Introduction to Data Science for Fintech

Cognitir (formerly Import Classes) was founded in August 2015 by David Haber and Neal Kumar. The company provides data science and web development courses designed for business and non-technical professionals. Given David's technical expertise and Neal's vast business & training experience both founders have combined their backgrounds to create world-class, practical training courses.

We always strive to create best-in-class products, so feel free to email us any comments on this course and/or new product ideas. Our email is **feedback@cognitir.com**.

We sincerely hope that you enjoy taking the course as much as we enjoyed creating it!

 *@cognitir*

 *cognitir*

 *cognitir*

Terms and Conditions

The notes and relevant analyses are proprietary to Cognitir LLC. Distributing, copying, sharing, duplicating, and/or altering this file in any way is prohibited without the expressed written consent of Cognitir LLC.

The datasets used in the analyses are public information and can be found in public websites including:

- UC Irvine's Center for Machine Learning and Intelligent Systems
- University of Southern California's Marshall School of Business
- Python's scikit-learn
- Introduction to Statistical Learning, James, Witten, Hastie, and Tibshirani
- Python for Data Analysis, McKinney

- Author(s): David Haber and Neal Kumar
- Version: 1.0
- www.cognitir.com

This Course

Designed for non-technical and business professionals:

IC provides a balanced framework of technical skills and business problem solving to enhance decision making.

Course Topics:

- Data Science / Machine Learning
- Applications
- Programming
- Statistics

Course Goal:

- Learn the basics of data science, machine learning, and Python programming to perform effective data analysis.

Course Outline

Morning Session (09:00 – 13:00)

- Introduction
- Predicting Credit Defaults: Classification in Python for Fintech
- The Data Science Process

Lunch (13:00 – 14:00)

Afternoon Session (14:00 – 16:45)

- Identifying Similar Stocks: Clustering in Python for Fintech
- Big Data: What is Big Data? What is not Big Data?

Wrap-up (16:45 – 17:00)

- Q&A
- Next Steps

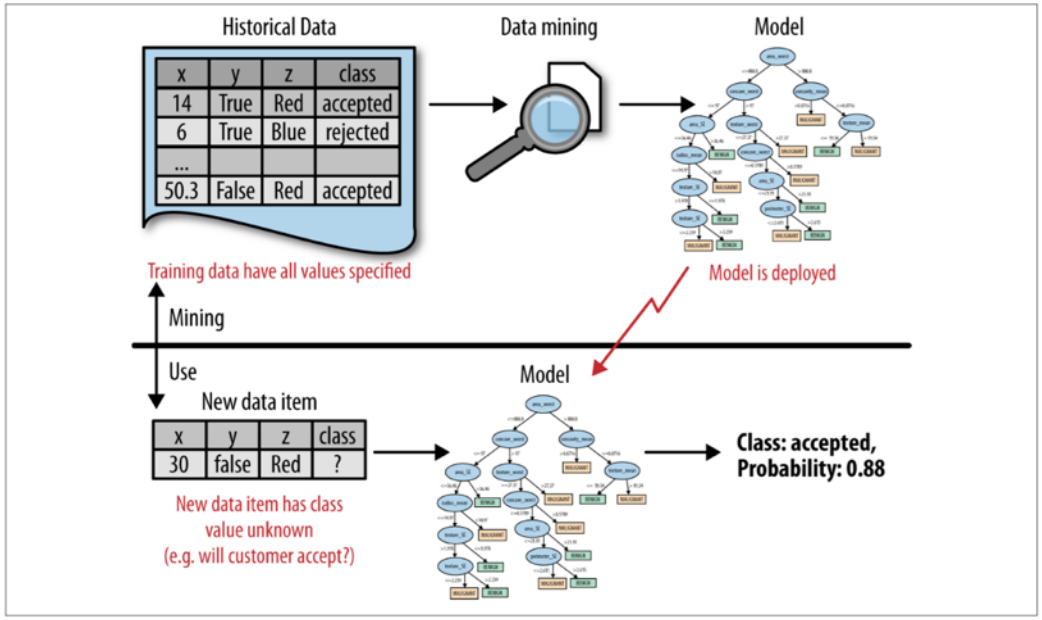
We create 2.5 quintillion bytes of data every day —
so much that 90% of the data in the world today has been created
in the last two years alone.

IBM

INTRODUCTION

What is Data Science?

- **Data Science** describes the art of extracting knowledge from data.
- It combines concepts from statistics, machine learning, and computer science.

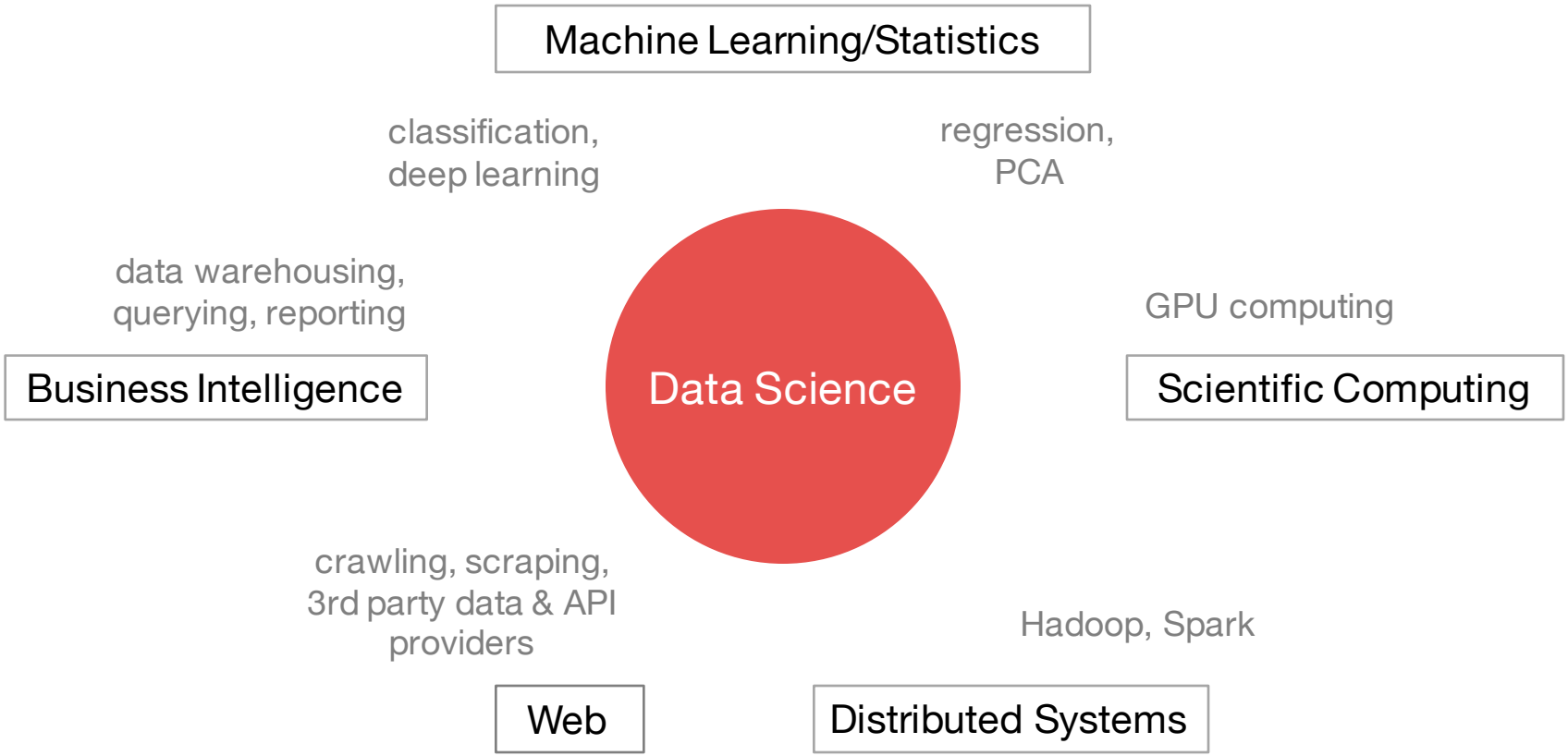


Source: Data Science for Business, Provost & Fawcett

Why is Data Science Important?

- Data Science uses the tremendous amounts of data available for **improved decision-making**.
- **Data-driven decision-making** (DDD) refers to the practice of basing decisions on the analysis of data, rather than purely on intuition.
  - *A company could select new products based on intuition and experience in the industry.*
  - *Using DDD, companies could base their selection on the analysis of data regarding how consumers react to different products.*

Data Science is **interdisciplinary**.



Why is Data Science Important? (cont'd)

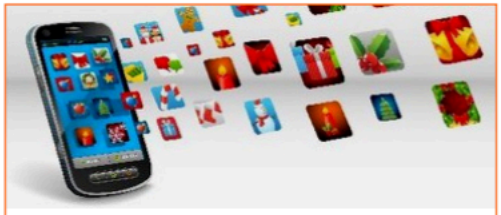
Big data can generate significant financial value across sectors

US health care

- $300 billion value per year
- ~0.7 percent annual productivity growth

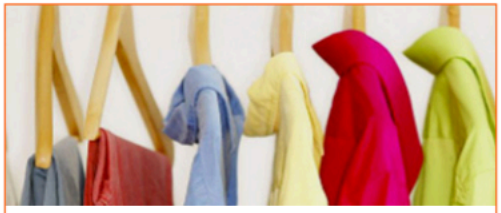
Europe public sector administration

- €250 billion value per year
- ~0.5 percent annual productivity growth



Global personal location data

- $100 billion+ revenue for service providers
- Up to $700 billion value to end users



US retail

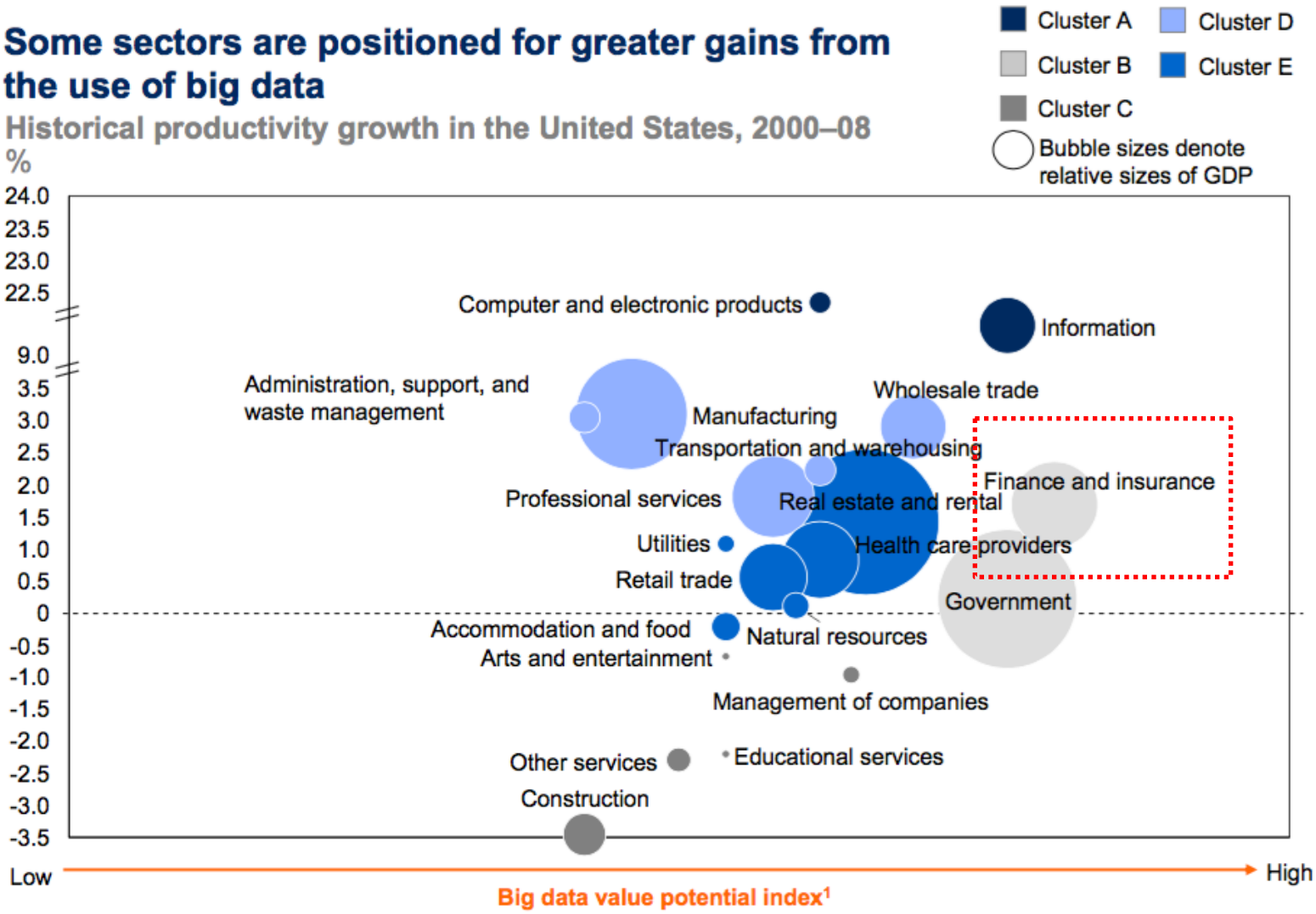
- 60+% increase in net margin possible
- 0.5–1.0 percent annual productivity growth

