Practices of Business Intelligence

预测性分析 I：
资料探勘流程、方法與演算法
(Predictive Analytics I: Data Mining Process, Methods, and Algorithms)

1071BI06
MI4 (M2084) (2888)
Wed, 7, 8 (14:10-16:00) (B217)

Min-Yuh Day
戴敏育
Assistant Professor
Dept. of Information Management, Tamkang University

http://mail.tku.edu.tw/myday/
<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2018/09/12</td>
<td>商業智慧實務課程介紹 (Course Orientation for Practices of Business Intelligence)</td>
</tr>
<tr>
<td>2</td>
<td>2018/09/19</td>
<td>商業智慧、分析與資料科學 (Business Intelligence, Analytics, and Data Science)</td>
</tr>
<tr>
<td>3</td>
<td>2018/09/26</td>
<td>人工智慧、大數據與雲端運算 (ABC: AI, Big Data, and Cloud Computing)</td>
</tr>
<tr>
<td>4</td>
<td>2018/10/03</td>
<td>描述性分析I：數據的性質、統計模型與可視化 (Descriptive Analytics I: Nature of Data, Statistical Modeling, and Visualization)</td>
</tr>
<tr>
<td>5</td>
<td>2018/10/10</td>
<td>國慶紀念日 (放假一天) (National Day) (Day off)</td>
</tr>
<tr>
<td>6</td>
<td>2018/10/17</td>
<td>描述性分析II：商業智慧與資料倉儲 (Descriptive Analytics II: Business Intelligence and Data Warehousing)</td>
</tr>
</tbody>
</table>
課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)

7 2018/10/24 預測性分析 I：資料探勘流程、方法與演算法
   (Predictive Analytics I: Data Mining Process, Methods, and Algorithms)

8 2018/10/31 預測性分析 II：文本、網路與社群媒體分析
   (Predictive Analytics II: Text, Web, and Social Media Analytics)

9 2018/11/07 期中報告 (Midterm Project Report)

10 2018/11/14 期中考試 (Midterm Exam)

11 2018/11/21 處方性分析：最佳化與模擬
    (Prescriptive Analytics: Optimization and Simulation)

12 2018/11/28 社會網絡分析
   (Social Network Analysis)
課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)
13 2018/12/05 機器學習與深度學習
      (Machine Learning and Deep Learning)
14 2018/12/12 自然語言處理
      (Natural Language Processing)
15 2018/12/19 AI交談機器人與對話式商務
      (AI Chatbots and Conversational Commerce)
16 2018/12/26 商業分析的未來趨勢、隱私與管理考量
      (Future Trends, Privacy and Managerial Considerations in Analytics)
17 2019/01/02 期末報告 (Final Project Presentation)
18 2019/01/09 期末考試 (Final Exam)
Business Intelligence (BI)

1. Introduction to BI and Data Science
2. Descriptive Analytics
3. Predictive Analytics
4. Prescriptive Analytics
5. Big Data Analytics
6. Future Trends
Predictive Analytics I: Data Mining Process, Methods, and Algorithms
Outline

• Data Mining Concepts and Applications
• Data Mining Applications
• Data Mining Process
• Data Mining Methods
• Data Mining Software Tools
• Data Mining Privacy Issues, Myths, and Blunders
Data Mining

• Data Mining
  – Discovering or “mining” knowledge from large amounts of data.

• Data mining is a misnomer
  – Mining of gold from within rocks or dirt is referred to as “gold” mining rather than “rock” or “dirt” mining.

  – Data mining
    • “knowledge mining”
    • “knowledge discovery”
Data Mining

• Knowledge extraction
• Pattern analysis
• Data archaeology
• Information harvesting
• Pattern searching
• Data dredging
Data Mining
Technical Definition

• Data mining is a process that uses statistical, mathematical, and artificial intelligence techniques to extract and identify useful information and subsequent knowledge (or patterns) from large sets of data.

• These patterns can be in the form of business rules, affinities, correlations, trends, or prediction models.

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Data Mining Definition
(Fayyad et al., 1996)

• “The nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data stored in structured databases.”
Data Mining

• **Process**
  – data mining comprises many iterative steps.

• **Nontrivial**
  – some experimentation-type search or inference is involved
  – it is not as straightforward as a computation of predefined quantities.

• **Valid**
  – the discovered patterns should hold true on new data with a sufficient degree of certainty.

• **Novel**
  – the patterns are not previously known to the user within the context of the system being analyzed.

• **Potentially useful**
  – the discovered patterns should lead to some benefit to the user or task.

• **Ultimately understandable**
  – the pattern should make business sense

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), *Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition*, Pearson
Data Mining
Is a Blend of Multiple Disciplines

Data Mining
(Knowledge Discovery)

Statistics

Management Science & Information Systems

Artificial Intelligence

Database Management & Data Warehousing

Machine Learning & Pattern Recognition

Information Visualization

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
## Data Mining Tasks & Methods

<table>
<thead>
<tr>
<th>Data Mining Tasks &amp; Methods</th>
<th>Data Mining Algorithms</th>
<th>Learning Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prediction</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classification</td>
<td>Decision Trees, Neural Networks, Support Vector Machines, kNN, Naive Bayes, GA</td>
<td>Supervised</td>
</tr>
<tr>
<td>Regression</td>
<td>Linear/Nonlinear Regression, ANN, Regression Trees, SVM, kNN, GA</td>
<td>Supervised</td>
</tr>
<tr>
<td>Time series</td>
<td>Autoregressive Methods, Averaging Methods, Exponential Smoothing, ARIMA</td>
<td>Supervised</td>
</tr>
<tr>
<td><strong>Association</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Link analysis</td>
<td>Expectation Maximization, Apriori Algorithm, Graph-Based Matching</td>
<td>Unsupervised</td>
</tr>
<tr>
<td>Sequence analysis</td>
<td>Apriori Algorithm, FP-Growth, Graph-Based Matching</td>
<td>Unsupervised</td>
</tr>
<tr>
<td><strong>Segmentation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clustering</td>
<td>k-means, Expectation Maximization (EM)</td>
<td>Unsupervised</td>
</tr>
<tr>
<td>Outlier analysis</td>
<td>k-means, Expectation Maximization (EM)</td>
<td>Unsupervised</td>
</tr>
</tbody>
</table>

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), *Business Intelligence, Analytics, and Data Science: A Managerial Perspective*, 4th Edition, Pearson
Data Mining Applications

- Customer Relationship Management (CRM)
- Banking
- Retailing and logistics
- Manufacturing and production
- Brokerage and securities trading
- Computer hardware and software
- Government and defense
- Travel industry (airlines, hotels/resorts, rental car companies)
- Healthcare
- Medicine
- Entertainment industry
- Homeland security and law enforcement
- Sports

The Six-Step CRISP-DM Data Mining Process

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Model Building
5. Testing and Evaluation
6. Deployment

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
SEMMA Data Mining Process

- **Sample**
  (Generate a representative sample of the data)

- **Assess**
  (Evaluate the accuracy and usefulness of the models)

- **Explore**
  (Visualization and basic description of the data)

- **Model**
  (Use variety of statistical and machine learning models)

- **Modify**
  (Select variables, transform variable representations)

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Ranking of Data Mining Methodologies/Processes

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
A Data Mining Methodology for Investigation of Comorbidity in Cancer Survivability

Source: Ramesh Sharda, Dursun Delen, and Efrem Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Data Mining Methods

• Classification
  – Classification
    • Class Label Prediction
  – Regression
    • Numeric Value Prediction

• Clustering

• Association

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Assessing the Classification Model

