

IS4303 Week 4

Data Preprocessing & Regression

Agenda

- Data Preprocessing
- Understand Linear Regression
- Understand Logistic Regression
 - Sigmoid Function
 - Maximum Likelihood
 - Gradient Ascent/Descent
- Python Jupyter Notebook

Data Preprocessing

A General Procedure:

- Read Dataset
 - The format of data file (e.g., type, separator, header, etc.);
 - Merge/Append/Concatenate
- Invalid values & Missing Values
- Categorical Variables
- Variables of Different Scales
- Outlier Detection
- Data Reduction
- Summary Statistics and Exploratory Data Analysis
- Train-Test Split (Question 11 in Homework 1)

Invalid & Missing Values

- Invalid Values:
 - Examples:
 - (1) Feature “**Interest Rate**” should be numerical (e.g., float), but becomes string value;
 - (2) Feature “**Age**” has some values: 999 or -1, which are unreasonable.
 - Data transformation; Remove values; Replace with “nearest” values
- Missing Values:
 - Delete rows/columns;
 - Data imputation;
 - Create dummy variables to indicate missing or not;
 - Change dataset: “**Garbage-In Garbage-Out**”

Categorical Variable

- Categorical Variable:
 - Categorical variable (a.k.a., nominal variable) is one that has two or more limited/fixed categories, but there is **no intrinsic ordering** to these categories.
 - Example:
(1) Feature “**Marital Status**” has 4 unique values: “Single”, “Married”, “Divorced” and “Widowed”

- One-Hot Encoding:

Name	Marital Status	Single	Married	Divorced	Widowed
AA	Single	1	0	0	0
BB	Divorced	0	0	1	0
CC	Widowed	0	0	0	1
DD	Single	1	0	0	0
EE	Married	0	1	0	0

Suppose a categorical feature has N unique values:

- (1) Create (N) dummy variables;
- (2) Include only ($N-1$) of them;
- (3) Guess why ?

Feature Rescaling

- Feature Rescaling/Normalization/Standardization:

- A method used to standardize the range of independent variables or features of data; Make features with different scales more balanced in contributing to the output.
- Methods:

(1) Z-score: $x' = \frac{x - \bar{x}}{\sigma}$

(2) Min-Max (e.g., to $[0, 1]$): $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$

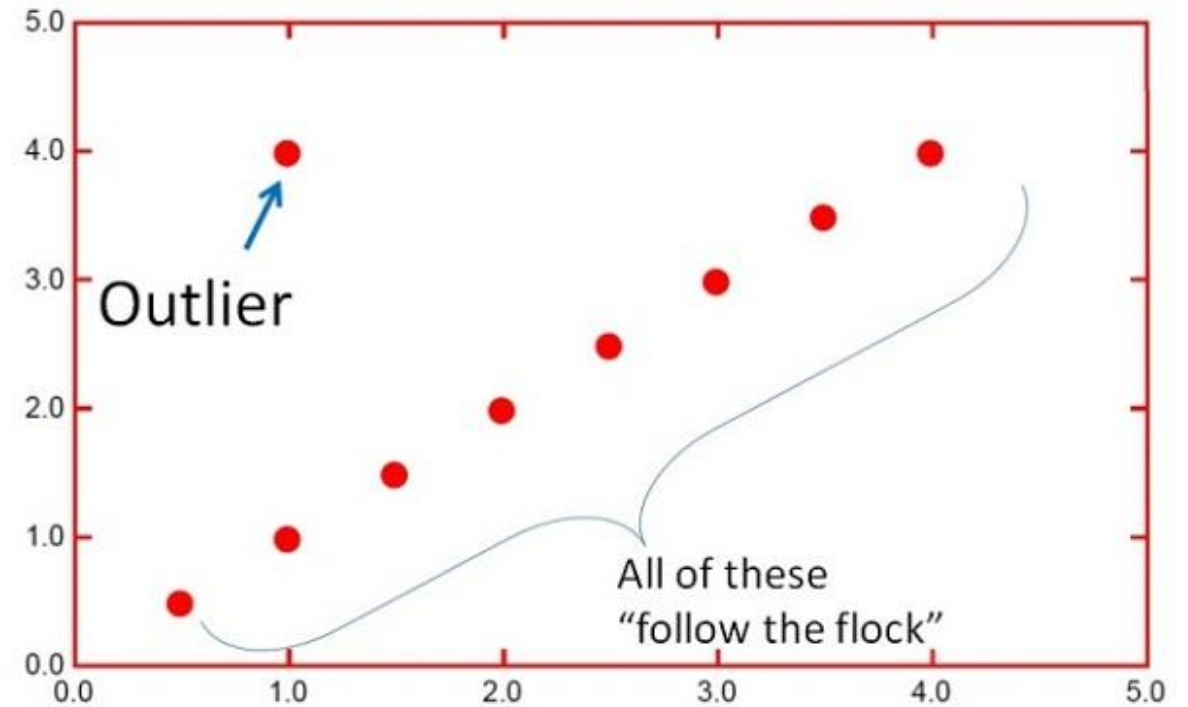
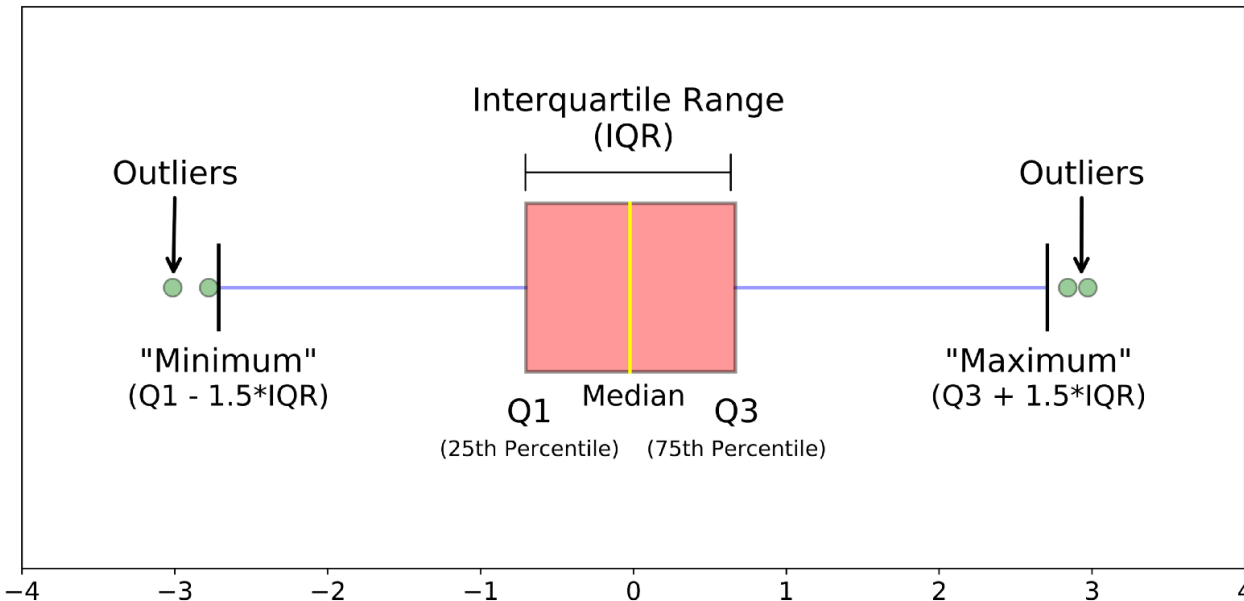
(3) Decimal Scaling: $v' = \frac{v}{10^j}$

- Advantages:

- (1) Improve convergence speed in optimization;
- (2) Some features are less important but have larger scale, so slight changes of them will still have greater impacts on the objective function for optimization than other more important features. It is problematic because it seems these less important features are dominant.