Manufacturing

- Up to 50 percent decrease in product development, assembly costs
- Up to 7 percent reduction in working capital

Source: McKinsey Global Institute

Why is Data Science Important? (cont'd)



Source: McKinsey Global Institute

Why is Data Science Important? (cont'd)

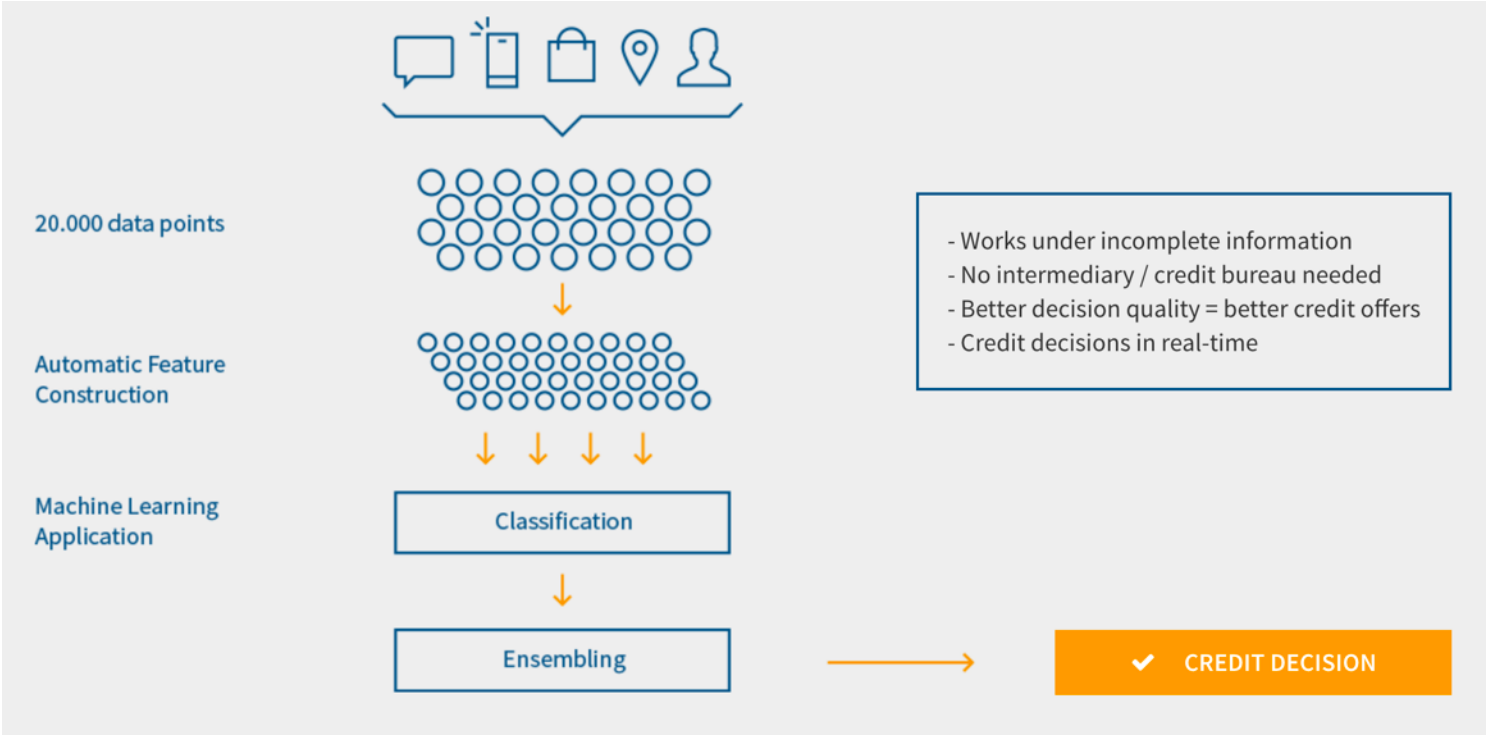
- Study by Brynjolfsson, Hitt, & Kim (2011) shows:
  - *Firms that adopt DDD have been able to increase output and productivity by 5-6%.*
  - *DDD yields higher returns on assets, equity, asset utilization, and market value.*
- Tambe (2012):
  - *Big data technologies are associated with significant productivity growth.*
- McKinsey Global Institute (2011):
  - *By 2018, the U.S. alone may face a 50 percent to 60 percent gap between supply and requisite demand of deep analytic talent.*

Example: Kreditech

Use of credit decision technology to provide access to credit for people with little or no credit history.

Kreditech

"Banking the Underbanked"



Source: kreditech.com

More Examples

Peer-to-peer money transfer service to avoid costly currency conversion and transfers crossing borders.

TransferWise

"The clever new way to beat bank fees"

Deposit, withdraw, transfer money, and pay for goods and services easily with a mobile device.

M-PESA

"9 million registered users transfer on average $320 million per month"

Management of personalized, globally diversified investment portfolio.

wealthfront

"The most tax-efficient, low-cost, hassle-free way to invest"

Credit Scoring

- Credit risk evaluation
- Analyzing customer data from multiple available sources

Marketing

- Improving customer acquisition through analysis of digital marketing channels
- Creating complete customer behavior/preference profiles
- Personalized, contextual offerings

Risk Management

- Enhanced fraud and authentication solutions
- Eradicating vulnerable digital access points
- Device identification, biometrics, behavior analysis

Investment Management

- Automated advisory solutions ("Robo-Advisors")
- Identifying anomalies
- Using social media data to spot trends

Data Science Questions

- *How will markets react to new information as it comes out?*
- *Where is market volatility heading in the near future?*
- *How can we predict key economic indicators before they are even released to the public?*
- *Can we find similarities/patterns in stock data?*
- *Among all our customers, which are likely to respond to a given offer?*
- *How much will a given customer use the service?*

Data Science Questions (cont'd)

- *How much can I sell property in New York City for / how much are people willing to pay for property in a given area?*
- *Do our customers form natural groups or segments?*
- *Which items are commonly purchased together?*
- *If we target customers x with new financial products y, how much will our sales likely increase?*
- *Which companies are similar to our best business customers so that we can focus our sales resources on the best opportunities?*

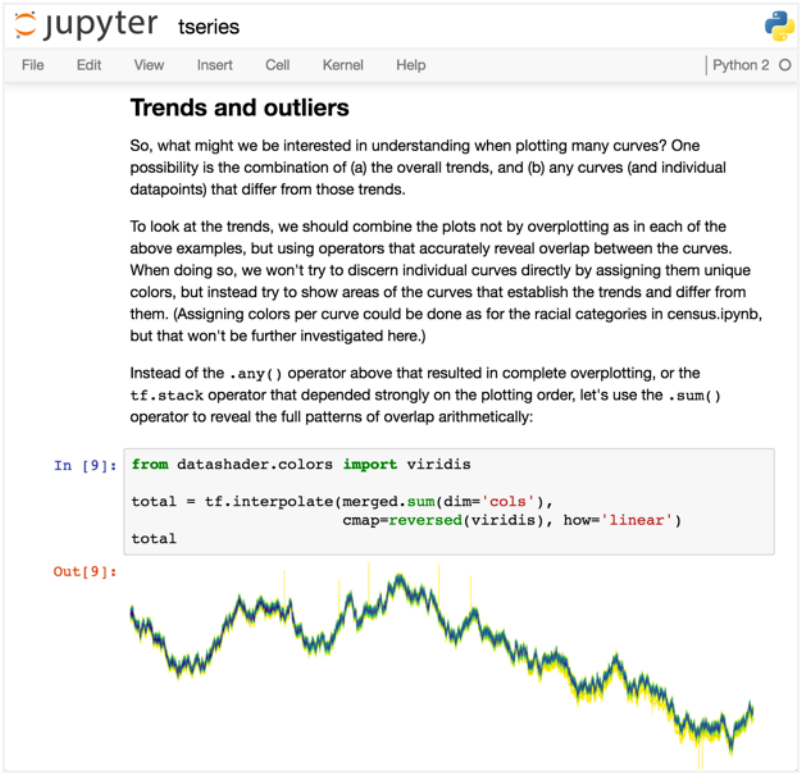
Data Science Questions (cont'd)

- *What are the characteristics of businesses that are considered our best customers?*
- *Did our advertisement really influence consumers to purchase?*
- *Can we find a subset of the data that contains enough of the important information of the larger set?*

- In this course, we will learn **data-driven approaches** to answering these questions.

Notebooks: A Powerful Tool

- **Jupyter Notebook** allows effective data analytics from your **web browser.**
- We can create and share documents that contain live **code**, **equations**, **visualizations**, and **explanatory text.**
- Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning, and much more.



Getting Up To Speed

Exercises:

1. In computer programming, what is a **variable** and how do you create it in Python?
2. Write Python code that **iterates** over the following list and prints the employee names:

```
employees = ["Hannah", "Max", "Daniel", "Christina"]
```

3. How do you print the **length** of the list in Question 3?
4. What is a Python **dict**?
5. What is a Python **function**? How do you define it?
6. What are Python **modules** and why are they useful?

You can have data without information, but you cannot have information without data.

Daniel Keys Moran

CLASSIFICATION

OVERVIEW

- The intuition behind Bayes' Theorem
- Classification using Naïve Bayes
- Techniques to evaluate classification performance
  - Accuracy
  - Confusion Matrix
  - Expected Value and Profit Curves
  - Receiver Operating Characteristic (ROC) and Area Under The Curve (AUC)
- Selecting informative attributes

Classification

- A **regression** model predicts **quantitative/continuous** target variables.
  - We can predict sales units as a function of budget spent on advertisement.
- When the target variables in question are **qualitative**, **categorical,** or **discrete**, we use **classification** methods.
- In this section, we will **learn** how to **predict** whether a customer will default on his credit card payments.

	Continuous Data	**Categorical Data**
Supervised	Regression	Classification
Unsupervised	Dimensionality Reduction	Clustering

Classification (cont'd)

- Questions we can answer with classification methods:
  - *Among all our customers, which are likely to respond to a given offer?*
  - *Which of our customers are likely to default on their credit card payments?*
  - *Which transactions are likely to be fraud?*
  - *Which of our applicants are likely to perform well on the job?*

- **Naïve Bayes** is a simple, yet powerful, classification technique.
- It uses probability estimation to classify an example, i.e., *what is the probability that our observation falls in category A versus category B*?

- There are other popular classification techniques including **Decision Trees**, **Support Vector Machines** and **Neural Networks**.

1. **Tossing a coin:**

   Probability of either side is **1/2**.

   We can write

   p(“Heads”) = 1/2

   p(“Tails”) = 1/2

   } Outcomes sum to 1.0

2. **Rolling a die:**

   Probability of a particular number is **1/6**.

   p(“One”) = 1/6

   …

   p(“Six”) = 1/6

   } Outcomes sum to 1.0

Joint Probability

Let's say we have two events, A and B. If we know p(A) and p(B), what is the probability p(A&B) that both A and B will occur?

Example:

- We roll a die twice. The first time we roll "One" and the second time we roll "Five".

  p("One") = **1/6** and p("Five") = **1/6**.

- The probability that we roll this combination is

  p("One" & "Five") = **1/6 * 1/6** = **1/36**.

p(A & B) is called **joint probability**.

Joint Probability (cont'd)

Two events are **independent** if knowing one event **does not** give you information on the probability of the other.

- If we had first rolled "Three", the probability of rolling "Five" would still be **1/6**.
- Knowing the value of the first roll tells us **nothing** about the value of the second.
- In this case, both events are independent and we can write **p(A & B) = p(A) * p(B)**.

Joint Probability (cont'd)

- Consider another example with two events:

  A = "it's cloudy today"

  B = "it will rain today"

- Clearly A and B are **not independent** events: knowing that it is cloudy today will make it more likely that it rains.
- The general formula for combining probabilities that **takes care of dependencies** between events is:

  P(A & B) = p(A) * p(B | A)

*In simple English: the **probability of A and B** equals the **probability of A** times the **probability of B given A***

Bayes' Theorem

Consider the following scenario:

- 1% of women have breast cancer (99% do not have breast cancer)
- 80% of mammograms detect breast cancer when it is there (20% miss it)
- 9.6% of mammograms detect breast cancer when it is not there (90.4% correctly return a negative result)

How **accurate** is this test? We are interested in finding out ***how likely*** *it is that you have breast cancer if the test indicates a positive result*.