• Predictive accuracy
  – Hit rate

• Speed
  – Model building; predicting

• Robustness

• Scalability

• Interpretability
  – Transparency, explainability

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Confusion Matrix for Tabulation of Two-Class Classification Results

<table>
<thead>
<tr>
<th>True/Observed Class</th>
<th>Predicted Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Positive</td>
</tr>
<tr>
<td>True Positive Count (TP)</td>
<td>False Positive Count (FP)</td>
</tr>
<tr>
<td>Negative</td>
<td>Negative</td>
</tr>
<tr>
<td>False Negative Count (FN)</td>
<td>True Negative Count (TN)</td>
</tr>
</tbody>
</table>

- **Accuracy** = \( \frac{TP + TN}{TP + TN + FP + FN} \)
- **True Positive Rate** = \( \frac{TP}{TP + FN} \)
- **True Negative Rate** = \( \frac{TN}{TN + FP} \)
- **Precision** = \( \frac{TP}{TP + FP} \)
- **Recall** = \( \frac{TP}{TP + FN} \)

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Accuracy vs. Precision

- **A**: High Accuracy, High Precision
- **B**: Low Accuracy, High Precision
- **C**: High Accuracy, Low Precision
- **D**: Low Accuracy, Low Precision
Accuracy vs. Precision

A

High Accuracy
High Precision
High Validity
High Reliability

B

Low Accuracy
High Precision
Low Validity
High Reliability

C

High Accuracy
Low Precision
High Validity
Low Reliability

D

Low Accuracy
Low Precision
Low Validity
Low Reliability
Accuracy vs. Precision

A
- High Accuracy
- High Precision
- High Validity
- High Reliability

B
- Low Accuracy
- High Precision
- Low Validity
- High Reliability

C
- High Accuracy
- Low Precision
- High Validity
- Low Reliability

D
- Low Accuracy
- Low Precision
- Low Validity
- Low Reliability
Sensitivity = True Positive Rate

Specificity = True Negative Rate
True Class (actual value)

<table>
<thead>
<tr>
<th>Predictive Class (prediction outcome)</th>
<th>True Class (actual value)</th>
<th>Predictive Class (prediction outcome)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Positive Positive (TP)</td>
<td>Negative False Positive (FP)</td>
</tr>
<tr>
<td>Negative</td>
<td>False Negative (FN)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>total</th>
<th>P</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>P'</td>
<td>N'</td>
</tr>
</tbody>
</table>

**Total**

- **P**
- **N**

**True Positive Rate (Sensitivity)**
$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

**True Negative Rate (Specificity)**
$$\text{True Negative Rate} = \frac{TN}{TN + FP}$$

**False Positive Rate**
$$\text{False Positive Rate} = \frac{FP}{FP + TN}$$

**False Positive Rate (1 - Specificity)**
$$\text{False Positive Rate (1 - Specificity)} = \frac{FP}{FP + TN}$$

**Accuracy**
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

**True Positive Rate**
$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

**True Negative Rate**
$$\text{True Negative Rate} = \frac{TN}{TN + FP}$$

**Precision**
$$\text{Precision} = \frac{TP}{TP + FP}$$

**Recall**
$$\text{Recall} = \frac{TP}{TP + FN}$$

True Positive Rate (Sensitivity) = \( \frac{TP}{TP + FN} \)

**Sensitivity**
- True Positive Rate
- Recall
- Hit rate
- \( TP / (TP + FN) \)
### Confusion Matrix Table

<table>
<thead>
<tr>
<th></th>
<th>True Class (actual value)</th>
<th>Predictive Class (prediction outcome)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
<td>Negative</td>
</tr>
<tr>
<td>Positive</td>
<td>True Positive (TP)</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>Negative</td>
<td>False Negative (FN)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

**Total**
- P: True Positive + False Positive
- N: True Negative + False Negative

**True Negative Rate**

\[
\text{True Negative Rate} = \frac{TN}{TN + FP}
\]

**Specificity**

\[
\text{Specificity} = \text{True Negative Rate} = \frac{TN}{N} = \frac{TN}{(TN + FP)}
\]

**False Positive Rate**

\[
\text{False Positive Rate} (1 - \text{Specificity}) = \frac{FP}{FP + TN}
\]

Precision
= Positive Predictive Value (PPV)

\[ \text{Precision} = \frac{TP}{TP + FP} \]

Recall
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate

\[ \text{Recall} = \frac{TP}{TP + FN} \]

F1 score (F-score)(F-measure)
is the harmonic mean of precision and recall
= \( \frac{2TP}{(P + P')} \)
= \( \frac{2TP}{(2TP + FP + FN)} \)

\[ F = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]

Recall = True Positive Rate (TPR) = Sensitivity = Hit Rate = TP / (TP + FN)

Specificity = True Negative Rate = TN / N = TN / (TN + FP)

TPR = 0.63
FPR = 0.28
PPV = 0.69

Precision = Positive Predictive Value (PPV)

F1 = 0.66
= 2*(0.63*0.69)/(0.63+0.69) = (2 * 63) / (100 + 91) = (0.63 + 0.69) / 2 =1.32 / 2 =0.66

ACC = 0.68
= (63 + 72) / 200 = 135/200 = 67.5

### A

<table>
<thead>
<tr>
<th></th>
<th>TP (63)</th>
<th>FP (28)</th>
<th>TN (72)</th>
<th>FN (37)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum</td>
<td>100</td>
<td>100</td>
<td>200</td>
<td>100</td>
</tr>
</tbody>
</table>

- TPR = 0.63
- FPR = 0.28
- PPV = 0.69
  
  \[
  \text{PPV} = \frac{TP}{TP + FP} = \frac{63}{63 + 28} = \frac{63}{91}
  \]

- F1 = 0.66
  
  \[
  F1 = \frac{2 \times (\text{TPR} \times \text{PPV})}{\text{TPR} + \text{PPV}} = \frac{2 \times (0.63 \times 0.69)}{0.63 + 0.69} = \frac{2 \times 63}{100 + 91} = \frac{63 + 72}{200} = \frac{135}{200} = 0.675
  \]

- ACC = 0.68
  
  \[
  ACC = \frac{TP + TN}{TP + FP + FN + TN} = \frac{63 + 72}{100 + 91} = \frac{135}{200} = 0.675
  \]

### B

<table>
<thead>
<tr>
<th></th>
<th>TP (77)</th>
<th>FP (23)</th>
<th>TN (77)</th>
<th>FN (23)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum</td>
<td>100</td>
<td>100</td>
<td>200</td>
<td>100</td>
</tr>
</tbody>
</table>

- TPR = 0.77
- FPR = 0.77
- PPV = 0.50
  
  \[
  \text{PPV} = \frac{TP}{TP + FP} = \frac{77}{77 + 23} = \frac{77}{100}
  \]

- F1 = 0.61
  
  \[
  F1 = \frac{2 \times (\text{TPR} \times \text{PPV})}{\text{TPR} + \text{PPV}} = \frac{2 \times (0.77 \times 0.50)}{0.77 + 0.50} = \frac{2 \times 77}{100 + 23}
  \]

- ACC = 0.50
  
  \[
  ACC = \frac{TP + TN}{TP + FP + FN + TN} = \frac{77 + 77}{100 + 23} = \frac{154}{200} = 0.77
  \]

---

**Recall**

\[
\text{Recall} = \frac{TP}{TP + FN}
\]

**Precision**

\[
\text{Precision} = \frac{TP}{TP + FP}
\]
<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>C'</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP</td>
<td>24</td>
<td>76</td>
</tr>
<tr>
<td>FP</td>
<td>88</td>
<td>12</td>
</tr>
<tr>
<td>FN</td>
<td>76</td>
<td>24</td>
</tr>
<tr>
<td>TN</td>
<td>100</td>
<td>88</td>
</tr>
</tbody>
</table>

