Big Data Mining

Machine Learning with Scikit-Learn in Python

1071BDM09
TLVXM1A (M2244) (8619) (Fall 2018)
(MBA, DBETKU) (3 Credits, Required) [Full English Course]
(Master’s Program in Digital Business and Economics)
Mon, 9, 10, 11, (16:10-19:00) (B206)

Min-Yuh Day, Ph.D.
Assistant Professor
Department of Information Management
Tamkang University

http://mail.tku.edu.tw/myday

2018-11-19
## Course Schedule (1/2)

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2018/09/10</td>
<td>Course Orientation for Big Data Mining</td>
</tr>
<tr>
<td>2</td>
<td>2018/09/17</td>
<td>ABC: AI, Big Data, Cloud Computing</td>
</tr>
<tr>
<td>3</td>
<td>2018/09/24</td>
<td>Mid-Autumn Festival (Day off)</td>
</tr>
<tr>
<td>4</td>
<td>2018/10/01</td>
<td>Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data</td>
</tr>
<tr>
<td>5</td>
<td>2018/10/08</td>
<td>Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem</td>
</tr>
<tr>
<td>6</td>
<td>2018/10/15</td>
<td>Foundations of Big Data Mining in Python</td>
</tr>
<tr>
<td>7</td>
<td>2018/10/22</td>
<td>Supervised Learning: Classification and Prediction</td>
</tr>
<tr>
<td>8</td>
<td>2018/10/29</td>
<td>Unsupervised Learning: Cluster Analysis</td>
</tr>
<tr>
<td>9</td>
<td>2018/11/05</td>
<td>Unsupervised Learning: Association Analysis</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Subject/Topics</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>2018/11/12</td>
<td>Midterm Project Report</td>
</tr>
<tr>
<td>11</td>
<td>2018/11/19</td>
<td>Machine Learning with Scikit-Learn in Python</td>
</tr>
<tr>
<td>12</td>
<td>2018/11/26</td>
<td>Deep Learning for Finance Big Data with TensorFlow</td>
</tr>
<tr>
<td>13</td>
<td>2018/12/03</td>
<td>Convolutional Neural Networks (CNN)</td>
</tr>
<tr>
<td>14</td>
<td>2018/12/10</td>
<td>Recurrent Neural Networks (RNN)</td>
</tr>
<tr>
<td>15</td>
<td>2018/12/17</td>
<td>Reinforcement Learning (RL)</td>
</tr>
<tr>
<td>16</td>
<td>2018/12/24</td>
<td>Social Network Analysis (SNA)</td>
</tr>
<tr>
<td>17</td>
<td>2018/12/31</td>
<td>Bridge Holiday (Extra Day Off)</td>
</tr>
<tr>
<td>18</td>
<td>2019/01/07</td>
<td>Final Project Presentation</td>
</tr>
</tbody>
</table>
Machine Learning with Scikit-Learn in Python
Outline

• Machine Learning with Scikit-Learn in Python

  – Machine Learning
  – Scikit-Learn
Artificial Intelligence (A.I.) Timeline

1950
TURING TEST
Computer scientist Alan Turing proposes a test for machine intelligence. If a machine can trick humans into thinking it is human, then it has intelligence.

1955
A.I. BORN
Term ‘artificial intelligence’ is coined by computer scientist John McCarthy to describe “the science and engineering of making intelligent machines.”

1961
UNIMATE
First industrial robot, Unimate, goes to work at GM replacing humans on the assembly line.

1964
ELIZA
Pioneering chatbot developed by Joseph Weizenbaum at MIT holds conversations with humans.

1966
SHAKEY
The ‘first electronic person’ from Stanford, Shakey is a general-purpose mobile robot that reasons about its own actions.

1997
DEEP BLUE
Deep Blue, a chess-playing computer from IBM defeats world chess champion Garry Kasparov.

1998
KISMET
Cynthia Breazeal at MIT introduces Kismet, an emotionally intelligent robot insofar as it detects and responds to people’s feelings.

1999
AIBO
Sony launches first consumer robot pet dog AIBO (A) robot with skills and personality that develop over time.

2002
ROOMBA
First mass produced autonomous robotic vacuum cleaner from iRobot learns to navigate and clean homes.

2011
SIRI
Apple integrates Siri, an intelligent virtual assistant with a voice interface, into the iPhone 4S.

2011
WATSON
IBM’s question answering computer Watson wins first place on popular $1M prize television quiz show Jeopardy.

2014
EUGENE
Eugene Goostman, a chatbot passes the Turing Test with a third of judges believing Eugene is human.

2014
ALEXA
Amazon launches Alexa, an intelligent virtual assistant with a voice interface that completes shopping tasks.

2016
TAY
Microsoft’s chatbot Tay goes rogue on social media making inflammatory and offensive racist comments.

2017
ALPHAGO
Google’s A.I. AlphaGo beats world champion Ke Jie in the complex board game of Go, notable for its vast number (2^170) of possible positions.

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Artificial Intelligence (AI)

Machine Learning (ML)

Supervised Learning

Unsupervised Learning

Deep Learning (DL)

CNN

RNN LSTM GRU

GAN

Semi-supervised Learning

Reinforcement Learning

Source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep_learning.html
3 Machine Learning Algorithms

Machine Learning Models

- Deep Learning
- Association rules
- Decision tree
- Clustering
- Bayesian
- Kernel
- Ensemble
- Dimensionality reduction
- Regression Analysis
- Instance based

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Machine Learning (ML) / Deep Learning (DL)

- Supervised Learning
  - Decision Tree Classifiers
  - Linear Classifiers
  - Rule-based Classifiers
  - Probabilistic Classifiers
  - Support Vector Machine (SVM)
  - Neural Network (NN)
  - Deep Learning (DL)
  - Naïve Bayes (NB)
  - Bayesian Network (BN)
  - Maximum Entropy (ME)

- Unsupervised Learning

- Reinforcement Learning

## 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><em>k</em>-means, Expectation Maximization (EM)</td>
<td>Unsupervised</td>
</tr>
<tr>
<td>Outlier analysis</td>
<td><em>k</em>-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 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>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>True Positive Count (TP)</td>
<td>False Positive Count (FP)</td>
</tr>
<tr>
<td>Negative</td>
<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

Precision

Validity

Reliability
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 Positive (TP)</th>
<th>False Positive (FP)</th>
<th>False Negative (FN)</th>
<th>True Negative (TN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Negative</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Total**

- **P’**
- **N’**

<table>
<thead>
<tr>
<th>total</th>
<th>P</th>
<th>N</th>
</tr>
</thead>
</table>

**Accuracy**

\[
Accuracy = \frac{TP + TN}{TP + TN + FP + FN}
\]

**True Positive Rate**

\[
True \ Positive \ Rate = \frac{TP}{TP + FN}
\]

**True Negative Rate**

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

**Precision**

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

**Recall**

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

---

**Source:** [http://en.wikipedia.org/wiki/Receiver_operating_characteristic](http://en.wikipedia.org/wiki/Receiver_operating_characteristic)
True Positive Rate (Sensitivity) = \( \frac{TP}{TP + FN} \)

**Sensitivity**

= True Positive Rate

= Recall

= Hit rate

= \( \frac{TP}{TP + FN} \)
**True Class (actual value)**

<table>
<thead>
<tr>
<th>Predictive Class</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>Negative</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

**Total**

<table>
<thead>
<tr>
<th>P</th>
<th>N</th>
</tr>
</thead>
</table>

**Specificity**

= True Negative Rate

= \( \frac{TN}{N} \)

= \( \frac{TN}{TN + FP} \)

**True Negative Rate (Specificity)**

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

**False Positive Rate (1 - Specificity)**

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

**True Class (actual value)**

<table>
<thead>
<tr>
<th></th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Positive</strong></td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td><strong>Negative</strong></td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

**Predictive Class (prediction outcome)**

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

**Total**

- **P**
- **N**
- **P’**
- **N’**
- **P + P’**
- **N + N’**

**Precision**

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

**Recall**

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

**F1 score (F-score)(F-measure)**

is the harmonic mean of precision and recall

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

**Source:** [http://en.wikipedia.org/wiki/Receiver_operating_characteristic](http://en.wikipedia.org/wiki/Receiver_operating_characteristic)
Recall = True Positive Rate (TPR) = Sensitivity = Hit Rate = TP / (TP + FN)

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

Accuracy = \frac{TP + TN}{TP + TN + FP + FN}

Precision = \frac{TP}{TP + FP}

F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}

TPR = 0.63
FPR = 0.28
PPV = 0.69
F1 = 0.66
ACC = 0.68

FP
TP
TP
Recall
Recall
Precision
Precision
F1 score (F-score) (F-measure) is the harmonic mean of precision and recall

Accuracy = \frac{TP + TN}{TP + TN + FP + FN}

Precision = \frac{TP}{TP + FP}

F1 = 2 \cdot \frac{0.63 \cdot 0.69}{0.63 + 0.69}

= (2 \cdot 63) / (100 + 91)

= (0.63 + 0.69) / 2 = 1.32 / 2 = 0.66

ACC = \frac{63 + 72}{200} = 135/200 = 67.5

**Recall**

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

**Precision**

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

**A**

\[
\begin{array}{ccc}
63 & 28 & 91 \\
37 & 72 & 109 \\
100 & 100 & 200
\end{array}
\]

- **TPR** = 0.63
- **FPR** = 0.28
- **PPV** = 0.69
- **F1** = 0.66
- **ACC** = 0.68

\[
= \frac{63}{63 + 28} = \frac{63}{91}
\]

\[
F1 = 2 \times \frac{0.63 \times 0.69}{0.63 + 0.69} = \frac{2 \times 63}{100 + 91} = \frac{63 + 72}{200} = \frac{135}{200} = 67.5
\]

**B**

\[
\begin{array}{ccc}
77 & 77 & 154 \\
23 & 23 & 46 \\
100 & 100 & 200
\end{array}
\]

- **TPR** = 0.77
- **FPR** = 0.77
- **PPV** = 0.50
- **F1** = 0.61
- **ACC** = 0.50

\[
Recall = \frac{77}{77 + 23} = \frac{77}{100}
\]

\[
Precision = \frac{77}{77 + 23} = \frac{77}{100}
\]

The image contains a table and a document that explains the concepts of recall, precision, and accuracy in the context of the Receiver Operating Characteristic (ROC) curve. The table includes counts and calculations for two different scenarios labeled as C and C'.

For scenario C:
- True Positive (TP): 24
- False Positive (FP): 76
- False Negative (FN): 12
- True Negative (TN): 88
- Total: 112
- True Positive Rate (TPR) = 0.24
- False Positive Rate (FPR) = 0.88
- Positive Predictive Value (PPV) = 0.21
- F1 Score = 0.22
- Accuracy (ACC) = 0.18

For scenario C':
- True Positive (TP): 76
- False Positive (FP): 24
- False Negative (FN): 12
- True Negative (TN): 88
- Total: 112
- True Positive Rate (TPR) = 0.76
- False Positive Rate (FPR) = 0.12
- Positive Predictive Value (PPV) = 0.86
- F1 Score = 0.81
- Accuracy (ACC) = 0.82

The formulas for recall and precision are:

- Recall = \( \frac{TP}{TP + FN} \)
- Precision = \( \frac{TP}{TP + FP} \)

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
Scikit-Learn
Machine Learning in 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/
Scikit-Learn Machine Learning Map

Scikit-Learn Machine Learning Map

Iris flower data set

setosa  versicolor  virginica

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


5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
5.0,3.4,1.6,0.4,Iris-setosa

setosa
virginica
versicolor
Iris Data Visualization

Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html
import seaborn as sns
iris = sns.load_dataset("iris")
g = sns.pairplot(iris, hue="species")

Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html
import seaborn as sns
sns.set(style="ticks", color_codes=True)
iris = sns.load_dataset("iris")
g = sns.pairplot(iris, hue="species")
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

# Import Libraries
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
print('imported')

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

import pandas as pd

df = pd.read_csv(url, names=names)

print(df.head(10))

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

df = pd.read_csv(url, names=names)
print(df.head(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>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>
</tbody>
</table>
```python
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>
```
```python
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>Iris-virginica</td>
</tr>
<tr>
<td>141</td>
<td>6.9</td>
<td>3.1</td>
<td>5.1</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>142</td>
<td>5.8</td>
<td>2.7</td>
<td>5.1</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>143</td>
<td>6.8</td>
<td>3.2</td>
<td>5.9</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>144</td>
<td>6.7</td>
<td>3.3</td>
<td>5.7</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>145</td>
<td>6.7</td>
<td>3.0</td>
<td>5.2</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>146</td>
<td>6.3</td>
<td>2.5</td>
<td>5.0</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>147</td>
<td>6.5</td>
<td>3.0</td>
<td>5.2</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>148</td>
<td>6.2</td>
<td>3.4</td>
<td>5.4</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>149</td>
<td>5.9</td>
<td>3.0</td>
<td>5.1</td>
<td>Iris-virginica</td>
</tr>
</tbody>
</table>
`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)
Machine Learning
Supervised Learning
Classification and Prediction
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.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()
```

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw
# 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()
scatter_matrix(df)
plt.show()

tsns.pairplot(df, hue="class", size=2).
# 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>
</tbody>
</table>

<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>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>
```python
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>

[Link to Google Colab](https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6I1nnZDlFF354Nf_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)
scorings = '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/1QE7fR2OxHiQO_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)
# 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,
mix_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')

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw
# 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]]
precision   recall  f1-score  support
Iris-setosa  1.00   1.00   1.00   7
Iris-versicolor 0.82   0.75   0.78   12
Iris-virginica 0.75   0.82   0.78   11
avg / total  0.84   0.83   0.83   30

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

```python
def make_predictions_on_validation_dataset(model, X_train, Y_train, X_validation):
m = LogisticRegression()
m.fit(X_train, Y_train)
predictions = m.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]]

precision    recall    f1-score    support

Iris-setosa    1.00    1.00    1.00      7
Iris-versicolor 0.88    0.58    0.70     12
Iris-virginica  0.67    0.91    0.77     11

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)
```

https://colab.research.google.com/drive/1QE7fR2OxHiQO_p6l1nnZDlFF354Nf_Lw
```python
# 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)

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>
<tr>
<td>avg / total</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>30</td>
</tr>
</tbody>
</table>

LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None, solver='svd', store_covariance=False, tol=0.0001)
```

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDlFF354Nf_Lw
# Make predictions on validation dataset
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]]

                precision  recall  f1-score  support

Iris-setosa       1.00    1.00    1.00        7
Iris-versicolor   1.00    0.75    0.86       12
Iris-virginica    0.79    1.00    0.88       11

avg / total     0.92    0.90    0.90       30

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)
Machine Learning
Unsupervised Learning
Cluster Analysis
K-Means Clustering
K-Means Clustering

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

```python
# importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

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

array = df.values
X = array[:,0:4]
Y = array[:,4]

# Finding the optimum number of clusters for k-means classification
from sklearn.cluster import KMeans
wcss = []

for i in range(1, 8):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Plotting the results onto a line graph, allowing us to observe 'The elbow'
plt.rcParams["figure.figsize"] = (10,8)
plt.plot(range(1, 8), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') # within cluster sum of squares
plt.show()
```

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw
#importing the libraries
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd

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

array = df.values
X = array[:,0:4]
Y = array[:,4]
# Finding the optimum number of clusters for k-means classification