Outlier Detection

- Outliers:
 - Observation points that are distant from other observations
 - Example:



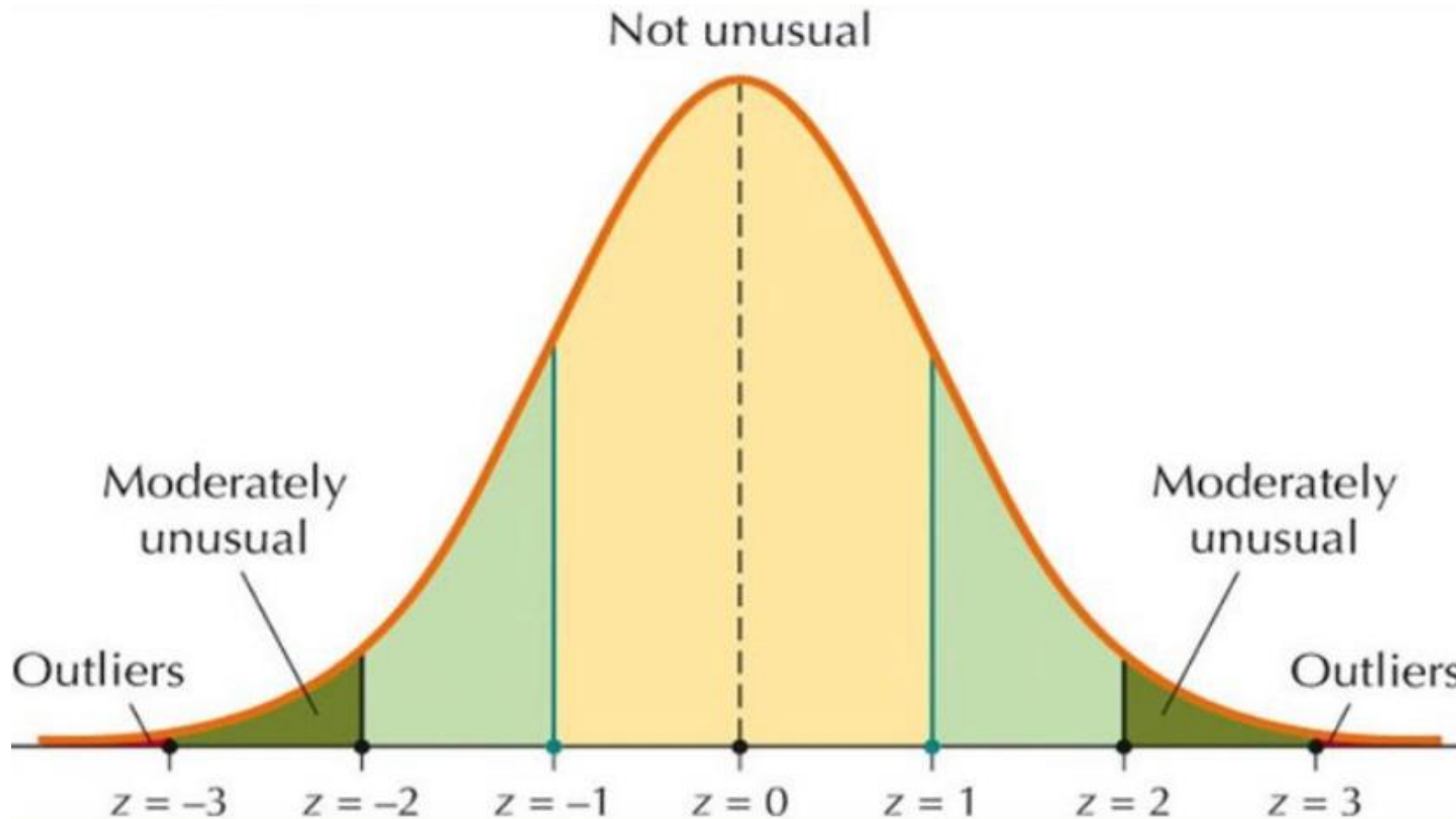
Never mind what the axes mean...

- Detection:
 - Boxplot
 - Univariate distribution
(e.g., $|z\text{-score of feature } X| > 3$)

Outlier Detection

- Outliers:
 - Observation points that are distant from other observations

- Detection:
 - Boxplot
 - Univariate distribution
(e.g., $|z\text{-score of feature } X| > 3$)



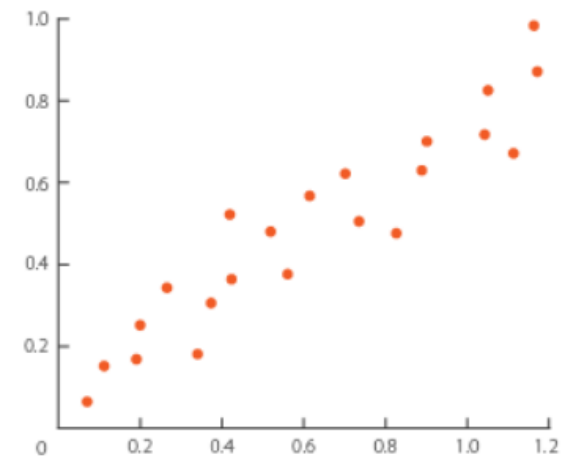
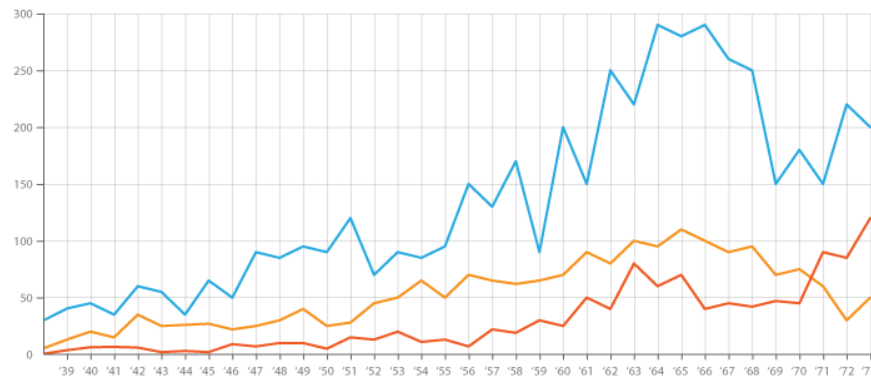
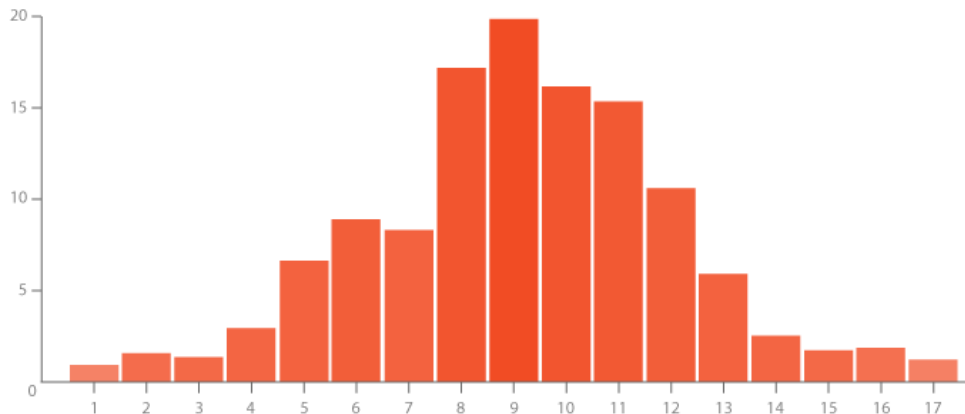
Feature Selection

Feature Selection:

- Brute-Force
 - Use simple logic or domain knowledge
- Filter method
 - correlations between feature X and output y
 - chi-square test, etc.
- Wrapper method
 - Sequential Feature Selection (e.g., stepwise regression)
- Embedded method
 - decision tree
 - LASSO, etc.

Summary Statistics and Exploratory Data Analysis

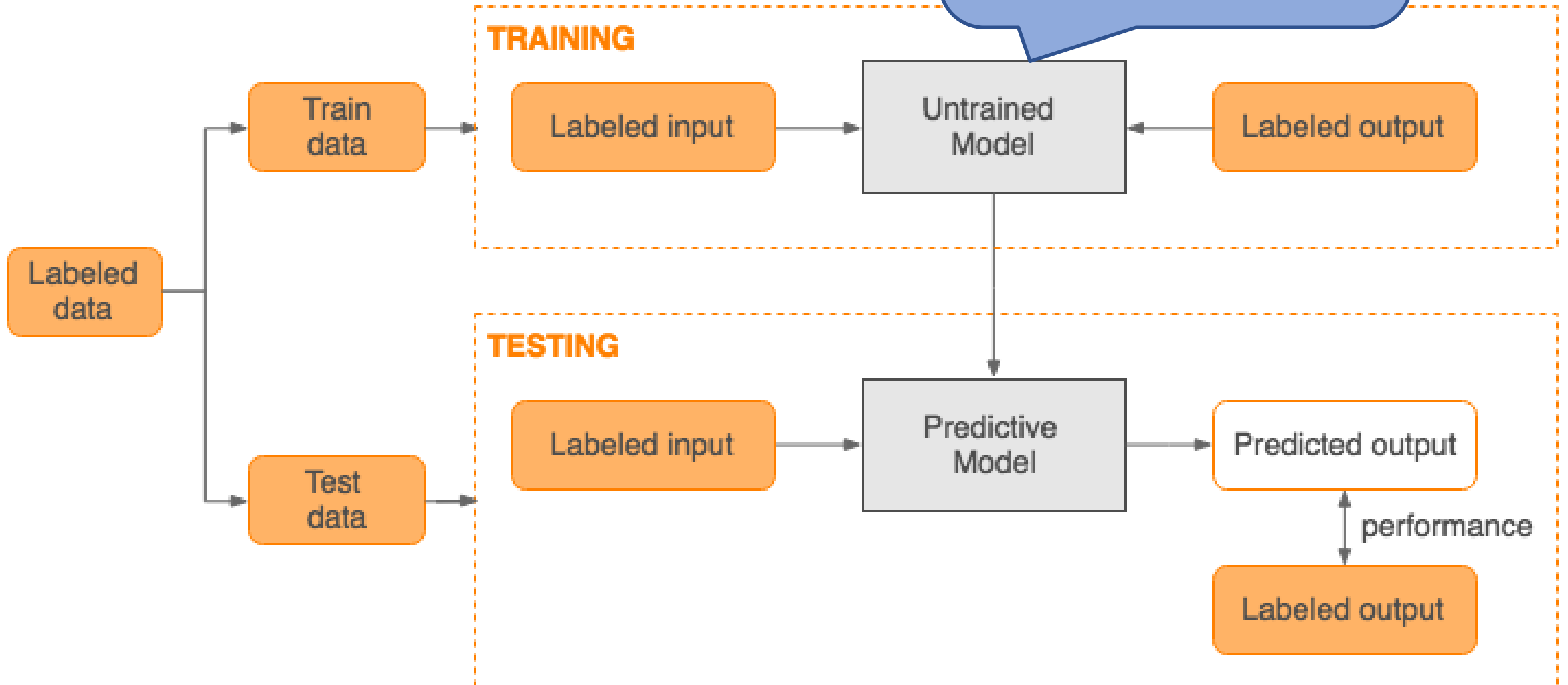
- Summary Statistics:
 - Help deep understanding of your data
 - Examples: mean, median, standard deviation, min/max value, etc.
- Exploratory Data Analysis:
 - Help deep understanding of your data
 - Visualization: <https://datavizcatalogue.com/>




Train-Test Split

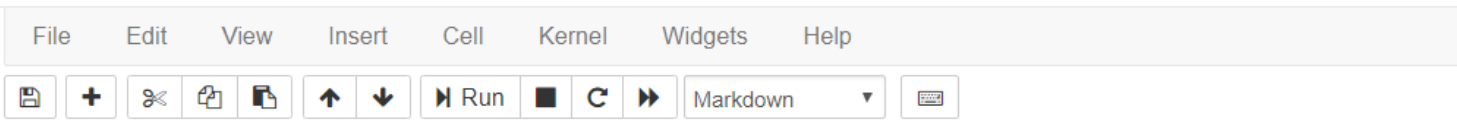
(Question 11 in Homework 1)

1. Decision Tree
2. Logistic Regression
3. Nearest Neighbor
4. Naïve Bayes Classifier
5. Support Vector Machine



Data Preprocessing in Python

 jupyter IS4303 Tutorial Week4 Last Checkpoint: 14 minutes ago (autosaved)



2 Data Preprocessing

```
In [2]: #!/usr/bin/env python
#-*- coding:utf-8 -*-
from __future__ import division, print_function
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from math import sqrt, log
from functools import reduce
from collections import defaultdict
from IPython.display import HTML
%matplotlib inline
```

Step 2.1: Read data into python pandas and named as "default".

Read Excel

Read CSV

Read Others

Note:

1. The type of your file?
2. Why do we set `header=1` ?
3. If you encounter error message, you may need to install or upgrade `xlrd` package in advance.

Linear Regression

Example: Predict price of a condo **Price**

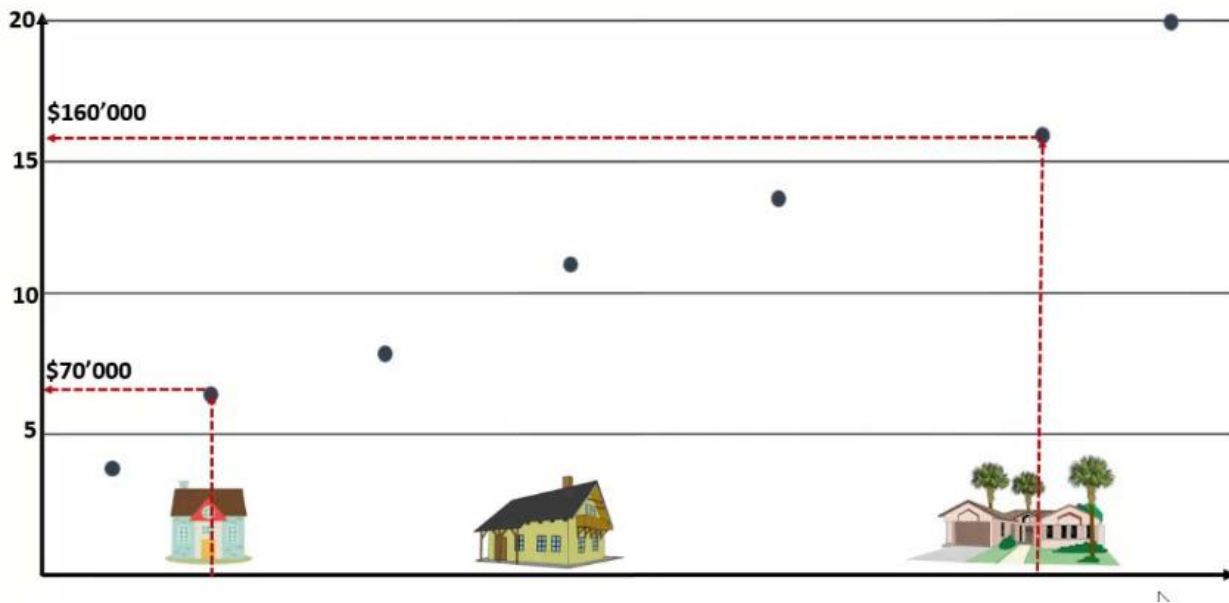
3 features:

$x = \{x_1: \text{size}, x_2: \text{\#bedroom}, x_3: \text{\#bathroom}\}$

Collect data from $N=1000$ condos ($i=1, \dots, 1000$):

Row	Size (m ²)	#Bedroom	#Bathroom	Price (k)
1	65	2	2	900
2	72	3	2	1200
.....
1000	50	1	1	500

$$\text{Price}_i = \beta_0 + \beta_1 \times \text{Size}_i + \beta_2 \times \text{Bedroom}_i + \beta_3 \times \text{Bathroom}_i + \varepsilon_i$$



Linear Regression

- Linear regression model of \mathbf{N} ($i=1,\dots,\mathbf{N}$) observations and \mathbf{p} ($j=1,\dots,\mathbf{p}$) predictors:

$$Y_{N \times 1} = (y_1, \dots, y_N)^T = X_{N \times (p+1)} \beta_{(p+1) \times 1} + \varepsilon_{N \times 1} = \beta_0 1_N + \sum_{j=1}^p x_j \beta_j + \varepsilon$$

- Where feature matrix: $X_{N \times (p+1)} = (1_N, x_1, \dots, x_p)$
- Feature x_j is a column vector: $x_j = (x_{1j}, x_{2j}, \dots, x_{Nj})^T$

**A column
vector of 1s**

- Parameters or coefficients: $\beta = (\beta_0, \beta_1, \dots, \beta_p)^T$

1 Intercept

p Slopes

- How to get parameter values? Answer: OLS (Ordinary Least Squares)

Linear Regression

- Minimize the loss function:

$$RSS(\beta) = \varepsilon^T \varepsilon = (Y - X \beta)^T (Y - X \beta) = \sum_{i=1}^N (y_i - f(X_i))^2 = \sum_{i=1}^N (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2$$

- Let the first-order derivative equal to 0 (Normal equations):

$$\hat{\beta}_{(p+1) \times 1} = (\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p) = (X^T X)^{-1} X^T Y$$

- Question: How to derive this formula ?

$$\min_{\beta} RSS(\beta) = \varepsilon^T \varepsilon = (Y - X \beta)^T (Y - X \beta) = Y^T Y - 2Y^T X \beta + \beta^T X^T X \beta$$

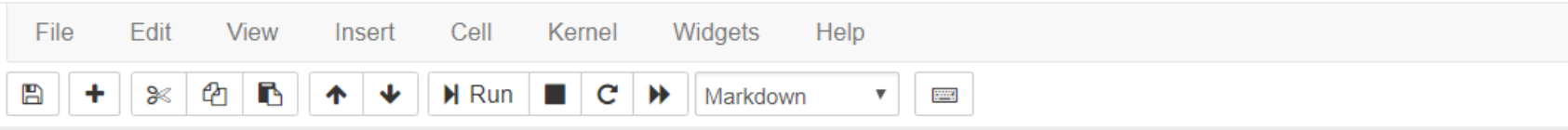
$$\rightarrow \text{FD: } -2X^T Y + 2X^T X \beta = 0$$

$$\rightarrow \text{OLS: } \beta = (X^T X)^{-1} X^T Y$$

- Matrix Differentiation: <https://www.comp.nus.edu.sg/~cs5240/lecture/matrix-differentiation-c.pdf>

Linear Regression in Python

jupyter IS4303 Tutorial Week4 Last Checkpoint: 13 minutes ago (autosaved)



3 Linear Regression

3.1 Summary of Multiple Linear Regression

From the lecture class, we know that a typical linear regression model of N observations and p predictors:

$$\begin{aligned} Y &= \beta_0 + \sum_{j=1}^p x_j \beta_j + \varepsilon \\ &= X\beta + \varepsilon \end{aligned}$$

And we aim to **Minimize** the loss function (i.e., Ordinary Least Squares, OLS):

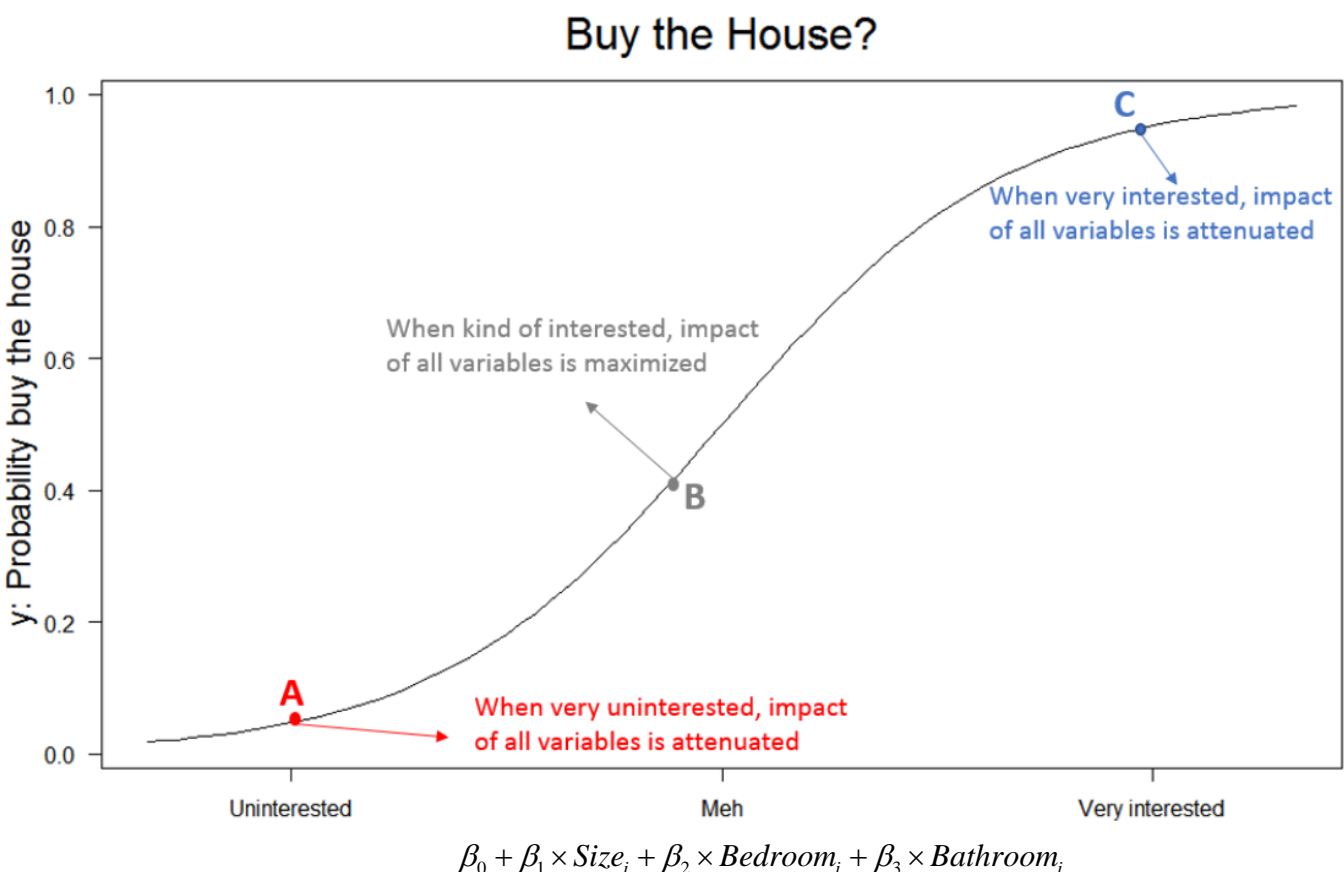
$$RSS(\beta) = \varepsilon^T \varepsilon = (Y - X\beta)^T (Y - X\beta) = \sum_{i=1}^N (y_i - 1 \times \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2$$

Solving this optimization problem, we obtain the normal equations for OLS estimations:

$$\beta_{OLS} = (X^T X)^{-1} X^T Y$$

Details of Linear Regression and OLS can be found here: https://en.wikipedia.org/wiki/Ordinary_least_squares

Logistic Regression



$$\log\left(\frac{\Pr(\text{Buy}_i = 1)}{1 - \Pr(\text{Buy}_i = 1)}\right) = \beta_0 + \beta_1 \times \text{Size}_i + \beta_2 \times \text{Bedroom}_i + \beta_3 \times \text{Bathroom}_i$$

Example: Predict whether to buy a condo

Buy

(=1, buy condo; =0, otherwise)

3 features:

$x = \{x_1: \text{size}, x_2: \text{\#bedroom}, x_3: \text{\#bathroom}\}$

Collect data from $N=1000$ people
($i=1, \dots, 1000$):

Row	Size (m ²)	#Bedroom	#Bathroom	Buy
1	65	2	2	1
2	72	3	2	0
.....
1000	50	1	1	0

Logistic Regression

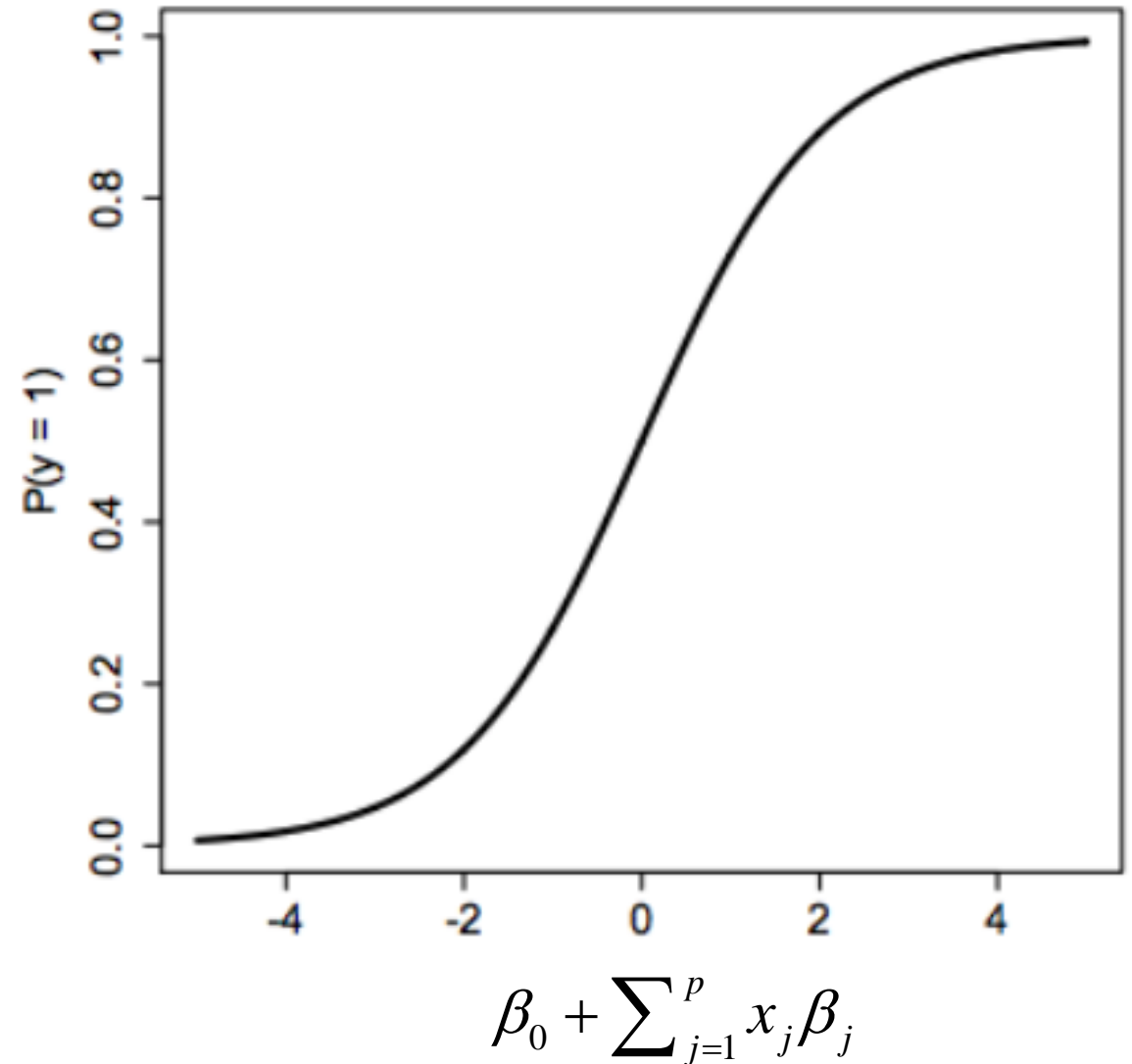
- Think about
 - General specification:

$$\text{Logodds}(\Pr(y_i = 1)) = \log\left(\frac{\Pr(y_i = 1)}{1 - \Pr(y_i = 1)}\right)$$

$$= \beta_0 + \sum_{j=1}^p x_j \beta_j$$

$$\Pr(y_i = 1) = \frac{\exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)}{1 + \exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)}$$

- Sigmoid function:
(Very Important Function)



Logistic Regression: Likelihood Function

We know: $\Pr(y_i = 1) = \frac{\exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)}{1 + \exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)}$ and $\Pr(y_i = 0) = 1 - \Pr(y_i = 1)$

So the probability of the occurrence of observation i ($=1, \dots, N$):

$$\begin{aligned} f(y_i) &= [\Pr(y_i = 1)]^{(y_i)} [\Pr(y_i = 0)]^{(1-y_i)} \\ &= \left[\frac{\exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)}{1 + \exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)} \right]^{(y_i)} \left[\frac{1}{1 + \exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)} \right]^{(1-y_i)} \end{aligned}$$

Logistic Regression: Likelihood Function

So the probability of the occurrence of all the N observations:

Likelihood Function:
$$l(\beta) = \prod_{i=1}^N \left[\frac{e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}}{1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}} \right]^{y_i} \left[\frac{1}{1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}} \right]^{(1-y_i)}$$

Take log and we will get Log-Likelihood Function:

$$ll(\beta) = \sum_{i=1}^N [-\log(1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}) + y_i (\beta_0 + \sum_{j=1}^p x_j \beta_j)]$$

Estimation

- Gradient Ascent
 - Maximizing (Log-)Likelihood:

$$l(\beta) = \prod_{i=1}^N \left[\frac{e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}}{1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}} \right]^{y_i} \left[\frac{1}{1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}} \right]^{(1-y_i)}$$

Score: $Pr(y_i=1/x, \beta)$

$$ll(\beta) = \sum_{i=1}^N [-\log(1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}) + y_i(\beta_0 + \sum_{j=1}^p x_j \beta_j)]$$

- Question: How to maximize this complex objective function?
 - Let first-order derivatives = 0 ?
 - Maybe you can take steps by steps (iteratively) to approach the (local) optimal value

Remember What We've Learned in Stats Modules

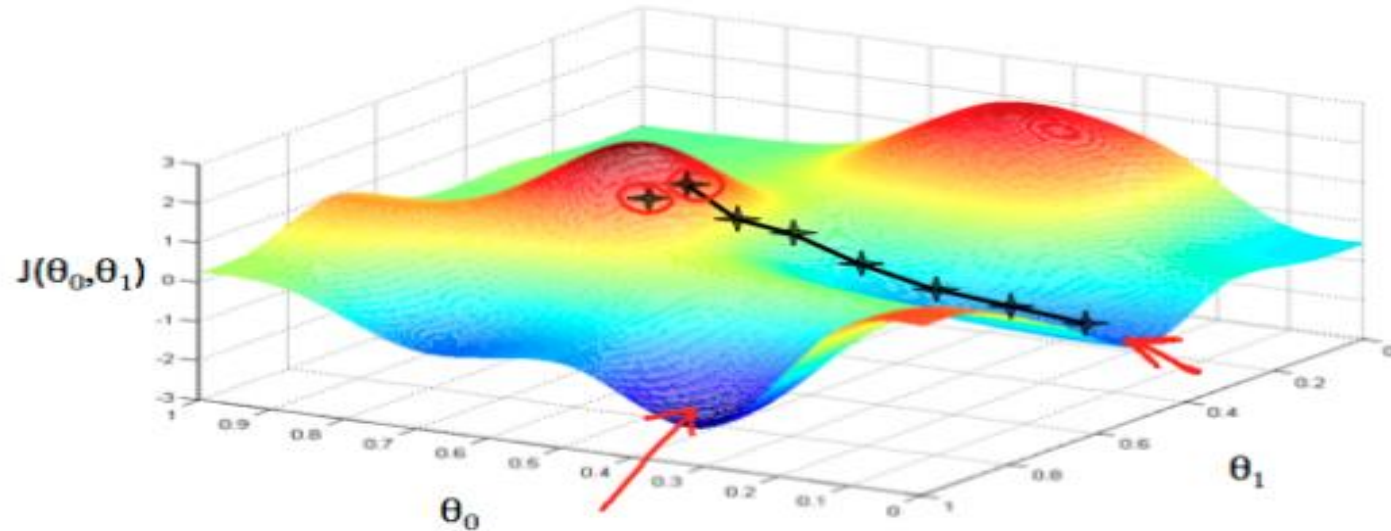
Gradient Descent

(For Minimizing)



Gradient Ascent

(For Maximizing)



Correct: Simultaneous update

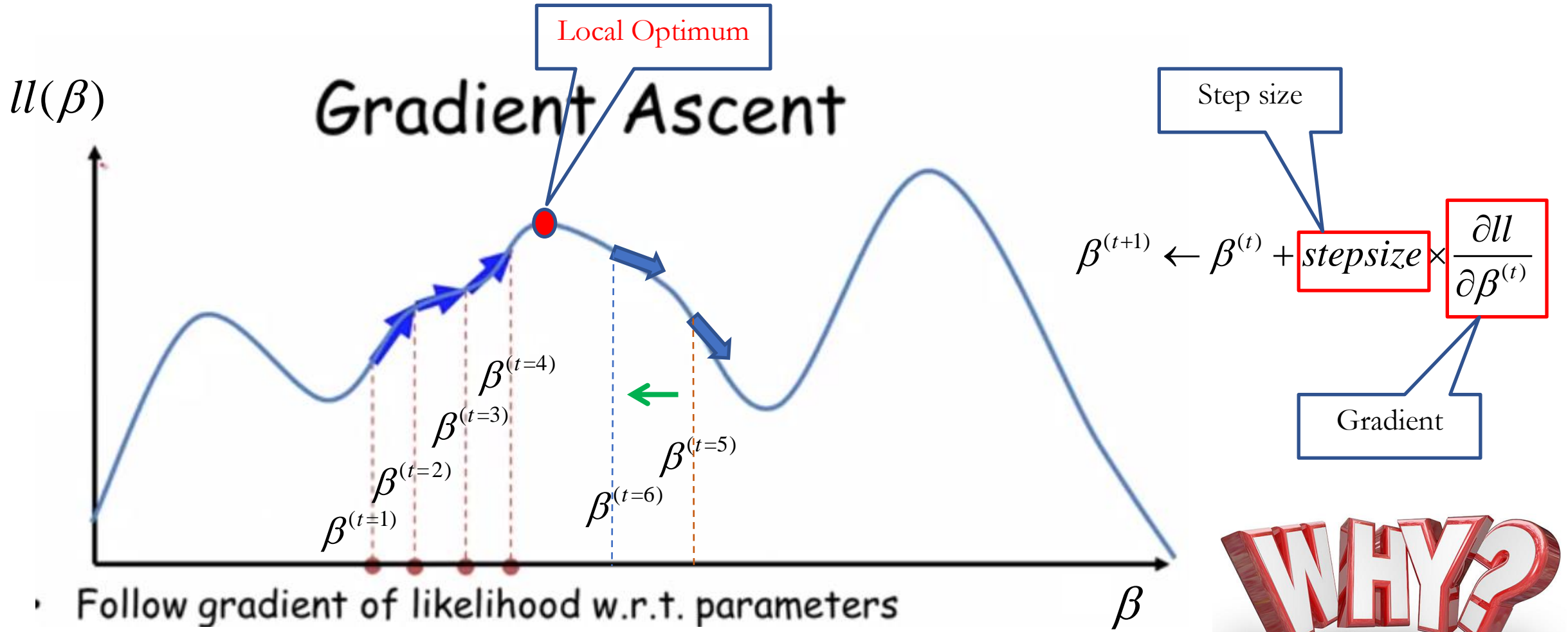
```
→ temp0 :=  $\theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$ 
→ temp1 :=  $\theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$ 
→  $\theta_0 := \text{temp0}$ 
→  $\theta_1 := \text{temp1}$ 
```

Incorrect:

```
→ temp0 :=  $\theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$ 
→  $\theta_0 := \text{temp0}$ 
→ temp1 :=  $\theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$ 
→  $\theta_1 := \text{temp1}$ 
```

- gradient *descent* aims at *minimizing* some objective function: $\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$
- gradient *ascent* aims at *maximizing* some objective function: $\theta_j \leftarrow \theta_j + \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

Overview of Gradient Ascent



Estimation

- Gradient Ascent
 - Maximizing (Log-)Likelihood:

$$ll(\beta) = \sum_{i=1}^N [-\log(1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}) + y_i(\beta_0 + \sum_{j=1}^p x_j \beta_j)]$$

Taylor Expansion

- Gradient Ascent:
 - Remember **Taylor Expansion**
 - Newton-Raphson Method
 - Hard to get $-H_t^{-1}$
 - Steepest Ascent Method:
 - Let: $-H_t^{-1} = \lambda I$
 - λ : step size
 - I : Identity Matrix

8.3.1. Newton–Raphson

To determine the best value of β_{t+1} , take a second-order Taylor's approximation of $LL(\beta_{t+1})$ around $LL(\beta_t)$:

(8.1)

$$LL(\beta_{t+1}) = LL(\beta_t) + (\beta_{t+1} - \beta_t)' g_t + \frac{1}{2}(\beta_{t+1} - \beta_t)' H_t (\beta_{t+1} - \beta_t).$$

Now find the value of β_{t+1} that maximizes this approximation to $LL(\beta_{t+1})$:

$$\frac{\partial LL(\beta_{t+1})}{\partial \beta_{t+1}} = g_t + H_t(\beta_{t+1} - \beta_t) = 0,$$

$$H_t(\beta_{t+1} - \beta_t) = -g_t,$$

$$\beta_{t+1} - \beta_t = -H_t^{-1} g_t,$$

$$\beta_{t+1} = \beta_t + (-H_t^{-1}) g_t.$$

Gradient Ascent

$$\beta_{t+1} \leftarrow \beta_t + \lambda \times g_t$$

Gradient Ascent

- Gradient Ascent

- Objective Function: $ll(\beta) = \sum_{i=1}^N [-\log(1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}) + y_i(\beta_0 + \sum_{j=1}^p x_j \beta_j)]$

- Gradient Ascent:

- (1) Initialize $\beta^{(0)} = (\beta_0^{(0)}, \beta_1^{(0)}, \dots, \beta_j^{(0)}) = (0, 0, \dots, 0), t = 1$

- (2) In step t, update coefficients:

$$\frac{\partial ll}{\partial \beta_j} \leftarrow \sum_{i=1}^N (y_i - \frac{e^{\beta_0^{(t-1)} + \sum_{j=1}^p x_j \beta_j^{(t-1)}}}{1 + e^{\beta_0^{(t-1)} + \sum_{j=1}^p x_j \beta_j^{(t-1)}}}) x_{ij}$$

Calculate gradients or first-order derivatives of coefficients

$$\beta_j^{(t)} \leftarrow \beta_j^{(t-1)} + \text{stepsize} \times \frac{\partial ll}{\partial \beta_j}$$

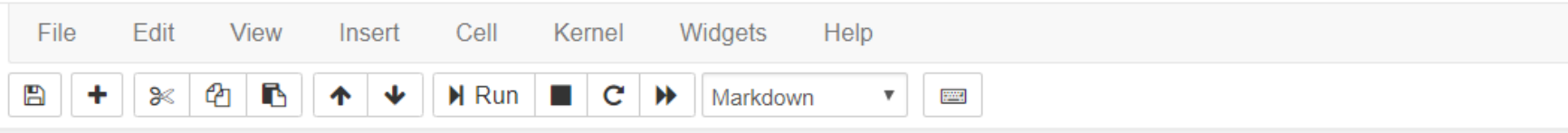
Update coefficients with Steepest Ascent Method

$$t \leftarrow t + 1$$

- (3) Check convergence condition $\|\nabla ll(\beta^{(t)})\| < \text{tolerance}$. If not, go back to (2) until (3) is satisfied

Logistic Regression in Python

jupyter IS4303 Tutorial Week4 Last Checkpoint: 12 minutes ago (autosaved)



4 Logistic Regression

4.1 Summary of Logistic Regression

From the lecture class, we know that a typical logistic regression model of N observations and p predictors:

$$\begin{aligned}\text{Logit}(P(y_i = 1)) &= \log\left(\frac{P(y_i = 1)}{1 - P(y_i = 1)}\right) \\ &= \beta_0 + \sum_{j=1}^p x_j \beta_j \\ &= X\beta\end{aligned}$$

Rewrite the model and we can get this function:

$$P(y_i = 1) = \frac{e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}}{1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}}$$

Homework Assignment 1

Revisions:

(1) Question 13:

Please fit "**rescaled** train data" with Logistic Regression model. Please report/print parameter values on the "**rescaled** train dataset". Please print the performance of model on "**rescaled** train data" using performance metric: `accuracy_score`.

(2) Question 14:

Please show model performance: `accuracy_score` of this Logistic Regression model on the "rescaled **test** dataset", not "rescaled **validation** dataset" because we have not created validation dataset in this homework.

Any Questions About Homework ?

Thank you!

Appendix:

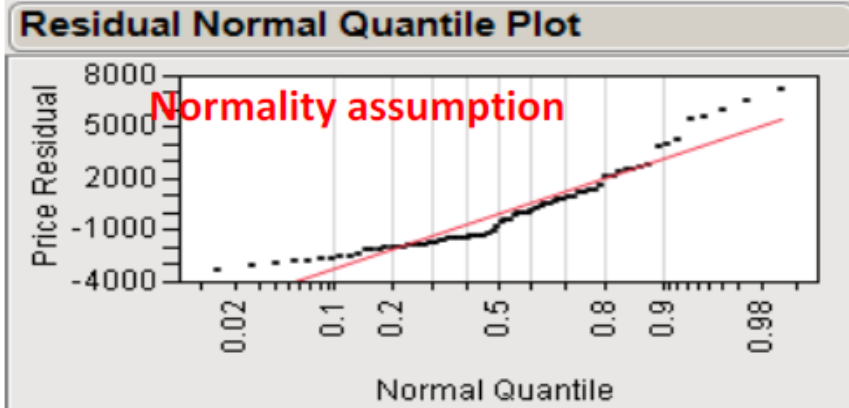
Linear Regression: Gauss-Markov Assumptions

- Linear relationship between X and Y (i.e., Linearity)
- Error term ε is normally distributed (i.e., Normality)
- $\text{Var}(\varepsilon_i)$ is constant (i.e., Homoskedasticity)
- $\text{Cov}(\varepsilon_i, \varepsilon_j) = 0, i \neq j$ (i.e., iid)
- $\text{Cov}(X_i, \varepsilon_i) = 0$ (i.e., Exogeneity)
- No multi-collinearity (i.e., Full column rank)

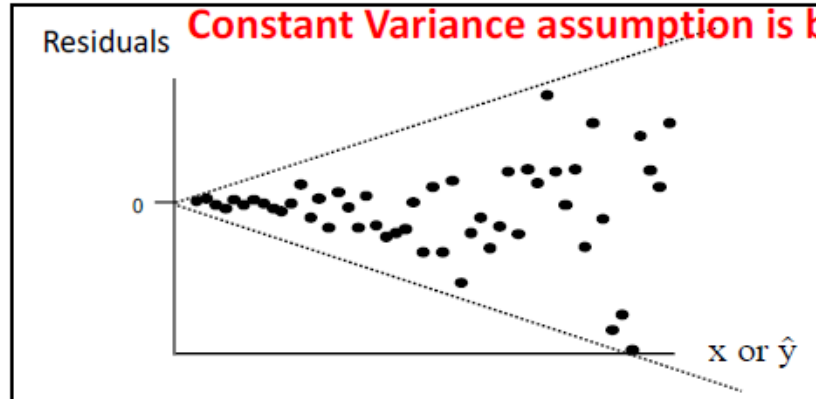
By analyzing residuals, we can check whether the assumptions of regression are satisfied.

Differing Variance →

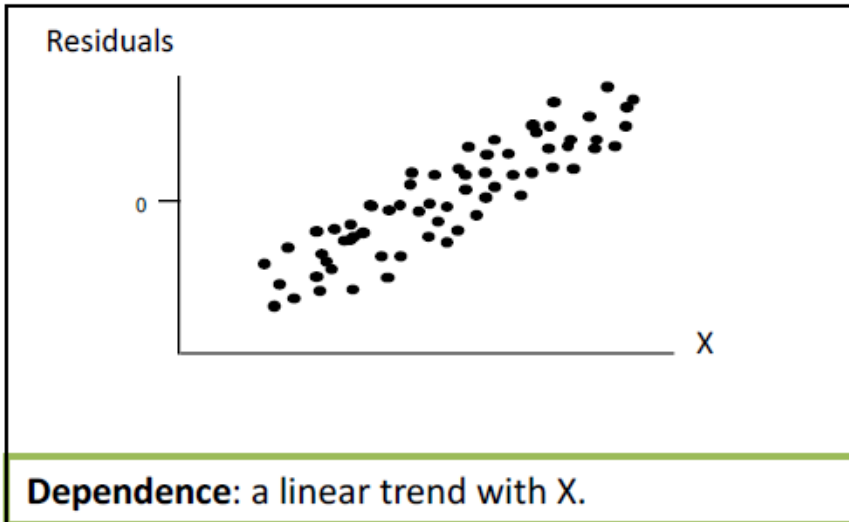
Constant Variance assumption is broken.



Normality: If data are distributed near to the red line, no problem with the normality assumption.

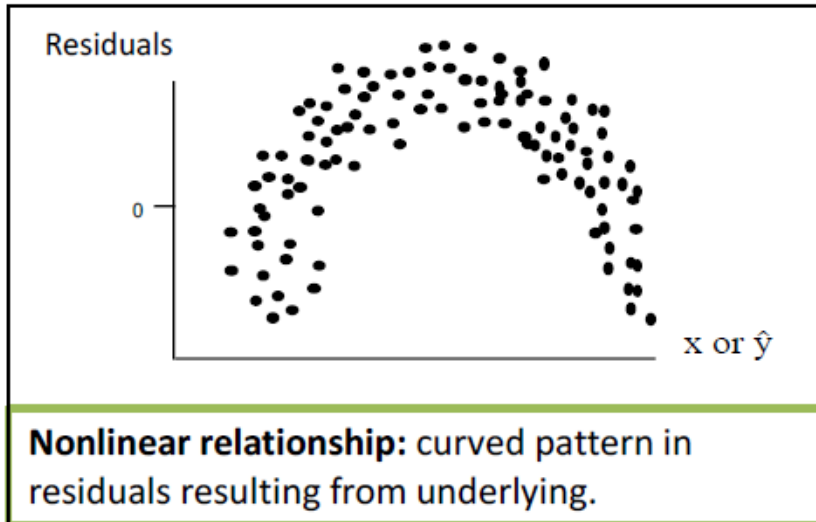


Heteroscedasticity: Variance of residuals changes when x changes.



Dependence: a linear trend with X.

Independence assumption is broken.



Nonlinear relationship: curved pattern in residuals resulting from underlying.

Linearity, Independence, Constant Variance assumptions are broken.

Appendix:

Logistic Regression: Maximum Likelihood Function

We know: $\Pr(y_i = 1) = \frac{\exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)}{1 + \exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)}$ and $\Pr(y_i = 0) = 1 - \Pr(y_i = 1)$

So the probability of the occurrence of observation i ($=1, \dots, N$):

$$\begin{aligned} f(y_i) &= [\Pr(y_i = 1)]^{(y_i)} [\Pr(y_i = 0)]^{(1-y_i)} \\ &= \left[\frac{\exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)}{1 + \exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)} \right]^{(y_i)} \left[\frac{1}{1 + \exp(\beta_0 + \sum_{j=1}^p x_j \beta_j)} \right]^{(1-y_i)} \end{aligned}$$

Appendix:

Logistic Regression: Maximum Likelihood Function

So the probability of the occurrence of all the N observations:

$$\text{Likelihood Function: } l(\beta) = \prod_{i=1}^N \left[\frac{e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}}{1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}} \right]^{y_i} \left[\frac{1}{1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}} \right]^{(1-y_i)}$$

Take log and we will get Log-Likelihood Function:

$$ll(\beta) = \sum_{i=1}^N [-\log(1 + e^{\beta_0 + \sum_{j=1}^p x_j \beta_j}) + y_i(\beta_0 + \sum_{j=1}^p x_j \beta_j)]$$