**TPR = 0.76**
**FPR = 0.12**
**PPV = 0.86**
**F1 = 0.81**
**ACC = 0.82**

**TPR = 0.24**
**FPR = 0.88**
**PPV = 0.21**
**F1 = 0.22**
**ACC = 0.18**

**Recall**
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate

**Precision**
= Positive Predictive Value (PPV)

Estimation Methodologies for Classification

- **Simple split** (or holdout or test sample estimation)
  - Split the data into 2 mutually exclusive sets: training (~70%) and testing (30%)
  - For ANN, the data is split into three sub-sets: (training [~60%], validation [~20%], testing [~20%])

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
$k$-Fold Cross-Validation

Repeated for all 10 folds

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Estimation Methodologies for Classification Area under the ROC curve

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Ensemble Models
Heterogeneous Ensemble

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Steps in the *k*-Means Algorithm

Identification of Frequent Itemsets in the Apriori Algorithm

<table>
<thead>
<tr>
<th>Raw Transaction Data</th>
<th>One-Item Itemsets</th>
<th>Two-Item Itemsets</th>
<th>Three-Item Itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transaction No</strong></td>
<td><strong>SKUs (Item No)</strong></td>
<td><strong>Itemset (SKUs)</strong></td>
<td><strong>Support</strong></td>
</tr>
<tr>
<td>1001234</td>
<td>1, 2, 3, 4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1001235</td>
<td>2, 3, 4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>1001236</td>
<td>2, 3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1001237</td>
<td>1, 2, 4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1001238</td>
<td>1, 2, 3, 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001239</td>
<td>2, 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
## Selected Data Mining Software

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Web Site (URL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM SPSS Modeler</td>
<td>www-01.ibm.com/software/analytics/spss/products/modeler/</td>
</tr>
<tr>
<td>IBM Watson Analytics</td>
<td>ibm.com/analytics/watson-analytics/</td>
</tr>
<tr>
<td>SAS Enterprise Miner</td>
<td>sas.com/en_id/software/analytics/enterprise-miner.html</td>
</tr>
<tr>
<td>Dell Statistica</td>
<td>statsoft.com/products/statistica/product-index</td>
</tr>
<tr>
<td>PolyAnalyst</td>
<td>megaputer.com/site/polyanalyst.php</td>
</tr>
<tr>
<td>CART, RandomForest</td>
<td>salford-systems.com</td>
</tr>
<tr>
<td>XLMiner</td>
<td>solver.com/xlminer-data-mining</td>
</tr>
<tr>
<td>SAP InfiniteInsight (KXEN)</td>
<td>help.sap.com/ii</td>
</tr>
<tr>
<td>GhostMiner</td>
<td>fqs.pl/ghostminer</td>
</tr>
<tr>
<td>SQL Server Data Mining</td>
<td>msdn.microsoft.com/en-us/library/bb510516.aspx</td>
</tr>
<tr>
<td>Knowledge Miner</td>
<td>knowledgeminer.com</td>
</tr>
<tr>
<td>Teradata Warehouse Miner</td>
<td>teradata.com/products-and-services/teradata-warehouse-miner/</td>
</tr>
<tr>
<td>Oracle Data Mining (ODM)</td>
<td>oracle.com/technetwork/database/options/odm/</td>
</tr>
<tr>
<td>FICO Decision Management</td>
<td>fico.com/en/analytics/decision-management-suite/</td>
</tr>
<tr>
<td>Orange Data Mining Tool</td>
<td>orange.biologlab.si/</td>
</tr>
<tr>
<td>Zementis Predictive Analytics</td>
<td>zementis.com</td>
</tr>
</tbody>
</table>

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Popular Data Mining Software Tools
(Poll Results)

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Process Flow Screenshot for the Box-Office Prediction System

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
## Tabulated Prediction Results for Individual and Ensemble Models

<table>
<thead>
<tr>
<th>Performance Measure</th>
<th>SVM</th>
<th>ANN</th>
<th>CART</th>
<th>Random Forest</th>
<th>Boosted Tree</th>
<th>Fusion (Average)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count (Bingo)</strong></td>
<td>192</td>
<td>182</td>
<td>140</td>
<td>189</td>
<td>187</td>
<td><strong>194</strong></td>
</tr>
<tr>
<td><strong>Count (1-Away)</strong></td>
<td>104</td>
<td>120</td>
<td>126</td>
<td>121</td>
<td>104</td>
<td><strong>120</strong></td>
</tr>
<tr>
<td><strong>Accuracy (% Bingo)</strong></td>
<td>55.49%</td>
<td>52.60%</td>
<td>40.46%</td>
<td>54.62%</td>
<td>54.05%</td>
<td><strong>56.07%</strong></td>
</tr>
<tr>
<td><strong>Accuracy (% 1-Away)</strong></td>
<td>85.55%</td>
<td>87.28%</td>
<td>76.88%</td>
<td>89.60%</td>
<td>84.10%</td>
<td><strong>90.75%</strong></td>
</tr>
<tr>
<td><strong>Standard deviation</strong></td>
<td>0.93</td>
<td>0.87</td>
<td>1.05</td>
<td>0.76</td>
<td>0.84</td>
<td><strong>0.63</strong></td>
</tr>
</tbody>
</table>

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), *Business Intelligence, Analytics, and Data Science: A Managerial Perspective*, 4th Edition, Pearson
Data Mining Privacy Issues

• De-identification

• Many publicly available data sources (e.g., CDC data, SEER data, UNOS data) are already de-identified.

## Data Mining Myths

<table>
<thead>
<tr>
<th>Myth</th>
<th>Reality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data mining provides instant, crystal-ball-like predictions.</td>
<td>Data mining is a multistep process that requires deliberate, proactive design and use.</td>
</tr>
<tr>
<td>Data mining is not yet viable for mainstream business applications.</td>
<td>The current state of the art is ready to go for almost any business type and/or size.</td>
</tr>
<tr>
<td>Data mining requires a separate, dedicated database.</td>
<td>Because of the advances in database technology, a dedicated database is not required.</td>
</tr>
<tr>
<td>Only those with advanced degrees can do data mining.</td>
<td>Newer Web-based tools enable managers of all educational levels to do data mining.</td>
</tr>
<tr>
<td>Data mining is only for large firms that have lots of customer data.</td>
<td>If the data accurately reflect the business or its customers, any company can use data mining.</td>
</tr>
</tbody>
</table>

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Foundations of Big Data Mining in Python

- **Python**
  - Programming language

- **Numpy**
  - Scientific computing

- **Pandas**
  - Data structures and data analysis tools
Welcome to Colab!

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. See our FAQ for more info.

Getting Started

- Overview of Colab
- Loading and saving data: Local files, Drive, Sheets, Google Cloud Storage
- Importing libraries and installing dependencies
- Using Google Cloud BigQuery
- Forms, Charts, Markdown, & Widgets
- TensorFlow with GPU
- Machine Learning Crash Course: Intro to Pandas & First Steps with TensorFlow

Highlighted Features

Seedbank

Looking for Colab notebooks to learn from? Check out Seedbank, a place to discover interactive machine learning examples.

