Python Scikit-Learn

(Machine Learning in Finance Application with Scikit-Learn In Python)

1081AI FA07
EMBA, IMTKU (M2457) (8413) (Fall 2019)
Fri 12,13,14 (19:20-22:10) (D301)

Min-Yuh Day
Associate Professor
Dept. of Information Management, Tamkang University

http://mail.tku.edu.tw/myday/
2019-11-22
<table>
<thead>
<tr>
<th>周次 (Week)</th>
<th>日期 (Date)</th>
<th>内容 (Subject/Topics)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2019/09/13</td>
<td>中秋節 (Mid-Autumn Festival) 放假一天 (Day off)</td>
</tr>
<tr>
<td>2</td>
<td>2019/09/20</td>
<td>人工智慧財務金融應用課程介紹 (Course Orientation for AI in Financial Application)</td>
</tr>
<tr>
<td>3</td>
<td>2019/09/27</td>
<td>人工智慧投資分析與機器人理財顧問 (Artificial Intelligence for Investment Analysis and Robo-Advisors)</td>
</tr>
<tr>
<td>4</td>
<td>2019/10/04</td>
<td>金融科技對話式商務與智慧型交談機器人 (Conversational Commerce and Intelligent Chatbots for Fintech)</td>
</tr>
<tr>
<td>5</td>
<td>2019/10/11</td>
<td>國慶日補假 (Bridge Holiday for National Day, Extra Day Off)</td>
</tr>
<tr>
<td>6</td>
<td>2019/10/18</td>
<td>財務金融事件研究法 (Event Studies in Finance)</td>
</tr>
<tr>
<td>週次 (Week)</td>
<td>日期 (Date)</td>
<td>內容 (Subject/Topics)</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>7</td>
<td>2019/10/25</td>
<td>人工智慧財務金融應用個案研究Ⅰ (Case Study on AI in Financial Application I)</td>
</tr>
<tr>
<td>8</td>
<td>2019/11/01</td>
<td>Python AI智慧金融分析基礎 (Foundations of AI in Finance Big Data Analytics with Python)</td>
</tr>
<tr>
<td>9</td>
<td>2019/11/08</td>
<td>Python Pandas 量化投資分析 (Quantitative Investing with Pandas in Python)</td>
</tr>
<tr>
<td>10</td>
<td>2019/11/15</td>
<td>期中報告 (Midterm Project Report)</td>
</tr>
<tr>
<td>11</td>
<td>2019/11/22</td>
<td>Python Scikit-Learn 機器學習財務金融應用 (Machine Learning in Finance Application with Scikit-Learn In Python)</td>
</tr>
<tr>
<td>12</td>
<td>2019/11/29</td>
<td>TensorFlow 深度學習財務金融應用Ⅰ (Deep Learning for Finance Application with TensorFlow I)</td>
</tr>
<tr>
<td>週次 (Week)</td>
<td>日期 (Date)</td>
<td>內容 (Subject/Topics)</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>----------------------</td>
</tr>
</tbody>
</table>
| 13         | 2019/12/06  | 人工智慧財務金融應用個案研究 II  
(Case Study on AI in Financial Application II) |
| 14         | 2019/12/13  | TensorFlow 深度學習財務金融應用 II  
(Deep Learning for Finance Application with TensorFlow II) |
| 15         | 2019/12/20  | TensorFlow 深度學習財務金融應用 III  
(Deep Learning for Finance Application with TensorFlow III) |
| 16         | 2019/12/27  | 社會網絡分析財務金融應用  
(Social Network Analysis for Finance Application) |
| 17         | 2020/01/03  | 期末報告 I (Final Project Presentation I) |
| 18         | 2020/01/10  | 期末報告 II (Final Project Presentation II) |
Machine Learning in Finance Application with Scikit-Learn In Python
Outline

• Machine Learning in Finance Application with Scikit-Learn In Python
  – Machine Learning
  – Scikit-Learn

https://github.com/ageron/handson-ml2

Artificial Intelligence
Machine Learning & Deep Learning

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
- Semi-supervised Learning
- Reinforcement Learning