Any guesses?

Bayes' Theorem (cont'd)

Let's summarize our probabilities:

p("Cancer") = 1%
p("No Cancer") = 99%

p("Positive" | "Cancer") = 80%
p("Negative" | "Cancer") = 20%

p("Positive" | "No Cancer") = 9.6%
p("Negative | "No Cancer") = 90.4%

We want to compute: **p("Cancer" | "Positive")**

Bayes' Theorem (cont'd)

- How do we combine probabilities to get **p("Cancer" | "Positive")?**
- **Bayes' Theorem** states:

$$p(A \mid B) = \frac{p(A)p(B \mid A)}{p(B)}$$

Thomas Bayes
(a.k.a. The Dude)

- For our example, we can write:

$$p("Cancer" \mid "Positive") = \frac{p("Cancer")p("Positive" \mid "Cancer")}{p("Positive")}$$

- We have all probabilities given except **p("Positive").**

Bayes' Theorem (cont'd)

	Cancer (1%)	No Cancer (99%)
Positive (?)	p("Positive" | "Cancer") = 80%	p("Positive" | "No Cancer") = 9.6%
Negative (?)	p("Negative" | "Cancer") = 20%	p("Negative" | "No Cancer") = 90.4%

	Cancer (1%)	No Cancer (99%)
Positive (10.3%)	p("Positive" & "Cancer") = 0.8%	p("Positive" & "No Cancer") = 9.5%
Negative (89.7%)	p("Negative" & "Cancer") = 0.2%	p("Negative" & "No Cancer") = 89.5%

We can now write:

$$p(\text{"}Cancer\text{"} \mid \text{"}Positive\text{"}) = \frac{0.01 * 0.8}{0.103} = 0.078$$

If your test gives a positive result, the probability of having cancer is only **around 8%**.

Bayes' Theorem For Classification

- We can use Bayes' Theorem for **classification**:

$$p(C = c|E) = \frac{p(C = c)p(E|C = c)}{p(E)}$$

- *P(C=c | E)* is the **posterior probability**. This is what we wish to infer.
- *p(C=c)* is the **prior probability**, i.e., the probability we would assign to the class before we see any evidence.
- *p(E | C=c)* is the **likelihood** of seeing evidence when the class C=c.
- *p(E)* is the **likelihood of the evidence**, i.e., how common is the feature representation E present in our data?

Naïve Bayes Equation

- Assuming that the attributes (features) are *conditionally independent*, we can rewrite **Bayes' Theorem for classification**:

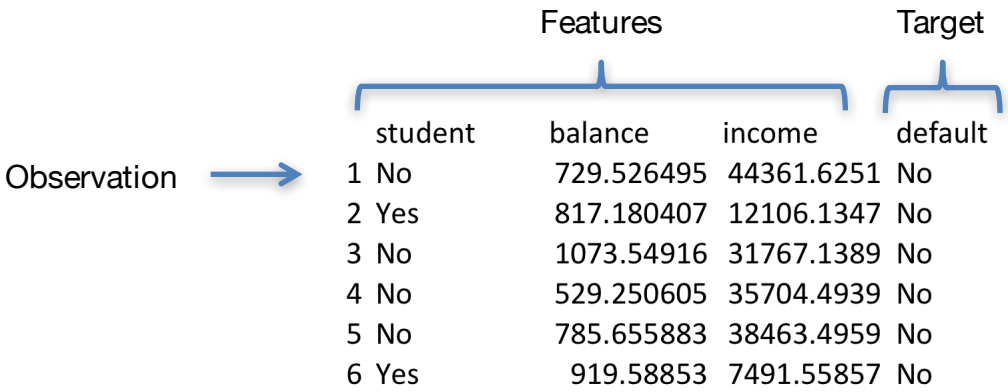
$$p(C = c|E) = \frac{p(e_1|c)p(e_2|c)\cdots p(e_k|c)p(c)}{p(E)}$$

- We call this the **Naïve Bayes (NB) classifier**.
- It classifies a test example by estimating the probability that the example belongs to each class and reports the class with the **highest probability**:

$$\underset{i=\{1,\ldots,N\}}{\operatorname{argmax}}\, p(C = c_i \mid E)$$

Naïve Bayes Classification in Action

- In this section, we will see how to use Naïve Bayes to predict if a customer will **default on his credit card payment**.
- The data:



- Input Variables (Features):
  - **Student**: Indicates if customer is a student (Yes / No)
  - **Balance**: Monthly card balance
  - **Income**: Individual's income
- Output Variable (Target):
  - **Default**: Customer defaults on credit card payment (Yes / No)

Naïve Bayes Classification in Action (cont'd)

- We have continuous data so we assume that the features follow a **Gaussian distribution.**
- Gaussian Naïve Bayes classification consists of the following steps:

1. Load and format data
2. Train classifier
   a) Split dataset in classes.
   b) Calculate class prior probability.
   c) For each attribute, calculate class conditional mean and standard deviation.
3. Predict values: $$Y_{pred} = \underset{i=\{1,\ldots,N\}}{\operatorname{argmax}}\, p(y_k) * \prod_i N(x_i, \mu, \sigma)$$
4. Evaluate accuracy

- Let's see how this is built in Python!

1. Load and format data

We use *pandas* to load the dataset:

```
from pandas import read_csv

df = read_csv('data/Default.csv', index_col=0)
```

Next, we replace strings with numerical values:

```
df = df.replace("Yes", 1)
df = df.replace("No", 0)
```

To assess accuracy, we split the dataset into training and test set:

```
from sklearn.cross_validation import train_test_split

# Create training and test sets
df_train, df_test = train_test_split(df, test_size=0.1)
```

2. Train Classifier

We can train our classification model (Naïve Bayes classifier) on the training set with *scikit-learn*:

```
from sklearn.naive_bayes import GaussianNB
import numpy as np

# Select feature and target columns
feature_cols = ["student", "balance", "income"]

# Separate training dataset into features (X) and target (Y)
X_train = df_train[feature_cols]
Y_train = df_train.default

# Train classifier
gnb = GaussianNB()
clf = gnb.fit(X_train, Y_train)
```

3. Predict Values

Our trained classifier can be used to predict target values for the observations in the test set:

```
# Separate test dataset into features (X) and target (Y)
X_test = df_test[feature_cols]
Y_test = df_test.default

# Predict values
Y_pred = clf.predict(X_test)
```

4. Evaluate Accuracy

We compare the predicted target values (*Y_pred*) against the actual target values (*Y_test*):

```
print "Number of mislabeled points out of a total %d points : %d"
        % (X_test.shape[0], (Y_test != Y_pred).sum())
```

Evaluating Classifiers

- There are various metrics to find out how well our classifier performs:
  - **Accuracy**
    - Plain accuracy = **number of correct classifications / total sample size**
  - **Confusion Matrix**

Predicted Class

True Class

	Negative	**Positive**
Negative	True Negative	False Positive
Positive	False Negative	True Positive

- We can compute the confusion matrix in Python:

```
from sklearn.metrics import confusion_matrix

print confusion_matrix(Y_test, Y_pred)
```

Expected Value (EV)

- EV is the **weighted average** of the values of **different possible outcomes** o_1, o_2, o_3, …:

$$EV = p(o_1) \cdot v(o_1) + p(o_2) \cdot v(o_2) + p(o_3) \cdot v(o_3) \ldots$$

- EV provides a framework that **aids decision-making while optimizing profits**.

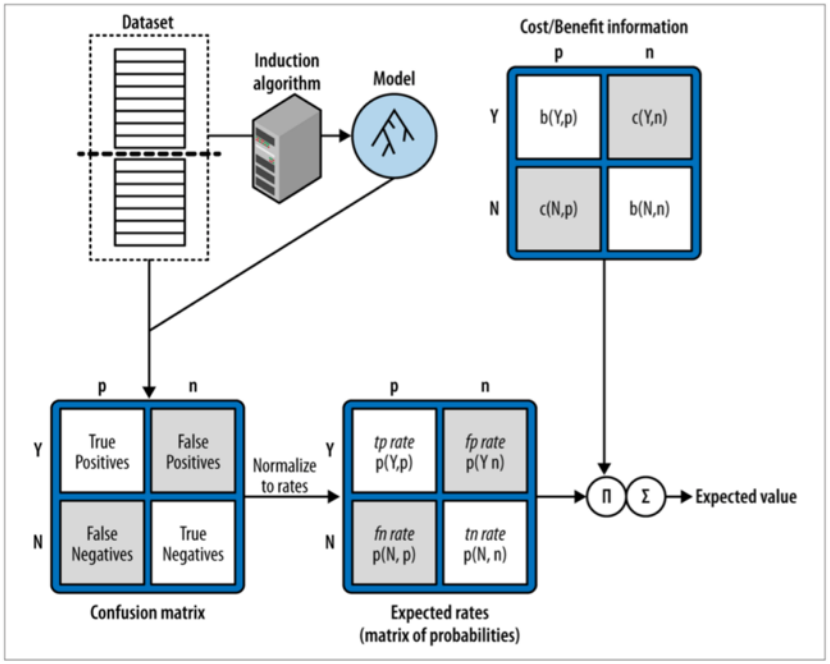
Example:
Should we target consumer **x**? A responding consumer yields a value (profit) of $\mathbf{v_R = \$99}$. If the consumer does not respond, we have a cost of $\mathbf{v_{NR} = \$1}$.

$$p_R(x) \cdot \$99 - [1 - p_R(x)] \cdot \$1 > 0$$
$$p_R(x) \cdot \$99 > [1 - p_R(x)] \cdot \$1$$
$$p_R(x) > 0.01$$

In this example, we should target the consumer as long as the estimated probability of responding is greater than 1%.

Expected Value to Evaluate Classifiers/Models

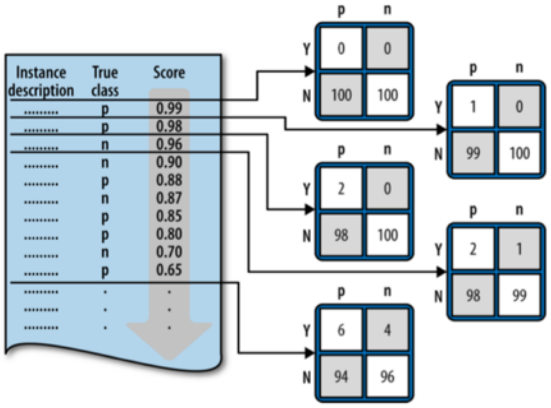
- Expected Value can be used to compare classification models. For example, does our data-driven model perform better than the hand-crafted model suggested by the marketing group?

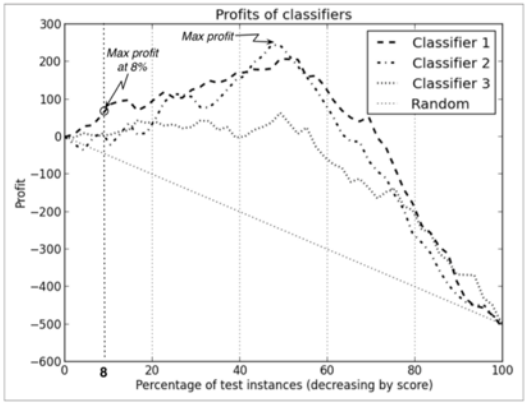


Source: Data Science for Business, Provost & Fawcett

Profit Curves

- All instances are **ordered from highest to lowest score**.
  - For example, consumers are ordered from highest to lowest probability of accepting an offer based on some model.
- We **apply successive thresholds** to divide the list of instances into sets of predicted **positive** and **negative** instances.
- As we **move the threshold down**, we get additional instances predicted as being positive rather than negative, i.e., **each threshold has a corresponding confusion matrix**.
- For each confusion matrix, we **record the expected profit** which gives us a **profit curve**.





Source: Data Science for Business, Provost & Fawcett

ROC Curve

– Receiver Operating Characteristic (ROC) is a plot of a binary (two-class) classifier with **false positive rate** on the x axis against **true positive rate** on the y axis.