TensorFlow execution

Colaboratory allows you to execute TensorFlow code in your browser with a single click. The example below adds two matrices.

\[
\begin{bmatrix}
1 & 1 & 1 \\
\end{bmatrix} + 
\begin{bmatrix}
1 & 2 & 3 \\
\end{bmatrix} = 
\begin{bmatrix}
2 & 3 & 4 \\
\end{bmatrix}
\]
Python in Google Colab

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

```python
print("hello, world")
```

```python
# comment
from platform import python_version
print("Python Version: ", python_version())
```

```
Python Version: 3.6.6
```

```python
# https://www.learnpython.org/en/
# LearnPython.org interactive Python tutorial
print("Hello World")
print("Hello World\nThis is a message")
x = 3
print(x)
x = 2
y = 3
print(x, ' ', y)
```

```
Hello World
Hello World
This is a message
3
2
3
```

```python
# Python Variables
x = 2
price = 2.5
word = 'Hello'
word = 'Hello'
word = "Hello"
word = "'Hello''"```
### Python in Google Colab

[https://colab.research.google.com/drive/1FEG6DnGvWFUbeo4zJ1zTunjMqf2RkCrT](https://colab.research.google.com/drive/1FEG6DnGvWFUbeo4zJ1zTunjMqf2RkCrT)

```python
# Future Value
pv = 100
r = 0.1
n = 7
fv = pv * ((1 + (r)) ** n)
print(round(fv, 2))
```

```
[11]
amount = 100
interest = 10 #10% = 0.01 * 10
years = 7

future_value = amount * ((1 + (0.01 * interest)) ** years)
print(round(future_value, 2))
```

```
[12]
# Python Function def
def getfv(pv, r, n):
    fv = pv * ((1 + (r)) ** n)
    return fv
fv = getfv(100, 0.1, 7)
print(round(fv, 2))
```

```
[13]
# Python if else
score = 80
if score >= 60 :
    print("Pass")
else:
    print("Fail").
```

Pass
Variables

```python
x = 2
price = 2.5
word = 'Hello'

word = 'Hello'
word = "Hello"
word = '''Hello'''

x = 2
x = x + 1
x = 5
```

Source: [http://pythonprogramminglanguage.com/](http://pythonprogramminglanguage.com/)
Python Basic Operators

```python
print('7 + 2 =', 7 + 2)
print('7 - 2 =', 7 - 2)
print('7 * 2 =', 7 * 2)
print('7 / 2 =', 7 / 2)
print('7 // 2 =', 7 // 2)
print('7 % 2 =', 7 % 2)
print('7 ** 2 =', 7 ** 2)
```
BMI Calculator in Python

```python
height_cm = float(input("Enter your height in cm: "))
weight_kg = float(input("Enter your weight in kg: "))

height_m = height_cm/100
BMI = (weight_kg/(height_m**2))

print("Your BMI is: " + str(round(BMI,1)))
```

Source: http://code.activestate.com/recipes/580615-bmi-code/
BMI Calculator in Python

```python
height_cm = float(input("Enter your height in cm: "))
weight_kg = float(input("Enter your weight in kg: "))

height_m = height_cm/100
BMI = (weight_kg/(height_m**2))

print("Your BMI is: " + str(round(BMI,1)))
```

Enter your height in cm: 170
Enter your weight in kg: 60
Your BMI is: 20.8
Future value of a specified principal amount, rate of interest, and a number of years
# How much is your $100 worth after 7 years?

```python
print(100 * 1.1 ** 7)
```

# output = 194.87
Future Value (FV)

\[ pv = 100 \]
\[ r = 0.1 \]
\[ n = 7 \]

\[ fv = pv \times ((1 + (r)) ^{n}) \]

print(round(fv, 2))

194.87
Future Value (FV)

```python
amount = 100
interest = 10 #10% = 0.01 * 10
years = 7

future_value = amount * ((1 + (0.01 * interest)) ** years)
print(round(future_value, 2))
```

194.87

Source: https://www.w3resource.com/python-exercises/python-basic-exercise-39.php
if statements

> greater than
< smaller than
== equals
!= is not

score = 80
if score >=60 :
   print("Pass")
else:
   print("Fail")

Pass

score = 80
if score >=60 :
   print("Pass")
else:
   print("Fail")
score = 90
grade = ""
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "E"
print(grade)
# grade = "A"

http://pythontutor.com/visualize.html
https://goo.gl/E6w5ph

Source: http://pythonprogramminglanguage.com/
for loops

for i in range(1,11):
    print(i)

1
2
3
4
5
6
7
8
9
10
for loops

```python
for i in range(1,10):
    for j in range(1,10):
        print(i, ' * ' , j , ' = ', i*j)
```

9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81

Source: [http://pythonprogramminglanguage.com/](http://pythonprogramminglanguage.com/)
while loops

age = 10

while age < 20:
    print(age)
    age = age + 1

10
11
12
13
14
15
16
17
18
19

Source: https://learnpython.trinket.io/learn-python-part-8-loops#_while-loops/about-while-loops
```python
def convertCMtoM(xcm):
    m = xcm/100
    return m

cm = 180
m = convertCMtoM(cm)
print(str(m))
```

1.8
Lists

```python
x = [60, 70, 80, 90]
print(len(x))
print(x[0])
print(x[1])
print(x[-1])
```

4
60
70
90
Tuples

A tuple in Python is a collection that cannot be modified. A tuple is defined using parenthesis.

\[ x = (10, 20, 30, 40, 50) \]

\[
\begin{align*}
&\text{print}(x[0]) \\
&\text{print}(x[1]) \\
&\text{print}(x[2]) \\
&\text{print}(x[-1])
\end{align*}
\]

Source: http://pythonprogramminglanguage.com/tuples/
Dictionary

```python
k = { 'EN':'English', 'FR':'French' }
print(k['EN'])
```

Source: http://pythonprogramminglanguage.com/dictionary/
Sets

animals = {'cat', 'dog'}

animals = {'cat', 'dog'}
print('cat' in animals)  # Check if an element is in a set; prints "True"
print('fish' in animals)  # prints "False"
animals.add('fish')       # Add an element to a set
print('fish' in animals)  # Prints "True"
print(len(animals))       # Number of elements in a set; prints "3"
animals.add('cat')        # Adding an element that is already in the set does nothing
print(len(animals))       # Prints "3"
animals.remove('cat')     # Remove an element from a set
print(len(animals))       # Prints "2"

True
False
True
3
3
2

animals = {'cat', 'dog'}
print('cat' in animals)
print('fish' in animals)
animals.add('fish')
print('fish' in animals)
print(len(animals))
animals.add('cat')
print(len(animals))
animals.remove('cat')
print(len(animals))

Source: http://cs231n.github.io/python-numpy-tutorial/
with open('myfile.txt', 'w') as file:
    file.write('Hello World
This is Python File Input Output')

with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)

Hello World
This is Python File Input Output

text

'Hello World
This is Python File Input Output'
with open('myfile.txt', 'a+') as file:  
    file.write('
' + 'New line')

with open('myfile.txt', 'r') as file:  
    text = file.read()  
print(text)

Hello World  
This is Python File Input Output  
New line

Source: https://github.com/TiesdeKok/LearnPythonforResearch/blob/master/0_python_basics.ipynb
Big Data Analytics with Numpy in Python
Numpy

NumPy
Base
N-dimensional array
package
NumPy is the fundamental package for scientific computing with Python.

Source: http://www.numpy.org/
NumPy

• NumPy provides a multidimensional array object to store homogenous or heterogeneous data; it also provides optimized functions/methods to operate on this array object.
NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

Getting Started

- Getting NumPy
- Installing the SciPy Stack
- NumPy and SciPy documentation page
- NumPy Tutorial
- NumPy for MATLAB© Users
- NumPy functions by category
- NumPy Mailing List

For more information on the SciPy Stack (for which NumPy provides the fundamental array data structure), see scipy.org.

http://www.numpy.org/
One-dimensional Array (1-D Array)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>m-1</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

Two-dimensional Array (2-D Array)
v = list(range(1, 6))
v
2 * v

import numpy as np
v = np.arange(1, 6)
v
2 * v
1 \( v = \text{list(range}(1, 6)) \)
2 \( v \)

[1, 2, 3, 4, 5]

1 \( 2 * v \)

[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]

1 \( \text{import numpy as np} \)
2 \( v = \text{np.arange}(1, 6) \)
3 \( v \)

array([1, 2, 3, 4, 5])

1 \( 2 * v \)

array([2, 4, 6, 8, 10])
```python
import numpy as np
a = np.array([1, 2, 3])