```python
from sklearn.cluster import KMeans

wcss = []

for i in range(1, 8):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Plotting the results onto a line graph, allowing us to observe 'The elbow'
plt.rcParams["figure.figsize"] = (10,8)
plt.plot(range(1, 8), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') # within cluster sum of squares
plt.show()
```

https://colab.research.google.com/drive/1QE7fR20xHiQ0_p6l1nnZDlFF354Nf_Lw
K-Means Clustering
The elbow method ($k=3$)
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(X)

#Applying kmeans to the dataset / Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(X).
# Visualising the clusters

```python
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')

# Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100, c = 'yellow', label = 'Centroids')
plt.legend()
```

# Plotting the centroids of the clusters

```python
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100, c = 'yellow', label = 'Centroids')
plt.legend()
```
# Applying kmeans to the dataset / Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(X).

# Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Iris-versicolor')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')

# Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = 'yellow', label = 'Centroids')
plt.legend()
# !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(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')
bottom.bar(df.index, df['Volume'])

# set the labels
top_axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_xlabel('Adj Close')
top.set_ylabel('Volume')
sns.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
df.plot(figsize=(12, 9), legend=True, title='AAPL')
df.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
Python Data Science Handbook in Google Colab

[Image of Python Data Science Handbook]

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb
Table of Contents

Preface

1. IPython: Beyond Normal Python
   • Help and Documentation in IPython
   • Keyboard Shortcuts in the IPython Shell
   • IPython Magic Commands
   • Input and Output History
   • IPython and Shell Commands
   • Errors and Debugging
   • Profiling and Timing Code
   • More IPython Resources

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb
2. Introduction to NumPy

- Understanding Data Types in Python
- The Basics of NumPy Arrays
- Computation on NumPy Arrays: Universal Functions
- Aggregations: Min, Max, and Everything In Between
- Computation on Arrays: Broadcasting
- Comparisons, Masks, and Boolean Logic
- Fancy Indexing
- Sorting Arrays
- Structured Data: NumPy's Structured Arrays

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb
3. Data Manipulation with Pandas
   • Introducing Pandas Objects
   • Data Indexing and Selection
   • Operating on Data in Pandas
   • Handling Missing Data
   • Hierarchical Indexing
   • Combining Datasets: Concat and Append
   • Combining Datasets: Merge and Join
   • Aggregation and Grouping
   • Pivot Tables
   • Vectorized String Operations
   • Working with Time Series
   • High-Performance Pandas: eval() and query()
   • Further Resources

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb
4. Visualization with Matplotlib

- Simple Line Plots
- Simple Scatter Plots
- Visualizing Errors
- Density and Contour Plots
- Histograms, Binnings, and Density
- Customizing Plot Legends
- Customizing Colorbars
- Multiple Subplots
- Text and Annotation
- Customizing Ticks
- Customizing Matplotlib: Configurations and Stylesheets
- Three-Dimensional Plotting in Matplotlib
- Geographic Data with Basemap
- Visualization with Seaborn
- Further Resources
5. Machine Learning

- What Is Machine Learning?
- Introducing Scikit-Learn
- Hyperparameters and Model Validation
- Feature Engineering
- In Depth: Naive Bayes Classification
- In Depth: Linear Regression
- In-Depth: Support Vector Machines
- In-Depth: Decision Trees and Random Forests
- In Depth: Principal Component Analysis
- In-Depth: Manifold Learning
- In Depth: k-Means Clustering
- In Depth: Gaussian Mixture Models
- In-Depth: Kernel Density Estimation
- Application: A Face Detection Pipeline
- Further Machine Learning Resources

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb
AI + VDI

POC
AI + VDI POS

TensorFlow Models

• M1: Basic Classification (Image Classification) (65 Seconds)

• M2: Basic Text Classification (Text Classification) (46 Seconds)

• M3: Basic Regression (Predict House Prices) (43 Seconds)

• M4: Pix2Pix Eager (Option) (7-8 Hours)

• M5: NMT with Attention (Option) (20-30 minutes)
Basic Classification

Fashion MNIST Image Classification

[Link to Google Colab Notebook](https://colab.research.google.com/drive/19PJOJi1vn1kjcvtlzNHjRSLbeVl4kd5z)

---

**Train your first neural network: basic classification**

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details, this is a fast-paced overview of a complete TensorFlow program with the details explained as we go.

This guide uses `tf.keras`, a high-level API to build and train models in TensorFlow.

```python
# memory footprint support libraries/code
1 import os
2
3 # install cudnn
4 !pip install cudnn
5 # install psutil
6 !pip install psutil
7 # install humanize
8
9 import psutil
10 import humanize
11
12 def printm():
13     process = psutil.Process(os.getpid())
14     print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Pro"
15     print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:.0f}% | Total {3:.0f}MB".format
16     printm()
```
Text Classification
IMDB Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsLIrLYtPCvCHaoO1W-i_gror

Copyright 2018 The TensorFlow Authors.
Licensed under the Apache License, Version 2.0 (the "License");
MIT License

Text classification with movie reviews
Download the IMDB dataset
Explore the data
Convert the integers back to words
Prepare the data
Build the model
Hidden units
Loss function and optimizer
Create a validation set
Train the model
Evaluate the model

This notebook classifies movie reviews as positive or negative using the text of the review. This is an example of binary—or two-class—classification, an important and widely applicable kind of machine learning problem.

We'll use the IMDB dataset that contains the text of 50,000 movie reviews from the Internet Movie Database. These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are balanced, meaning they contain an equal number of positive and negative reviews.

This notebook uses tf.keras, a high-level API to build and train models in TensorFlow. For a more advanced text classification tutorial using tf.keras, see the MLCC Text Classification Guide.

```python
# memory footprint support libraries/code
!ln -sI /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
tlutil install psutil
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printt():
    process = psutil.Process(os.getpid())
```
Basic Regression
Predict House Prices

https://colab.research.google.com/drive/1v4c8ZHTnRtgd2_25K_AURjR6SCVBRdlj

Copyright 2018 The TensorFlow Authors.

Predict house prices: regression

The Boston Housing Prices dataset

Examples and features

Labels

Normalize features

Create the model

Train the model

Predict

Conclusion

Predict house prices: regression

In a regression problem, we aim to predict the output of a continuous value, like a price or a probability. Contrast this with a classification problem, where we aim to predict a discrete label (for example, where a picture contains an apple or an orange).

This notebook builds a model to predict the median price of homes in a Boston suburb during the mid-1970s. To do this, we'll provide the model with some data points about the suburb, such as the crime rate and the local property tax rate.

This example uses the tf.keras API, see this guide for details.
AI+VDI POC
ISAC+TKU Test

• AI+VDI POC Folder (3+1 ipynb) (v3.0.20181120)
  – https://drive.google.com/open?id=1qHOemktbEmUz-ot8eFxIkbgWjvXlrjtc

• run3models.ipynb
  – https://colab.research.google.com/drive/1HQ1GrIqQUUPCct7_AVgoMwMrh0UqMm0f

• AI+VDI POC ISAC+TKU Test Report (Example)
  – https://docs.google.com/spreadsheets/d/1meMwqn15PSuTk6d5TgendDpdDX6L3OfHM4E0Slkq1Zk/edit?usp=sharing
Summary

• Machine Learning with Scikit-Learn in Python
  – Machine Learning
  – Scikit-Learn
References

• Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
• Michael Heydt (2015), Mastering Pandas for Finance, Packt Publishing
• Michael Heydt (2015), Learning Pandas - Python Data Discovery and Analysis Made Easy, Packt Publishing
• James Ma Weiming (2015), Mastering Python for Finance, Packt Publishing
• Fabio Nelli (2015), Python Data Analytics: Data Analysis and Science using PANDAs, matplotlib and the Python Programming Language, Apress
• Skikit-learn, http://scikit-learn.org/
• Data School (2015), Machine learning in Python with scikit-learn, https://www.youtube.com/playlist?list=PL5-da3qGB5ICeMbQuqbbCOQWcS6OYBr5A
• Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson.