# CS 484
# Introduction to Machine Learning

**Week 1, January 21, 2021**

Spring Semester 2021

ILLINOIS TECH

College of Computing

# The Instruction Team

**Instructor**

Dr. Ming-Long Lam

mlam5@iit.edu

**Teaching Assistant / Grader**

Mr. Olugbenga Abdulai

oabdulai@hawk.iit.edu

# About the Instructor: Ming-Long Lam

- Twenty-nine years of experience in practicing data science for:
  - Predictive analytics software development
  - Property and casualty insurance ratemaking
  - Human resources analytics for retail banking service
- Passion and expertise
  - Develop analytics algorithms that enable machines to learn
  - Educate humans to mine data for insights
  - Develop commercial analytics software
- Ph.D. in Statistics from The University of Chicago
- Currently a Principal Research Statistician Developer of the SAS Institute, Cary, North Carolina
- Previously held senior R&D technical positions in Chase, Allstate, and SPSS
- ResearchGate profile: https://www.researchgate.net/profile/Ming_Long_Lam
- LinkedIn profile: https://www.linkedin.com/pub/ming-long-lam/10/a73/657

# This is a Survey Course

**We Will**

- Learn the primary topics of a broad field of knowledge
- Understand the main machine learning algorithms in each topic
- Develop computer codes to implement or exercise these algorithms

**We Won't**

- Visit every machine learning topic (new ideas are constantly introduced)
- Go through every algorithm in each topic
- Teach basic Calculus and Probability (you need to refresh these topics)

# Course Schedule

1. Introduction

*Unsupervised Learning*

2. Memory Based Learner*
3. Association Rules
4. Clustering*

*Supervised Learning*

5. Decision Trees
6. Multinomial Logistic Regression*
7. Learner Evaluation and Comparison

*Advanced Topics*

8. Feature Selection
9. Naïve Bayes
10. Neural Networks*
11. Support Vector Machines
12. Bagging, Adaptive Boosting*
13. Gradient Boosting

* Assignment Week

# Software and Textbook for This Section

- **Mandatory**: Python 3.7 or above
- Standard modules: numpy, pandas, scipy, sklearn, and statsmodel


- Available from Anaconda https://www.anaconda.com/distribution


- **Recommended**: Ming-Long Lam (2020). *A Practitioner's Guide to Machine Learning,* Kendall Hunt Publishing


- https://he.kendallhunt.com/product/practitioners-guide-machine-learning

# Evaluation


## Homework Assignments

- Five Assignments
- Each Assignment Weight 10% Toward Final Grade

## Mid-Term

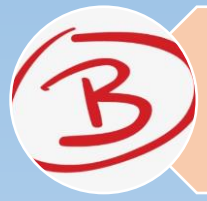
- Weight 25%
- Conduct Only Online After the March 4 Lecture

## Final Examination

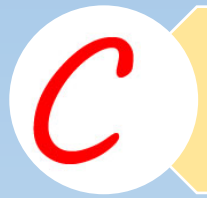
- Weight 25%
- Conduct Only Online After the Last Lecture

# Score to Grade


$$90\% \leq S$$




$$75\% \leq S < 90\%$$




$$60\% \leq S < 75\%$$


$$S < 60\%$$

# Assignments

- Five assignments will be given on
  - **Week 2**, **Week 4**, **Week 6**, **Week 10**, and **Week 12**
  - Each assignment's grade count towards 10% of your course grade
- Submit your assignment as **PDF** documents
  - Each assignment carries a maximum of 100 points
- Submit your assignments to the Blackboard site before the due date
  - Due date is the end of Sunday after the next class day
  - For example, the January 28 assignment is due at 11:59 pm on February 7.
- Whenever applicable, properly documented and error-free Python program codes and output should be included your submission

# Late Submission Policy

- If you turn in an assignment late, then we reserve our rights to deduct **5 points** from the total score for each 24-hour period after the deadline.
- Assignments turned in more than 7 days late will not receive credit.
- In the case of unexpected events (e.g., sickness, emergency, computer crash, loss of internet, etc.), you must contact the instructor and explain your situation before the assignment due date in order to receive a grace period.

# Quick Knowledge Poll on Blackboard

**Statistics**

Linear Regression

Normal Distribution

**Matrix Algebra**

Matrix Multiplication and Inversion

Eigenvalues and Eigenvectors

**Programming**

Python

**Calculus**

Differentiation

Integration

**Optimization**

Newton-Raphson or Gradient Descent

Maximum Likelihood Estimation

# What is Machine Learning?

C-3PO is a master of natural language processing and he knows over six million forms of communication, including English, in the galaxy

R2-D2 analyzes a problem logically, learns from past experiences, selects the right tool for the problem, and courageously reach the solution.

# What is Data Mining?

- Originally called Knowledge Discovery Process (KDD), Data Mining is a field of science that discover or find out the properties of datasets.
- Large amounts of observations (or records) from relational database management system (RDMS) in data warehouses are analyzed (a.k.a. mined) to identify interesting correlations and formulate patterns among the data items.
- The term “Knowledge Discovery in Databases” was coined by Gregory Piatetsky-Shapiro in 1989. The term “data mining” appeared in the database community in 1990. https://www.kdnuggets.com/gps.html

# What is Machine Learning?

The term Machine Learning was coined by Arthur L. Samuel (1901 – 1990) in his 1959 paper.

- Received a master's degree in Electrical Engineering from MIT in 1926
- Developed the first Checkers program on IBM's first commercial computer in 1952
- Retired from IBM in 1966 and joined Stanford University as a professor
- Worked on the TeX project around 1978

Arthur Samuel playing Checkers on the IBM 701 in 1956.

Source: http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/ibm700series/impacts/

# Historical Definitions

- A field of study that gives computers the ability to learn without being explicitly programmed. – Arthur L. Samuel (1959). "Some Studies in Machine Learning Using the Game of Checkers". *IBM Journal of Research and Development*.
- A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. – Tom Mitchell (1997). Machine Learning, New York: McGraw-Hill
  - Tom Mitchell (1951 – ), currently Professor, Computer Science Department, Carnegie Mellon University

# Definitions in the Dictionary

- Machine Learning is the process by which a computer is able to improve its own performance (as in analyzing image files) by continuously incorporating new data into an existing statistical model. — **Merriam-Webster**, https://www.merriam-webster.com/dictionary/machine%20learning

- Machine learning, in artificial intelligence (a subject within computer science), is a discipline concerned with the implementation of computer software that can learn autonomously. – **Britannica**, https://www.britannica.com/technology/machine-learning

# Definitions on the Internet

- Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. – **Wikipedia**, https://en.wikipedia.org/wiki/Machine_learning

# Definitions from Tech Company

- A program or system that builds (trains) a predictive model from input data. The system uses the learned model to make useful predictions from new (never-before-seen) data drawn from the same distribution as the one used to train the model. Machine learning also refers to the field of study concerned with these programs or systems. – **Google**, https://developers.google.com/machine-learning/glossary/#m

- Machine learning is a form of AI that enables a system to learn from data rather than through explicit programming. However, machine learning is not a simple process. As the algorithms ingest training data, it is then possible to produce more precise models based on that data. A machine-learning model is the output generated when you train your machine-learning algorithm with data. — **IBM**, https://www.ibm.com/analytics/machine-learning

# Definitions from Tech Company

- Machine learning (ML) is the process of using mathematical models of data to help a computer learn without direct instruction. It's considered a subset of artificial intelligence (AI). Machine learning uses algorithms to identify patterns within data, and those patterns are then used to create a data model that can make predictions. With increased data and experience, the results of machine learning are more accurate — much like how humans improve with more practice. — **Microsoft**, https://azure.microsoft.com/en-us/overview/what-is-machine-learning-platform/

- Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. – **SAS Institute**, https://www.sas.com/en_us/insights/analytics/machine-learning.html#machine-learning-importance

# Machine Learning (Tom Mitchell,1997)

- A **computer program** is said to **learn from experience** E with respect to **some task** T and **some performance measure** P, if its performance on T, as measured by P, improves with experience E.
- When a machine learns, it involves three components.
  - Experience – What does a machine learn from?
  - Task – What does a machine learn for?
  - Performance Measure – How well did a machine learn?

# Data Mining versus Machine Learning

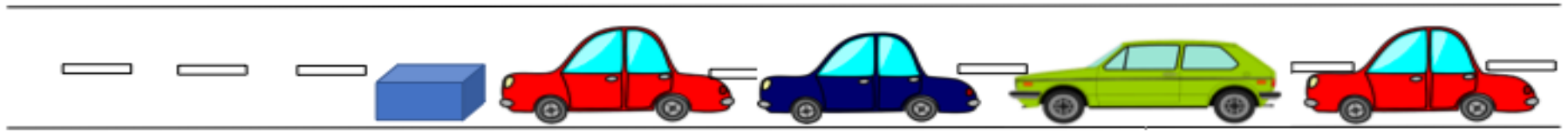
- Machine learning and data mining often use the same algorithms to discover patterns in the data.

- Machine learning uses algorithms to improve its performance at a task with experience over time.
- Emphasis is on the accuracy and the ability to reuse the algorithm outcomes in the future.

- Data mining uses algorithms to generate insights for humans to identify and interpret the data patterns.
- Emphasis is on the ability to maximize the utility of an algorithm for the current data.

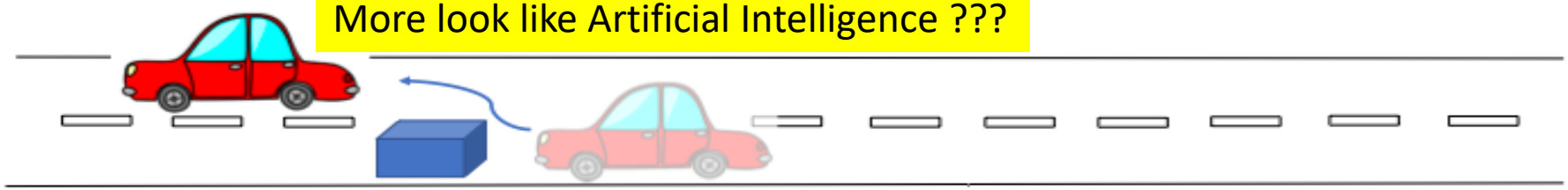
# Data Mining versus Machine Learning

Data Mining vs. Machine Learning Illustration:



Data Mining: It takes time to batch and cluster data. If an obstacle is encountered, it cannot be dealt with in real time. It needs to wait for an analysis to be initiated by a person.

More look like Artificial Intelligence ???



Machine Learning: The machine detects what is relevant to the task. If an obstacle is encountered, and analysis can be done inline, without a human stepping into the loop.

Source: https://www.guavus.com/artificial-intelligence-vs-machine-learning-vs-data-mining-101-whats-big-difference

# A Human Learning Activity

- A human learning activity also contains these three components.
- A **person** is said to **learn from experience** E with respect to **some task** T and **some performance measure** P, if its performance on T, as measured by P, improves with experience E.
- A person learns to
  - EITHER do something better (i.e., improve benefits, accuracy, or profit)
  - OR avoid a mistake (i.e., reduce cost, error, or loss)

# Three Components of a Learning Activity

- For a student in this CS 484 course
  - Experience – lecture, textbook, assignments, tests, and exam
  - Task – learn various topics in machine learning
  - Performance Measure – Letter grade (A > B > C > E)
- For a loan officer
  - Experience – historical data where 20% of loans have defaulted
  - Task – learn to identify loans whose default rates are substantially higher or lower than 20%
  - Performance Measure – misclassification rate which is the fraction of observations that are misclassified by your model, and lower is better

# Before a Learning Activity


- Before we dive into any learning activity,
- Take a minute or two to ask yourself this question.
  - What is the primary motivation for this learning activity?
  - In other words, Why am I Learning This?
- Some possible answers are:
  - Identify loans that have high likelihoods of default
  - Establish a profile of the most valuable customers
  - Estimate the current value of a real estate property
  - Study browsing history of users who eventually land on Amazon

# The Loan Officer Learning Example

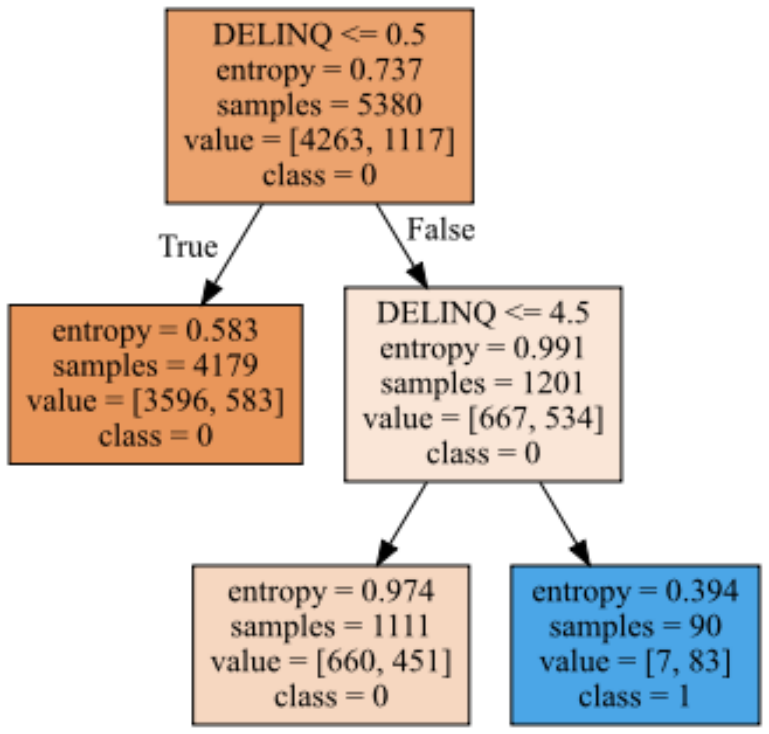
- Experience tells us that 20% of loans have defaulted.
- If we do not have access to the data (i.e., the experience), then we will say that the Loan Default probability is 0.2.
  - This is a One-Size-Fits-All solution.
- Suppose we will declare (i.e., classify) a loan is going to default if its Loan Default probability is greater than or equal to $p_0$.
  - If $p_0 \leq 0.2$, then all the loans will be classified as defaulted. As a result, 80% = (1 – 0.2 = 0.8) of the loans will be misclassified.
  - If $p_0 > 0.2$, then none of the loans will be classified as defaulted. As a result, 20% of loans will be misclassified.

# The Loan Officer Learning Example

- Experience tells us that 20% of loans have defaulted.
- If we have access to the data (i.e., the experience), then we will divide the data into segments. The average Loan Default probabilities in some segments are higher than 0.2 and some are lower than 0.2. The data is in `hmeq.csv`.
- We will use the Classification Tree algorithm to learn from the data. We will cover this popular algorithm in a few weeks.
- Run the Python code `Week 1 MyFirstDecisionTree.py` to check if you have the right environment

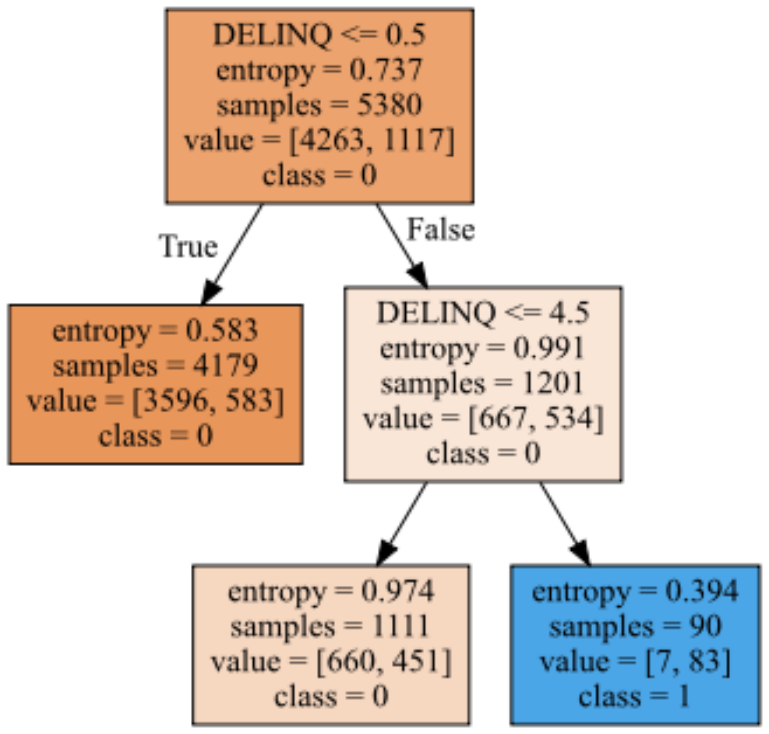
# The Loan Officer Learning Example

- The DELINQ variable is the number of credit inquiries (0, 1, …,15)
- Value is the pair of [number of active loans, number of defaulted loans]
- The training data contains 5,380 non-missing observations which have 1,117 defaulted loans. Thus, the default rate is 1117/5380 = 0.2076.



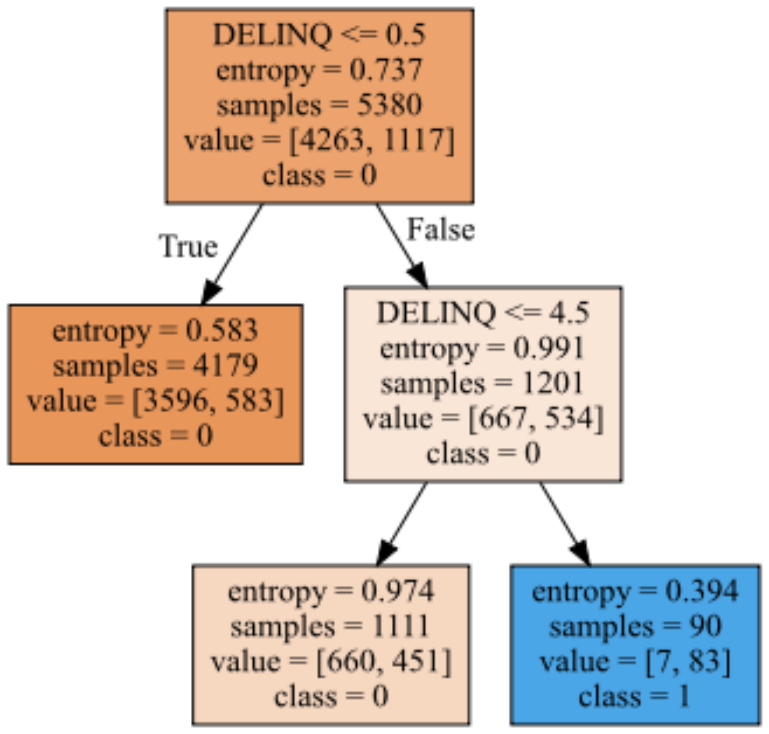
# The Loan Officer Learning Example

- There are 4,179 observations without any inquiries (i.e., DELINQ <= 0.5). The default rate is 583/4179 = 13.95%.
- There are 1,201 observations which have at least one inquiry (i.e., DELINQ > 0.5). The default rate is 534/1201 = 44.46%.



# The Loan Officer Learning Example

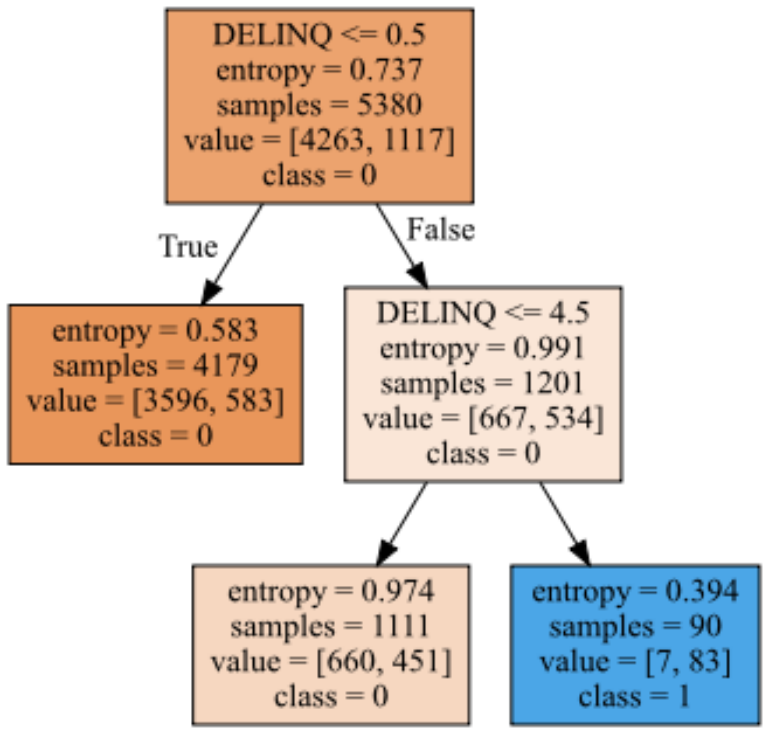
- There are 1,111 observations which have 1, 2, 3 or 4 inquiries (i.e., 0.5 < DELINQ <= 4.5). The default rate is 451/1111 = 40.59%.
- There are 90 observations which have at least five inquiries (i.e., DELINQ > 4.5). The default rate is 83/90 = 92.22%.



# The Loan Officer Learning Example

After learning from the data, we produced these rules.

1. If DELINQ = 0, then the predicted Default Likelihood = 13.95%.
2. If DELINQ = 1, 2, 3 or 4, then the predicted Default Likelihood = 40.59%.
3. If DELINQ > 4, then the predicted Default Likelihood = 92.22%.



# The Loan Officer Learning Example

Suppose we will declare (i.e., classify) a loan is going to default if its Loan Default probability is greater than or equal to $p_0$.

- If $p_0 \leq 0.1395$, then all the loans → defaulted. The Misclassification Rate is 4263/5380 = 79.24%.
- If $0.1395 < p_0 \leq 0.4059$, then the loans under Rule 2 and Rule 3 → defaulted. In other words, if DELINQ > 0 → defaulted. The Misclassification Rate is 23.23%.

1. If DELINQ = 0, then the predicted Default Likelihood = 13.95%.
2. If DELINQ = 1, 2, 3 or 4, then the predicted Default Likelihood = 40.59%.
3. If DELINQ > 4, then the predicted Default Likelihood = 92.22%.

# The Loan Officer Learning Example

- If $0.4059 < p_0 \leq 0.9222$, then the loans under Rule 3 → defaulted. In other words, if DELINQ > 4 → defaulted. The Misclassification Rate is 19.35%.
- If $0.9222 < p_0$, then all loans will be classified as non-default. The Misclassification Rate is 20.76%.

1. If DELINQ = 0, then the predicted Default Likelihood = 13.95%.
2. If DELINQ = 1, 2, 3 or 4, then the predicted Default Likelihood = 40.59%.
3. If DELINQ > 4, then the predicted Default Likelihood = 92.22%.

# The Loan Officer Learning Example

- Without learning from the experience (i.e., the data), the misclassification rate is either 80% or 20%.
- After learning from the experience by using the classification tree algorithm, the misclassification rate is 79.24%, 23.23%, 19.35%, and 20.76%.
- The benefits of performing this learning activity are:
  1. We get away from the One-Size-Fits-All solution, have more risk levels for consideration even although we still cannot attain 0% misclassification rate.
  2. We can identify a small segment of loans (DELINQ > 4 in 1.5% of the loans) which exhibit a very high default likelihood (92.22%). In other words, we have found a credible indicator of default for a loan.

# After a Learning Activity

- After we wrap up any learning activity,
- Take a few minutes to answer this question.
  - What is the main outcome of this learning activity?
  - In other word, What Did I Learn?
- Some possible answers are:
  - The most valuable customers bought something recently, very often, and expensive items.
  - I conclude that I cannot learn anything using the data that I have (e.g., all customers eventually bought something from Amazon)
- If you cannot figure out what you've learned or cannot learn, then the learning activity is simply an activity for you to spend some time.

# Follow-Up a Learning Activity

**Reproducibility**

- If we start from the original data, use the same algorithms, execute the same tasks on the same or compatible machine, the machine learning activity will reproduce the same results (within the machine precisions) and arrive at the same conclusions.
- If the expected results cannot be reproduced, this indicates there are some unexplained (intentional or random) interactions among the data, the algorithm, and the machine.
- Common causes are uninitialized variables in the codes or incomplete or inaccurate documentation of the activity.

# Follow-Up a Learning Activity

**Replicability**

- If a researcher runs the machine learning activity (i.e., use the same algorithms, execute the same tasks) using a different data on some compatible machine, the researcher can obtain results that lead to the same conclusions (e.g., the list of selected features, the distribution of the predicted outcomes).
- If the expected conclusions cannot be replicated, this indicates there are some design flaws in the tasks. For example, some correlations among the features are not accounted for, the algorithms need further tunings, additional tasks are required, or different algorithms should be considered.
- We will come back to this topic when we talk about Learner Evaluation and Comparison

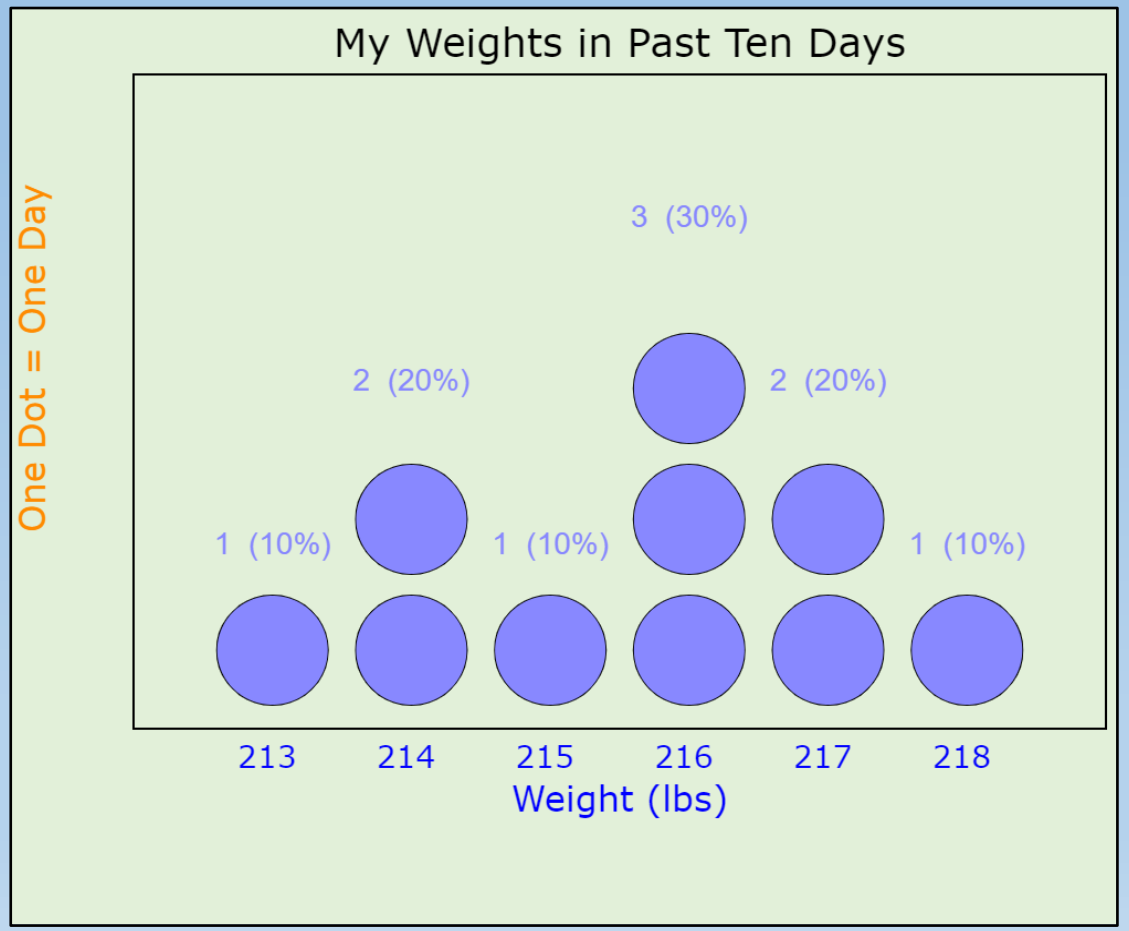
# A Simple Machine Learning Activity

**Goal**: Estimate Empirical Density Function

- **Idea**: Use a histogram
- **Activity**: Generate a histogram
- **Khan Academy**: A histogram is a graphical display of data using bars of different heights. In a histogram, each bar groups numbers into ranges. Taller bars show that more data falls in that range. A histogram displays the shape and spread of continuous sample data.

# Represent Empirical Density Function

- Suppose my weights (pounds) of the past ten days are:
  213, 214, 214, 215, 216, 216, 216, 217, 217, 218
- How my weights vary?
- Use a Dot Plot
- https://www.mathsisfun.com/data/data-graph.php



# Represent Empirical Density Function

- Frequency Table
  - List the unique values and their counts (i.e., number of observations that have the unique values)
  - Useful for knowing what values we have in the data and their counts
  - However, the table does not easily show the distances among values

- Histogram
  - Display a macro-level view of how the values distribute
  - Useful for getting a glimpse of the data distribution
  - Easy to draw a histogram, but need skills to make an informative histogram

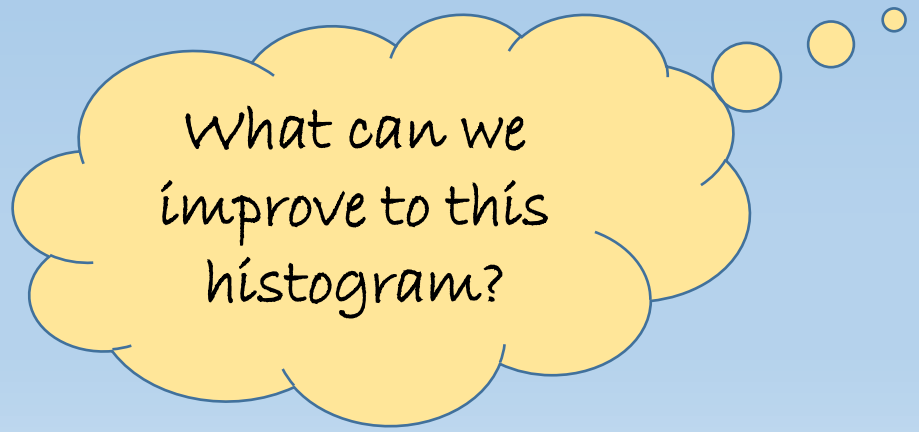
# Represent Empirical Density Function

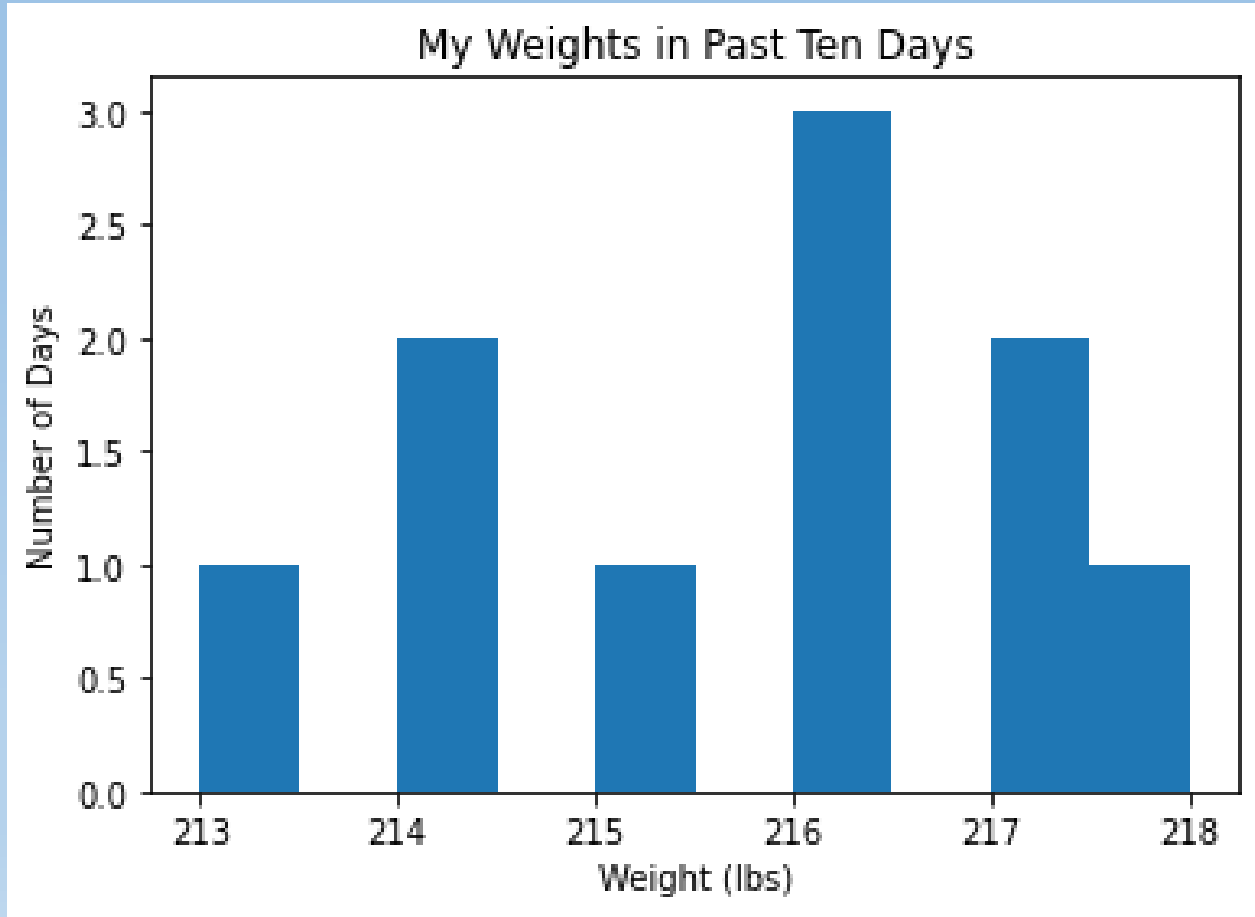
Week 1 Simple Histogram.py

```
import matplotlib.pyplot as plt
import numpy

x = numpy.array([213,214,214,215,216,216,216,217,217,218])

# Generate a frequence table
uvalue, ucount = numpy.unique(x, return_counts = True)
print('Unique Values:\n', uvalue)
print('Unique Counts:\n', ucount)

# Draw a properly labeled histogram with default specification
plt.hist(x)
plt.title('My Weights in Past Ten Days')
plt.xlabel('Weight (lbs)')
plt.ylabel('Number of Days')
plt.show()
```

# Represent Empirical Density Function

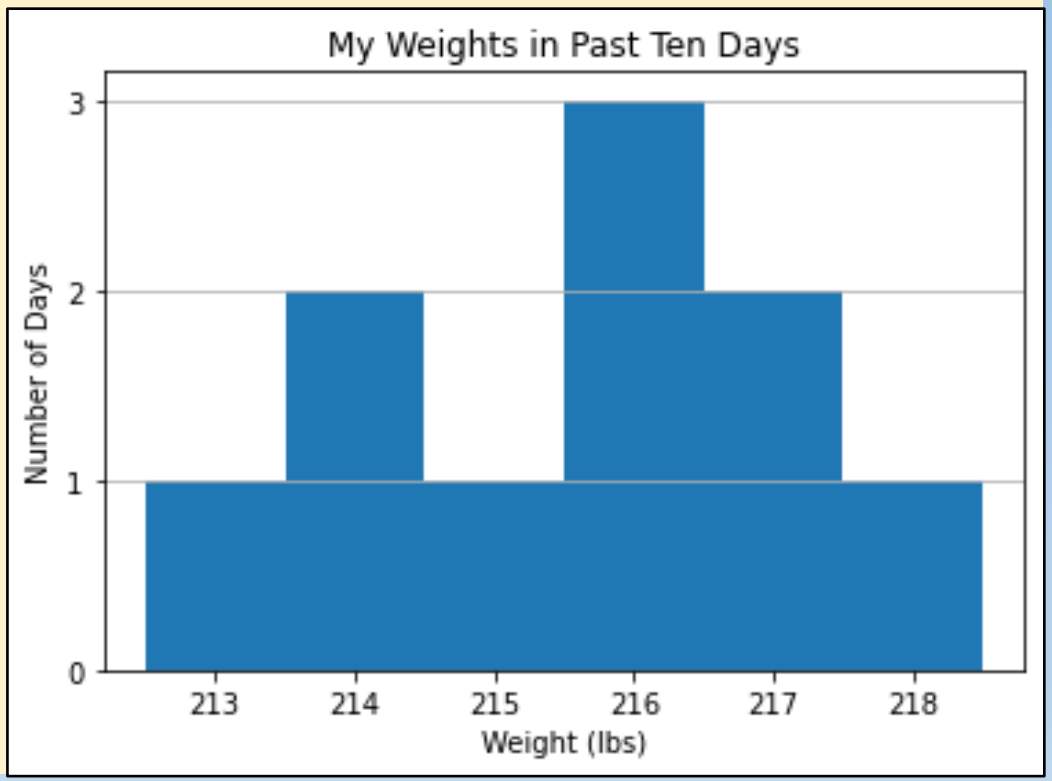
```
Unique Values:
 [213 214 215 216 217 218]
Unique Counts:
 [1 2 1 3 2 1]
```



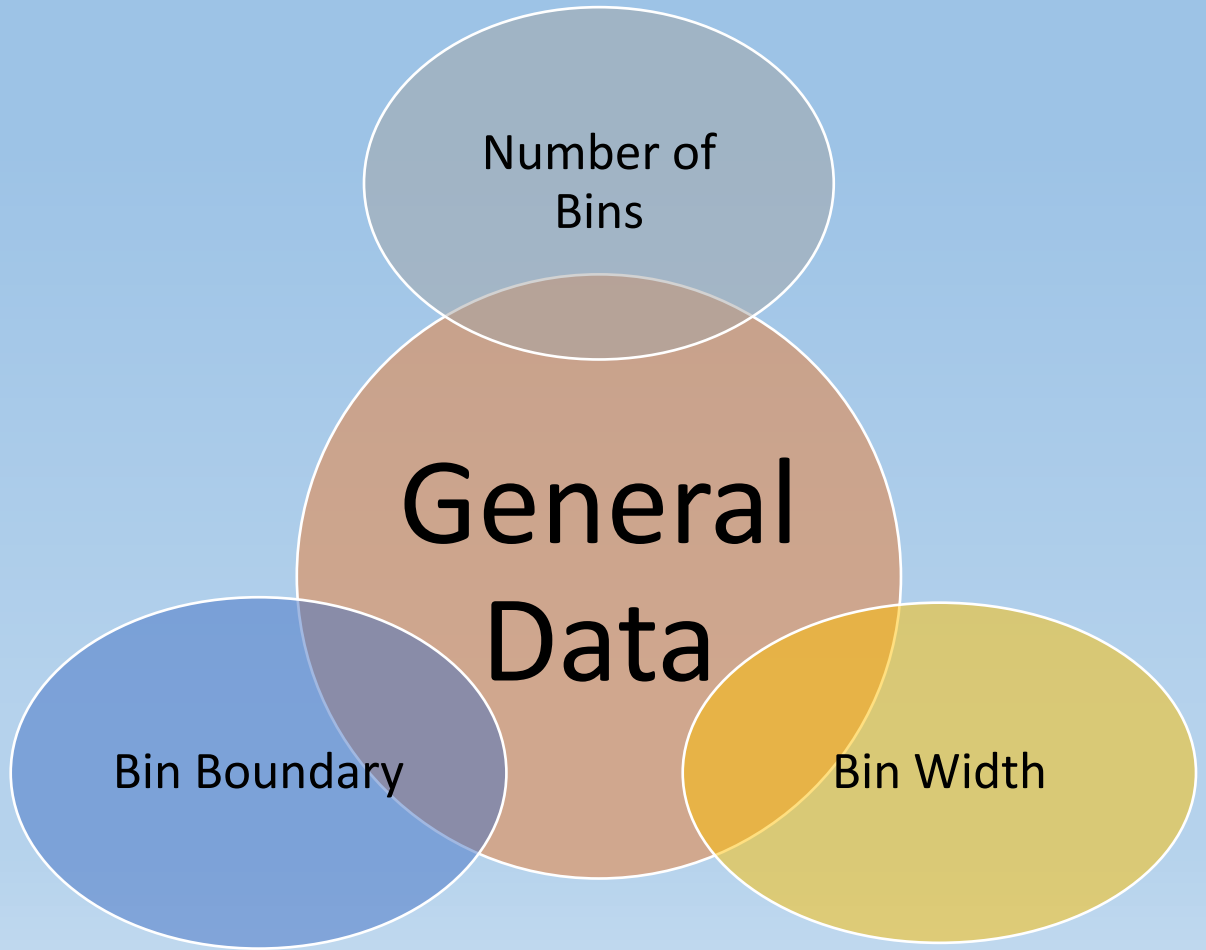



# Represent Empirical Density Function

```
# Draw a better labeled histogram with specified bin boundaries (bin width = 1)
plt.hist(x, bins = [212.5,213.5,214.5,215.5,216.5,217.5,218.5], align='mid')

# Specify title, labels
plt.title('My Weights in Past Ten Days')
plt.xlabel('Weight (lbs)')
plt.ylabel('Number of Days')

# Specify y-axis tick values
plt.yticks(range(4))

# Show grid line on y-axis
plt.grid(axis = 'y')

# Show the histogram
plt.show()
```



# Represent Empirical Density Function




Three Related Specifications

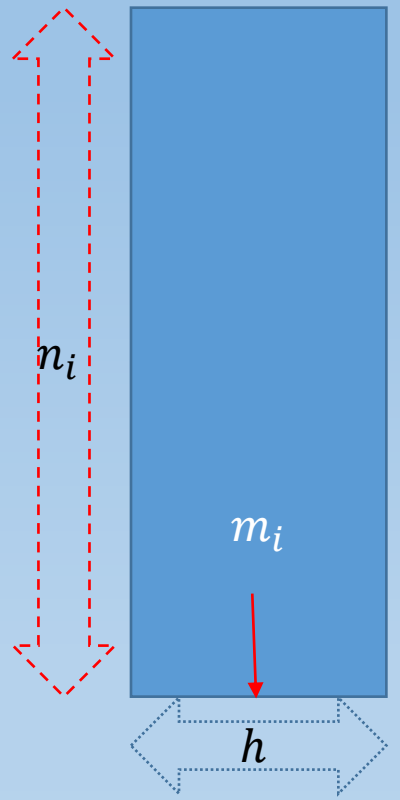
1. Number of Bins
   - Avoid empty bins
2. Bin Width
   - Fixed or Variable
   - Total span of bins should represent the data range
3. Bin Boundary
   - Nice looking and intelligent
   - Resemble data values

# A Procedure for Constructing a Histogram

1. Select a positive bin width.
2. Select the mid-points of the bins
3. Construct the bins
   a) Left-open and right-closed, i.e., (a,b], except for first bin [a, b]
   b) Left-closed and right-open, i.e., [a,b), except for last bin [a, b]
4. Count the number of observations in each individual bin
5. Plot the numbers of observations against the bin definitions, either horizontally or vertically.

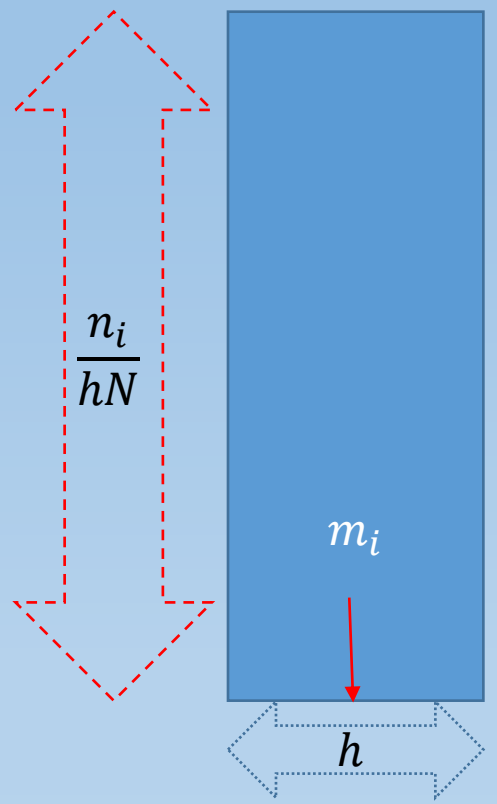
# Use a Histogram to Estimate the Density

- Let $h > 0$ be the bin-width.
- Let $m_i$ be the mid-point of the $i^{\text{th}}$ bin. Let us represent the bin by this left-open-right-close interval $(m_i - h/2\,, m_i + h/2]$.
- Let $n_i \geq 0$ be the number of observations in the $i^{\text{th}}$ bin. Let $N = \sum_i n_i$.
- Thus, the area of the rectangle that represents the $i^{\text{th}}$ bin is $hn_i$ (base width multiple the rectangle height)



# Use a Histogram to Estimate the Density

- The total area of all the rectangles is $\sum_i (hn_i) = h\sum_i n_i = hN$.
- Since the area under the density polygon must be one, we need to divide the area of each rectangle by $hN$.
- Since the base of the rectangle is always $h$, therefore, the height of the $i^{\text{th}}$ rectangle in the density estimator is $\frac{n_i}{hN}$.



# Use a Histogram to Estimate the Density

- The density estimate is $\hat{p}(m_i) = \frac{\#\left\{x: x \in (m_i - \frac{h}{2}, m_i + \frac{h}{2}]\right\}}{Nh}$ where #() is the count of observation in the interval
- Suppose $x_j, j = 1, \ldots, N$ are the observations
- Alternatively, the density estimate is $\hat{p}(m_i) = \frac{1}{Nh} \sum_{j=1}^{N} w\left(\frac{x_j - m_i}{h}\right)$
- The weight function is $w(u) = \begin{cases} 1, & -\frac{1}{2} < u \leq \frac{1}{2} \\ 0, & \text{otherwise} \end{cases}$

# Density Estimation Example

- The observations are 0.4, 0.6, 0.7, 1.9, 2.4, 6.1, 6.2, and 7.3 ($N = 8$)
- For $h = 2$, choose mid-points as 1, 3, 5, and 7.
- Consider $m_1 = 1$.

| $x_i$ | 0.4 | 0.6 | 0.7 | 1.9 | 2.4 | 6.1 | 6.2 | 7.3 |
|---|---|---|---|---|---|---|---|---|
| $u = (x_i - m_1)/2$ | -0.3 | -0.2 | -0.15 | 0.45 | 0.7 | 2.55 | 2.6 | 3.15 |
| $w(u)$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

- The density estimate for $m_1 = 1$ is $\hat{p}(1) = \Sigma\, w(u) / (Nh)$ $= 4 / (8*2) = 0.25$.

# Density Estimation Example

- The observations are 0.4, 0.6, 0.7, 1.9, 2.4, 6.1, 6.2, and 7.3 ($N = 8$)
- For $h = 2$, choose mid-points as 1, 3, 5, and 7.
- Consider $m_1 = 3$.

| $x_i$ | 0.4 | 0.6 | 0.7 | 1.9 | 2.4 | 6.1 | 6.2 | 7.3 |
|---|---|---|---|---|---|---|---|---|
| $u = (x_i - m_1)/2$ | -1.3 | -1.2 | -1.15 | -0.55 | -0.3 | 1.55 | 1.6 | 2.15 |
| $w(u)$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

- The density estimate for $m_1 = 3$ is $\hat{p}(3) = \Sigma\, w(u) / (Nh) = 1 / (8*2) = 0.0625$.
- The density estimates are $\hat{p}(1) = 0.25$, $\hat{p}(3) = 0.0625$, $\hat{p}(5) = 0$, and $\hat{p}(7) = 0.1875$.

# Density Estimation Example

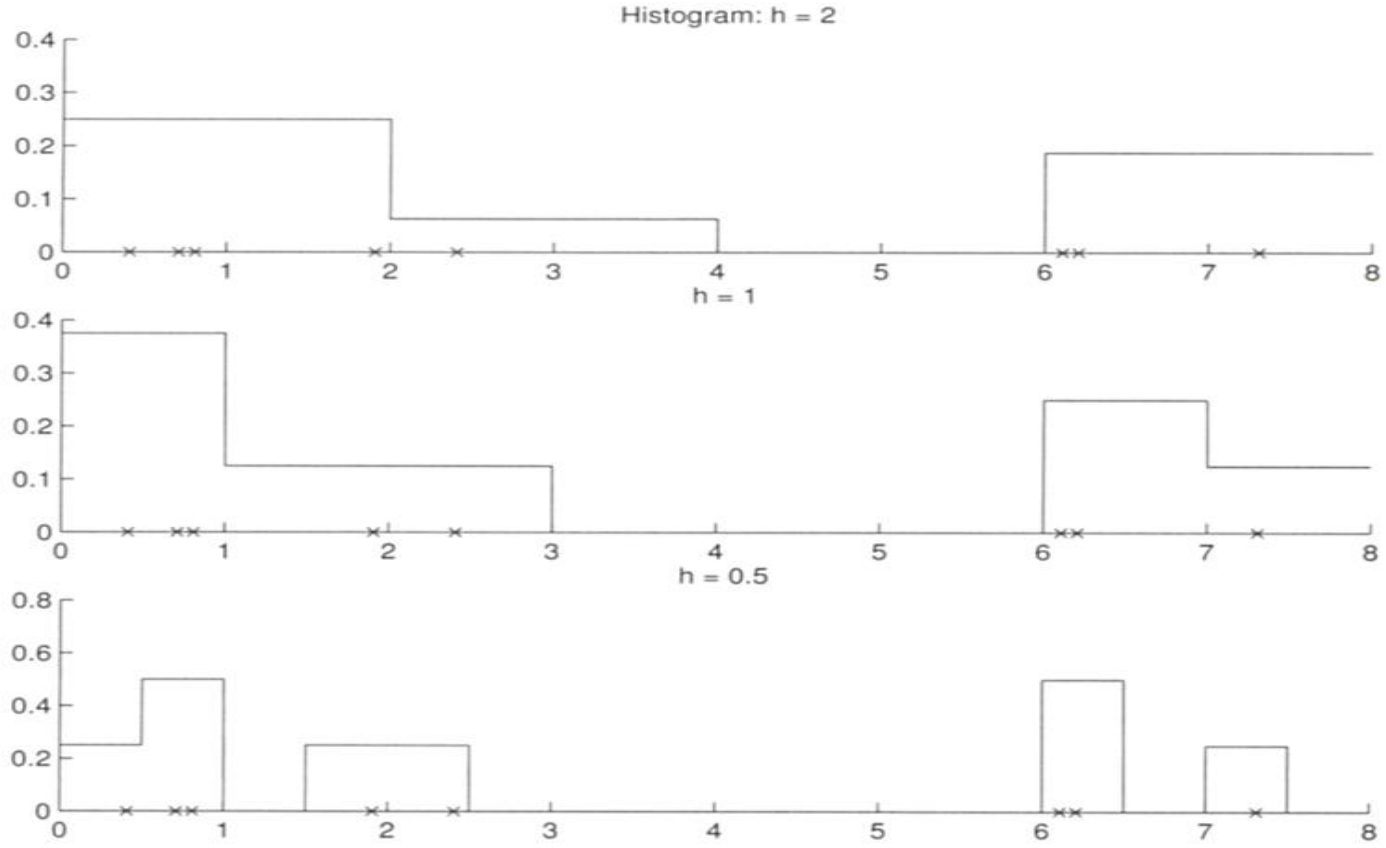



**Figure 8.1** Histograms for various bin lengths. '×' denote data points.

| h= 2 | |
|---|---|
| $m_i$ | $p(m_i)$ |
| 1 | 0.25 |
| 3 | 0.0625 |
| 5 | 0 |
| 7 | 0.1875 |

| h = 1 | |
|---|---|
| $m_i$ | $p(m_i)$ |
| 0.5 | 0.375 |
| 1.5 | 0.125 |
| 2.5 | 0.125 |
| 3.5 | 0 |
| 4.5 | 0 |
| 5.5 | 0 |
| 6.5 | 0.25 |
| 7.5 | 0.125 |

| h = 0.5 | |
|---|---|
| $m_i$ | $p(m_i)$ |
| 0.25 | 0.25 |
| 0.75 | 0.5 |
| 1.25 | 0 |
| 1.75 | 0.25 |
| 2.25 | 0.25 |
| 2.75 | 0 |
| 3.25 | 0 |
| 3.75 | 0 |
| 4.25 | 0 |
| 4.75 | 0 |
| 5.25 | 0 |
| 5.75 | 0 |
| 6.25 | 0.5 |
| 6.75 | 0 |
| 7.25 | 0.25 |
| 7.75 | 0 |

# How to Specify the Bin-Width?

- Izenman, A. J. (1991). Recent developments in nonparametric density estimation. *Journal of the American Statistical Association*, Volume 86, Number 413, Pages 205 – 224.
- Specify $h = 2(IQR)N^{-1/3}$ where $N$ is the number of observations and $IQR$ is the Interquartile Range. IQR is the Third Quartile minus the First Quartile.
- In practice, we may round *h* to some nice value.
  1. Round to the nearest integer
  2. Ceiling, i.e., round to the next larger integer (e.g., 4.567 to 5)
  3. Floor, i.e., truncate to the next smaller integer (e.g., 4.567 to 4)

# How to Specify the Bin-Width?

- The observations are 0.4, 0.6, 0.7, 1.9, 2.4, 6.1, 6.2, and 7.3 ($N$ = 8)
- In our previous example, Q3 = 6.15 and Q1 = 0.65. Thus IQR = 6.15 – 0.65 = 5.5. The suggested bin-width is h = 2 * 5.5 * $8^{-1/3}$ = 5.5.
- The *nice* bin-width is 5. Two bin-widths exceed the range, so reduce the bin-width to h = 4.
- The mid-points are 2 and 6.
- The density will have two rectangles, $\hat{p}(2)$ = 0.15625, $\hat{p}(6)$ = 0.09375.
- This may be a good choice as the observations seem to form two groups {0.4, 0.6, 0.7, 1.9, 2.4} and {6.1, 6.2, 7.3}

# Break-Out Room Team Exercise

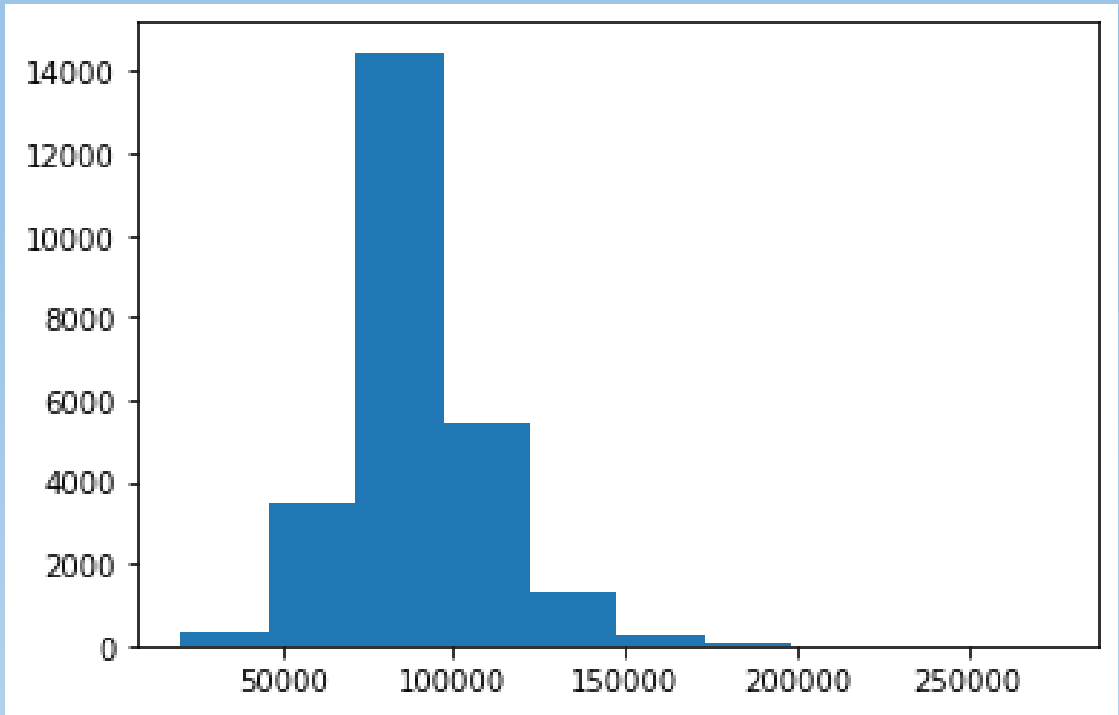
- Each student will join one of the Break-Out rooms
- Each room will download the Column_Y.csv file from Blackboard
- The CSV file contains one column, namely, Y
- Read the data into a Pandas dataframe

```
import matplotlib.pyplot as plt
import numpy
import pandas

inData = pandas.read_csv('C:\\IIT\\Machine Learning\\Data\\Column_Y.csv')
Y = inData['Y']
```

# Break-Out Room Team Exercise

```
print(Y.describe())

plt.hist(Y)
plt.show()
```

```
count      25297.000000
mean       89609.219514
std        20913.191206
min        20400.000000
25%        76266.000000
50%        90024.000000
75%        98496.000000
max       275004.000000
Name: Y, dtype: float64
```




Your team will spend 15 minutes in recommending a bin width such that the histogram can best represent the empirical density function.

# How to Specify the Bin-Width?

- Shimazaki H. and Shinomoto S. (2007). A method for selecting the bin size of a time histogram. *Neural Computation*, Volume 19, Issue 6, Pages 1503 – 1527
- *"From the observed data only, the method estimates a binwidth that minimizes expected L2 loss between the histogram and an unknown underlying density function. An assumption made here is merely that samples are drawn from the density independently each other."* – https://www.neuralengine.org/res/histogram.html

# How to Specify the Bin-Width?

**Shimazaki H. and Shinomoto S. (2007):**

1. Divide the data range into $m$ bins of width $d$.
2. Count the number of observations $n_i$ that enter the $i^{\text{th}}$ bin.
3. Calculate the mean and variance of the number of observations as $\bar{n} = \frac{1}{m}\sum_{i=1}^{m} n_i$ and $v = \frac{1}{m}\sum_{i=1}^{m}(n_i - \bar{n})^2$
4. Compute a formula $C(d) = (2\bar{n} - v)/d^2$
5. Repeat Steps 1 to 4 for different bin widths.
6. Determine the optimal bin width as the $d$ that minimizes $C(d)$.

# How to Specify the Bin-Width?

- The Shimazaki H. and Shinomoto S. (2007) method is straightforward except one issue: Divide the data range into $m$ bins of width $d$.

- We need a few sub-tasks to address the above issue:
  1. We will find the optimal width $d$ by the Grid Search, so what values of $d$ should we try? Too small width consumes more resources and time, too large width may not represent the data well.
  2. How should we determine the bin boundaries? For example, these two bins (0.1, 0.6] and (0.0, 0.5] both have width 0.5, but the number of observations in the bins (i.e., counts) may be different.

# How to Specify the Bins' Boundaries?

*Since the Mean often represents the Center of Gravity of the data, we use the Mean as our guiding post.*

- Let $\bar{y}$ denotes the Mean of the data
- Round $\bar{y}$ as an integral multiple of the bin width $d$.
  - The result is $d \times \text{round}(\bar{y}/d)$ where round() function rounds a value to the nearest integer
  - Suppose $\bar{y} = 12.34567$ and d = 0.2, then the result is 12.4
  - Suppose $\bar{y} = 1234.567$ and d = 600, then the result is 1200.
- Use this rounded mean as $b_0$ the boundary of a central bin

# How to Specify the Bins' Boundaries?

Use this rounded mean as $b_0$ the boundary of a central bin

- Add bins to the left of $b_0$ and the left boundaries of these bins are $b_0 - j_L \times d$ where $j_L = 1, 2, \ldots$
- Let $m_L$ denotes the number of bins on the left of $b_0$.
- $m_L$ is the smallest integer such that $b_0 - m_L \times d \leq y_{min}$ where $y_{min}$ is the minimum value of the data.

- Add bins to the right of $b_0$ and the right boundaries of these bins are $b_0 + j_R \times d$ where $j_R = 1, 2, \ldots$
- Let $m_R$ denotes the number of bins on the right of $b_0$.
- $m_R$ is the smallest integer such that $b_0 + m_R \times d \geq y_{max}$ where $y_{max}$ is the maximum value of the data.

# Python Codes for Shimazaki and Shinomoto (2007) Method

Week 1 Shimazaki Bin Width.py

```
def calcCD (Y, delta):
   maxY = numpy.max(Y)
   minY = numpy.min(Y)
   meanY = numpy.mean(Y)

   # Round the mean to integral multiples of delta
   middleY = delta * numpy.round(meanY / delta)

   # Determine the number of bins on both sides of the rounded mean
   nBinRight = numpy.ceil((maxY - middleY) / delta)
   nBinLeft = numpy.ceil((middleY - minY) / delta)
   lowY = middleY - nBinLeft * delta

   # Assign observations to bins starting from 0
   m = nBinLeft + nBinRight
   BIN_INDEX = 0;
   boundaryY = lowY
   for iBin in numpy.arange(m):
      boundaryY = boundaryY + delta
      BIN_INDEX = numpy.where(Y > boundaryY, iBin+1, BIN_INDEX)

   # Count the number of observations in each bins
   uBin, binFreq = numpy.unique(BIN_INDEX, return_counts = True)
```

```
   # Calculate the average frequency
   meanBinFreq = numpy.sum(binFreq) / m
   ssDevBinFreq = numpy.sum((Y - meanBinFreq)**2) / m
   CDelta = (2.0 * meanBinFreq - ssDevBinFreq) / (delta * delta)
   return(m, middleY, lowY, CDelta)
```

# Python Codes for Shimazaki and Shinomoto (2007) Method

```
result = pandas.DataFrame()
deltaList = [1000, 2000, 5000, 10000, 20000, 50000]

for d in deltaList:
   nBin, middleY, lowY, CDelta = calcCD(Y,d)
   highY = lowY + nBin * d
   result = result.append([[d, CDelta, lowY, middleY, highY, nBin]], ignore_index = True)

   binMid = lowY + 0.5 * d + numpy.arange(nBin) * d
   plt.hist(Y, bins = binMid, align='mid')
   plt.title('Delta = ' + str(d))
   plt.ylabel('Number of Observations')
   plt.grid(axis = 'y')
   plt.show()

result = result.rename(columns = {0:'Delta', 1:'CDelta', 2:'Low Y', 3:'Middle Y', 4:'High Y', 5:'N Bin'})
```

# Optimal Bin Width for Column Y

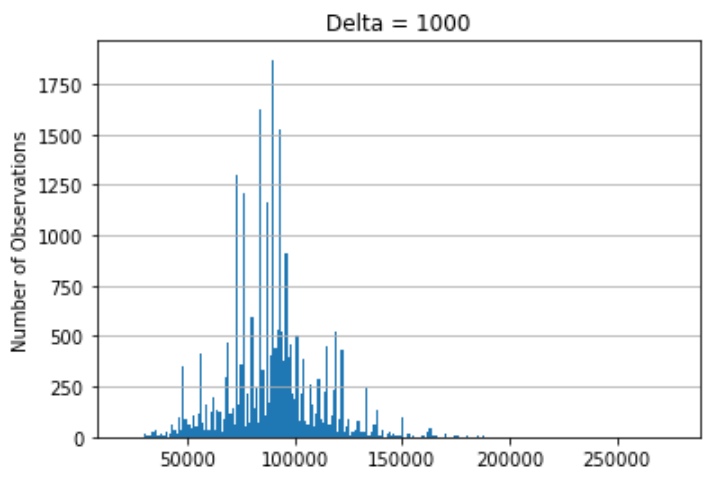
| Delta | C(Delta) | Low Y | Middle Y | High Y | N Bin |
|---|---|---|---|---|---|
| 1000 | -834945.0 | 20000 | 90000 | 276000 | 256 |
| 2000 | -416599.0 | 20000 | 90000 | 276000 | 128 |
| 5000 | -163072.0 | 20000 | 90000 | 280000 | 52 |
| 10000 | -80694.8 | 20000 | 90000 | 280000 | 26 |
| 20000 | -39512.9 | 20000 | 80000 | 280000 | 13 |
| 50000 | -13035.2 | 0 | 100000 | 300000 | 6 |

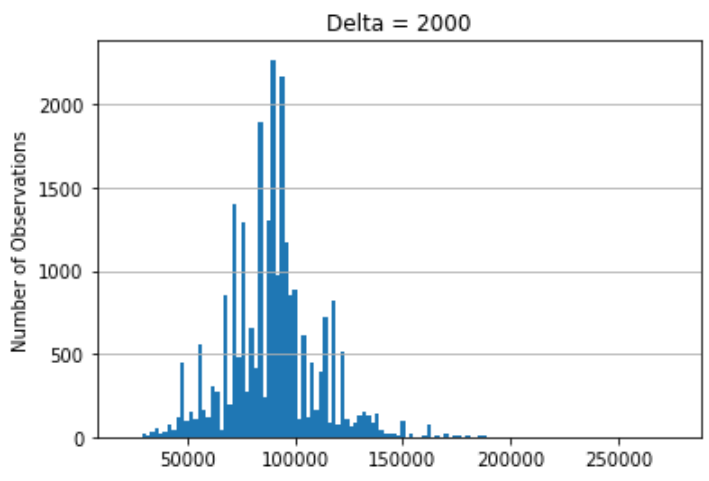
Finding
C(Delta) is lowest when Delta = 1000.

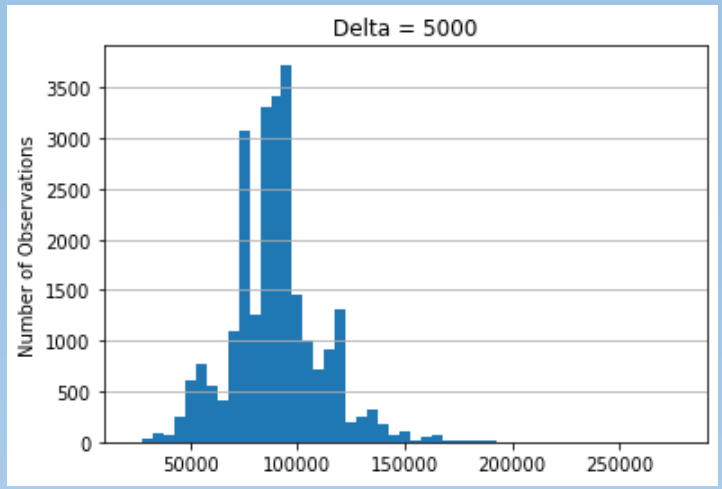
Concern
Is 256 bins too many when Delta = 1000?

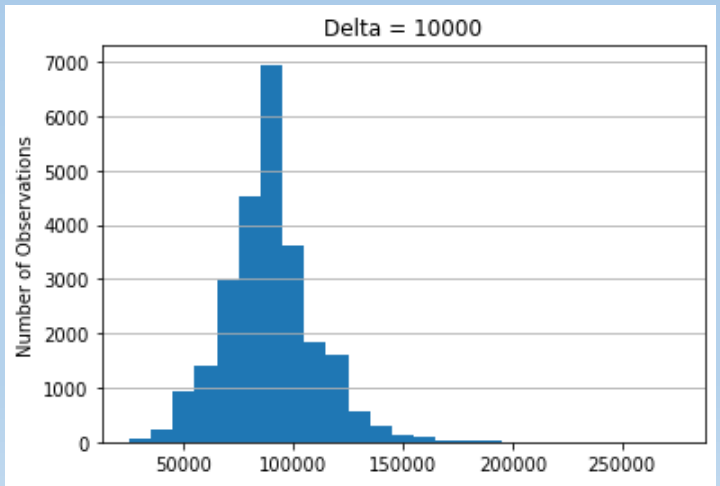
Consideration
Should we set maximum and minimum of bins?

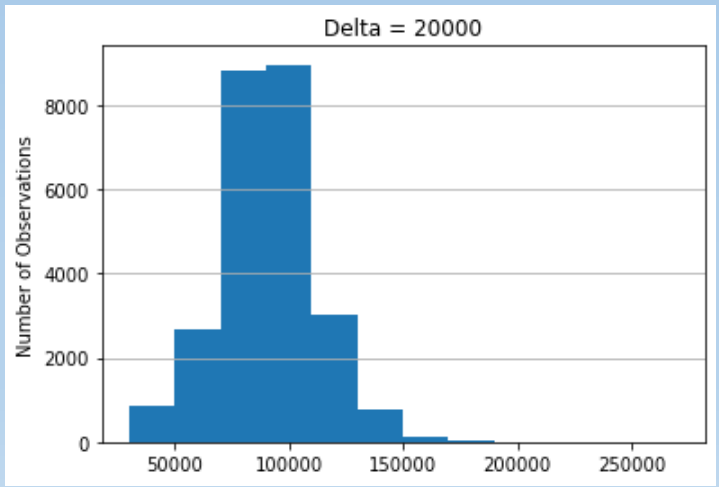
# Optimal Bin Width for Column Y

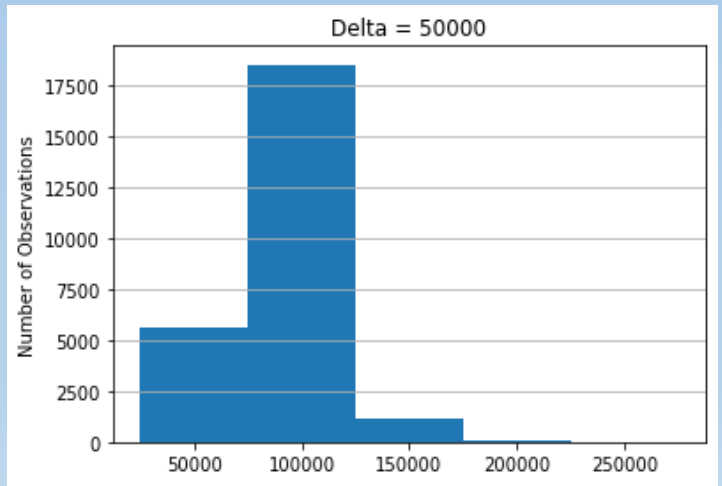


# An Afterthought …

We like the idea of making the $d$ values the multiples of 1, 2, and 5


Can we *examine* the data to determine the candidate list of $d$ values instead?

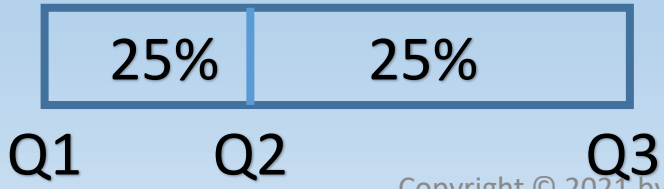
Would you like to suggest a strategy for us?

# Detect Statistical Outliers

- Statistical outliers are not necessarily incorrect values.
- Statistical outliers mean the values are significantly separated from the main body of data values.
- So, what is the main body of data values?
- What should the main body of data values include?
- What about a box that represents the central 50% of data values?

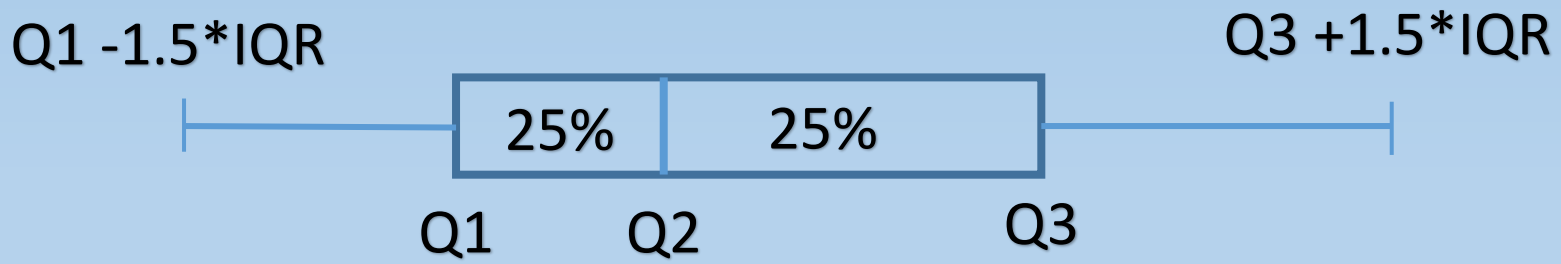
# A Procedure for Constructing a Box-Plot

1. Calculate the five-number summary which is the minimum, the first quartile, the median, the third quartile, and the maximum.
   a) The first quartile (Q1) is the $25^{th}$ percentile
   b) The median (Q2) is the $50^{th}$ percentile
   c) The third quartile (Q3) is the $75^{th}$ percentile
2. Draw a box, either horizontally or vertically
   a) The centerline represents the median
   b) The left or the lower box boundary represents the first quartile.
   c) The right or the upper box boundary represents the third quartile



# A Procedure for Constructing a Box-Plot

1. Calculate the Interquartile Range IQR = Q3 – Q1.
2. Draw two whiskers which extend from the ends of the box
   a) The left or the lower whisker extends to the **larger** of Q1 – 1.5 * IQR and the minimum.
   b) The right or the upper whisker extends to the **smaller** of Q3 + 1.5 * IQR and the maximum.



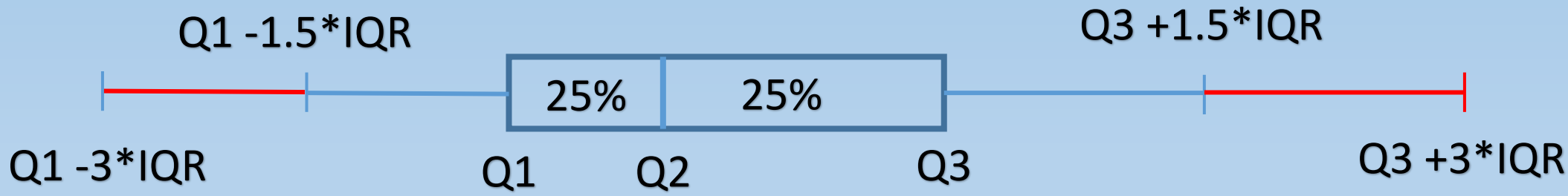
# Why 1.5 Times of the IQR?

For a standard normal distribution (i.e., mean = 0 and standard deviation = 1)

- Q1 = -0.67449, Q2 = 0, and Q3 = 0.67449.
- Therefore IQR = Q3 – Q1 = 1.34898 and 1.5*IQR = 2.02347.
- The lower whisker is Q1 – 1.5*IQR = -2.69796 and the upper whisker is Q3 + 1.5*IQR = 2.69796.
- Common wisdom says any values more than three standard deviations from the mean are outliers.
- If the multiplier is 2, then the upper whisker is 3.37245 > 3.
- If the multiplier is 1, then the upper whisker is 2.02347 << 3.
- The 1.5 multiplier is sort of the compromise.

# Detect Outliers Using a Box-Plot

1. Observations whose values lie outside the whiskers Q1 – 1.5*IQR and Q3 + 1.5*IQR are considered **outliers**.
2. Actually, there are another set of whiskers which are Q1 – 3*IQR and Q3 + 3*IQR. Observations whose values lie outside these whiskers are considered **extreme values**.



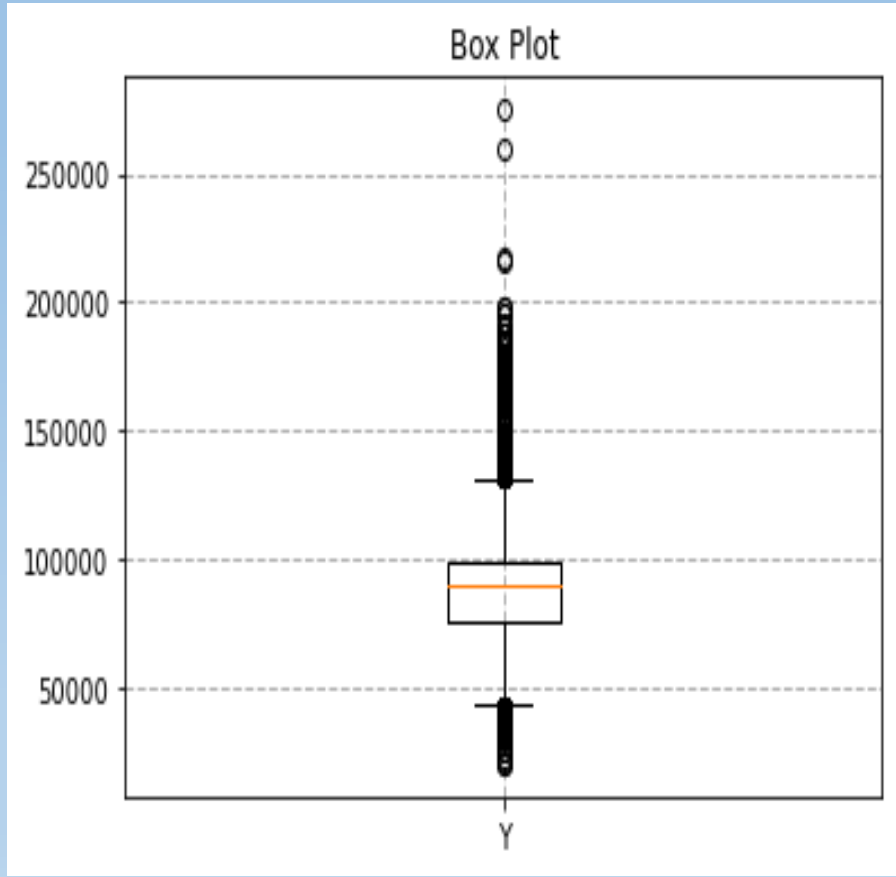
# Statistical Outliers in Column Y

- Q1 = 76,266, Q2 = 90,024, Q3 = 98,496
- IQR = Q3 – Q1 = 22,230, 1.5 × IQR = 33,345
- Q1 – 1.5 × IQR = 49,921, Q1 – 3 × IQR = -23,769
- Q3 + 1.5 × IQR = 131,841, Q3 + 3 × IQR = 198,531

# Statistical Outliers in Column Y

- Q1 = 76,266, Q2 = 90,024, Q3 = 98,496
- IQR = Q3 – Q1 = 22,230, 1.5 × IQR = 33,345
- Q1 – 1.5 × IQR = 49,921, Q1 – 3 × IQR = -23,769
- Q3 + 1.5 × IQR = 131,841, Q3 + 3 × IQR = 198,531

```
fig1, ax1 = plt.subplots()
ax1.set_title('Box Plot')
ax1.boxplot(Y, labels = ['Y'])
ax1.grid(linestyle = '--', linewidth = 1)
plt.show()
```



# What is Column Y?


- Source: https://data.cityofchicago.org/Administration-Finance/Current-Employee-Names-Salaries-and-Position-Title/xzkq-xp2w
- Data: Current Employee Names, Salaries, and Position Titles
- Column Y: **Annual Salary** of the salaried employees
- Four Statistical Outliers: Annual Salary > 198,531

| Name | Job Titles | Department | Full or Part-Time | Salary or Hourly | Annual Salary |
|---|---|---|---|---|---|
| RHEE, JAMIE L | COMMISSIONER OF AVIATION | AVIATION | F | Salary | 275,004.00 |
| BROWN, DAVID O | SUPERINTENDENT OF POLICE | POLICE | F | Salary | 260,004.00 |
| FORD II, RICHARD C | FIRE COMMISSIONER | FIRE | F | Salary | 217,728.00 |
| LIGHTFOOT, LORI E | MAYOR | MAYOR'S OFFICE | F | Salary | 216,210.00 |

# Preparation For Week 2

- Get the Machine Learning book
- Get your Python environment ready
  - Run the Week 1 MyFirstDecisionTree.py
- Read Chapter 1 and Chapter 2 of the Machine Learning book
- Ask yourself:
  - Why do you need to read this chapter?
  - What problems can I apply the algorithms in this chapter?