Deep Learning (DL)

- CNN
- RNN
- LSTM
- GRU
- GAN

Source: https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/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
- **Unsupervised Learning**
- **Reinforcement Learning**

**Deep Learning (DL)**
- Support Vector Machine (SVM)
- Neural Network (NN)
- Deep Learning (DL)
- Naïve Bayes (NB)
- Bayesian Network (BN)
- Maximum Entropy (ME)

# 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, Naïve 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 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
Evaluation
(Accuracy of Classification Model)
Assessing the Classification Model

• Predictive accuracy
  – Hit rate

• Speed
  – Model building; predicting

• Robustness

• Scalability

• Interpretability
  – Transparency, explainability

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
Confusion Matrix for Tabulation of Two-Class Classification Results

True/Observed Class

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

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

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

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

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

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
Sensitivity = True Positive Rate

Specificity = True Negative Rate
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

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

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

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

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

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

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

**Sensitivity**

- True Positive Rate
- Recall
- Hit rate
- \( \frac{TP}{TP + FN} \)

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

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

**Specificity**

- True Negative Rate
- \( \frac{TN}{N} \)
- \( \frac{TN}{TN + FP} \)

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

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

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

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

<table>
<thead>
<tr>
<th>A</th>
<th>63 (TP)</th>
<th>28 (FP)</th>
<th>91</th>
</tr>
</thead>
<tbody>
<tr>
<td>37 (FN)</td>
<td>72 (TN)</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>200</td>
<td></td>
</tr>
</tbody>
</table>

**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

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

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

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

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

**PPV** = Positive Predictive Value

\[
F \text{ score (F-score) (F-measure)} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

\[
F1 = \frac{2\times(0.63\times0.69)/(0.63+0.69)}{(2 \times 63)/(100 + 91)} = (0.63 + 0.69) / 2 = 1.32 / 2 = 0.66
\]

**ACC** = 0.68

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

\[
= \frac{63 + 72}{200} = \frac{135}{200} = 0.675
\]

A

\[
\begin{array}{cc}
63 & 28 \\
37 & 72 \\
\end{array}
\]

TPR = 0.63

FPR = 0.28

PPV = 0.69

\[
F1 = \frac{2 \times (0.63 \times 0.69)}{(0.63 + 0.69)} = \frac{2 \times 63}{100 + 91} = \frac{0.63 + 0.69}{2} = 0.66
\]

ACC = 0.68

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

B

\[
\begin{array}{cc}
77 & 77 \\
23 & 23 \\
\end{array}
\]

TPR = 0.77

FPR = 0.77

PPV = 0.50

\[
F1 = \frac{2 \times (0.63 \times 0.69)}{(0.63 + 0.69)} = \frac{2 \times 63}{100 + 91} = \frac{0.63 + 0.69}{2} = 0.66
\]

ACC = 0.50

\[
ACC = \frac{100 + 100}{200} = \frac{200}{200} = 100
\]

Recall

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

Precision

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

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

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

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

Precision = Positive Predictive Value (PPV)
The Quant Finance PyData Stack

PyThalesians
Zipline
DX Analytics
PyAlgoTrade
QuantLib

Quantopian

PyTables
NetworkX
scikits-image

StatsModels
Statistics in Python

SciPy

matplotlib
pandas

\[ y_{it} = \beta x_{it} + \mu_i + \epsilon_{it} \]

NumPy

Python

IPython

Ipython

jupyter

Jake VanderPlas

Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#5
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.

Applications: 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

Scikit-Learn Machine Learning Map

Scikit-Learn Machine Learning Map

Iris Classification

iris.data


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
4.7,3.0,1.4,0.1,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
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")
```

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")

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

# 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']
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)).
```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>
```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())
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)
```python
print(df.groupby('class').size())
```

```
class
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
dtype: int64
```
```python
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_p6I1nnZDlFF354Nf_Lw

Data Mining and Machine Learning in Google Colab

```python
[17] # 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_p6I1nnZDlFF354Nf_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()
sns.pairplot(df, hue="class", size=2).

<table>
<thead>
<tr>
<th></th>
<th>sepal-length</th>
<th>sepal-width</th>
<th>petal-length</th>
<th>petal-width</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>1</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>2</td>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>3</td>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4</td>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5</td>
<td>5.4</td>
<td>3.9</td>
<td>1.7</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>6</td>
<td>4.6</td>
<td>3.4</td>
<td>1.4</td>
<td>0.3</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>7</td>
<td>5.0</td>
<td>3.4</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>8</td>
<td>4.4</td>
<td>2.9</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>9</td>
<td>4.9</td>
<td>3.1</td>
<td>1.5</td>
<td>0.1</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>140</td>
<td>6.7</td>
<td>3.1</td>
<td>5.6</td>
<td>2.4</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>141</td>
<td>6.9</td>
<td>3.1</td>
<td>5.1</td>
<td>2.3</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>142</td>
<td>5.8</td>
<td>2.7</td>
<td>5.1</td>
<td>1.9</td>
<td>Iris-virginica</td>
</tr>
</tbody>
</table>
```python
# 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>
```r
1 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>
# 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')
# 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)
```python
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)
```

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDIFF354Nf_Lw
# 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>
<tr>
<td>avg / total</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
<td>30</td>
</tr>
</tbody>
</table>

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')

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDlFF354Nf_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]]

<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>0.82</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
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  1 10]]

<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>
<tr>
<td>avg / total</td>
<td>0.83</td>
<td>0.80</td>
<td>0.80</td>
<td>30</td>
</tr>
</tbody>
</table>

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)
```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]]

            precision  recall  f1-score  support

Iris-setosa       1.00   1.00    1.00        7
Iris-versicolor   1.00   0.92    0.96       12
Iris-virginica    0.92   1.00    0.96       11

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
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>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>
</tr>
<tr>
<td>Iris-versicolor</td>
<td>1.00</td>
<td>0.75</td>
<td>0.86</td>
</tr>
<tr>
<td>Iris-virginica</td>
<td>0.79</td>
<td>1.00</td>
<td>0.88</td>
</tr>
</tbody>
</table>

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
#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
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().
# 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

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()
**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)
# 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-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()
#K-Means Clustering

```python
#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-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()
```

https://colab.research.google.com/drive/1QE7fR2OxHiQ0_p6l1nnZDlFF354Nf_Lw
# Python in Google Colab

[Link to the Google Colab notebook](https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT)

```python
# !pip install pandas_datareader
1 import numpy as np
2 import pandas as pd
3 import pandas_datareader.data as web
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import datetime as dt
7 import matplotlib inline
8
9 # Read Stock Data from Yahoo Finance
10 end = dt.datetime.now()
11 start = dt.datetime(end.year-2, end.month, end.day)
12 df = web.DataReader("^TWII", 'yahoo', start, end)  # ^TWII #2330.TW #^DJI #AAPL
13 df.to_csv('TWII.csv')
14 print(df.head())
15 print(df.tail())
16 df2 = pd.read_csv('TWII.csv') #df.from_csv('AAPL.csv')
17 print(df2.head())
18
19 df['Adj Close'].plot(legend=True, figsize=(12, 8), title='TWII', label='Adj Close')
20 plt.figure(figsize=(12,9))
21 top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
22 bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
23 top.plot(df.index, df['Adj Close'], color='blue')  # df.index gives the dates
24 bottom.bar(df.index, df['Volume'])
25
26 # set the labels
27 top.set_xlabel().set_visible(False)
28 top.set_title('TWII')
29 top.set_ylabel('Adj Close')
30 bottom.set_ylabel('Volume')
```
np.where
(df['MA20'] > df['MA60'],
12000,
9000)

# 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['Positions'] = np.where(df['MA20'] > df['MA60'], 12000, 9000)
df2 = pd.DataFrame({
    'Adj Close': df['Adj Close'],
    'MA05': df['MA05'],
    'MA20': df['MA20'],
    'MA60': df['MA60'],
    'Positions': df['Positions']
})
df2.plot(figsize=(12, 9), legend=True, title='AAPL', secondary_y='Positions').legend(bbox_to_anchor=(1.2, 0.5))
np.where
(df['MA20'] > df['MA60'],
1,
0)

# 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['Positions'] = np.where(df['MA20'] > df['MA60'], 1, 0)
df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'],
'MA20': df['MA20'], 'MA60': df['MA60'], 'Positions': df['Positions']})
Yves Hilpisch (2018),
Python for Finance: Mastering Data-Driven Finance, O'Reilly

https://github.com/yhilpisch/py4fi2nd

Source: https://www.amazon.com/Python-Finance-Mastering-Data-Driven/dp/1492024333
Aurélien Géron (2019),
O’Reilly Media, 2019

https://github.com/ageron/handson-ml2

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

https://github.com/ageron/handson-ml2
Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

Notebooks
1. The Machine Learning landscape
2. End-to-end Machine Learning project
3. Classification
4. Training Models
5. Support Vector Machines
6. Decision Trees
7. Ensemble Learning and Random Forests
8. Dimensionality Reduction
9. Unsupervised Learning Techniques
10. Artificial Neural Nets with Keras
11. Training Deep Neural Networks
12. Custom Models and Training with TensorFlow
13. Loading and Preprocessing Data
14. Deep Computer Vision Using Convolutional Neural Networks
15. Processing Sequences Using RNNs and CNNs
16. Natural Language Processing with RNNs and Attention
17. Representation Learning Using Autoencoders
18. Reinforcement Learning
19. Training and Deploying TensorFlow Models at Scale

https://github.com/ageron/handson-ml2
Papers with Code
State-of-the-Art (SOTA)

Browse State-of-the-Art

- 1509 leaderboards • 1327 tasks • 1347 datasets • 17810 papers with code

Follow on Twitter for updates

Computer Vision

- Semantic Segmentation
  - 33 leaderboards
  - 667 papers with code
- Image Classification
  - 52 leaderboards
  - 564 papers with code
- Object Detection
  - 54 leaderboards
  - 467 papers with code
- Image Generation
  - 51 leaderboards
  - 231 papers with code
- Pose Estimation
  - 40 leaderboards
  - 231 papers with code

› See all 707 tasks

Natural Language Processing

- Machine Translation
- Language Modelling
- Question Answering
- Sentiment Analysis
- Text Generation

https://paperswithcode.com/sota
Papers with Code
Stock Market Prediction

Leaderboards

No evaluation results yet. Help compare methods by submit evaluation metrics.

Subtasks

- Stock Price Prediction: 3 papers with code
- Stock Trend Prediction: 2 papers with code
- Stock Prediction: 1 paper with code

https://paperswithcode.com/task/stock-market-prediction
The Quant Finance PyData Stack

- PyThalesians
- Zipline
- DX Analytics
- PyAlgoTrade
- QuantLib

- Quantopian
- PyTables
- NetworkX
- scikits-image
- PyMC

- StatsModels
- matplotlib
- pandas
- NumPy
- SymPy

Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#5
Summary

• Machine Learning in Finance Application with Scikit-Learn In Python
  – Machine Learning
  – Scikit-Learn
References

- Ties de Kok (2017), Learn Python for Research, https://github.com/TiesdeKok/LearnPythonforResearch
- Python Programming, https://pythonprogramming.net/
- Python, https://www.python.org/
- Pandas, http://pandas.pydata.org/
- Skikit-learn, http://scikit-learn.org/
- Data School (2015), Machine learning in Python with scikit-learn, https://www.youtube.com/playlist?list=PL5-da3qGB5ICeMbQubbCOQWcS6OYBr5A