b = np.array([4, 5, 6])

c = a * b

c
```

Source: Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
import numpy as np

a = np.zeros((2,2))  # Create an array of all zeros
print(a)            # Prints "[[' 0.  0.]
                     #   [ 0.  0.]]"

b = np.ones((1,2))  # Create an array of all ones
print(b)            # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7)  # Create a constant array
print(c)            # Prints "[[' 7.  7.]
                     #   [ 7.  7.]]"

d = np.eye(2)       # Create a 2x2 identity matrix
print(d)            # Prints "[[' 1.  0.]
                     #   [ 0.  1.]]"

e = np.random.random((2,2))  # Create an array filled with random values
print(e)            # Might print "[[' 0.91940167  0.08143941]
                     #   [ 0.68744134  0.87236687]]"
Quickstart Tutorial

Prerequisites

Before reading this tutorial you should know a bit of Python. If you would like to refresh your memory, take a look at the Python tutorial.

If you wish to work the examples in this tutorial, you must also have some software installed on your computer. Please see http://scipy.org/install.html for instructions.

The Basics

NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called axes. The number of axes is rank.

For example, the coordinates of a point in 3D space \([1, 2, 1]\) is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```python
[[ 1., 0., 0.],
 [ 0., 1., 2.]]
```

NumPy’s array class is called ndarray. It is also known by the alias array. Note that numpy.array is not the same as the Standard Python Library class array.array, which only handles one-dimensional arrays and offers less functionality. The more important attributes of an ndarray object are:

- `ndarray.ndim`:
  - the number of axes (dimensions) of the array. In the Python world, the number of dimensions is referred to as rank.

- `ndarray.shape`: 

https://docs.scipy.org/doc/numpy-dev/user/quickstart.html
import numpy as np
a = np.arange(15).reshape(3, 5)

a.shape
a.ndim
a.dtype.name
Matrix

An $m$-by-$n$ matrix is a rectangular array of numbers or elements, organized in $m$ rows and $n$ columns. The elements are denoted as $a_{i,j}$, where $i$ represents the row and $j$ represents the column.

Source: https://simple.wikipedia.org/wiki/Matrix_(mathematics)
NumPy ndarray:
Multidimensional Array Object
One-dimensional Array  
(1-D Array)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Two-dimensional Array  
(2-D Array)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

NumPy ndarray
import numpy as np

a = np.array([1, 2, 3, 4, 5])

One-dimensional Array (1-D Array)

|   | 0 | 1 | 2 | 3 | 4 | 5 |
---|---|---|---|---|---|---|
|   | 1 | 2 | 3 | 4 | 5 |   |

a = np.array([1, 2, 3, 4, 5])
a

array([1, 2, 3, 4, 5])
Two-dimensional Array (2-D Array)

\[ a = \text{np.array}([[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15],[16,17,18,19,20]]) \]
```python
import numpy as np
a = np.array([[0, 1, 2, 3],
              [10, 11, 12, 13],
              [20, 21, 22, 23]])
a
```

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
</tbody>
</table>
a = np.array([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])

print(a.ndim)
2

print(a.shape)
(3, 4)
NumPy Basics: Arrays and Vectorized Computation
NumPy Array

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,0</td>
<td>0,1</td>
<td>0,2</td>
</tr>
<tr>
<td>1</td>
<td>1,0</td>
<td>1,1</td>
<td>1,2</td>
</tr>
<tr>
<td>2</td>
<td>2,0</td>
<td>2,1</td>
<td>2,2</td>
</tr>
</tbody>
</table>
```

# Numpy Array

<table>
<thead>
<tr>
<th>Expression</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>arr[:2, 1:]</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>arr[2]</td>
<td>(3,)</td>
</tr>
<tr>
<td>arr[2, :]</td>
<td>(3,)</td>
</tr>
<tr>
<td>arr[2:, :]</td>
<td>(1, 3)</td>
</tr>
<tr>
<td>arr[:, :2]</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>arr[1, :2]</td>
<td>(2,)</td>
</tr>
<tr>
<td>arr[1:2, :2]</td>
<td>(1, 2)</td>
</tr>
</tbody>
</table>

Materials and IPython notebooks for "Python for Data Analysis" by Wes McKinney, published by O'Reilly Media

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 months ago</td>
<td>Add Kaggle titanic dataset</td>
</tr>
<tr>
<td>4 months ago</td>
<td>Remove sex column from tips dataset</td>
</tr>
<tr>
<td>2 years ago</td>
<td>Add gitignore</td>
</tr>
<tr>
<td>a month ago</td>
<td>Use MIT license for code examples</td>
</tr>
<tr>
<td>19 days ago</td>
<td>Add launch in Azure Notebooks button</td>
</tr>
<tr>
<td>a month ago</td>
<td>Make more cells markdown instead of raw</td>
</tr>
<tr>
<td>a month ago</td>
<td>Make more cells markdown instead of raw</td>
</tr>
<tr>
<td>a month ago</td>
<td>Make more cells markdown instead of raw</td>
</tr>
<tr>
<td>a month ago</td>
<td>Convert all notebooks to v4 format</td>
</tr>
<tr>
<td>a month ago</td>
<td>Make more cells markdown instead of raw</td>
</tr>
<tr>
<td>a month ago</td>
<td>Make more cells markdown instead of raw</td>
</tr>
<tr>
<td>a month ago</td>
<td>Convert all notebooks to v4 format</td>
</tr>
<tr>
<td>a month ago</td>
<td>Make more cells markdown instead of raw</td>
</tr>
<tr>
<td>a month ago</td>
<td>Make more cells markdown instead of raw</td>
</tr>
<tr>
<td>a month ago</td>
<td>Make more cells markdown instead of raw</td>
</tr>
<tr>
<td>a month ago</td>
<td>Make more cells markdown instead of raw</td>
</tr>
</tbody>
</table>
NumPy Basics: Arrays and

```
In [ ]:
import numpy as np
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 5))
np.set_printoptions(precision=4, suppress=True)
```

```
In [ ]:
import numpy as np
my_arr = np.arange(1000000)
my_list = list(range(1000000))
```

```
In [ ]:
%time for _ in range(10): my_arr2 = my_arr * 2
%time for _ in range(10): my_list2 = [x * 2 for x in my_list]
```

The NumPy ndarray: A Multidimensional Array Object

```
In [ ]:
import numpy as np
# Generate some random data
data = np.random.randn(2, 3)
data
```
Python Pandas
Python Data Analysis Library

*pandas* is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the *Python* programming language.

*pandas* is a NUMFocus sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project.

A Fiscally Sponsored Project of NUMFocus

**0.19.2 Final (December 24, 2016)**

This is a minor bug-fix release in the 0.19.x series and includes some small regression fixes, bug fixes and performance improvements.

Highlights include:

- Compatibility with Python 3.6

http://pandas.pydata.org/
pandas
Python Data Analysis Library
providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Source: http://pandas.pydata.org/
Creating `pd.DataFrame`

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
</tr>
</tbody>
</table>

```
import pandas as pd
df = pd.DataFrame(
    {
        "a": [4, 5, 6],
        "b": [7, 8, 9],
        "c": [10, 11, 12],
    },
    index = [1, 2, 3])
```

Pandas DataFrame

type(df)

type(df)
pandas.core.frame.DataFrame
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
print('pandas imported')

s = pd.Series([1,3,5,np.nan,6,8])
s

dates = pd.date_range('20181001', periods=6)
dates
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
print('pandas imported.

s = pd.Series([1,3,5,np.nan,6,8]).
s
0   1.0
1   3.0
2   5.0
3   NaN
4   6.0
5   8.0
dtype: float64
dates = pd.date_range('20181001', periods=6)
dates
DatetimeIndex(['2018-10-01', '2018-10-02', '2018-10-03', '2018-10-04',
               '2018-10-05', '2018-10-06'],
dtype='datetime64[ns]', freq='D')
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
df
```python
df = pd.DataFrame(np.random.randn(3,5),
                 index=['student1','student2','student3'],
                 columns=list('ABCDE'))
df
```

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-0.346884</td>
<td>-1.232934</td>
<td>-0.302072</td>
<td>-1.345084</td>
<td>-0.723880</td>
</tr>
<tr>
<td>B</td>
<td>1.090955</td>
<td>-0.010483</td>
<td>1.280072</td>
<td>-0.253958</td>
<td>-0.030604</td>
</tr>
<tr>
<td>C</td>
<td>0.325660</td>
<td>0.808956</td>
<td>-0.395820</td>
<td>-1.498926</td>
<td>1.603471</td>
</tr>
</tbody>
</table>
df2 = pd.DataFrame({'A': 1., 'B': pd.Timestamp('20181001'), 'C': pd.Series(2.5, index=list(range(4)), dtype='float32'), 'D': np.array([3] * 4, dtype='int32'), 'E': pd.Categorical(['test', 'train', 'test', 'train']), 'F': 'foo'})
df2

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>test</td>
<td>foo</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>train</td>
<td>foo</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>test</td>
<td>foo</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>train</td>
<td>foo</td>
</tr>
</tbody>
</table>
```
df2.dtypes

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>float64</td>
</tr>
<tr>
<td>B</td>
<td>datetime64[ns]</td>
</tr>
<tr>
<td>C</td>
<td>float32</td>
</tr>
<tr>
<td>D</td>
<td>int32</td>
</tr>
<tr>
<td>E</td>
<td>category</td>
</tr>
<tr>
<td>F</td>
<td>object</td>
</tr>
</tbody>
</table>

dtype: object
```
Python Pandas for Finance
! pip install pandas_datareader

Collecting pandas_datareader
  Downloading https://files.pythonhosted.org/packages/cc/5c/ea5b6dcfd0f55c5fbb1e37fb45335ec01ccec0a11bb8a79339137f5ed269e0/pandas_datareader-0.8.1.tar.gz (112kB)  [100%]
Collecting lxml (from pandas_datareader)
  Downloading https://files.pythonhosted.org/packages/03/a4/9e0a8035fc7c7670e5eab97f34ff2ef0d8d78a491bf96df5ac63b8e63f5/lxml-4.2.5-cp31-cp31abi-winx64.whl (5.8MB)

Requirement already satisfied: pandas>=0.19.2 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (0.22.0)
Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (2.18.4)
Requirement already satisfied: wrapt in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (1.10.11)
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/python3.6/dist-packages (from pandas-datareader) (2.8.0)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from pandas-datareader) (1.1.14)
Requirement already satisfied: idna<2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (2.5.0)
Requirement already satisfied: charsetprep<=3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (3.1.1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (2017.4.17)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (1.21.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2->pandas-datareader) (1.12.0)
Installing collected packages: lxml, pandas-datareader
Successfully installed lxml-4.2.5 pandas-datareader-0.8.1
conda install pandas-datareader

[IMyday-MacBook-Pro:~ imyday$ conda install pandas-datareader
Fetching package metadata ..........  
Solving package specifications: .

Package plan for installation in environment /Users/imoyna/anaconda:

The following NEW packages will be INSTALLED:

    pandas-datareader: 0.2.1-py36_0
    requests-file: 1.4.1-py36_0

Proceed ([y]/n)? y

requests-file- 100% |#.#.#.#.#.#.#.#.#.#.#.#.#.#.#.#.#.| Time: 0:00:00  1.55 MB/s
pandas-datareader 100% |#.#.#.#.#.#.#.#.#.#.#.#.#.#.#.#.#.| Time: 0:00:00 409.66 kB/s

[IMyday-MacBook-Pro:~ imyday$ conda list
# packages in environment at /Users/imoyna/anaconda:
#
  _license  1.1  py36_1
  alabaster  0.7.9  py36_0
  anaconda  4.3.1  np111py36_0
  anaconda-client  1.6.0  py36_0
  anaconda-navigator  1.5.0  py36_0
  anaconda-project  0.4.1  py36_0
Finance Data from Yahoo Finance

# !pip install pandas_datareader
import pandas_datareader.data as web
import datetime as dt
#Read Stock Data from Yahoo Finance
end = dt.datetime(2017, 12, 31)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))

top = plt.subplot2grid((12,9), (0, 0),
                      rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0),
                          rowspan=2, colspan=9)

top.plot(df.index, df['Adj Close'],
         color='blue')  # df.index gives the dates
bottom.bar(df.index, df['Volume'])
# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
# simple moving averages

df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days

df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days

df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days

df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})

df2.plot(figsize=(12, 9), legend=True, title='AAPL')

df2.to_csv('AAPL_MA.csv')

fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)

plt.show()
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

# Read Stock Data from Yahoo Finance
end = dt.datetime.now()
# start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') # df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

# Read Stock Data from Yahoo Finance
end = dt.datetime.now()
# start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv("AAPL.csv")
df.from_csv("AAPL.csv")
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') # df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() # 5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() # 20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() # 60 days
df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```python
# ! pip install quandl
import quandl
# quandl.ApiConfig.api_key = "YOURAPIKEY"
df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-12-22</td>
<td>174.68</td>
<td>175.424</td>
<td>174.500</td>
<td>175.01</td>
<td>16052615.0</td>
<td>0.0</td>
<td>1.0</td>
<td>174.68</td>
<td>175.424</td>
<td>174.500</td>
<td>175.01</td>
<td>16052615.0</td>
</tr>
<tr>
<td>2017-12-26</td>
<td>170.80</td>
<td>171.470</td>
<td>169.679</td>
<td>170.57</td>
<td>32968167.0</td>
<td>0.0</td>
<td>1.0</td>
<td>170.80</td>
<td>171.470</td>
<td>169.679</td>
<td>170.57</td>
<td>32968167.0</td>
</tr>
<tr>
<td>2017-12-27</td>
<td>170.10</td>
<td>170.780</td>
<td>169.710</td>
<td>170.60</td>
<td>21672062.0</td>
<td>0.0</td>
<td>1.0</td>
<td>170.10</td>
<td>170.780</td>
<td>169.710</td>
<td>170.60</td>
<td>21672062.0</td>
</tr>
<tr>
<td>2017-12-28</td>
<td>171.00</td>
<td>171.850</td>
<td>170.480</td>
<td>171.08</td>
<td>15997739.0</td>
<td>0.0</td>
<td>1.0</td>
<td>171.00</td>
<td>171.850</td>
<td>170.480</td>
<td>171.08</td>
<td>15997739.0</td>
</tr>
<tr>
<td>2017-12-29</td>
<td>170.52</td>
<td>170.590</td>
<td>169.220</td>
<td>169.23</td>
<td>25643711.0</td>
<td>0.0</td>
<td>1.0</td>
<td>170.52</td>
<td>170.590</td>
<td>169.220</td>
<td>169.23</td>
<td>25643711.0</td>
</tr>
</tbody>
</table>

Source: https://www.quandl.com/tools/python
Scikit-Learn

scikit-learn
Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: SVM, nearest neighbors, random forest, ...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, ridge regression, Lasso, ...

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, ...

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: PCA, feature selection, non-negative matrix factorization.

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: grid search, cross validation, metrics.

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

Modules: preprocessing, feature extraction.

Source: http://scikit-learn.org/
Iris flower data set

setosa  versicolor  virginica

Source: https://en.wikipedia.org/wiki/Iris_flower_data_set
Iris Classification

<table>
<thead>
<tr>
<th>Sepal length</th>
<th>Sepal width</th>
<th>Petal length</th>
<th>Petal width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.9</td>
<td>1.7</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.4</td>
<td>1.4</td>
<td>0.3</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.4</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.4</td>
<td>2.9</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.1</td>
<td>1.5</td>
<td>0.1</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.7</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.8</td>
<td>3.4</td>
<td>1.6</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.8</td>
<td>3.0</td>
<td>1.4</td>
<td>0.1</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.3</td>
<td>3.0</td>
<td>1.1</td>
<td>0.1</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.8</td>
<td>4.0</td>
<td>1.2</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.7</td>
<td>4.4</td>
<td>1.5</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.9</td>
<td>1.3</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.3</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.7</td>
<td>3.8</td>
<td>1.7</td>
<td>0.3</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.1</td>
<td>3.8</td>
<td>1.5</td>
<td>0.3</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.4</td>
<td>1.7</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.1</td>
<td>3.7</td>
<td>1.5</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.6</td>
<td>1.0</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.1</td>
<td>3.3</td>
<td>1.7</td>
<td>0.5</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.8</td>
<td>3.4</td>
<td>1.9</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.0</td>
<td>1.6</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.4</td>
<td>1.6</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
</tbody>
</table>

Iris Data Visualization

Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html
Data Visualization in Google Colab

```python
import seaborn as sns
iris = sns.load_dataset("iris")
g = sns.pairplot(iris, hue="species")
```
```python
import seaborn as sns
sns.set(style="ticks", color_codes=True)
iris = sns.load_dataset("iris")
g = sns.pairplot(iris, hue="species")
```

Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

# Load dataset
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

print(df.head(10))
print(df.tail(10))
print(df.describe())
print(df.info())
print(df.shape)
print(df.groupby('class').size())

plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()

df.hist()
plt.show()

scatter_matrix(df)
plt.show()
	sns.pairplot(df, hue="class", size=2)
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

df = pd.read_csv(url, names=names)
print(df.head(10))
```python
print(df.describe())
```

<table>
<thead>
<tr>
<th></th>
<th>sepal-length</th>
<th>sepal-width</th>
<th>petal-length</th>
<th>petal-width</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>150.000000</td>
<td>150.000000</td>
<td>150.000000</td>
<td>150.000000</td>
</tr>
<tr>
<td>mean</td>
<td>5.843333</td>
<td>3.054000</td>
<td>3.758667</td>
<td>1.198667</td>
</tr>
<tr>
<td>std</td>
<td>0.828066</td>
<td>0.433594</td>
<td>1.764420</td>
<td>0.763161</td>
</tr>
<tr>
<td>min</td>
<td>4.300000</td>
<td>2.000000</td>
<td>1.000000</td>
<td>0.100000</td>
</tr>
<tr>
<td>25%</td>
<td>5.100000</td>
<td>2.800000</td>
<td>1.600000</td>
<td>0.300000</td>
</tr>
<tr>
<td>50%</td>
<td>5.800000</td>
<td>3.000000</td>
<td>4.350000</td>
<td>1.300000</td>
</tr>
<tr>
<td>75%</td>
<td>6.400000</td>
<td>3.300000</td>
<td>5.100000</td>
<td>1.800000</td>
</tr>
<tr>
<td>max</td>
<td>7.900000</td>
<td>4.400000</td>
<td>6.900000</td>
<td>2.500000</td>
</tr>
</tbody>
</table>
```
print(df.tail(10)).
```

<table>
<thead>
<tr>
<th>sepal-length</th>
<th>sepal-width</th>
<th>petal-length</th>
<th>petal-width</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>140</td>
<td>6.7</td>
<td>3.1</td>
<td>5.6</td>
<td>2.4 Iris-virginica</td>
</tr>
<tr>
<td>141</td>
<td>6.9</td>
<td>3.1</td>
<td>5.1</td>
<td>2.3 Iris-virginica</td>
</tr>
<tr>
<td>142</td>
<td>5.8</td>
<td>2.7</td>
<td>5.1</td>
<td>1.9 Iris-virginica</td>
</tr>
<tr>
<td>143</td>
<td>6.8</td>
<td>3.2</td>
<td>5.9</td>
<td>2.3 Iris-virginica</td>
</tr>
<tr>
<td>144</td>
<td>6.7</td>
<td>3.3</td>
<td>5.7</td>
<td>2.5 Iris-virginica</td>
</tr>
<tr>
<td>145</td>
<td>6.7</td>
<td>3.0</td>
<td>5.2</td>
<td>2.3 Iris-virginica</td>
</tr>
<tr>
<td>146</td>
<td>6.3</td>
<td>2.5</td>
<td>5.0</td>
<td>1.9 Iris-virginica</td>
</tr>
<tr>
<td>147</td>
<td>6.5</td>
<td>3.0</td>
<td>5.2</td>
<td>2.0 Iris-virginica</td>
</tr>
<tr>
<td>148</td>
<td>6.2</td>
<td>3.4</td>
<td>5.4</td>
<td>2.3 Iris-virginica</td>
</tr>
<tr>
<td>149</td>
<td>5.9</td>
<td>3.0</td>
<td>5.1</td>
<td>1.8 Iris-virginica</td>
</tr>
</tbody>
</table>
print(df.info())
print(df.shape)

print(df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal-length    150 non-null float64
sepal-width     150 non-null float64
petal-length    150 non-null float64
petal-width     150 non-null float64
class           150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
None

print(df.shape)

(150, 5)
df.groupby('class').size()

print(df.groupby('class').size())

class
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
dtype: int64
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
df.hist()
plt.show()
scatter_matrix(df)
plt.show()
sns.pairplot(df, hue="class", size=2)
Classification and Prediction

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw

Data Mining and Machine Learning in Google Colab

```python
# Import libraries
import numpy as np
import pandas as pd
import matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

# Import sklearn
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier

print("Imported")

# Load dataset
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

print(df.head(10))
print(df.tail(10))
print(df.describe())
print(df.info())
print(df.shape)
print(df.groupby('class').size())

plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
df.hist()
plt.show()
```
# Load dataset
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

print(df.head(10))
print(df.tail(10))
print(df.describe())
print(df.info())
print(df.shape)
print(df.groupby('class').size())

plt.rcParams["figure.figsize"] = (10, 8)
df.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)
plt.show()

df.hist()
plt.show()

scatter_matrix(df)
plt.show()

sns.pairplot(df, hue="class", size=2).

<table>
<thead>
<tr>
<th>sepal-length</th>
<th>sepal-width</th>
<th>petal-length</th>
<th>petal-width</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.9</td>
<td>1.7</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.4</td>
<td>1.4</td>
<td>0.3</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.4</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.4</td>
<td>2.9</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.1</td>
<td>1.5</td>
<td>0.1</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>6.7</td>
<td>3.1</td>
<td>5.6</td>
<td>2.4</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>6.9</td>
<td>3.1</td>
<td>5.1</td>
<td>2.3</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>5.8</td>
<td>2.7</td>
<td>5.1</td>
<td>1.9</td>
<td>Iris-virginica</td>
</tr>
</tbody>
</table>
# Load dataset
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

print(df.head(10))
print(df.tail(10))
print(df.describe())
print(df.info())
print(df.shape)
print(df.groupby('class').size())

plt.rcParams["figure.figsize"] = (10, 8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()

df.hist()
plt.show()

scatter_matrix(df)
plt.show()

sns.pairplot(df, hue="class", size=2).
```python
import pandas as pd

df = pd.read_csv('iris.csv')

# Calculate correlation matrix
df_corr = df.corr()

# Display correlation matrix
df_corr
```

<table>
<thead>
<tr>
<th></th>
<th>sepal-length</th>
<th>sepal-width</th>
<th>petal-length</th>
<th>petal-width</th>
</tr>
</thead>
<tbody>
<tr>
<td>sepal-length</td>
<td>1.000000</td>
<td>-0.109369</td>
<td>0.871754</td>
<td>0.817954</td>
</tr>
<tr>
<td>sepal-width</td>
<td>-0.109369</td>
<td>1.000000</td>
<td>-0.420516</td>
<td>-0.356544</td>
</tr>
<tr>
<td>petal-length</td>
<td>0.871754</td>
<td>-0.420516</td>
<td>1.000000</td>
<td>0.962757</td>
</tr>
<tr>
<td>petal-width</td>
<td>0.817954</td>
<td>-0.356544</td>
<td>0.962757</td>
<td>1.000000</td>
</tr>
</tbody>
</table>

Source: [Colab Notebook](https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw)
# Split-out validation dataset
array = df.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = 
model_selection.train_test_split(X, Y, 
test_size=validation_size, random_state=seed)
scoring = 'accuracy'

len(Y_validation).
# Models
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10,
random_state=seed)
    cv_results =
model_selection.cross_val_score(model,
X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "\%s: \%.4f (\%.4f)" " % (name,
    cv_results.mean(), cv_results.std())
    print(msg)
# Models
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))

# evaluate each model in turn
results = []
names = []
for name, model in models:
kfold = model_selection.KFold(n_splits=10, random_state=seed)
cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
results.append(cv_results)
names.append(name)
msg = "%s: %.4f (\%.4f)" % (name, cv_results.mean(), cv_results.std())
print(msg)

LR: 0.9667 (0.0408)
LDA: 0.9750 (0.0382)
KNN: 0.9833 (0.0333)
DT: 0.9750 (0.0382)
NB: 0.9750 (0.0534)
SVM: 0.9917 (0.0250)
# Make predictions on validation dataset
model = KNeighborsClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
# Make predictions on validation dataset
model = KNeighborsClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)

0.9000
[[ 7  0  0]
 [ 0 11  1]
 [ 0  2  9]]

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris-setosa</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>7</td>
</tr>
<tr>
<td>Iris-versicolor</td>
<td>0.85</td>
<td>0.92</td>
<td>0.88</td>
<td>12</td>
</tr>
<tr>
<td>Iris-virginica</td>
<td>0.90</td>
<td>0.82</td>
<td>0.86</td>
<td>11</td>
</tr>
<tr>
<td>avg / total</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
<td>30</td>
</tr>
</tbody>
</table>

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=1, n_neighbors=5, p=2,
weights='uniform')

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6I1nnZDlFF354Nf_Lw
# Make predictions on validation dataset
model = SVC()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
model = SVC()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

# Make predictions on validation dataset
model = SVC()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)

0.9333
[[ 7  0  0]
 [ 0 10  2]
 [ 0  0 11]]

precision   recall   f1-score   support

Iris-setosa  1.00      1.00      1.00      7
Iris-versicolor 1.00      0.83      0.91      12
Iris-virginica 0.85      1.00      0.92      11

avg / total  0.94      0.93      0.93      30

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)
```python
# Make predictions on validation dataset
model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

0.9000

```
[[ 7  0  0]
 [ 0 11  1]
 [ 0  2  9]]
```

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris-setosa</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>7</td>
</tr>
<tr>
<td>Iris-versicolor</td>
<td>0.85</td>
<td>0.92</td>
<td>0.88</td>
<td>12</td>
</tr>
<tr>
<td>Iris-virginica</td>
<td>0.90</td>
<td>0.82</td>
<td>0.86</td>
<td>11</td>
</tr>
</tbody>
</table>

avg / total    | 0.90      | 0.90   | 0.90     | 30      |

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=None, splitter='best')
# Make predictions on validation dataset
model = GaussianNB()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)

0.8333
[[7 0 0]
 [0 9 3]
 [0 2 9]]

<table>
<thead>
<tr>
<th>class</th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris-setosa</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>7</td>
</tr>
<tr>
<td>Iris-versicolor</td>
<td>0.82</td>
<td>0.75</td>
<td>0.78</td>
<td>12</td>
</tr>
<tr>
<td>Iris-virginica</td>
<td>0.75</td>
<td>0.82</td>
<td>0.78</td>
<td>11</td>
</tr>
<tr>
<td>avg / total</td>
<td>0.84</td>
<td>0.83</td>
<td>0.83</td>
<td>30</td>
</tr>
</tbody>
</table>

GaussianNB(priors=None)
# Make predictions on validation dataset

```python
model = LogisticRegression()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.8000
[[ 7  0  0]
 [ 0  7  5]
 [ 0 11  0]]

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris-setosa</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>7</td>
</tr>
<tr>
<td>Iris-versicolor</td>
<td>0.88</td>
<td>0.58</td>
<td>0.70</td>
<td>12</td>
</tr>
<tr>
<td>Iris-virginica</td>
<td>0.67</td>
<td>0.91</td>
<td>0.77</td>
<td>11</td>
</tr>
</tbody>
</table>

avg / total    | 0.83      | 0.80   | 0.80     | 30      |
```

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1, penalty='l2', random_state=None, solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
# Make predictions on validation dataset
```
model = LinearDiscriminantAnalysis()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```python
0.9667
[[ 7  0  0]
 [ 0 11  1]
 [ 0  0 11]]
```

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris-setosa</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>7</td>
</tr>
<tr>
<td>Iris-versicolor</td>
<td>1.00</td>
<td>0.92</td>
<td>0.96</td>
<td>12</td>
</tr>
<tr>
<td>Iris-virginica</td>
<td>0.92</td>
<td>1.00</td>
<td>0.96</td>
<td>11</td>
</tr>
</tbody>
</table>

| avg / total      | 0.97      | 0.97   | 0.97     | 30      |

LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None, solver='svd', store_covariance=False, tol=0.0001)
# Make predictions on validation dataset

```python
model = MLPClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model).
```

0.9000

```
[[ 7  0  0]
 [ 0  9  3]
 [ 0  0 11]]
```

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris-setosa</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>7</td>
</tr>
<tr>
<td>Iris-versicolor</td>
<td>1.00</td>
<td>0.75</td>
<td>0.86</td>
<td>12</td>
</tr>
<tr>
<td>Iris-virginica</td>
<td>0.79</td>
<td>1.00</td>
<td>0.88</td>
<td>11</td>
</tr>
<tr>
<td>avg / total</td>
<td>0.92</td>
<td>0.90</td>
<td>0.90</td>
<td>30</td>
</tr>
</tbody>
</table>

MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(100,), learning_rate='constant', learning_rate_init=0.001, max_iter=200, momentum=0.9, nesterovs_momentum=True, power_t=0.5, random_state=None, shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False)
Summary

• Data Mining Concepts and Applications
• Data Mining Applications
• Data Mining Process
• Data Mining Methods
• Data Mining Software Tools
• Data Mining Privacy Issues, Myths, and Blunders
References

• Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson.