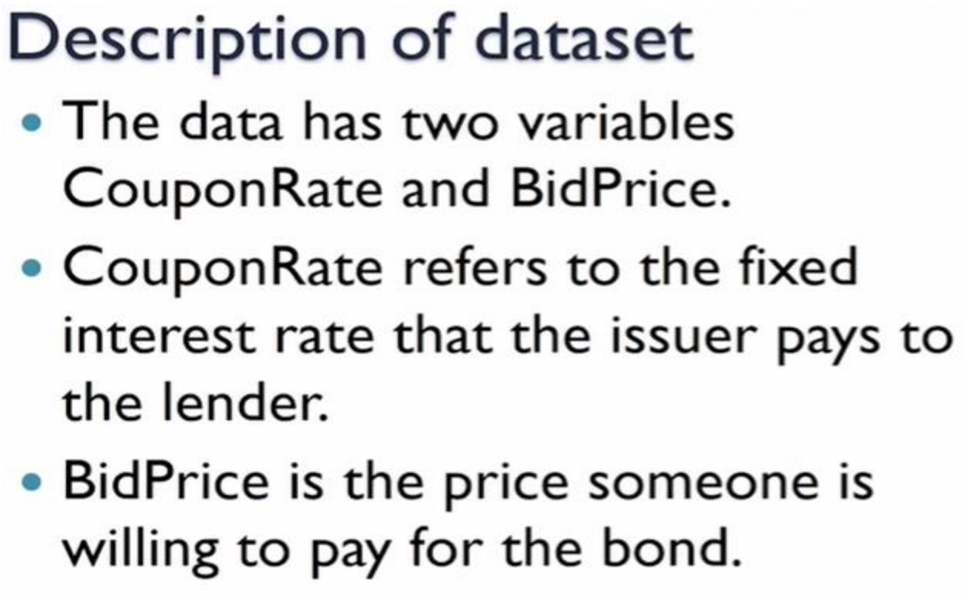


Simple linear regression

- Loading the data from .txt file
- Plot the data
- Build linear model
- Look at summary of the model



```
bonds <- read.delim("bonds.txt", row.names=1)
```

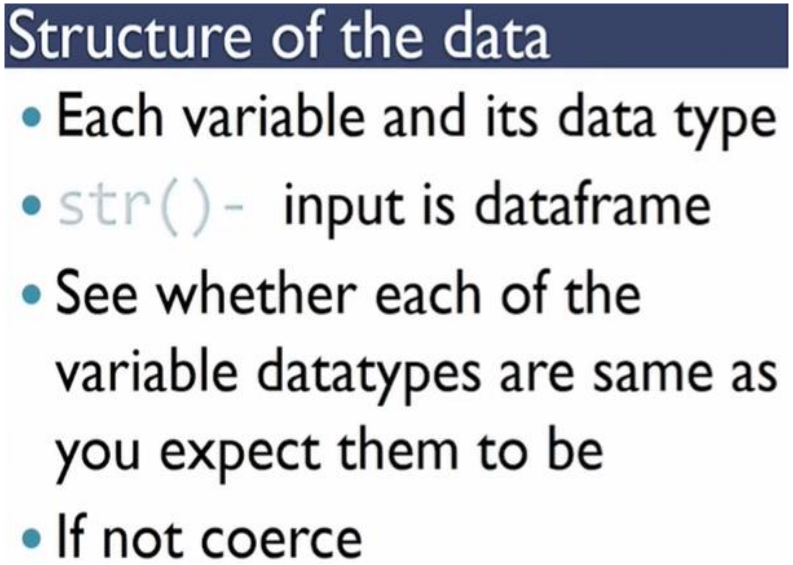


Description of dataset

- The data has two variables CouponRate and BidPrice.
- CouponRate refers to the fixed interest rate that the issuer pays to the lender.
- BidPrice is the price someone is willing to pay for the bond.