- **False Positive Rate**: $\frac{FP}{N} = \frac{FP}{FP + TN}$

  (*false alarm rate*, what percentage of actual negative examples does the classifier get wrong)

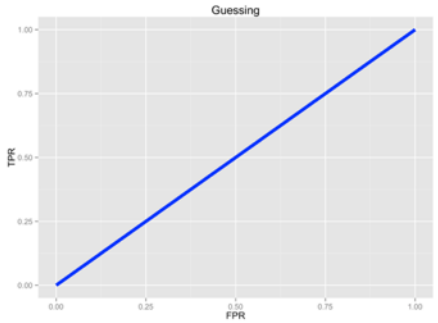
- **True Positive Rate**: $\frac{TP}{P} = \frac{TP}{TP + FN}$

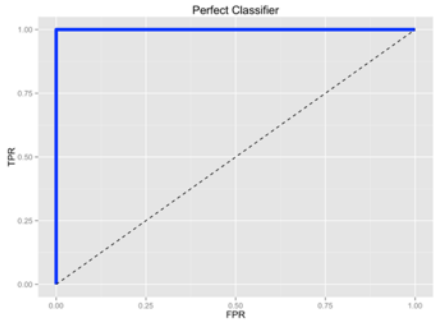
  (*hit rate, sensitivity or recall*, what percentage of the actual positives does the classifier get right)

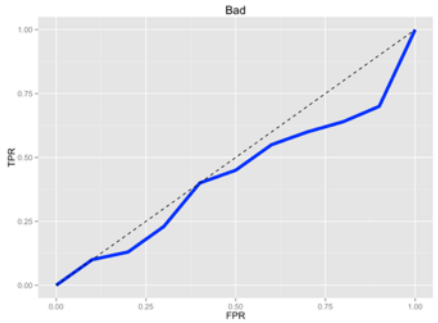
Evaluating Classifiers (cont'd)

- A ROC plot depicts **relative trade-offs that a classifier makes** between benefits (true positives) and costs (false positives).
- ROC graphs **decouple classifier performance** from the conditions under which the classifiers will be used.

"The more "up and left", the better."







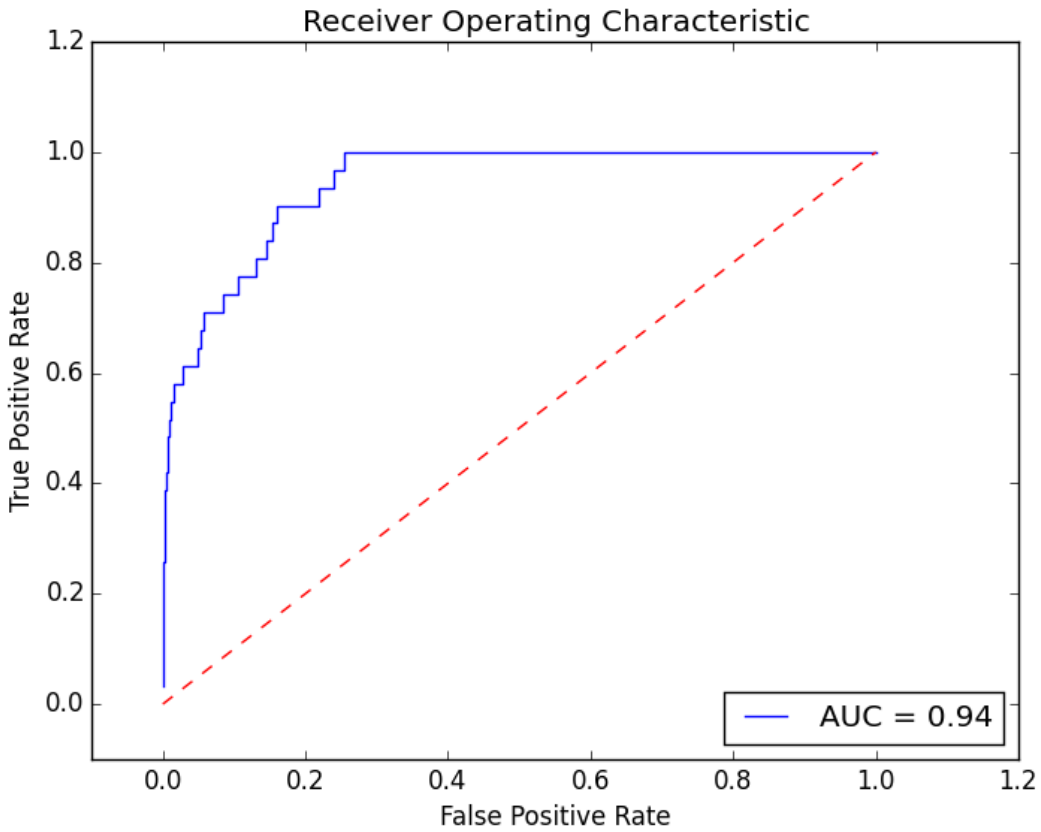
Evaluating Classifiers (cont'd)

- To better understand the performance of our classifier, we can plot the ROC curve in Python:

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Get sample scores for ROC curve
Y_score = clf.predict_proba(X_test)

# Compute ROC curve
false_positive_rate, true_positive_rate, thresholds =
                            roc_curve(Y_test, Y_score[:, 1], pos_label=1)
roc_auc = auc(false_positive_rate, true_positive_rate)

# Plot ROC curve
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',
                            label='AUC = %0.2f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Evaluating Classifiers (cont'd)

- The below plot shows the **ROC curve** for the classifier we trained on the **loan defaulting dataset**:



Area under the ROC curve (AUC):

$$A_{ROC} = P(random\ positive\ example > random\ negative\ example)$$

- This is equal to the **probability** that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.
- It ranges from 0 to 1:
  - **AUC = 1** -> *perfect*
  - **AUC = 0** -> *bad*
- The AUC is useful when a **single number** is needed to summarize performance.

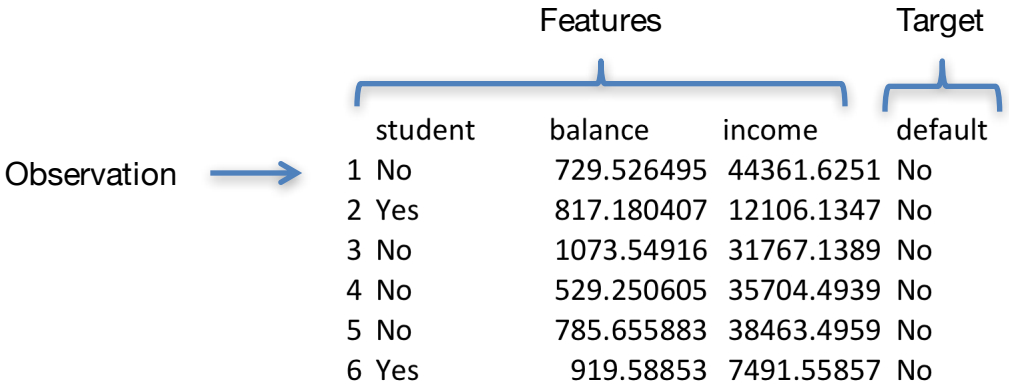
Exercise

Additional Exercises:

1. Create a new Notebook in the Notebook application.
2. Use the data from **"Diabetes.csv"** to build a classifier that predicts if patients have diabetes (i.e., predicts the "Class" column). You can reference the code from **"Classification.ipynb"** and the previous slides for this task.
3. How accurate is your classifier?

Selecting Informative Attributes

- Which **features** should we use to best predict which customers will default on their credit?
  - Do we achieve the best prediction performance if we use all features?
  - Is there a subset of features that gives better results?
- Ideally, we want to find a combination of attributes that creates a **pure/perfect segmentation**.



Selecting Informative Attributes (cont'd)

- There are several **complications**:
  - In real data, we rarely have a variable that will make the segments **pure**, i.e., split the group perfectly.
  - Not all attributes are binary. Some take numeric values. It does not make sense to create a segment for every numeric value.

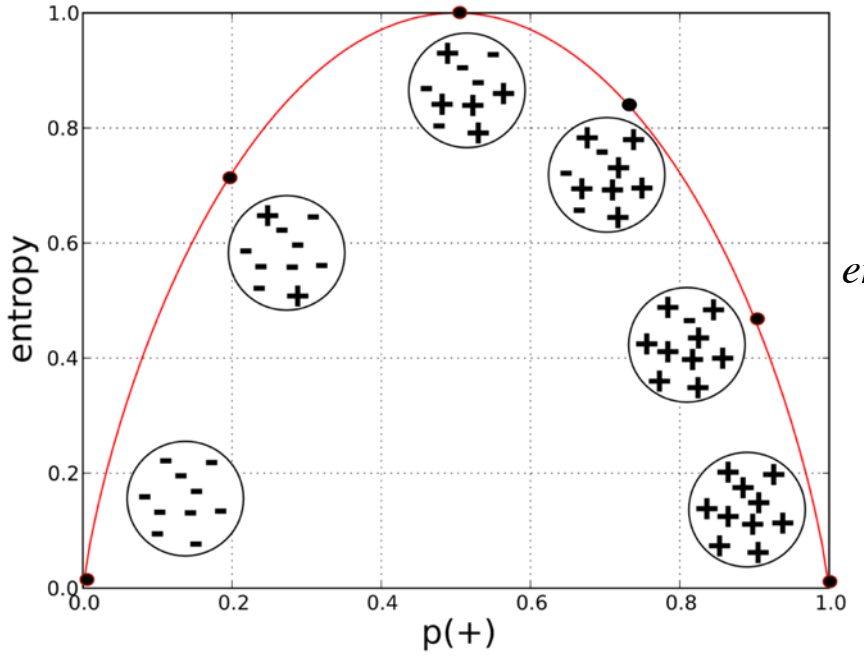
Additional Exercises: Experimenting with features

1. Open **"Classification.ipynb"** in the Notebook application.
2. Remove "**balance**" and "**income**" from the feature list. Run the prediction script and observe its accuracy.
3. Add "**balance**" or "**income**" to the list of features. How does it change the classification accuracy?
4. Experiment with **other combinations**. What can we say about the predictive "power" of our features?

Bonus: Experiment with the **features** for the diabetes classifier.

Selecting Informative Attributes (cont'd)

- A more formal procedure to select informative attributes is based on **information gain,** which uses a purity measure called **entropy**.
- We would like to measure how *informative* an attribute is with respect to our target. How much gain in information does it give us about the value of the target variable?
- **Entropy** describes *"how much information is missing"*.



$$entropy(S) = \; 0.7 \;\; \log_2(0.7) \;\; 0.3 \;\; \log_2(0.3)$$
$$0.7 \;\; (\; 0.51) \;\; 0.3 \;\; (\; 1.74)$$
$$0.88$$

Source: Data Science for Business, Provost & Fawcett

Selecting Informative Attributes (cont'd)

- The general definition of **entropy** is:

$$entropy(S) = -\sum_i p_i * \log_2(p_i)$$

  Each p_i is the probability (the relative percentage) of property/class *i* within the set.
- We define **information gain** to measure how much an attribute improves (decreases) entropy over the whole segmentation it creates.
- Information gain measures the **change in entropy** as a result of new information being added.
- When splitting a dataset into partitions, we define information gain as:

$$IG(parent, children) = entropy(parent) - [p(c_1) * entropy(c_1) + \cdots + p(c_n) * entropy(c_n)]$$

Proportion of instances belonging to child c_1

Recipe: Classification in Python

1. Import required Python modules
2. Load data
3. Select a classification model
   (e.g., sklearn.naive_bayes, sklearn.tree, sklearn.svm)
4. Fit model to the data
5. Get predictions
6. Evaluate results

Section Recap: What have we learned?

SUMMARY

- The intuition behind Bayes' Theorem ✓
- Classification using Naïve Bayes ✓
- Techniques to evaluate classification performance ✓
  - Accuracy
  - Confusion Matrix
  - Expected Value and Profit Curves
  - Receiver Operating Characteristic (ROC) and Area Under The Curve (AUC)
- Selecting informative attributes ✓

MORE INFO

- Scikit-learn Naïve Bayes
  http://scikit-learn.org/stable/modules/naive_bayes.html
- Scikit-learn Receiver Operating Characteristic
  http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
- "Twitter as a Corpus for Sentiment Analysis and Opinion Mining.", Pak, Paroubek, 2010
  http://lrec-conf.org/proceedings/lrec2010/pdf/385_Paper.pdf
- Supervised Learning Lecture on Coursera
  https://www.coursera.org/learn/machine-learning/lecture/1VkCb/supervised-learning

Data is the new oil!

Clive Humby

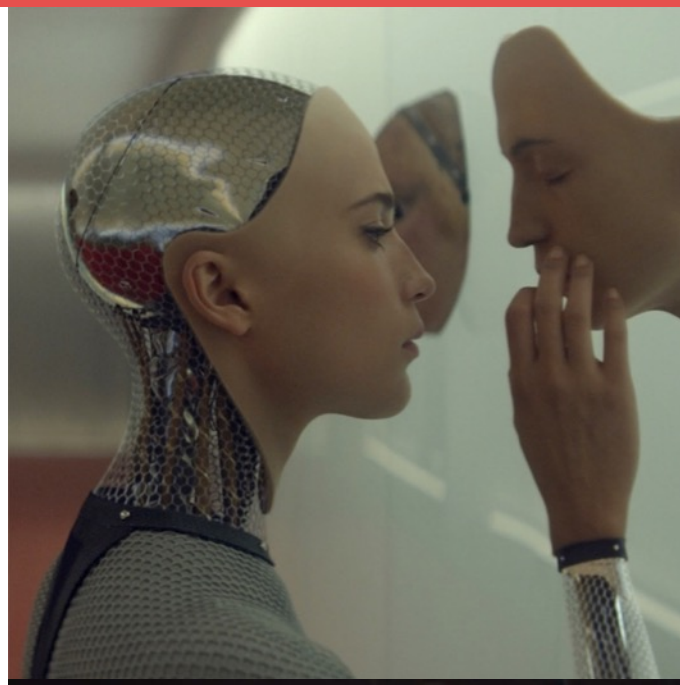
DATA SCIENCE METHODS

OVERVIEW

- How are Artificial Intelligence (AI), Machine Learning, and Data Science similar/different?
- What is Data Mining?
- Overview of data science methods
- Data Science Process
- What is the difference between supervised and unsupervised learning?

Artificial Intelligence (AI)
Turing Test (1950)

- **Natural Language Processing**
  to enable successful communication.
- **Knowledge Representation**
  to store information provided before or during the interrogation.
- **Automated Reasoning**
  to use the stored information to answer questions and to draw new conclusions.
- **Machine Learning**
  to adapt to new circumstances and to detect and extrapolate patterns.



Machine Learning

A subfield of AI that deals with methods for improving knowledge or performance over time, in response to experiences in the world.

Data Science/Data Mining

A set of fundamental principles that guide the extraction of knowledge or unknown patterns from data.

- **Regression**
  - *What is the relationship between different stocks and the market?*
  - *What makes a "good" credit rating?*
  - *How much will a given customer use the service?*
  - *How much is my property worth?*
  - *If we target customers x with new financial products y, how much will our sales likely increase?*
- Regression involves **numeric/continuous** target values.

- **Classification**
  - *Among all our customers, which are likely to respond to a given offer?*
  - *Which of our customers are likely to default on their credit card payments?*
  - *What do people on social media say about a particular company?*
  - *Which of our applicants are likely to perform well on the job?*
- Classification involves a **categorical** (often binary) target.

- **Similarity Matching**
  - *Which companies are similar to our best business customers so that we can focus our sales capacities on the best opportunities?*
  - *Amazon's product recommendation system: Which people are similar to a given customer in terms of the products they have purchased?*
- Similarity matching attempts to identify similar individuals/entities based on **known** data.

- **Clustering**
  - *Do our customers form natural groups or segments?*
  - *Can we find natural segments in the stock market to improve portfolio decisions?*
- Clustering attempts to **group** individuals in a population together by their **similarity**.

- **Co-occurrence grouping**
  - *Amazon's recommendation system: "Which items are commonly purchased together?"*

  - *Which financial products are commonly purchased together?*
- Attempts to find associations between entities based on **transactions** involving them.

- **Profiling**
  - *What is the product usage of this customer segment?*
  - *What characterizes our typical customer?*
- Profiling attempts to characterize the **typical behavior** of an individual, group, or population.

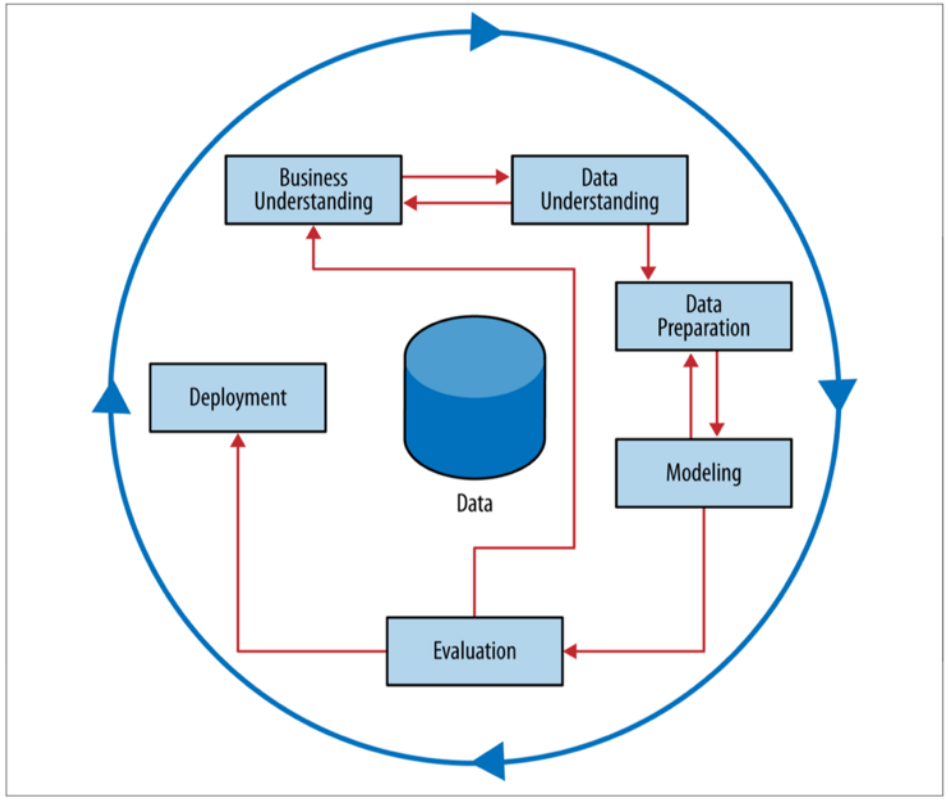
- **Link Prediction**
  - *Since you and Tom share 10 friends, maybe you'd like to be Tom's friend?*
- Link prediction attempts to **predict connections** between data items, usually by suggesting that a link should exist.

- **Data Reduction**
  - *A massive data set may be reduced to a much smaller dataset, which then*
    a. *better **reveals** the information and*
    b. *is **easier** to process.*
- Data Reduction attempts to take a large set of data and replace it with a smaller set of data that contains much of the important information in the larger set.
- Data Reduction usually involves a **loss of information** as a tradeoff for **improved insight**.

- **Causal Modeling**
  - *Did our increase in advertisement budgets really influence the consumers to purchase?*
  - *Placebo effect in medicine*
- Causal modeling attempts to help us understand what events or actions actually **influence** others.

Correlation does not imply causation!

The Data Science Process: From Problems to Tasks



Source: Data Science for Business, Provost & Fawcett

An **iterative** data science process

*"A critical skill in data science is the ability to **decompose** a data-analytics problem into pieces such that each piece matches a **known task** for which tools are available. Recognizing familiar problems and their solutions avoids wasting time and resources reinventing the wheel."*

Business Understanding

At the beginning of the process, it is necessary to **first understand** the problem to be solved.

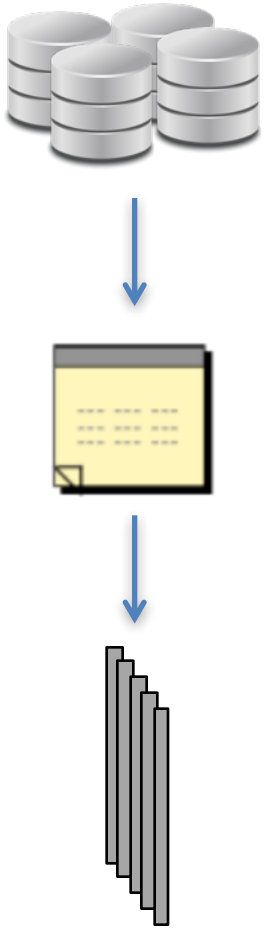
You got your data – what's next?

Data Understanding

In most cases, data is collected for purposes **unrelated** to the current business problem or for no explicit purpose at all. We need to understand the data and its limitations.

Data Preparation

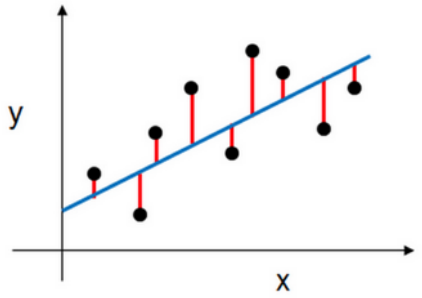
- The data need to be **manipulated** and **converted** into a format that is appropriate for further processing.
- Typical examples of data preparation include:
  - converting data into tabular format
  - inferring missing values
  - converting data types
  - rescaling of numerical values so that they can be compared
  - eliminating outliers
  - generating features



*Data Understanding and Data Preparation often take a **significant** amount of time.*

Modeling

- This step produces a model that captures **regularities** in the data.
- Yesterday, we fit a *regression model* to predict sales based on advertising budgets.



Evaluation

- The results produced by the model have to be assessed and validity needs to be **checked**.
- This step also helps ensure that the model satisfies the **original business goals**.

Deployment

- This final step puts the results into **real use** in order to realize some **return on investment**.
- Example:
  - We implement our learned predictive model in our company's HR information system. Results are visualized in a real-time web dashboard.
  - The learned model revealed that we can optimize a specific business process in our company division.
- Full production deployment often incurs substantial cost.
- In many cases the *data science team* is responsible for producing a working prototype before it is passed on to the *software development* team.

Supervised vs. Unsupervised Learning

- Let's consider the following example:
  - *Can we learn to predict if a financial transaction is fraudulent based on past labeled transaction data?*
- We want to learn a relationship between transactions (features) and their outcome (labels).
- We have a **specific target** ("fraud"/"no fraud").
- If we have **labeled data** we speak of a **supervised** learning problem.

Supervised vs. Unsupervised Learning (cont'd)

- Let's consider two other examples:
  - *Do our customers naturally fall into different groups?*
  - *Is there any particular structure in the data describing our company's assets/portfolio?*
- We cannot pinpoint a relationship, but we can try to group our observations based on common characteristics (features).
- We also say that the data is **unlabeled**.
- In this case the problem is called **unsupervised**.

Supervised vs. Unsupervised Learning (cont'd)

	Continuous Data	**Categorical Data**
Supervised	Regression	Classification
Unsupervised	Dimensionality Reduction	Clustering

Supervised vs. Unsupervised Learning (cont'd)

Exercises: Which of the following are **supervised or unsupervised** learning problems?

1. Given past patient data and test outcomes, how likely is it that the test will be positive for one of my new patients?
2. Are individuals who voted for Obama in the last Presidential election similar in their social media behavior?
3. An online banking service must be able to determine whether a transaction being performed on the site is fraudulent. To make a decision, the system uses the user's IP address, past transaction history, etc.
4. Do our website visitors form groups that show similar behavior?

Supervised vs. Unsupervised Learning (cont'd)

Exercises: Which of the following are **supervised or unsupervised** learning problems?

1. Given past patient data and test outcomes, how likely is it that the test will be positive for one of my new patients?
   Supervised – we base our decision on historical patient and test data (positive test/negative test).
2. Are individuals who voted for Obama in the last Presidential election similar in their social media behavior?
   Unsupervised – we perform an exploratory search in data set, without "knowing the truth".
3. An online banking service must be able to determine whether a transaction being performed on the site is fraudulent. To make a decision, the system uses the user's IP address, past transaction history, etc.
   Supervised – we base our decision on historical transaction and user data (fraudulent/not fraudulent).
4. Do our website visitors form groups that show similar behavior?
   Unsupervised – we do an exploratory analysis of website data without "knowing the truth".

Section Recap: What have we learned?

SUMMARY

- How are Data Science, AI, and Machine Learning similar/different? ✓
- What is Data Mining? ✓
- Overview of data science methods ✓
- Data Science Process ✓
- What is the difference between supervised and unsupervised learning? ✓

MORE INFO

- Data Mining Process and CRISP DM
  https://www.youtube.com/watch?v=nNc_q08yWxw
- 9 Most Common Data Science Tasks
  https://www.youtube.com/watch?v=3ri9H5iWeS4
- What is the difference between supervised learning and unsupervised learning?
  http://stackoverflow.com/questions/1832076/what-is-the-difference-between-supervised-learning-and-unsupervised-learning
- Artificial Intelligence: A Modern Approach (Russel, Norvig)
  http://aima.cs.berkeley.edu/

Data beats emotions.

Sean Rad, founder of Ad.ly

CLUSTERING

OVERVIEW

- Unsupervised modeling
- When to use clustering
- What is similarity and how do we measure it?
- The intuition behind k-means
- How to implement k-means clustering in Python
- How to improve your clustering model
- Using similarity for predictive modeling (classification)

Unsupervised Modeling

- So far, we have looked at **supervised** methods that model relationships between feature and target variables.
  - For example, we predicted credit defaulting behavior based on customer information.
- Sometimes it's good to take a step back and do a more general analysis.
  - For example, we may want to consider our marketing efforts more broadly.
  - Can we find natural subgroups in our customer data to develop better targeted campaigns?
- In this section, we will look the idea of finding natural subgroups in our data.
- This idea is called **unsupervised modeling** or **clustering**.

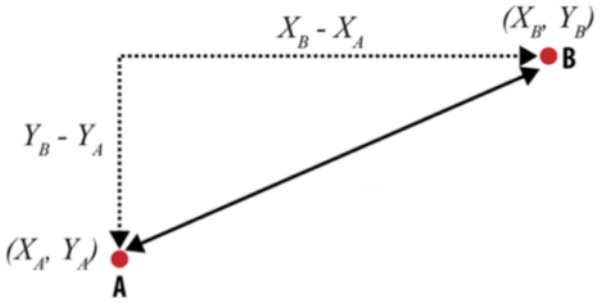
	Continuous Data	**Categorical Data**
Supervised	Regression	Classification
Unsupervised	Dimensionality Reduction	Clustering

Clustering

- Questions we can answer with **clustering**:
  - *Do our customers form natural groups of segments? What do these groups have in common?*
  - *What products should we offer or develop?*
  - *How should our customer care teams (or sales teams) be structured?*
  - *What other products should we recommend to existing customers?*
- In this section, we will cluster stock data to find **similarities in historical quotes**.

Similarity

- Before we generate clusters, we need to know what it means for two entities/data points (e.g., stocks) to be **similar**.
- We have two points in a XY coordinate system. What is the **distance** between the two points?



- Object A has coordinates X_A and Y_A, object B has coordinates X_B and Y_B.
- We can use the basic **Euclidean distance** formula to measure similarity:

$$Distance(A,B) = \sqrt{(X_A - X_B)^2 + (Y_A - Y_B)^2}$$

- Let's consider the following customer example:

Attribute	**Customer A**	**Customer B**
Age	26	29
Level	14	10

- We can calculate the similarity (distance) between the two customers using the **Euclidean distance formula** from the previous slide:

$$
\begin{aligned}
Distance(A,B) &= \sqrt{(26-29)^2 + (14-10)^2} \\
&= \sqrt{9+16} \\
&= \sqrt{25} \\
&= 5
\end{aligned}
$$

Similarity (cont'd)

- In many real-world clustering tasks, our data points are represented by more than two attributes.
- Let's add a third attribute to our customers:

Attribute	**Customer A**	**Customer B**
Age	26	29
Level	14	10
Activity	2	4

- We extend the Euclidean formula from before and write:

$$\begin{aligned} Distance(A,B) &= \sqrt{(26-29)^2 + (14-10)^2 + (2-4)^2} \\ &= \sqrt{9+16+4} \\ &= \sqrt{29} \\ &= 5.39 \end{aligned}$$

Similarity (cont'd)

- When our data points have *n* attributes, the general form of the Euclidean distance measure becomes:

$$Distance(A,B) = \sqrt{(d_{1,A} - d_{1,B})^2 + (d_{2,A} - d_{2,B})^2 + \cdots + (d_{n,A} - d_{n,B})^2}$$

Exercise:

1. Implement the Euclidean distance between two points with two attributes each in a new Python script.
2. Use the script created in step 1 to compute the distance between the following data points:

Attribute	**Visitor A**	**Visitor B**
Age	36	22
Visits	8	16

Manhattan ("City Block") Distance

$$Distance_{Manhattan}(A,B) = |d_{1,A} - d_{1,B}| + |d_{2,A} - d_{2,B}| + \ldots + |d_{n,A} - d_{n,B}|$$
$$= \sum_{i=1}^{n} |d_{i,A} - d_{i,B}|$$

Jaccard Distance

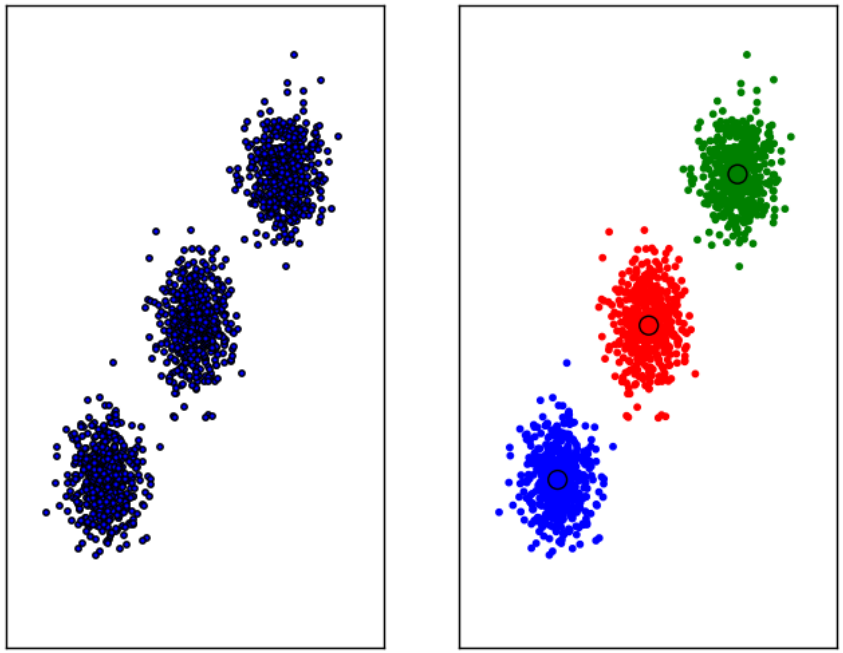
$$Distance_{Jaccard}(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Cosine Distance

$$Distance_{Cosine}(A,B) = 1 - \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

where $\|X\| = \sqrt{X \cdot X} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$

- k-means is a simple, yet powerful, **clustering technique** to partition a data set into k distinct, non-overlapping **groups**.

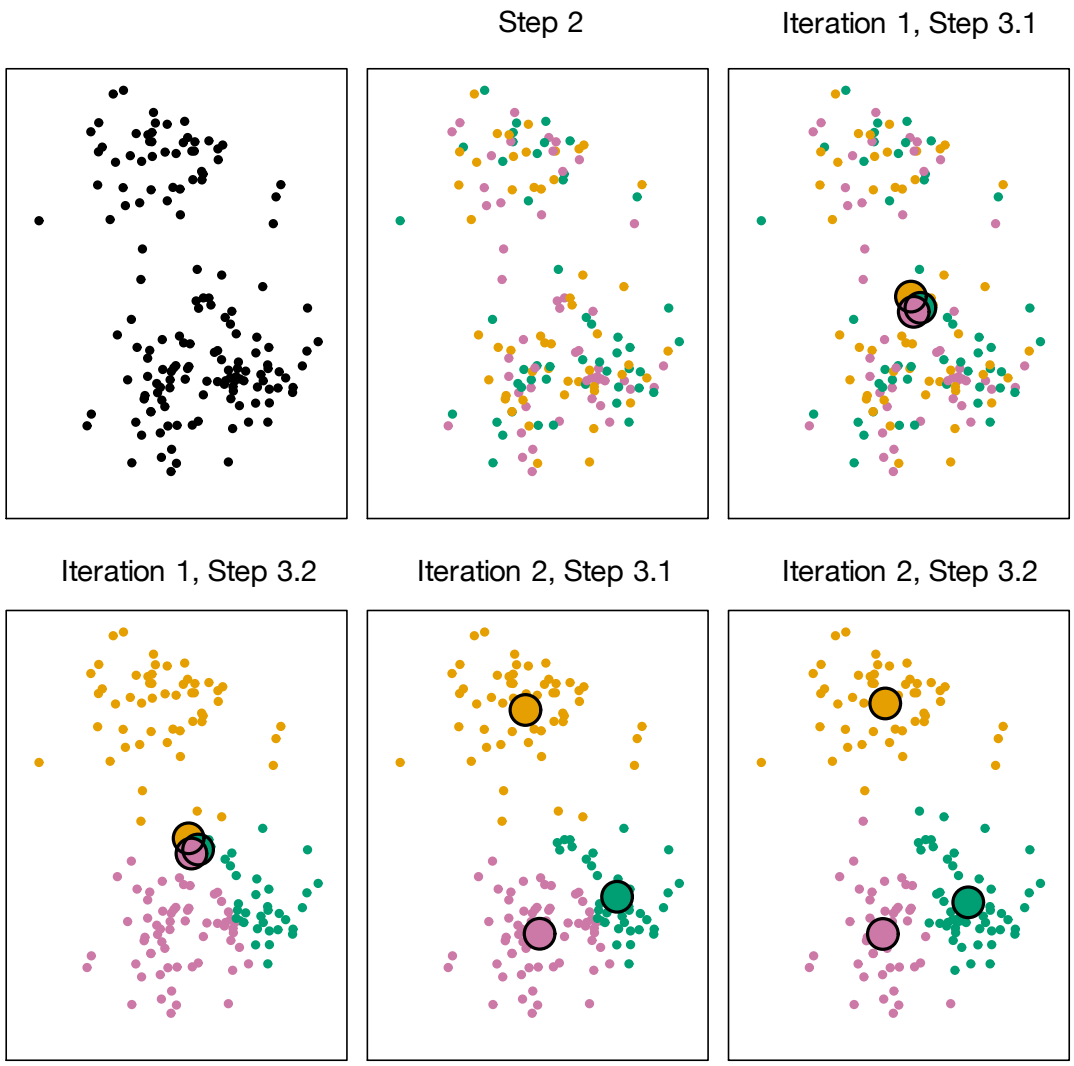


- k-means consists of the following **steps**:
  1. Choose k (number of clusters).
  2. Randomly assign a cluster label to each observation.
  3. **Repeat until clusters become stable:**
     1. Move each cluster center to the "center of gravity" of the data assigned to it (**Refitting step**).
     2. Assign data points to their closest cluster center (**Assign step**).

Exercise:

1. Where does '**k-means**' derive its name from?

k-means (cont'd)



k-means (cont'd)

- Mathematically speaking, k-means minimizes the **variation within clusters**.
- For a particular cluster, C_k, we can define $W(C_k)$ as the amount by which the data points within a cluster **differ** from each other.
- k-means then solves the following problem:

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

- We use the **Euclidean distance** as a measure of variation within the cluster:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

k-means in Action

Exercises: Perform **k-means clustering**, with k = 2, on a small example with n = 6 observations and p = 2 features. The observations are as follows:

Observation #	**X1**	**X2**
1	1	4
2	1	3
3	0	4
4	5	1
5	6	2
6	4	0

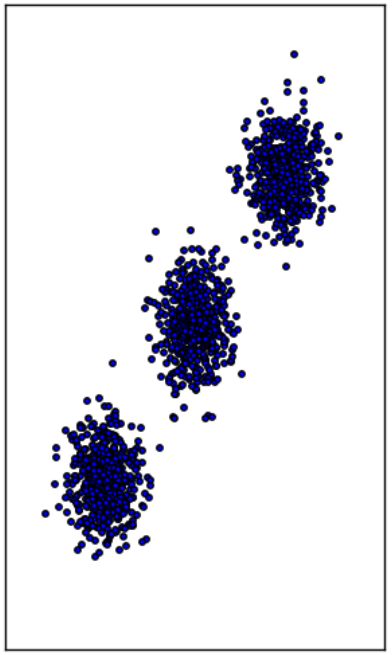
1. Plot the observations.
2. Randomly assign a cluster label to each observation.
3. Compute the **centers** for each cluster (does not have to be exact)
4. Assign each observation to the cluster which is closest, in terms of Euclidean distance.
5. Repeat steps 3) and 4) until the clusters stop changing.

k-means in Action (cont'd)

Use this space to solve the problem on the previous slide.

Implementing k-means in Python

- Let's see how we can implement **k-means in Python**.
- Python's **scikit-learn** library provides convenient tools to perform clustering.
  - We could also implement the k-means algorithm ourselves without any libraries!
- We will learn how to implement the following in Python:
  1. Create data / load dataset
  2. Prepare data (format, standardization, …)
  3. Perform k-means clustering on dataset
  4. Plot results
  5. Assess accuracy of our results

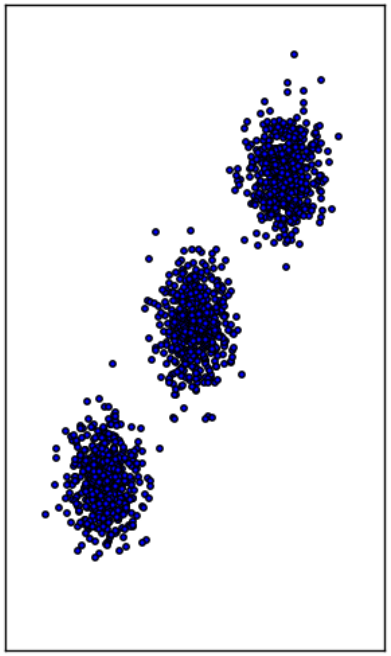


Implementing k-means in Python (cont'd)

- Let's start with the "toy data" shown on the right.
- **Scikit-learn** provides us with convenient mechanisms to create toy data sets.
  - This is useful when testing/debugging our algorithms.
- The following creates toy data ("blobs"):

```
from sklearn import datasets

# Define size of data set
n_samples = 1500

# Generate some normalized toy data
centers = [(-5, -5), (0, 0), (5, 5)]
n_clusters = len(centers)
X, y = datasets.make_blobs(n_samples=n_samples,
                           cluster_std=1.0,
                           centers=centers,
                           shuffle=False,
                           random_state=42)
```

Setting random_state to reproduce experiments.



More on creating toy data here: http://scikit-learn.org/stable/modules/classes.html#samples-generator

Implementing k-means in Python (cont'd)

- Before we run k-means on our data, it is important to **standardize our dataset**.
  - We should always standardize incomparable units (e.g., height in cm and weight in kg).
  - We should always standardize variables with different variances.
- We can standardize our dataset with the following code:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X = scaler.fit_transform(X)
```

- To run k-means on our dataset we use the following code:

```
from sklearn.cluster import KMeans

n_clusters = 5

# Configure our k-means clustering algorithm
kmeans = KMeans(n_clusters)

kmeans.fit(X)
```

We need to define n_clusters as the number of clusters k-means should use.

Visualizing Stock Market Structure

- We want to employ k-means clustering to **extract structure** from variations in historical stock prices.
  - Can be helpful in determining which stocks move similarly (e.g., cyclical stocks, growth stocks)
    - From an **investment banking** content, this could be another "test" to use when identifying comparable companies for **public comps** spreading.
    - From a **portfolio management** context, this could be helpful in deciding which stocks to add to portfolios to maximize diversification benefits since stocks that move similarly provide **less diversification** benefits than those that do not move similarly.

Exercises: Clustering Stock Data

1. Inspect "**Clustering.ipynb**". What does the existing code do?
2. Complete the implementation.
3. Experiment with the **number of clusters** used for k-means and observe your results.

Assessing the Quality of Our Results

- k-means guarantees **some** solution, i.e., it always generates k clusters for us.
- But, how well do the clusters represent true subgroups in the data? This answer is not always easy to answer!
- **Scikit-learn's** clustering object provides us with an attribute called "**inertia_**", which gives us the **sum of distances** of samples to their closest cluster center:

```
kmeans.inertia_
```

Finding the Optimal k in k-means

- It's generally **not easy to choose k**, so we need to conduct experimentation and use our own judgement!
- The following are commonly used **guidelines**:
  - Rule of Thumb: **k ~ sqrt(n/2)**, where n is the number of observations.
  - Run algorithm for several values of k. Plot number of clusters against average variance. We want to **minimize** both parameters ("Elbow method").
  - Start with one cluster, then keep **splitting** clusters until the data points assigned to **each cluster** have a **Gaussian distribution** (Hamerly, Elkan, NIPS, 2004).

When to Use k-means?

- **Advantages** of k-means:
  - Simple and efficient
  - Scales well
  - Always converges (we always get *a* solution)
    - In both the assign and refitting steps, the sum of squared distances of data points to their assigned clusters is **reduced**.
- **Disadvantages** of k-means:
  - Not always clear how to select k
  - Can get stuck in local minimum (we get a suboptimal result)
  - Can produce distorted clusters if outliers are present
- **When** should we use k-means?
  - If similarity of data points can be measured with **Euclidean distance or similar metric** (e.g., Manhattan, Jaccard, or Cosine distance)

Recipe: Clustering in Python

1. Import required Python modules
2. Get your data
3. Select a clustering algorithm
   Check sklearn.cluster for common clustering algorithms (e.g., KMeans, MiniBatchKMeans, SpectralClustering)
4. Fit your model to the data
5. Get your predictions
6. Evaluate your results

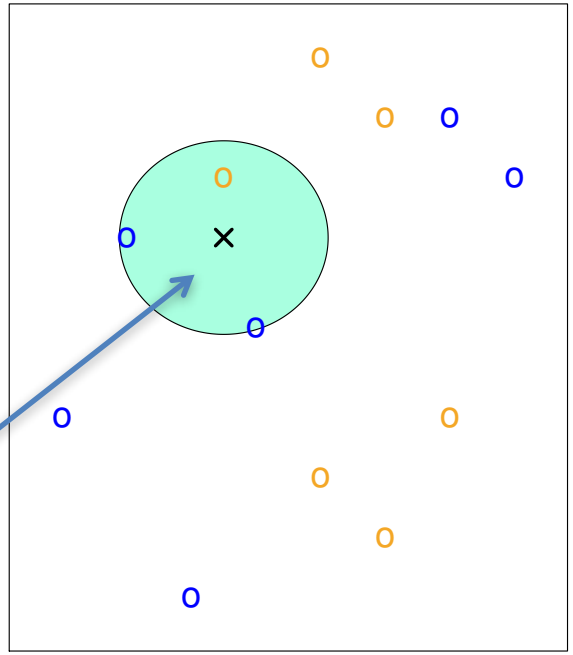
Using Similarity for Predictive Modeling

- Once we have generated our clusters and are confident that they represent subgroups in the data we can use them for **predictive modeling**.
- Consider the following **examples**:
  - A doctor may draw conclusions about a new difficult case by recalling a similar case and its diagnosis.
  - A lawyer may argue cases by citing similar historical cases.

Using Similarity for Predictive Modeling (cont'd)

- Given a new data point whose target variable we want to predict, we:
  - Scan through all the present data points and chose several (k) that are most similar to the new data point.
  - Predict the new data point's target value using the **nearest neighbors'** known target values.

  - For a **classification task**, we find the new target using majority voting among the nearest neighbors.
  - For a **regression task**, we find the new target using the average or median of the nearest neighbors' target values.



Classifying new data point with k = 3

Section Recap: What have we learned?

SUMMARY

- Unsupervised modeling ✓
- When to use clustering ✓
- What is similarity and how do we measure it? ✓
- The intuition behind k-means ✓
- How to implement k-means clustering in Python ✓
- How to improve your clustering model ✓
- Using similarity for predictive modeling (Classification) ✓

MORE INFO

- Scikit-learn Clustering
  http://scikit-learn.org/stable/modules/clustering.html
- Unsupervised Learning Lecture on Coursera
  https://www.coursera.org/learn/machine-learning/lecture/olRZo/unsupervised-learning
- Nature Article on Machine Intelligence and Deep Learning
  http://www.nature.com/news/computer-science-the-learning-machines-1.14481

Every day, we create 2.5 quintillion bytes of data — so much that 90% of the data in the world today has been created in the last two years alone.

IBM

BIG DATA

OVERVIEW

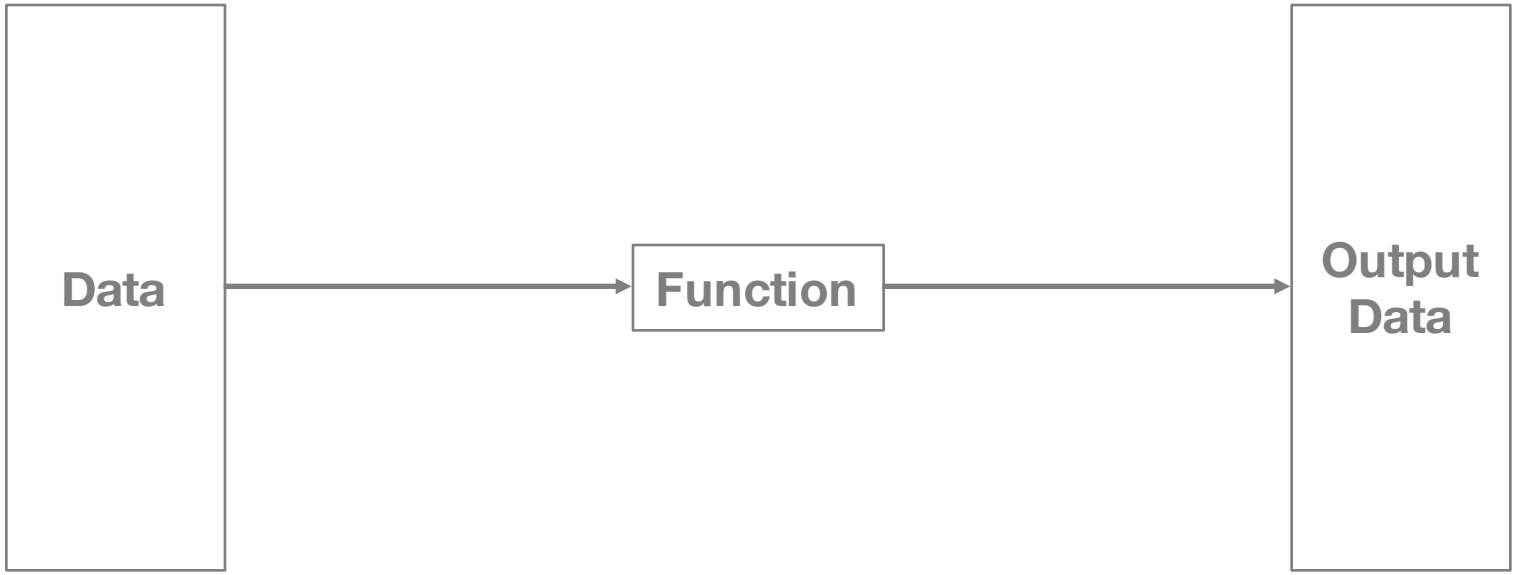
- What is Big Data? Why is it relevant?
- Big Data concepts
- What are common Big Data technologies?

Big Data refers to datasets that are **too large** for traditional data processing systems to process.

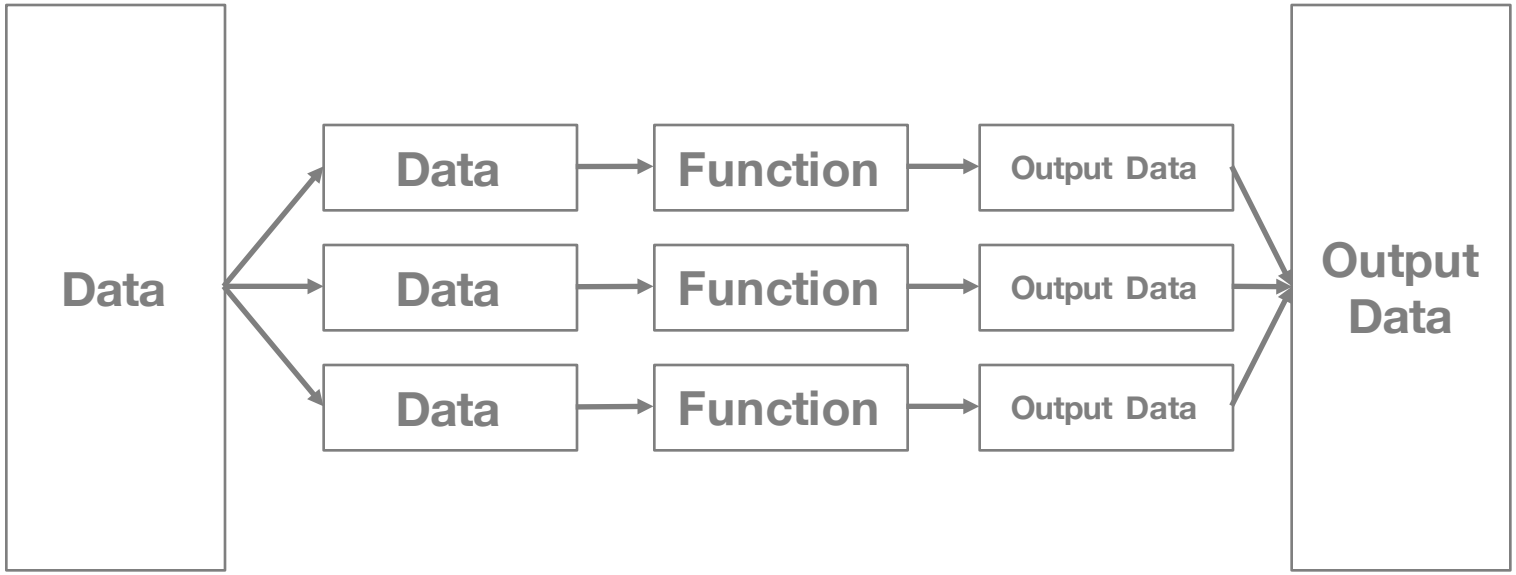
Three Vs:

- **Volume**
  Facebook processes more than 500TB of data daily.
- **Velocity**
  Frequency stock trading algorithms reflect market changes within microseconds.
- **Variety**
  Audio, photos, videos, time series, geospatial data, 3D data, and unstructured text, etc.

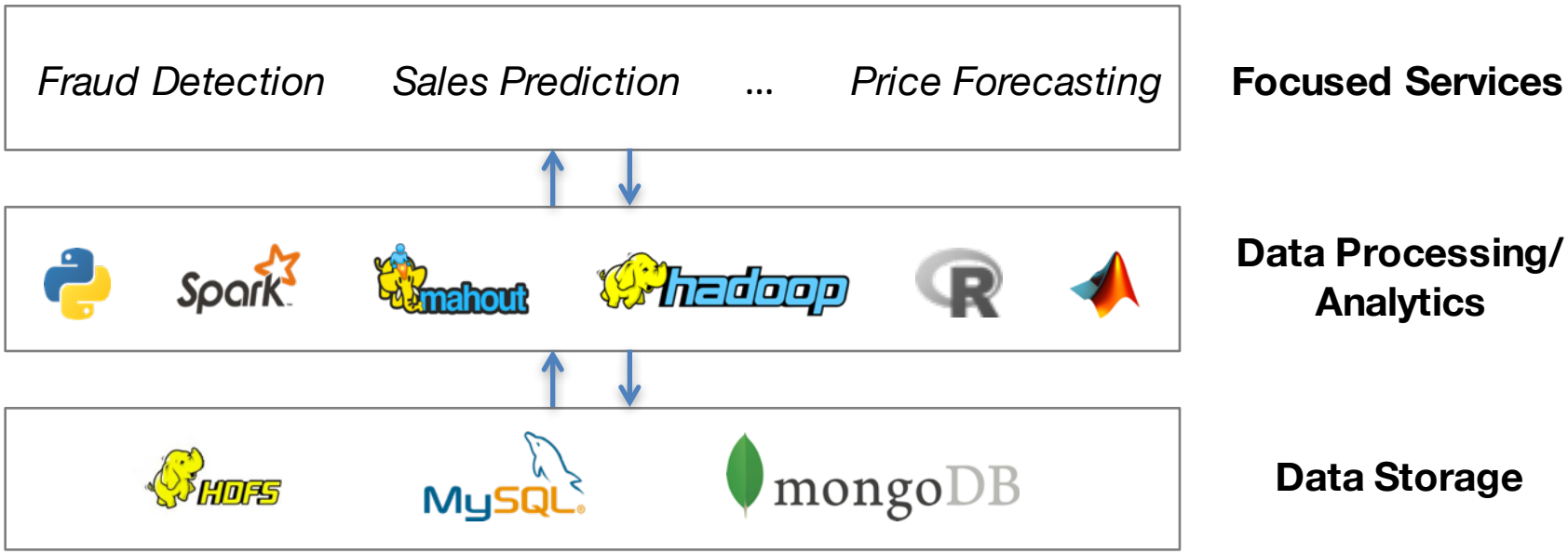
Traditional Data Processing



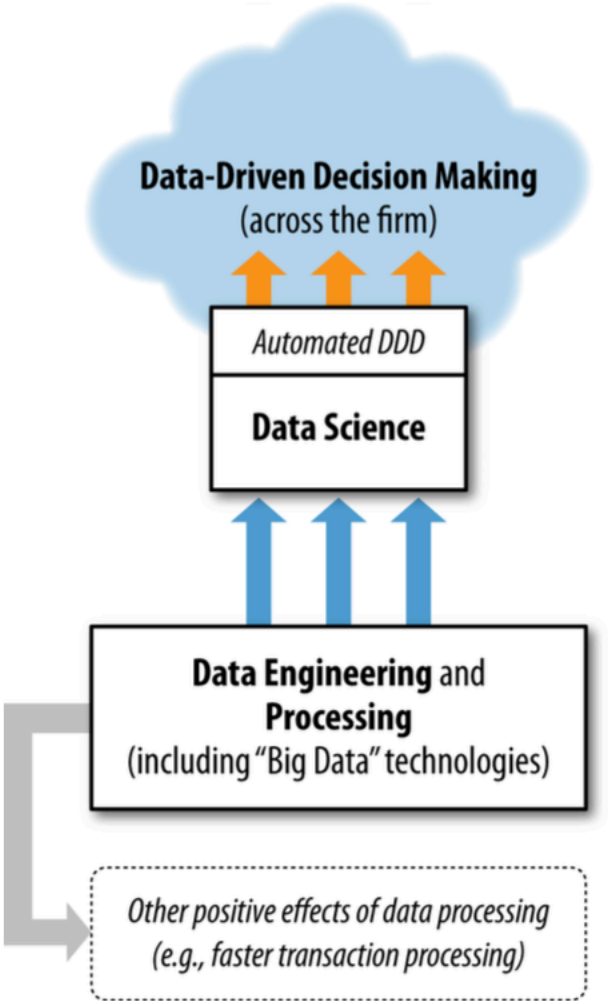
MapReduce Data Processing



Big Data requires new **distributed/parallel** computing architectures and **complex technologies**.



Data-Driven Decision Making



Source: Data Science for Business, Provost & Fawcett

Section Recap: What have we learned?

SUMMARY

- What is Big Data? Why is it relevant? ✓
- Big Data concepts ✓
- What are common Big Data technologies? ✓

MORE INFO

- "Big data: The next frontier for innovation, competition, and productivity"
  http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation
- "Big Data Explained"
  https://www.mongodb.com/big-data-explained
- Apache Hadoop
  https://hadoop.apache.org/
- Data Science in the Data Analytics Process
  https://www.youtube.com/watch?v=S6PbOP-8lBw

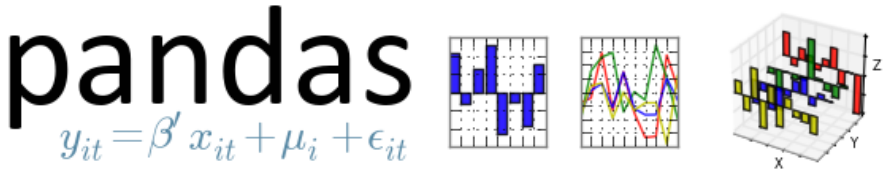
WHAT HAVE WE LEARNED?

What Have We Learned?

- What is Data Science and why is it important?
- Basics of how to use Python programming for data analysis
- Different types of data science tools for solving business problems
- What is Big Data?

	Continuous Data	**Categorical Data**
Supervised	Regression	Classification
Unsupervised	Dimensionality Reduction	Clustering

- We have learned how to use tools for **effective data analysis**:



WHERE TO GO FROM HERE?

Reading:

- Data Science from Scratch, Grus, O'Reilly
- Python for Data Analysis, McKinney, O'Reilly
- Introduction to Statistical Learning, James, Witten, Hastie, Tibshirani, Springer
- Pattern Recognition and Machine Learning, Bishop

Hands-on data science training:

- http://www.kaggle.com/competitions

Where To Go From Here (cont'd)

Visit our website www.cognitir.com for **advanced data science training** courses.

This course includes **post-workshop support** for three months after the event. You can **e-mail us** course-related questions to *support@cognitir.com* and we will get back to you within two business days.

To **stay up to date with our courses and free resource offerings**, sign up to our **newsletter** on *www.cognitir.com* and follow us on **social media**.

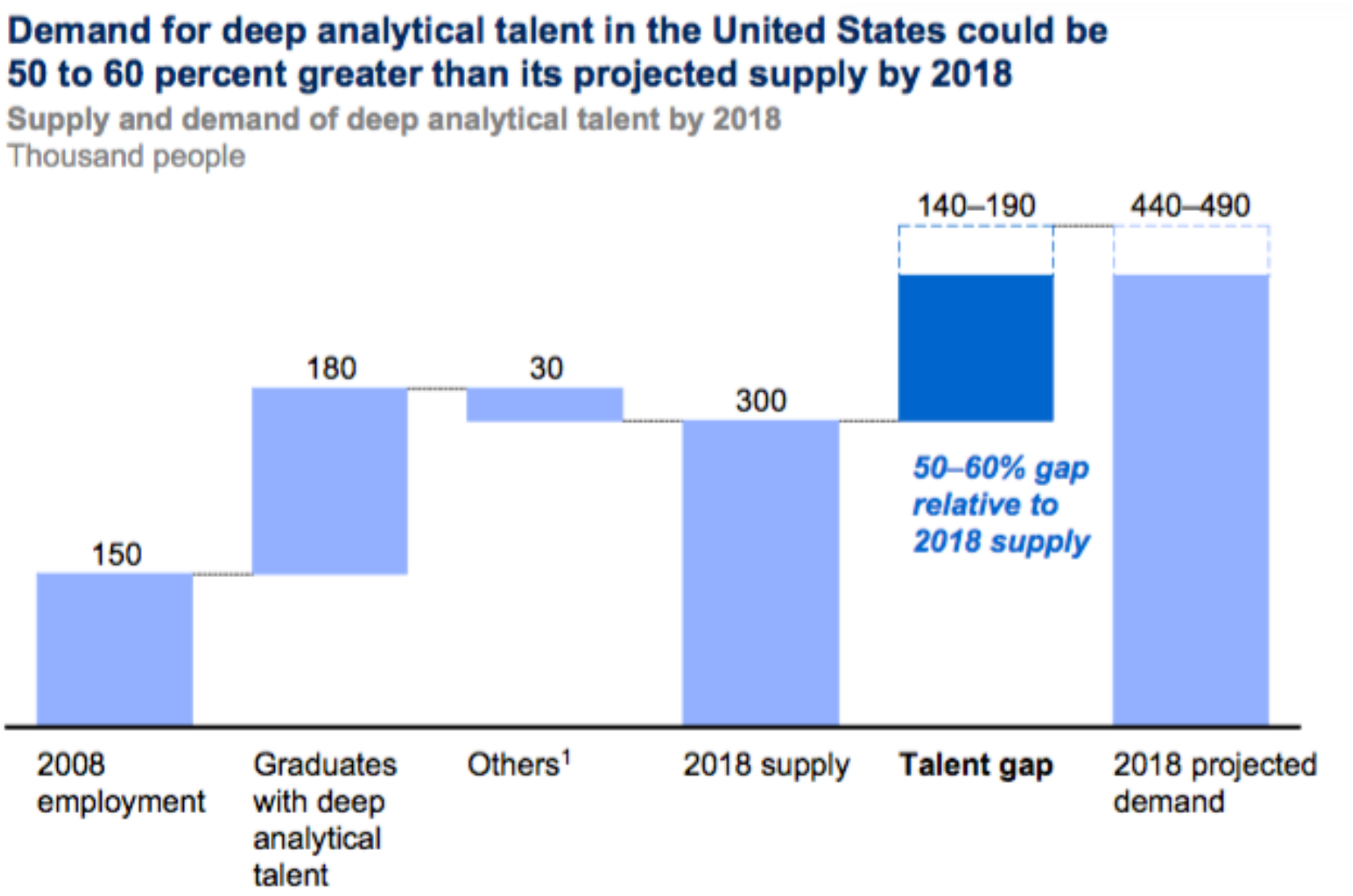
Please send **feedback** to *feedback@cognitir.com*.

 *@cognitir*

 *cognitir*

 *cognitir*

Your Opportunity!



Source: McKinsey Global Institute

The journey of a thousand miles begins with one step.

Lao Tzu

CONGRATULATIONS!

APPENDIX

PYTHON LIBRARIES

Python Library: pandas

pandas is an open source Python library for **effective data analysis**.

It offers data structures and operations for manipulating **numerical tables** and **time series**.

Features include:

- **Reading and writing** data between in-memory data structures and different formats (CSV, Microsoft Excel, SQL databases, HDF5, etc.)
- Rich data structures and operations to for easy and fast **data manipulations**
- **Time series functionalities** such as date range generation, frequency conversion, moving window statistics, moving window linear regressions, etc.

More information available at http://pandas.pydata.org.

Python Library: NumPy

NumPy is one of Python's fundamental packages for **scientific computing**.

Provides the primary container for data to be passed between algorithms: **ndarray**.

Features include:

- A fast and efficient multidimensional array object ndarray
- Functions for performing element-wise computations with arrays or mathematical operations between arrays
- Tools for reading and writing array-based data sets to disk
- Linear algebra operations, Fourier transform, and random number generation
- Tools for integrating connecting C, C++, and Fortran code to Python

More information available at http://www.numpy.org.

Python Library: matplotlib

matplotlib is a popular plotting library for the Python programming language.

It was designed to closely resemble **MATLAB**'s plotting functionalities. Plots can be exported as **PNG**, **PDF**, **SVG** or **PS** and directly used for presentations, on websites, etc.

Matplotlib supports plots including:

- Line Charts
- Histograms
- Power Spectra
- Bar Charts
- Error Charts
- Scatter Plots

More information available at http://www.matplotlib.org.
An alternative for interactive visualization is Bokeh (by our partners Continuum Analytics): http://bokeh.pydata.org/

Python Library: scikit-learn

scikit-learn is an open source **machine learning library** for the Python programming language.

It is built on **NumPy**, **matplotlib**, and **SciPy**. Companies such as Spotify and Evernote use scikit-learn for a variety of data science/machine learning tasks.

Features include:

- Classification
- Regression
- Clustering
- Dimensionality Reduction
- Model Selection
- Preprocessing

More information available at http://scikit-learn.org.

Statsmodels provides a complement to scikit-learn/scipy for statistical computations: http://statsmodels.sourceforge.net