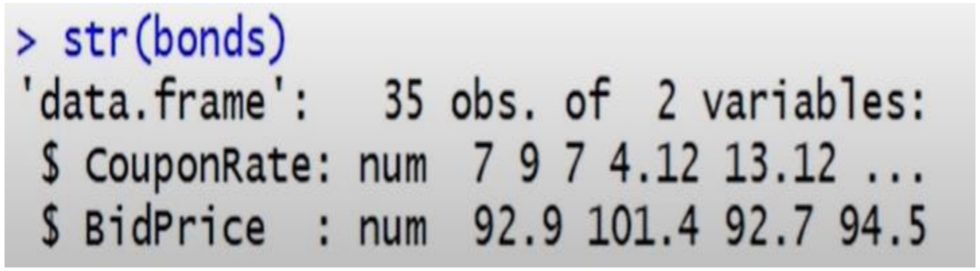
https://gattonweb.uky.edu/sheather/book/data_sets.php

Check data

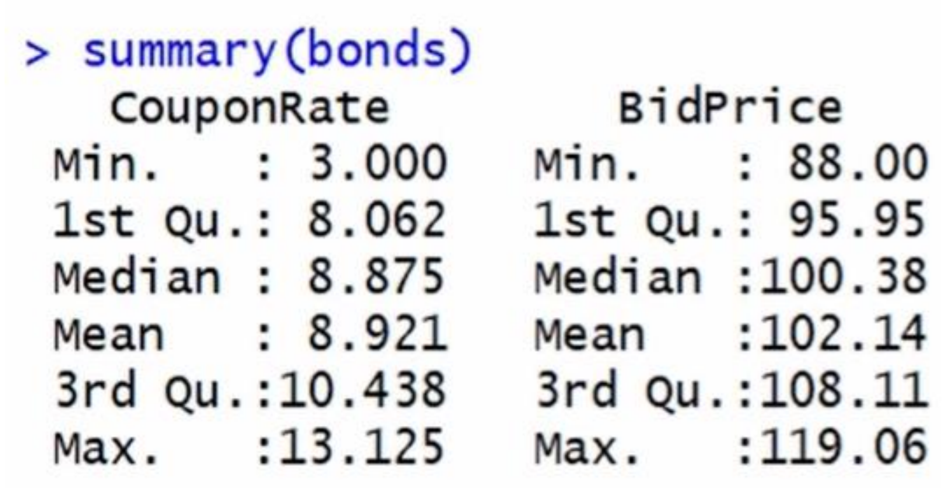


Structure of the data

- Each variable and its data type
- str()- input is dataframe
- See whether each of the variable datatypes are same as you expect them to be
- If not coerce



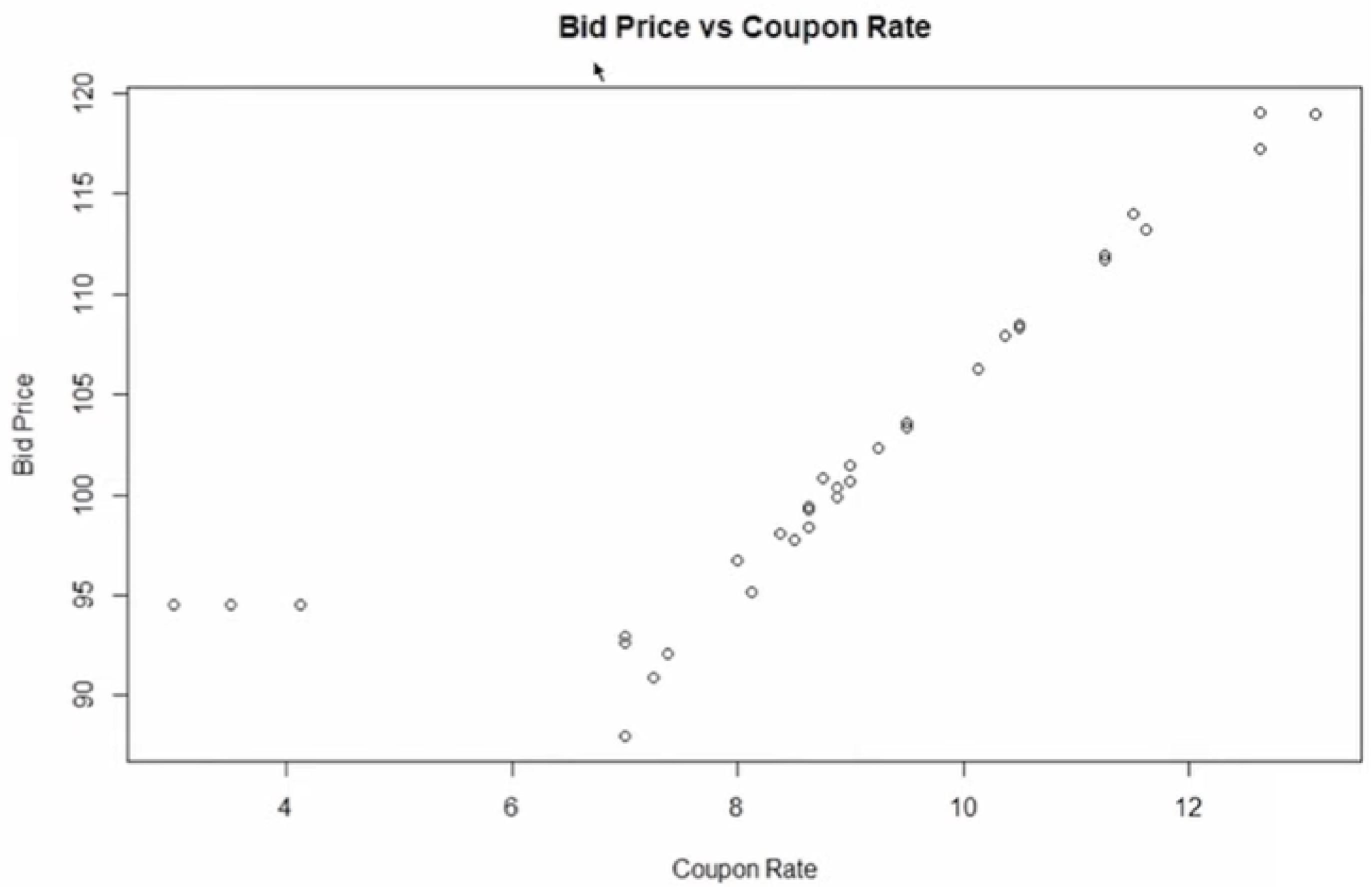
```
> str(bonds)
'data.frame':   35 obs. of  2 variables:
 $ CouponRate: num  7 9 7 4.12 13.12 ...
 $ BidPrice  : num  92.9 101.4 92.7 94.5
```



```
> summary(bonds)
   CouponRate         BidPrice     
 Min.   : 3.000   Min.   : 88.00  
 1st Qu.: 8.062   1st Qu.: 95.95  
 Median : 8.875   Median :100.38  
 Mean   : 8.921   Mean   :102.14  
 3rd Qu.:10.438   3rd Qu.:108.11  
 Max.   :13.125   Max.   :119.06  
```

Plot data

```
plot(bonds$CouponRate,bonds$BidPrice,
     main = "Bid Price vs Coupon Rate",
     xlab = "Coupon Rate",
     ylab = "Bid Price")
```



Build linear model

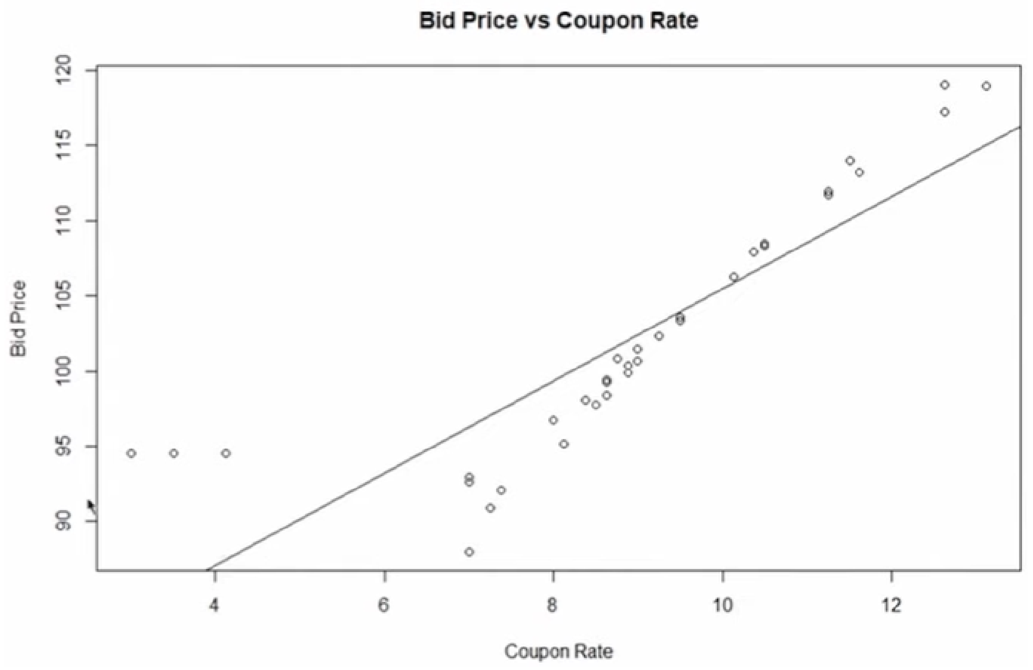
```
bondsmod <- lm(bonds$BidPrice~bonds$CouponRate)
```

or

```
bondsmod <- lm(BidPrice~CouponRate,data = bonds)
```

$$\hat{y}_i = \underbrace{\hat{\beta}_0}_{\text{Intercept}} + \underbrace{\hat{\beta}_1}_{\text{Slope}} x_i + \epsilon_i$$

```
plot(bonds$CouponRate,bonds$BidPrice,
     main = "Bid Price vs Coupon Rate",
     xlab = "Coupon Rate",
     ylab = "Bid Price")
abline(bondsmod)
```



Case Study: Polynomial Linear Regression

Mathematically the model: $\hat{y}_i = Wx_i + b$

Model parameter: $(W, b) \equiv (w_1, w_2 \cdots w_N, b)$

Where $x_i = \left\{ \left(x_i^j\right)_{j=1\cdots N} \right\}$

***N*: No. of Features.** i: i^{th} **data**

The cost function is simply the averaged $Loss(\hat{y}, y)$, **over all the training examples**

$$\left\{ \left\{ \left(x_i^j\right)_{j=1\cdots N} \right\}, y_i \right\}_{i=1\cdots n}$$

$$Cost = \frac{1}{2n} \sum_{i=1}^{n} Loss(\hat{y}_i, y_i)$$

$$Cost(w_1, w_2 \cdots w_N, b) = \frac{1}{2n} \sum_{j=1}^{n} Loss(\hat{y}_i, y_i)$$

$$= \frac{1}{2n} \sum_{i=1}^{n} \left(\sum_{j=1}^{N} w_i x_i^j + b - y_i \right)^2$$

Considering $(\hat{y}_i - y_i)$ positive, means $\frac{\delta cost}{\delta b}$ is positive. So, we can say if b is increasing $cost$ is increasing and if b is decreasing $cost$ is decreasing . Apparent!!!

Goal: To choose $w_1, w_2 \cdots w_N, b$ **so that** $Cost$ **is minimum**

$$\frac{\delta cost}{\delta w_j} = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{N} w_i x_i^j + b - y_i \right) \left(x_i^j\right) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)\left(x_i^j\right)$$

$$\frac{\delta cost}{\delta b} = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{N} w_i x_i^j + b - y_i \right) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

Linear Regression – if many data points, more parameters, solving set of linear equations is difficult

To achieve *Cost* minimum we may equate $\frac{\delta cost}{\delta w_j}$ to zero and we can solve a system of simultaneous equations. Getting closed form of system of simultaneous equations for higher order polynomial regression is difficult. Let us try with **Gradient Descent** iterative algorithm.

$$w_{j+1} = w_j - \alpha \frac{\delta cost}{\delta w_j}$$

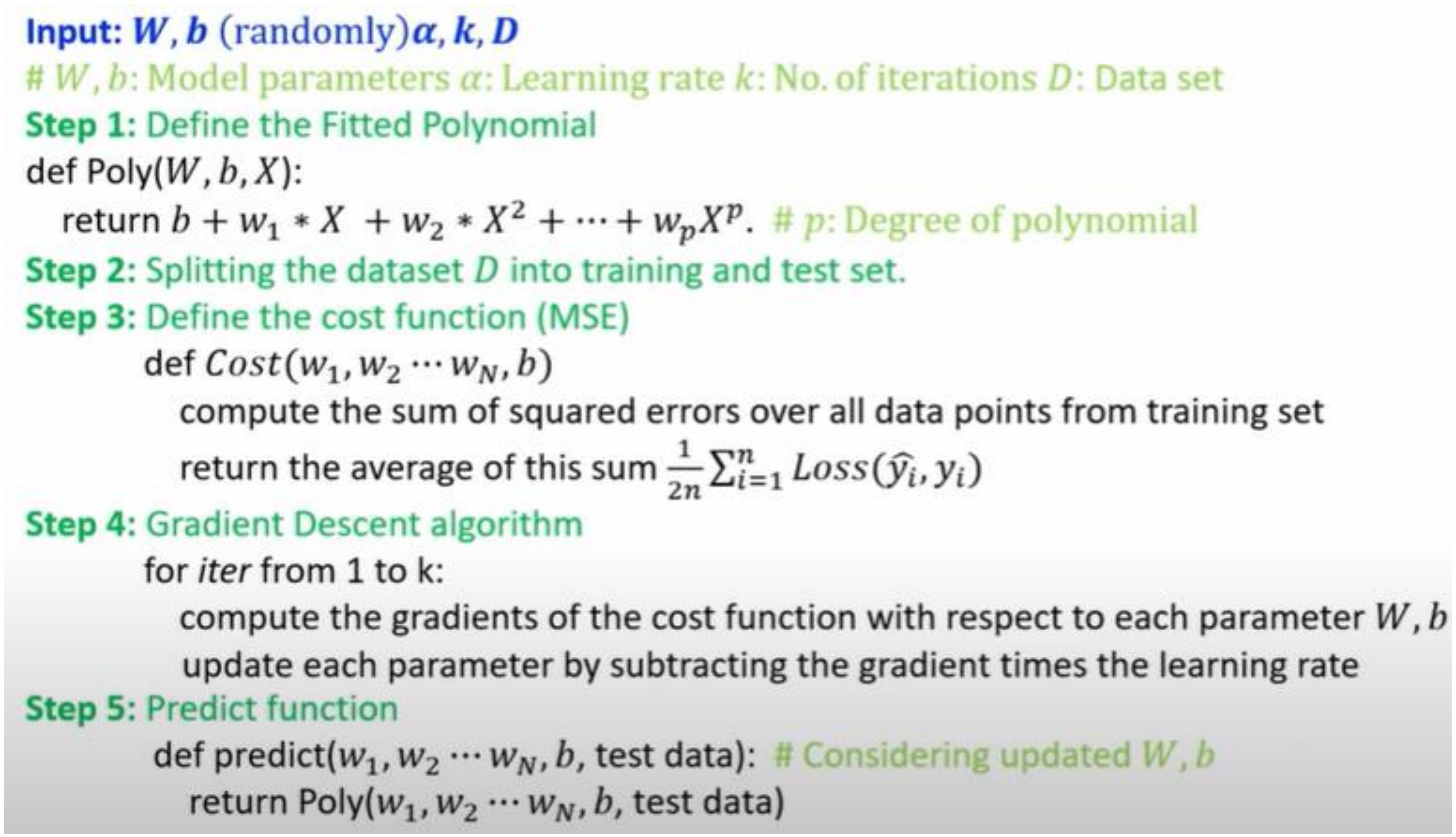
$$w_{j+1} = w_j - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)\left(x_i^j\right)$$

Polynomial Regression: Mathematically the model: $\widehat{y_i} = W_i \phi(x_i) + b$ Where $x_i = \left\{\left(x_i^j\right)_{j=1\cdots N}\right\}$

$\phi(x) = (1, x, x^2 \cdots x^p)^T$ p: degree of polynomial

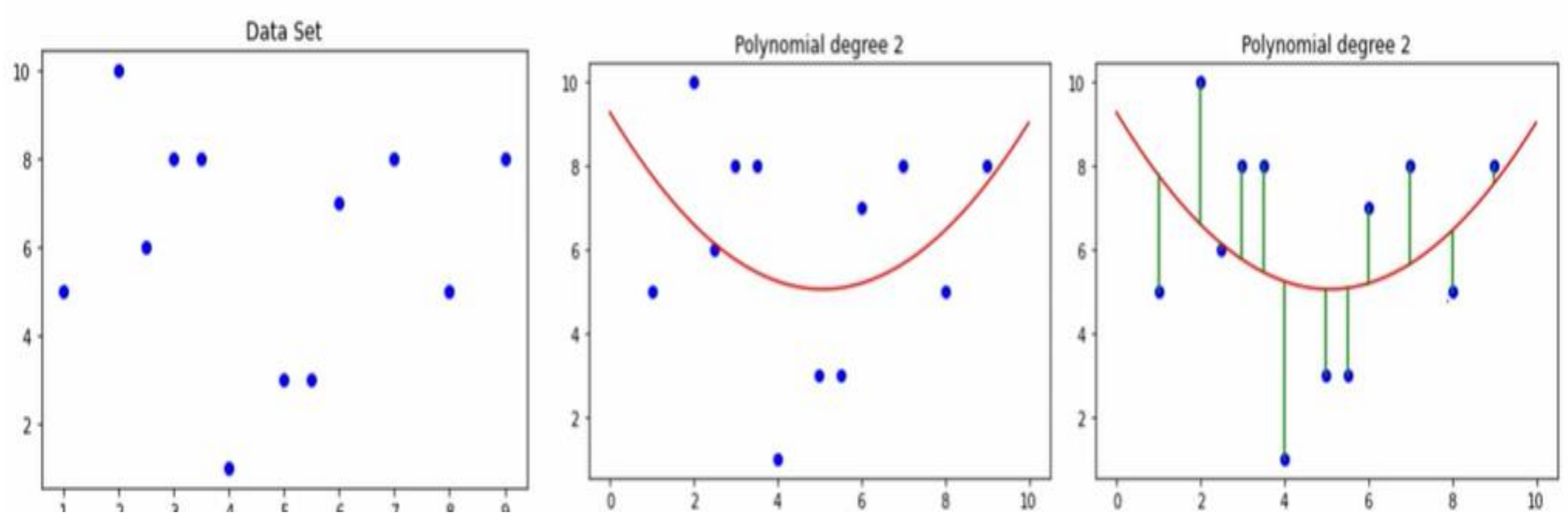
$$\hat{y_i} = w_1 * x_i + w_2 * x_i^2 + \cdots + w_p x_i^p + b$$

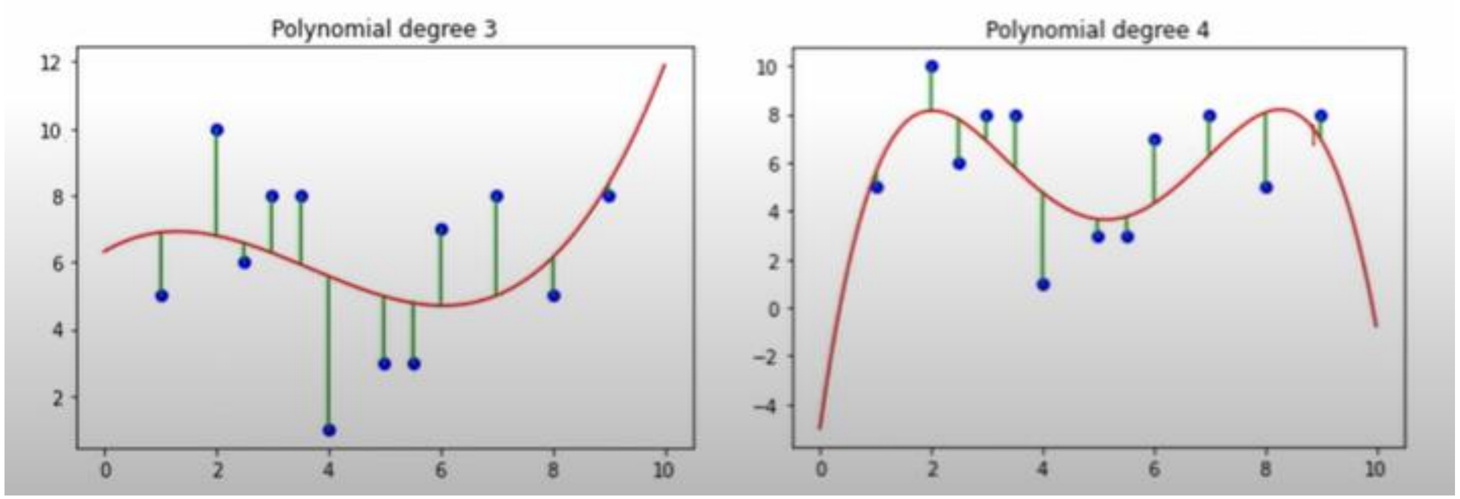
Polynomial Regression



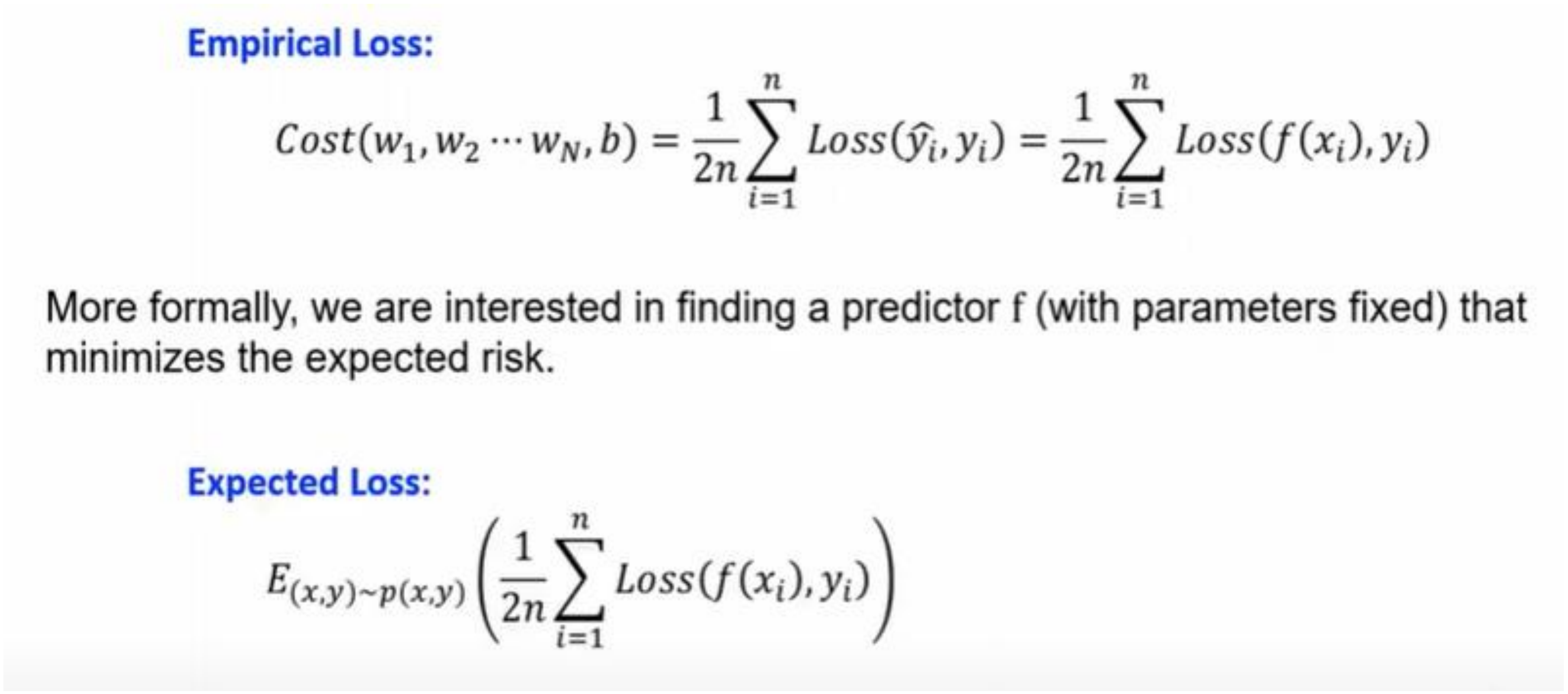
```
Input: $\boldsymbol{W}, \boldsymbol{b}$ (randomly) $\boldsymbol{\alpha}, \boldsymbol{k}, \boldsymbol{D}$
# $W, b$: Model parameters $\alpha$: Learning rate $k$: No. of iterations $D$: Data set
Step 1: Define the Fitted Polynomial
def Poly($W, b, X$):
  return $b + w_1 * X + w_2 * X^2 + \cdots + w_p X^p$.  # $p$: Degree of polynomial
Step 2: Splitting the dataset $D$ into training and test set.
Step 3: Define the cost function (MSE)
        def $Cost(w_1, w_2 \cdots w_N, b)$
           compute the sum of squared errors over all data points from training set
           return the average of this sum $\frac{1}{2n} \sum_{i=1}^{n} Loss(\hat{y}_i, y_i)$
Step 4: Gradient Descent algorithm
        for *iter* from 1 to k:
           compute the gradients of the cost function with respect to each parameter $W, b$
           update each parameter by subtracting the gradient times the learning rate
Step 5: Predict function
         def predict($w_1, w_2 \cdots w_N, b$, test data):  # Considering updated $W, b$
            return Poly($w_1, w_2 \cdots w_N, b$, test data)
```

Polynomial Regression





Polynomial Regression



Empirical Loss:

$$Cost(w_1, w_2 \cdots w_N, b) = \frac{1}{2n}\sum_{i=1}^{n} Loss(\hat{y}_i, y_i) = \frac{1}{2n}\sum_{i=1}^{n} Loss(f(x_i), y_i)$$

More formally, we are interested in finding a predictor f (with parameters fixed) that minimizes the expected risk.

Expected Loss:

$$E_{(x,y)\sim p(x,y)}\left(\frac{1}{2n}\sum_{i=1}^{n} Loss(f(x_i), y_i)\right)$$

Specifically, we need to minimize the expected loss on test data set.
Empirical loss looks at a training data samples.
The expected loss looks at all theoretically possible data (with weighted according to probability of occurrence).
The MSE, in the case of linear regression, is the empirical loss. It is computed on the data.

Thank you

- ???