

# Kecerdasan Bisnis Terapan

## Prescriptive Analytics

# Optimization and Simulation

Husni  
Lab. Riset JTIF UTM

# Business Intelligence (BI)

**1** Introduction to BI and Data Science

**2** Descriptive Analytics

**3** Predictive Analytics

**④** Prescriptive Analytics

**5** Big Data Analytics

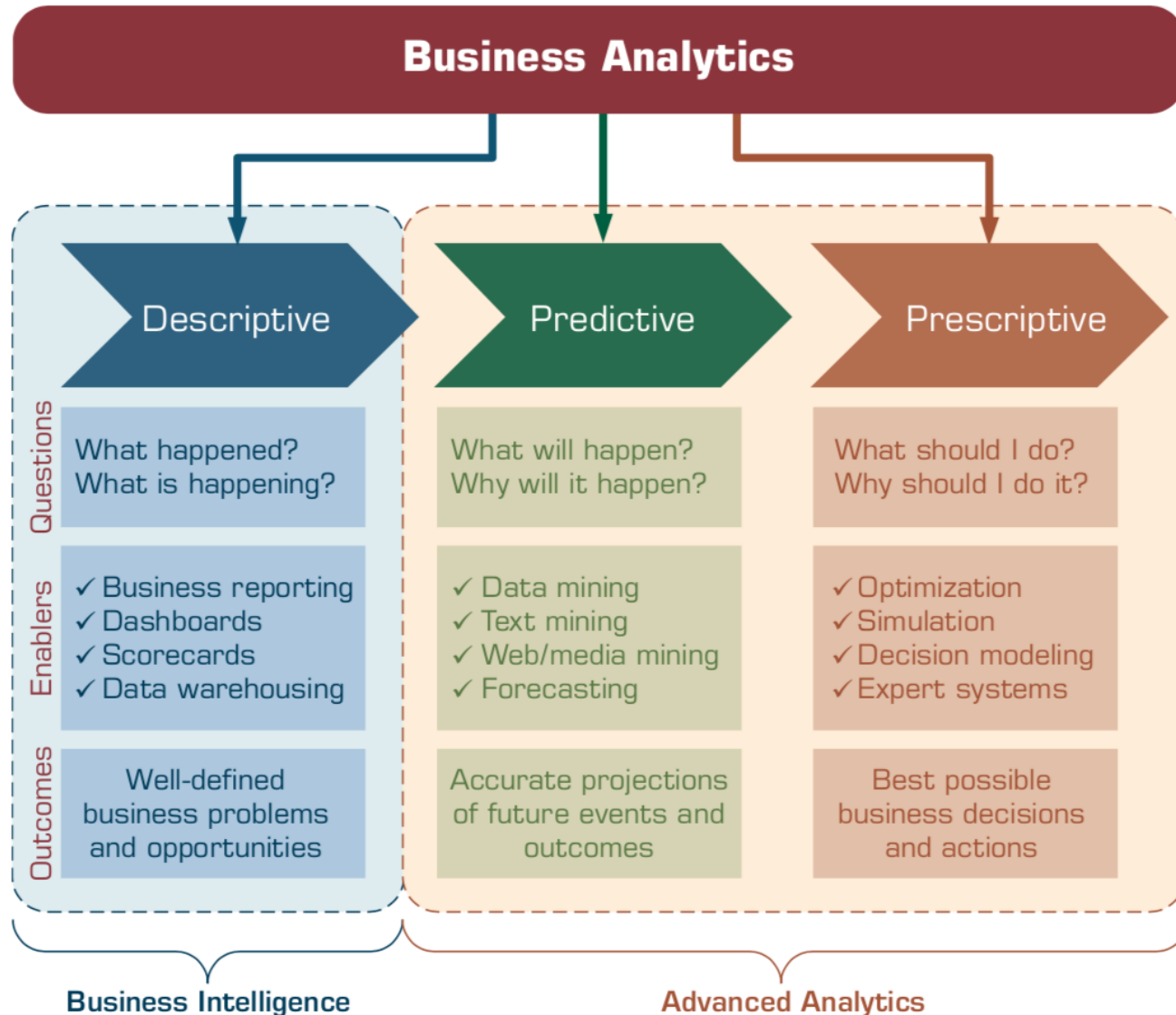
**6** Future Trends

# **Prescriptive Analytics: Optimization and Simulation**

# Outline

- Prescriptive Analytics
- Model-Based Decision Making
- Structure of Mathematical Models for Decision Support
- Certainty, Uncertainty, and Risk
- Mathematical Programming Optimization
- Simulation

# Business Analytics and Prescriptive Analytics



# Prescriptive Analytics Model Examples

- **Modeling** is a key element for **prescriptive analytics**.
- Employ a **mathematical model** to be able to recommend a decision for any realistic problem.
- Building a **probability-based response maximization model** with the budget as a constraint would give us the information we are seeking.

# Identification of the Problem and Environmental Analysis

- Environmental scanning and analysis
  - Monitoring, scanning, and interpretation of collected information
- Business intelligence/business analytics (BI/BA) tools can help identify problems by scanning for them.
- Variable Identification
- Forecasting (Predictive Analytics)

# Categories of Models

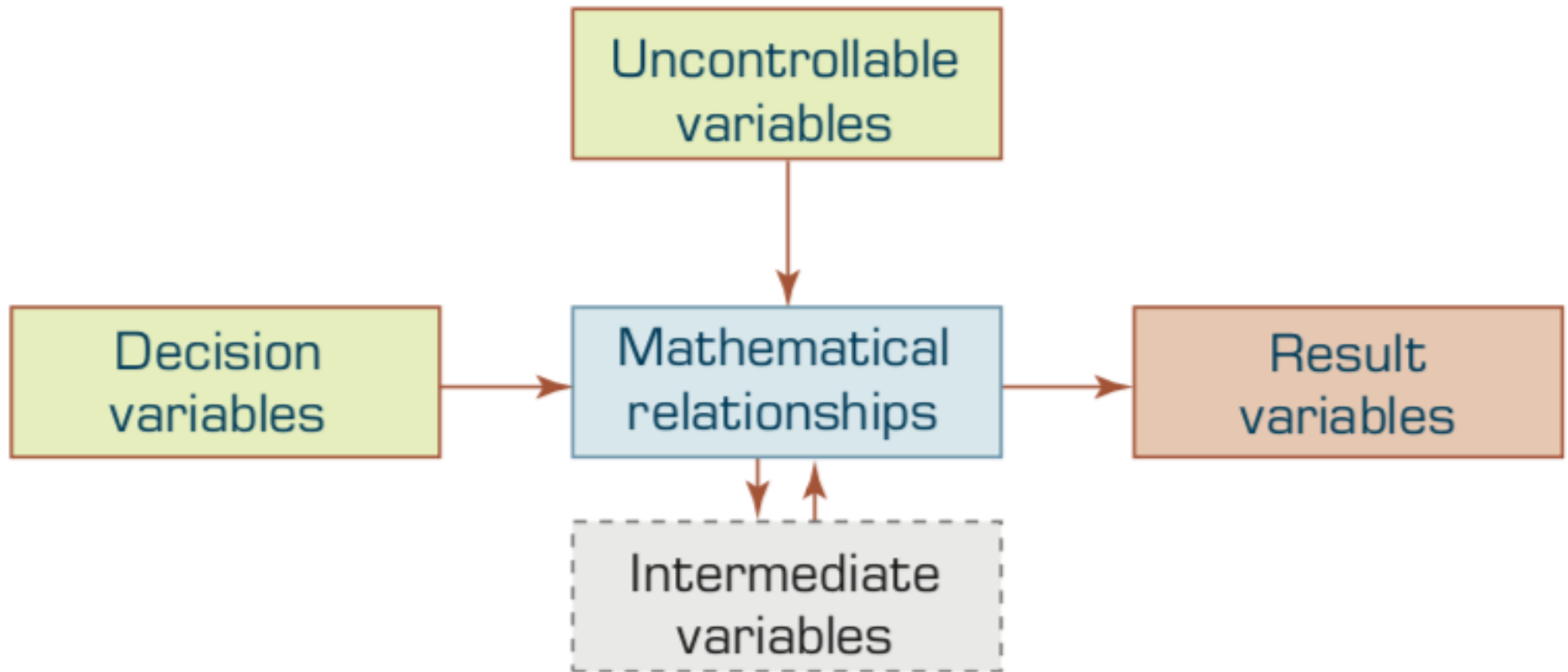
Category	Process and Objective	Representative Techniques
Optimization of problems with few alternatives	Find the best solution from a small number of alternatives	Decision tables, decision trees, analytic hierarchy process
Optimization via algorithm	Find the best solution from a large number of alternatives, using a step-by-step improvement process	Linear and other mathematical programming models, network models
Optimization via an analytic formula	Find the best solution in one step, using a formula	Some inventory models
Simulation	Find a good enough solution or the best among the alternatives checked, using experimentation	Several types of simulation
Heuristics	Find a good enough solution, using rules	Heuristic programming, expert systems
Predictive models	Predict the future for a given scenario	Forecasting models, Markov analysis
Other models	Solve a what-if case, using a formula	Financial modeling, waiting lines



# Structure of Mathematical Models for Decision Support

- The Components of Decision Support Mathematical Models
- The Structure of Mathematical Models

# The General Structure of a Quantitative Model



# Examples of Components of Models

Area	Decision Variables	Result Variables	Uncontrollable Variables and Parameters
Financial investment	Investment alternatives and amounts	Total profit, risk Rate of return on investment (ROI) Earnings per share Liquidity level	Inflation rate Prime rate Competition
Marketing	Advertising budget Where to advertise	Market share Customer satisfaction	Customer's income Competitor's actions
Manufacturing	What and how much to produce Inventory levels Compensation programs	Total cost Quality level Employee satisfaction	Machine capacity Technology Materials prices
Accounting	Use of computers Audit schedule	Data processing cost Error rate	Computer technology Tax rates Legal requirements
Transportation	Shipments schedule Use of smart cards	Total transport cost Payment float time	Delivery distance Regulations
Services	Staffing levels	Customer satisfaction	Demand for services

# The Structure of Mathematical Models

A very simple financial model is

$$***P = R - C***$$

where  $P$  = profit,  $R$  = revenue, and  $C$  = cost.

# The Structure of Mathematical Models

Financial model (present-value cash flow model)

$$P = F / (1 + i)^n$$

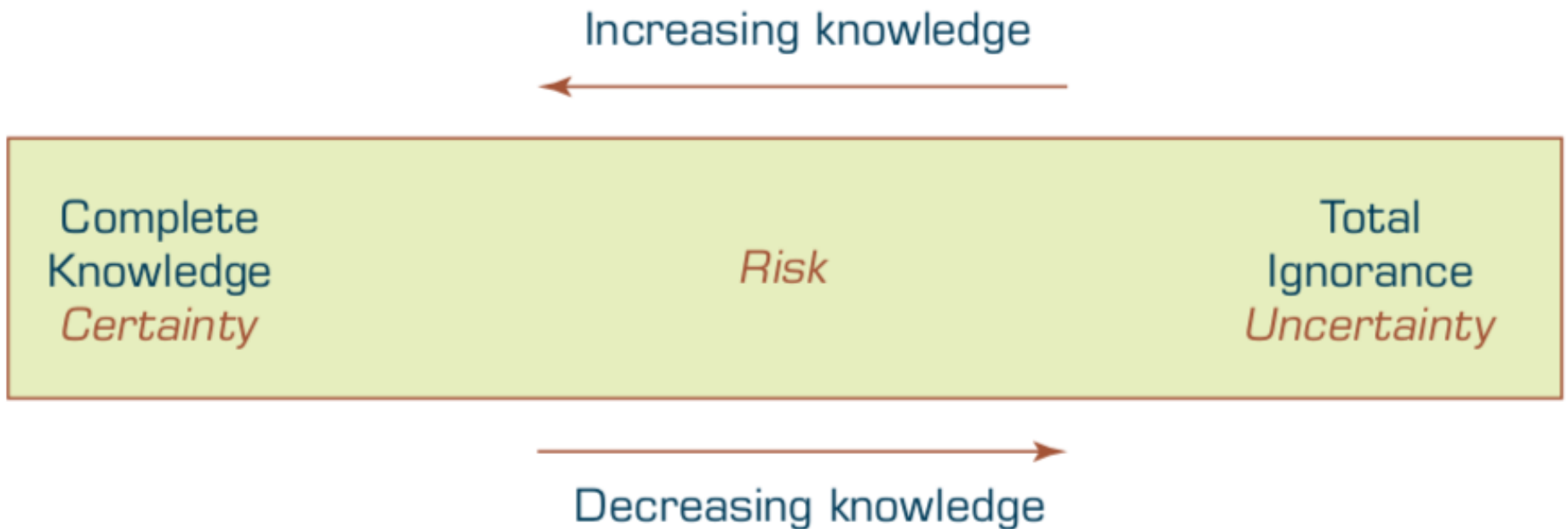
where  $P$  = present value,  $F$  = a future single payment in dollars,  $i$  = interest rate (percentage), and  $n$  = number of years.

$$P = 100,000 / (1 + 0.1)^5 = 62,092$$

# Certainty, Uncertainty, and Risk

- Decision Making under Certainty
- Decision Making under Uncertainty
- Decision Making under Risk (Risk Analysis)

# The Zones of Decision Making

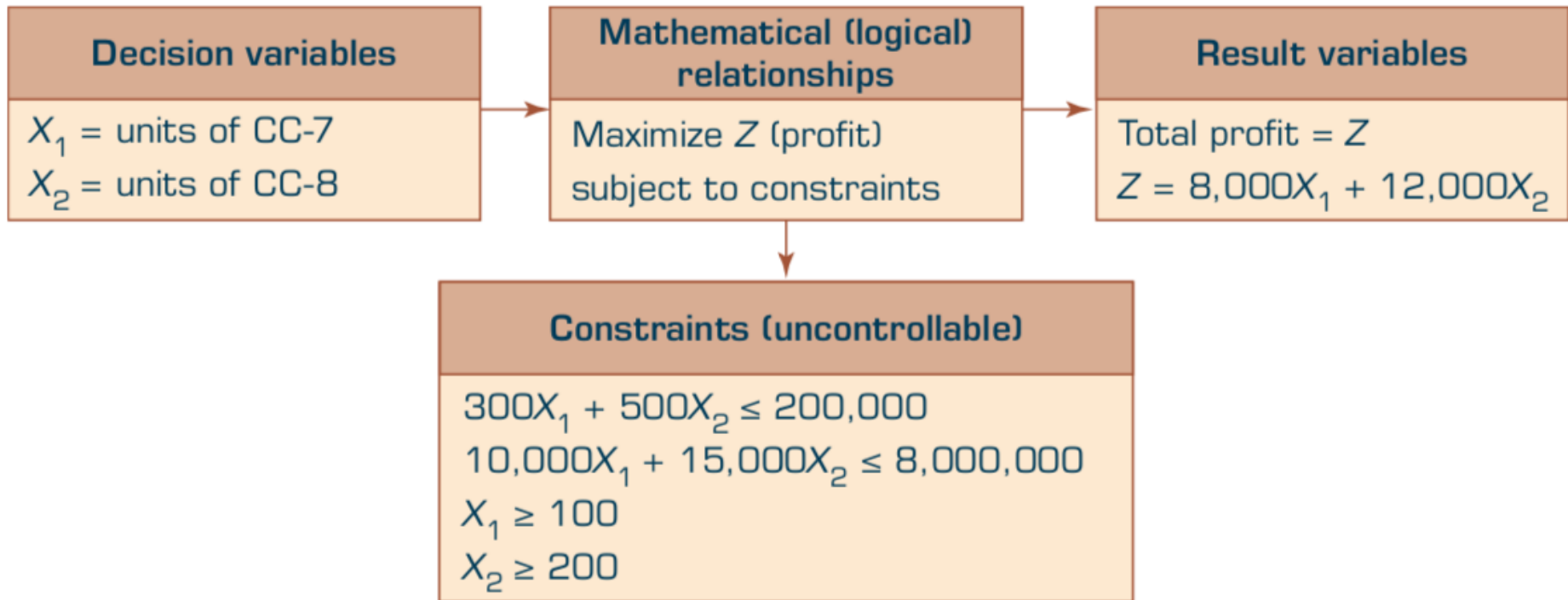


# Mathematical Programming Optimization

- **Mathematical programming**
  - a family of tools designed to help **solve managerial problems** in which the decision maker must **allocate scarce resources** among **competing activities** to **optimize a measurable goal**.
- **Linear programming (LP)** is the best-known technique in a family of optimization tools called *mathematical programming*.



# Mathematical Model of a Product-Mix Example



# Prescriptive Analytics, Operations research/ Management Science

- Multiple Goals
- Sensitivity Analysis
- What-If Analysis
- Goal Seeking

# Decision Analysis with Decision Tables and Decision Trees

- **Decision tables**
  - conveniently organize information and knowledge in a systematic, tabular manner to prepare it for analysis.
- **Decision tree**
  - shows the relationships of the problem graphically and can handle complex situations in a compact form.

# Investment Problem Decision Table Model

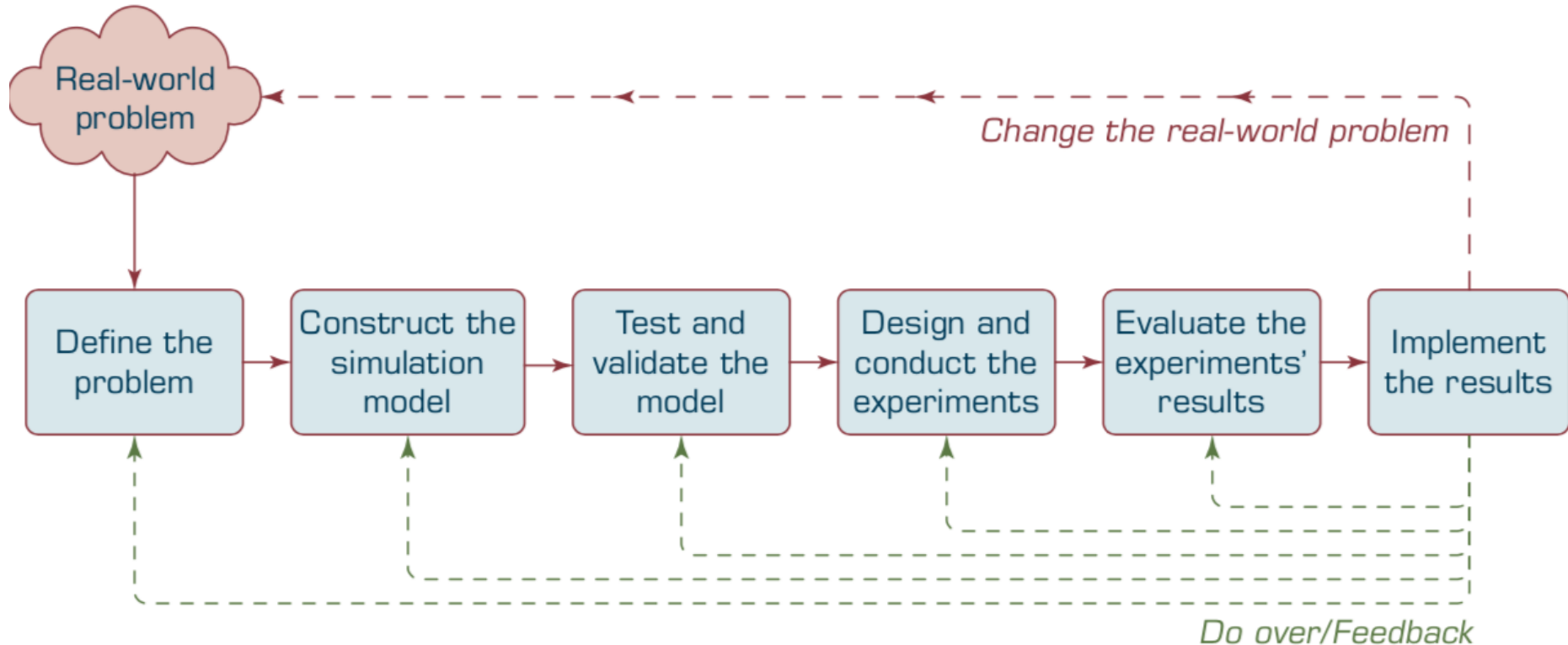
## State of Nature (Uncontrollable Variables)

Alternative	Solid Growth (%)	Stagnation (%)	Inflation (%)
Bonds	12.0	6.0	3.0
Stocks	15.0	3.0	-2.0
CDs	6.5	6.5	6.5

# Multiple Goals

Alternative	Yield (%)	Safety	Liquidity
Bonds	8.4	High	High
Stocks	8.0	Low	High
CDs	6.5	Very high	High

# The Process of Simulation



# Discrete versus Continuous Probability Distributions

Daily Demand	Discrete Probability	Continuous Probability
5	0.10	Daily demand is normally distributed with a mean of 7 and a standard deviation of 1.2
6	0.15	
7	0.30	
8	0.25	
9	0.20	

# Simulation Types

- Monte Carlo Simulation
- Discrete Event Simulation



# Visual Interactive Simulation

- Conventional Simulation Inadequacies
- Visual Interactive Simulation (VIS)
  - Visual Interactive Modeling (VIM)
- Visual Interactive Models and DSS

# Simulation Software

- Analytica  
(Lumina Decision Systems, [lumina.com](http://lumina.com))
- Excel add-ins Crystal Ball
- @RISK (Palisade Corp., [palisade.com](http://palisade.com))
- Arena
- Simio ([simio.com](http://simio.com))
- ExtendSim ([extendsim.com](http://extendsim.com))
- SAS JMP

# Monte Carlo Simulation

# Monte Carlo Simulation to Forecast Stock Prices

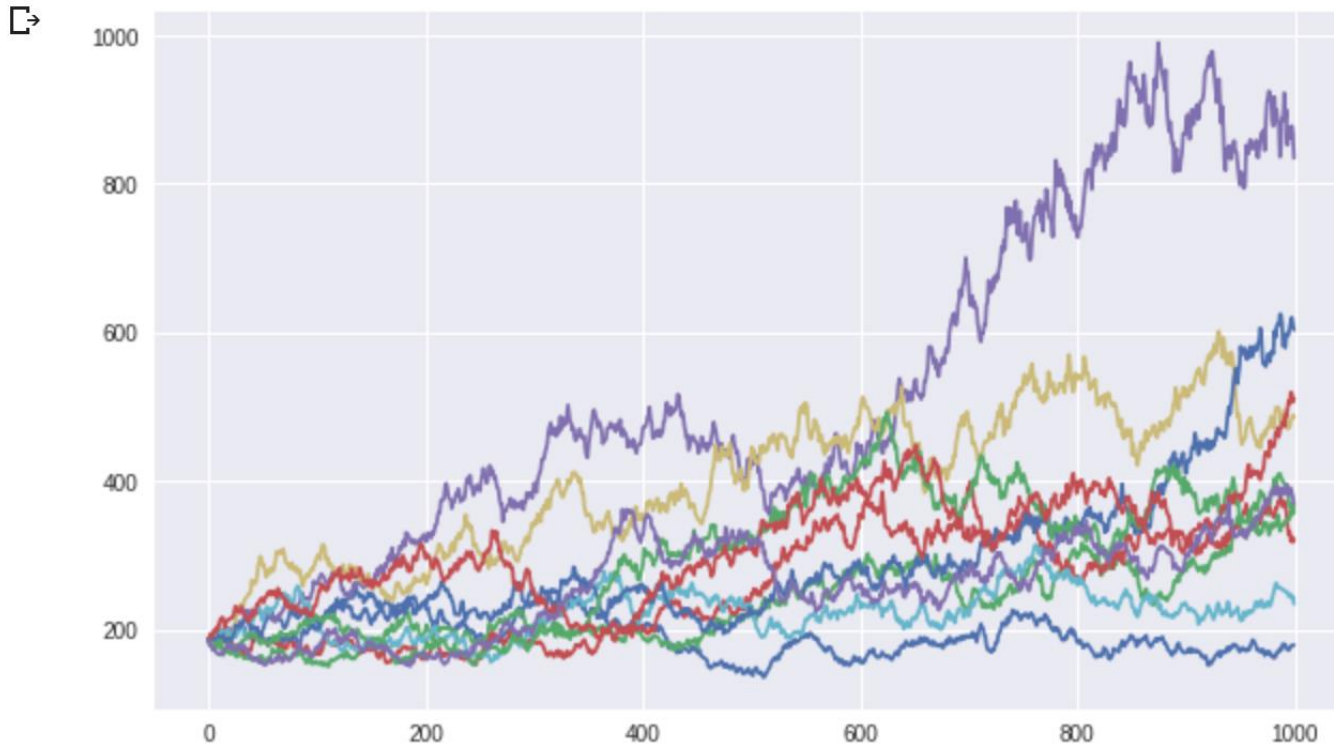


Stock\_Prices\_Monte\_Carlo\_Simulation.ipynb ☆

File Edit View Insert Runtime Tools Help

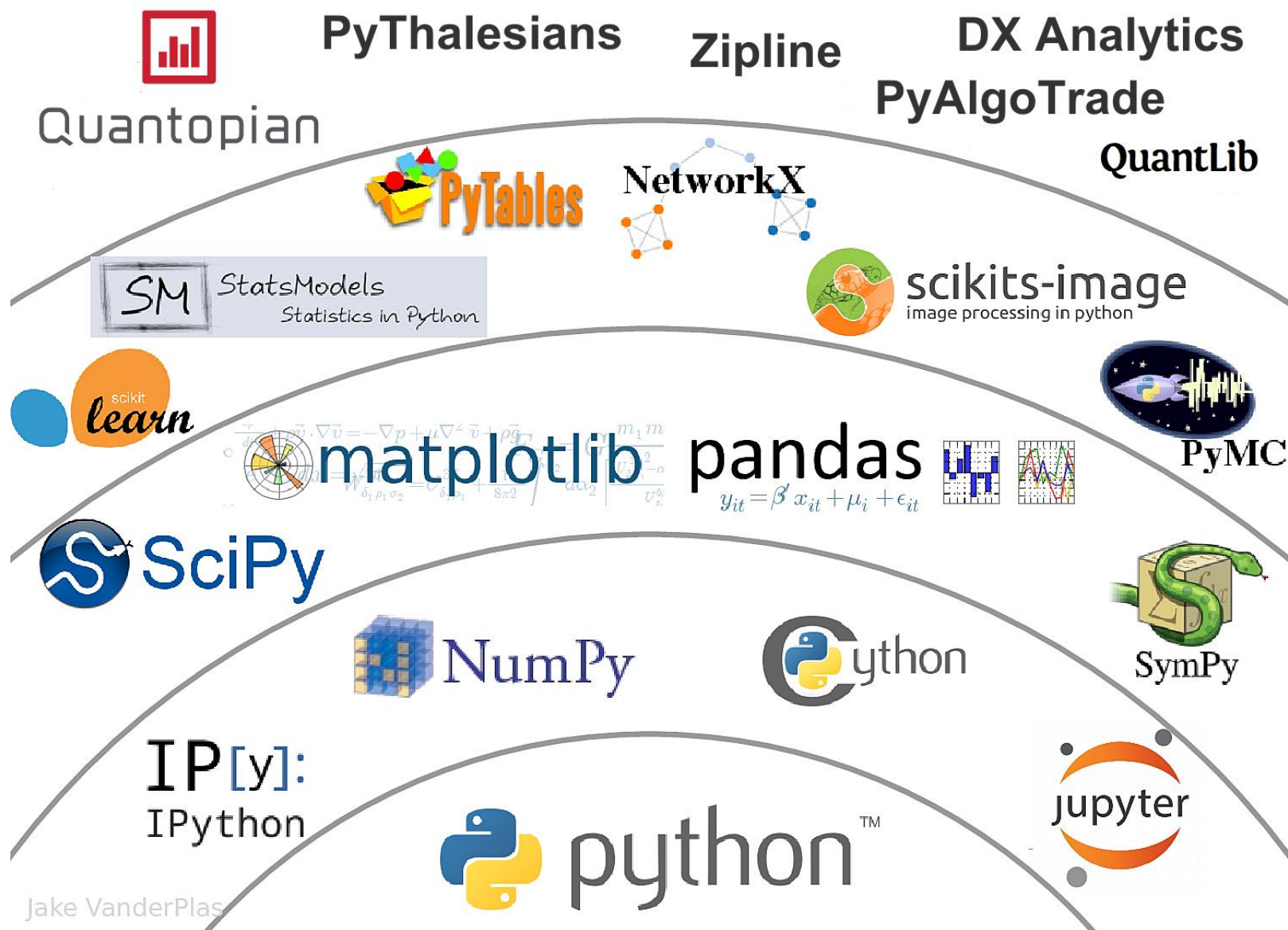
+ CODE + TEXT ↑ CELL ↓ CELL

```
1 plt.figure(figsize=(10,6))  
2 plt.plot(price_list);
```



# Python Pandas for Finance

# The Quant Finance PyData Stack



Jake VanderPlas

Source: [http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview\\_slides.ipynb/#/5](http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb/#/5)

# AAPL



# Python in Google Colab

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



python101.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT

SHARE

A

CODE TEXT CELL CELL

CONNECTED

EDITING

```
1 # !pip install pandas_datareader
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 %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 start = dt.datetime(2016, 1, 1)
13 df = web.DataReader("AAPL", 'yahoo', start, end)
14 df.to_csv('AAPL.csv')
15 df.from_csv('AAPL.csv')
16 df.tail()
17
18 df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
19 plt.figure(figsize=(12,9))
20 top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
21 bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
22 top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
23 bottom.bar(df.index, df['Volume'])
24
25 # set the labels
26 top.axes.get_xaxis().set_visible(False)
27 top.set_title('AAPL')
28 top.set_ylabel('Adj Close')
29 bottom.set_ylabel('Volume')
30
31 plt.figure(figsize=(12,9))
32 sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
33
34 # simple moving averages
35 df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
36 df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
37 df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
38 df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})
39 df2.plot(figsize=(12, 9), legend=True, title='AAPL')
40 df2.to_csv('AAPL_MA.csv')
41 fig = plt.gcf()
42 fig.set_size_inches(12, 9)
43 fig.savefig('AAPL_plot.png', dpi=300)
```



# ! pip install pandas\_datareader

```
1 ! pip install pandas_datareader
```

```
Collecting pandas_datareader
  Downloading https://files.pythonhosted.org/packages/cc/5c/ea5b6dcfd0f55c5fb1e37fb45335ec01ccec199b8a79339137f5ed269e0/pandas\_datarea
    100% |████████████████████| 112kB 2.7MB/s
Collecting lxml (from pandas_datareader)
  Downloading https://files.pythonhosted.org/packages/03/a4/9eea8035fc7c7670e5eab97f34ff2ef0ddd78a491bf96df5accedb0e63f5/lxml-4.2.5-cp3
    100% |████████████████████| 5.8MB 7.5MB/s
Requirement already satisfied: pandas>=0.19.2 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (0.22.0)
Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (2.18.4)
Requirement already satisfied: wrapt in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (1.10.11)
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (2.6.0)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (1.14.6)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (2018.5)
Requirement already satisfied: idna<2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (2.6)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (2018.8.24)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (1.24.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2->pandas>=0.19.2->pandas_datareader) (1.11.0)
Installing collected packages: lxml, pandas-datareader
Successfully installed lxml-4.2.5 pandas-datareader-0.7.0
```

# conda install pandas-datareader

```
imyday — -bash — 80x24
[iMyday-MacBook-Pro:~ imyday$ conda install pandas-datareader
Fetching package metadata .....
Solving package specifications: .

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

The following NEW packages will be INSTALLED:

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

Proceed ([y]/n)? y

requests-file- 100% |#####| Time: 0:00:00 1.55 MB/s
pandas-datarea 100% |#####| Time: 0:00:00 409.66 kB/s
[iMyday-MacBook-Pro:~ imyday$ conda list
# packages in environment at /Users/imyday/anaconda:
#
_license          1.1                py36_1
alabaster          0.7.9              py36_0
anaconda           4.3.1              np111py36_0
anaconda-client    1.6.0              py36_0
anaconda-navigator 1.5.0              py36_0
anaconda-project   0.4.1              py36_0
```

# AAPL



# Finance Data from Yahoo Finance

```
# !pip install pandas_datareader
import pandas_datareader.data as web
import datetime as dt
#Read Stock Data from Yahoo Finance
end = dt.datetime(2017, 12, 31)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

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

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

```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```



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



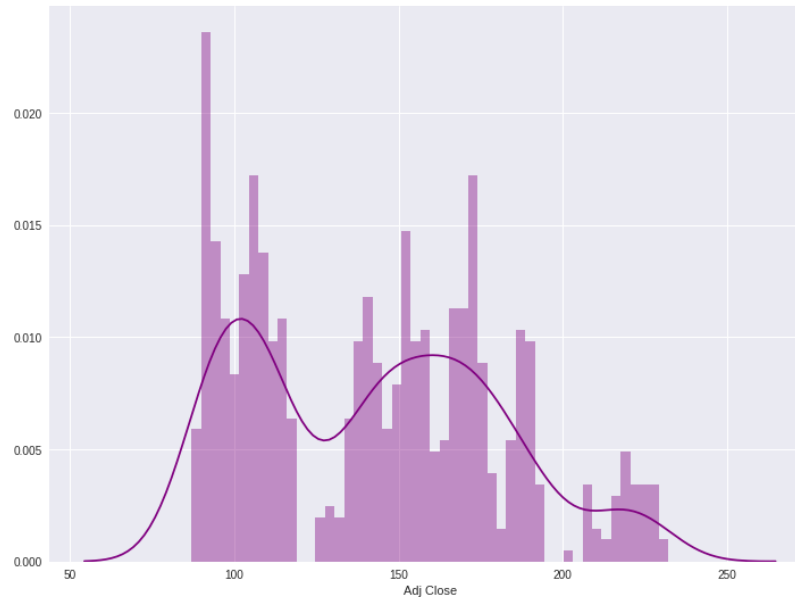
# AAPL

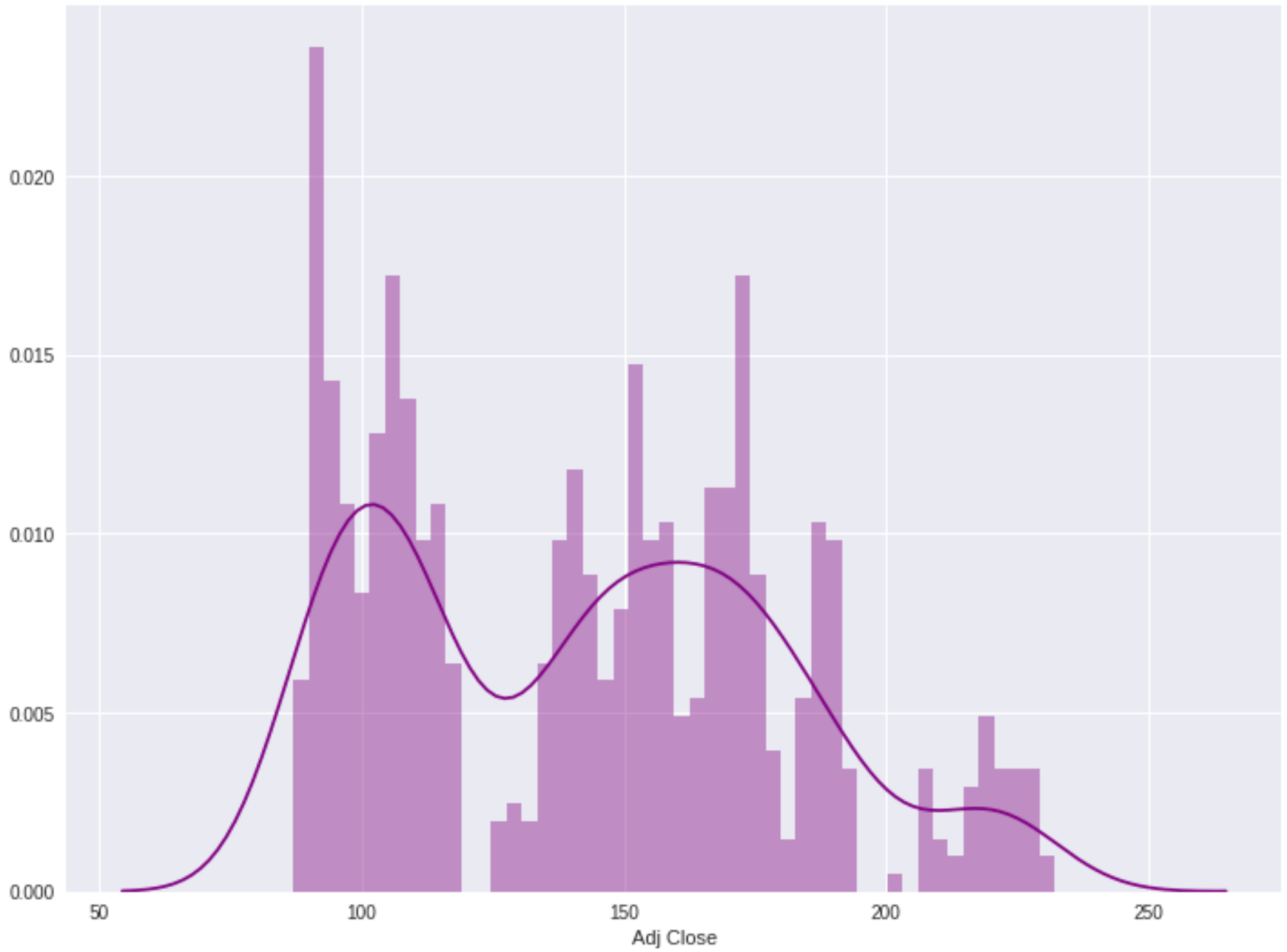




```
# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

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





```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean()
#5 days
df['MA20'] = df['Adj
Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj
Close'].rolling(60).mean() #60 days
df2 = pd.DataFrame({'Adj Close': df['Adj
Close'], 'MA05': df['MA05'], 'MA20': df['MA20'],
'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True,
title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```

# AAPL



```

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

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

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

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

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

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05'],'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()

```

```

1 # !pip install pandas_datareader
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 %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 start = dt.datetime(2016, 1, 1)
13 df = web.DataReader("AAPL", 'yahoo', start, end)
14 df.to_csv('AAPL.csv')
15 df.from_csv('AAPL.csv')
16 df.tail()
17
18 df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
19 plt.figure(figsize=(12,9))
20 top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
21 bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
22 top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
23 bottom.bar(df.index, df['Volume'])
24
25 # set the labels
26 top.axes.get_xaxis().set_visible(False)
27 top.set_title('AAPL')
28 top.set_ylabel('Adj Close')
29 bottom.set_ylabel('Volume')
30
31 plt.figure(figsize=(12,9))
32 sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
33
34 # simple moving averages
35 df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
36 df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
37 df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
38 df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})
39 df2.plot(figsize=(12, 9), legend=True, title='AAPL')
40 df2.to_csv('AAPL_MA.csv')
41 fig = plt.gcf()
42 fig.set_size_inches(12, 9)
43 fig.savefig('AAPL_plot.png', dpi=300)
44 plt.show()

```

# Finance Data from Quandl

```
# ! pip install quandl
import quandl
# quandl.ApiConfig.api_key = "YOURAPIKEY"
df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

```
1 # ! pip install quandl
2 import quandl
3 # quandl.ApiConfig.api_key = "YOURAPIKEY"
4 df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")
5 df.to_csv('AAPL.csv')
6 df.from_csv('AAPL.csv')
7 df.tail()
```

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:5: FutureWarning: from\_csv is deprecated. Please use read\_csv(...) instead  
"""

	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
Date												
2017-12-22	174.68	175.424	174.500	175.01	16052615.0	0.0	1.0	174.68	175.424	174.500	175.01	16052615.0
2017-12-26	170.80	171.470	169.679	170.57	32968167.0	0.0	1.0	170.80	171.470	169.679	170.57	32968167.0
2017-12-27	170.10	170.780	169.710	170.60	21672062.0	0.0	1.0	170.10	170.780	169.710	170.60	21672062.0
2017-12-28	171.00	171.850	170.480	171.08	15997739.0	0.0	1.0	171.00	171.850	170.480	171.08	15997739.0
2017-12-29	170.52	170.590	169.220	169.23	25643711.0	0.0	1.0	170.52	170.590	169.220	169.23	25643711.0

# Yahoo Finance Symbols: AAPL Apple Inc. (AAPL)

Home Mail Flickr Tumblr News Sports Finance Celebrity Answers Groups Mobile More ▾

**YAHOO!**  
FINANCE

Search for news, symbols or companies

Search

Finance Home Originals Events Personal Finance Technology Markets Industries **NEW** My Screeners My Portfolio

**S&P 500**

2,344.02

-29.45 (-1.24%)



**Dow 30**

20,668.01

-237.85 (-1.14%)



**Nasdaq**

5,793.83

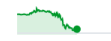
-107.70 (-1.83%)



**Crude Oil**

47.50

+0.16 (+0.34%)



**Gold**

1,245.40

-1.10 (-0.09%)



## Quote Lookup

Search for symbols or companies: YHOO, GOOG, DIS



## Symbols similar to 'aapl'

All Markets ▾

All (9)

Stocks (6)

Mutual Funds (0)

ETFs (1)

Indices (2)

Futures (0)

Currencies (0)

Symbol	Company Name	Last Price	Industry / Category	Type	Exchange
<a href="#">AAPL</a>	Apple Inc.	139.84	Electronic Equipment	Stocks	NMS
<a href="#">AAPL.SW</a>	Apple Inc.	140.70	N/A	Stocks	EBS
<a href="#">AAPL.MX</a>	Apple Inc.	2678.68	Electronic Equipment	Stocks	MEX
<a href="#">AAPL34F.SA</a>	Apple Inc.	0.00	N/A	Stocks	SAO
<a href="#">AAPL34.SA</a>	Apple Inc.	43.14	Electronic Equipment	Stocks	SAO

<http://finance.yahoo.com/q?s=AAPL>



# Dow Jones Industrial Average (^DJI)

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

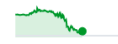
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,244.90

-1.60 (-0.13%)



Dow Jones Industrial Average (^DJI) 20,668.01 -237.85 (-1.14%) As of 4:36PM EDT. Market closed.



<http://finance.yahoo.com/chart/^DJI>

# TSEC weighted index (^TWII) - Taiwan

<b>S&amp;P 500</b> 2,344.02 -29.45 (-1.24%)	<b>Dow 30</b> 20,668.01 -237.85 (-1.14%)	<b>Nasdaq</b> 5,793.83 -107.70 (-1.83%)	<b>Crude Oil</b> 47.50 +0.16 (+0.34%)	<b>Gold</b> 1,245.10 -1.40 (-0.11%)
---	--	---	---	---

**TSEC weighted index (^TWII)** 9,868.95 **-103.54 (-1.04%)** As of 10:38AM CST. Taiwan Delayed Price. Market open.



# Yahoo Finance Charts

## TSMC (2330.TW)

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

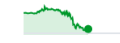
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.00

-1.50 (-0.12%)



Taiwan Semiconductor Manufacturing Company Limited (2330.TW) 192.00 -3.00 (-1.54%)

As of 10:29AM CST. Taiwan Delayed Price. Market open.



<http://finance.yahoo.com/chart/2330.TW>

```
import pandas as pd
import pandas_datareader.data as web
df = web.DataReader('AAPL', data_source='yahoo',
start='1/1/2010', end='3/21/2017')
df.to_csv('AAPL.csv')
df.tail()
```

```
import pandas as pd
import pandas_datareader.data as web
#df = web.DataReader('AAPL', 'yahoo')
df = web.DataReader('AAPL', data_source='yahoo', start='1/1/2010', end='3/21/2017')
#df = web.DataReader('AAPL', data_source='google', start='1/1/2010', end='3/21/2017')
df.to_csv('AAPL.csv')
df.tail()
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2017-03-15	139.410004	140.750000	139.029999	140.460007	25566800	140.460007
2017-03-16	140.720001	141.020004	140.259995	140.690002	19132500	140.690002
2017-03-17	141.000000	141.000000	139.889999	139.990005	43597400	139.990005
2017-03-20	140.399994	141.500000	140.229996	141.460007	20213100	141.460007
2017-03-21	142.110001	142.800003	139.729996	139.839996	39116800	139.839996

```
df = web.DataReader('GOOG',  
data_source='yahoo', start='1/1/1980',  
end='3/21/2017')  
df.head(10)
```

```
df = web.DataReader('GOOG', data_source='yahoo', start='1/1/1980', end='3/21/2017')  
df.head(10)
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2004-08-19	100.000168	104.060182	95.960165	100.340176	44871300	50.119968
2004-08-20	101.010175	109.080187	100.500174	108.310183	22942800	54.100990
2004-08-23	110.750191	113.480193	109.050183	109.400185	18342800	54.645447
2004-08-24	111.240189	111.600192	103.570177	104.870176	15319700	52.382705
2004-08-25	104.960181	108.000187	103.880180	106.000184	9232100	52.947145
2004-08-26	104.950180	107.950188	104.660179	107.910182	7128600	53.901190
2004-08-27	108.100185	108.620186	105.690180	106.150181	6241200	53.022069
2004-08-30	105.280178	105.490184	102.010172	102.010172	5221400	50.954132
2004-08-31	102.300173	103.710180	102.160177	102.370175	4941200	51.133953
2004-09-01	102.700174	102.970180	99.670169	100.250171	9181600	50.075011

# df.tail(10)

```
df.tail(10)
```

	<b>Open</b>	<b>High</b>	<b>Low</b>	<b>Close</b>	<b>Volume</b>	<b>Adj Close</b>
<b>Date</b>						
<b>2017-03-08</b>	833.510010	838.150024	831.789978	835.369995	988700	835.369995
<b>2017-03-09</b>	836.000000	842.000000	834.210022	838.679993	1259900	838.679993
<b>2017-03-10</b>	843.280029	844.909973	839.500000	843.250000	1701100	843.250000
<b>2017-03-13</b>	844.000000	848.684998	843.250000	845.539978	1149500	845.539978
<b>2017-03-14</b>	843.640015	847.239990	840.799988	845.619995	779900	845.619995
<b>2017-03-15</b>	847.590027	848.630005	840.770020	847.200012	1379600	847.200012
<b>2017-03-16</b>	849.030029	850.849976	846.130005	848.780029	970400	848.780029
<b>2017-03-17</b>	851.609985	853.400024	847.109985	852.119995	1712300	852.119995
<b>2017-03-20</b>	850.010010	850.219971	845.150024	848.400024	1190300	848.400024
<b>2017-03-21</b>	851.400024	853.500000	829.020020	830.460022	2442900	830.460022

# df.count()

```
df.count()
```

```
Open          3169  
High          3169  
Low           3169  
Close         3169  
Volume        3169  
Adj Close     3169  
dtype: int64
```

# df.ix['2015-12-31']

```
df.ix['2015-12-31']
```

```
Open          7.695000e+02
High          7.695000e+02
Low           7.583400e+02
Close         7.588800e+02
Volume        1.489600e+06
Adj Close     7.588800e+02
Name: 2015-12-31 00:00:00, dtype: float64
```



```
df.to_csv('2330.TW.Yahoo.Finance.Data.csv')
```

2330.TW.Yahoo.Finance.Data.csv ×

```
1 Date,Open,High,Low,Close,Volume,Adj Close
2 2010-01-01,64.5,64.5,64.5,64.5,0,52.8308
3 2010-01-04,65.0,65.0,64.0,64.9,39407000,53.1584
4 2010-01-05,65.0,65.1,63.9,64.5,37138000,52.8308
5 2010-01-06,64.5,64.9,63.7,64.9,49261000,53.1584
6 2010-01-07,64.9,65.0,64.2,64.2,42134000,52.5851
7 2010-01-08,63.5,64.3,63.5,64.0,46076000,52.4213
8 2010-01-11,64.0,64.9,63.5,64.5,36799000,52.8308
9 2010-01-12,64.4,64.4,63.3,63.6,49853000,52.0936
10 2010-01-13,63.0,63.1,62.6,62.8,47976000,51.4384
11 2010-01-14,63.6,63.6,63.0,63.2,36149000,51.766
12 2010-01-15,62.9,63.5,62.8,63.5,47852000,52.0117
13 2010-01-18,62.8,63.1,62.8,62.9,30136000,51.5203
14 2010-01-19,63.0,63.2,62.0,62.5,47202000,51.1926
15 2010-01-20,62.9,63.2,62.2,63.0,52281000,51.6022
```

```
import fix_yahoo_finance as yf
data = yf.download("^TWII", start="2017-07-01", end="2017-11-15")
data.to_csv('TWII_201707_201711.csv')
data.tail()
```

```
import fix_yahoo_finance as yf
data = yf.download("^TWII", start="2017-07-01", end="2017-11-15")
data.to_csv('TWII_201707_201711.csv')
data.tail()
```

```
[*****100%*****] 1 of 1 downloaded
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-11-08	10839.440430	10844.740234	10806.009766	10818.990234	10818.990234	2438000
2017-11-09	10802.950195	10831.379883	10721.870117	10743.269531	10743.269531	2917800
2017-11-10	10713.669922	10742.610352	10659.290039	10732.669922	10732.669922	2277000
2017-11-13	10728.219727	10749.389648	10683.919922	10683.919922	10683.919922	2765100
2017-11-14	10716.589844	10735.080078	10654.580078	10687.179688	10687.179688	2596600

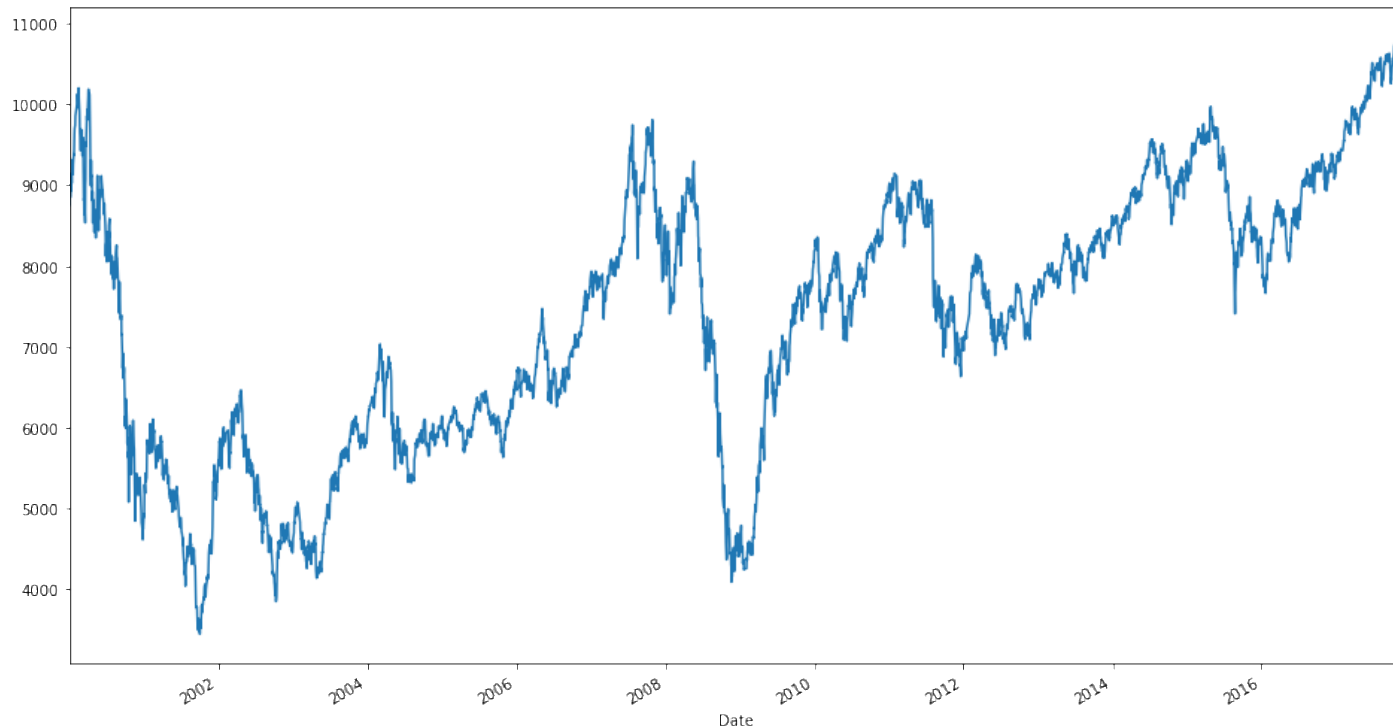
# df.loc[start:end]

```
df = df.loc['2017-10-01' : '2017-11-15']
```

```
import matplotlib.pyplot as plt
%matplotlib inline
import fix_yahoo_finance as yf

df = yf.download("^TWII", start="2000-01-01", end="2017-11-15")
df.to_csv('YF_TWII_2000_2017.csv')
print(df.head())

fig = plt.figure(figsize=(16,9))
df["Adj Close"].plot()
fig.show()
```



# candlestick\_ohlc

```
import matplotlib.pyplot as plt  
  
from matplotlib.finance  
import candlestick_ohlc
```

```
import matplotlib.pyplot as plt
from matplotlib.finance import candlestick_ohlc
```



# daily\_to\_weekly

```
#Convert Daily Data to Weekly Data
def daily_to_weekly(df):
    #dfWeekly = daily_to_weekly(df)
    #df.sort_index(axis=0, level=None, ascending=True, inplace=True)
    Open = df.Open.resample('W-Fri').first() #W #W-MON #W-Fri
    High = df.High.resample('W-Fri').max()
    Low = df.Low.resample('W-Fri').min()
    Close = df.Close.resample('W-Fri').last()
    Volume = df.Volume.resample('W-Fri').sum()
    Adj_Close = df["Adj Close"].resample('W-Fri').last()
    dfWeekly = pd.concat([Open, High, Low, Close, Volume, Adj_Close], axis=1)
    dfWeekly = dfWeekly[pd.notnull(dfWeekly['Adj Close'])]
    return dfWeekly
```

# daily\_to\_monthly

```
#Convert Daily Data to Monthly Data
def daily_to_monthly(df):
    #dfMonthly = daily_to_monthly(df)
    Open = df.Open.resample('M').first()
    High = df.High.resample('M').max()
    Low = df.Low.resample('M').min()
    Close = df.Close.resample('M').last()
    Volume = df.Volume.resample('M').sum()
    Adj_Close = df["Adj Close"].resample('M').last()
    dfMonthly = pd.concat([Open, High, Low, Close, Volume, Adj_Close], axis=1)
    dfMonthly = dfMonthly[pd.notnull(dfMonthly['Adj Close'])]
    return dfMonthly
```



# TA-Lib:

# Technical Analysis Library



## TA-Lib : Technical Analysis Library

Home

Products  
Downloads  
Purchase  
Support

Function List

Source Code  
Community Forum  
Useful Links

About Us

TA-Lib websites, products and trademarks are owned by TicTacTec LLC.

### Multi-Platform Tools for Market Analysis ...

TA-Lib is widely used by trading software developers requiring to perform technical analysis of financial market data.

- Includes 200 indicators such as ADX, MACD, RSI, Stochastic, Bollinger Bands etc... ([more info](#))
- Candlestick pattern recognition
- Open-source API for C/C++, Java, Perl, Python and 100% Managed .NET

### Free Open-Source Library

TA-Lib is available under a BSD License allowing it to be integrated in your own open-source or commercial application. ([more info](#))

### Commercial Application

TA-Lib is also available as an easy to install Excel Add-Ins. Try it for free!

# Stochastic Oscillator (KD)

```
#Stochastic oscillator %D
def KDJ(df, n, m1, m2):
    #KDJ(df, 9, 3, 3)
    KDJ_n = n
    KDJ_m1 = m1
    KDJ_m2 = m2

    df['Low_n'] = pd.rolling_min(df['Low'], KDJ_n)
    df['Low_n'].fillna(value=pd.expanding_min(df['Low']), inplace=True)
    df['High_n'] = pd.rolling_max(df['High'], KDJ_n)
    df['High_n'].fillna(value=pd.expanding_max(df['High']), inplace=True)

    df['RSV'] = (df['Close'] - df['Low_n']) / (df['High_n'] - df['Low_n']) * 100

    df['KDJ_K'] = pd.ewma(df['RSV'], KDJ_m1)
    df['KDJ_D'] = pd.ewma(df['KDJ_K'], KDJ_m2)
    df['KDJ_J'] = 3 * df['KDJ_K'] - 2 * df['KDJ_D']
    return df
```

# Bollinger Bands

```
#Bollinger Bands
def BBANDS20(df, n):
    MA = pd.Series(pd.rolling_mean(df['Close'], n))
    MSD = pd.Series(pd.rolling_std(df['Close'], n))
    b1 = 4 * MSD / MA
    B1 = pd.Series(b1, name = 'BollingerB_' + str(n))
    df = df.join(B1)
    b2 = (df['Close'] - MA + 2 * MSD) / (4 * MSD)
    B2 = pd.Series(b2, name = 'Bollinger%b_' + str(n))
    df = df.join(B2)
    return df
```

# Bollinger Bands

```
#BB Bollinger Bands BB_20
def BB_20(df):
    df['BB_MA20'] = pd.stats.moments.rolling_mean(df["Adj Close"], 20)
    df['BB_SD20'] = pd.stats.moments.rolling_std(df['Adj Close'],20)
    df['BB_UpperBand'] = df['BB_MA20'] + (df['BB_SD20']*2) # Default 2*SD
    df['BB_LowerBand'] = df['BB_MA20'] - (df['BB_SD20']*2)
    df['BB_PB'] = (df['Adj Close'] - df['BB_LowerBand']) / (df['BB_UpperBand'] -
df['BB_LowerBand'])
    df['BB_BW'] = (df['BB_UpperBand'] - df['BB_LowerBand']) / df['BB_MA20']
    df['BB_UpperBand_1SD'] = df['BB_MA20'] + (df['BB_SD20'])
    df['BB_LowerBand_1SD'] = df['BB_MA20'] - (df['BB_SD20'])
    #BB_PB: Bollinger Band Percent b (PB)
    #BB_BW: Bollinger Band Band Width (BW)
    return df
```

**AI + VDI**

**POC**

# AI + VDI POS

## TensorFlow Models

- M1: Basic Classification (Image Classification) (65 Seconds)
  - [https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic\\_classification.ipynb](https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_classification.ipynb)
- M2: Basic Text Classification (Text Classification) (46 Seconds)
  - [https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic\\_text\\_classification.ipynb](https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb)
- M3: Basic Regression (Predict House Prices) (43 Seconds)
  - [https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic\\_regression.ipynb](https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_regression.ipynb)
- M4: Pix2Pix Eager (Option) (7-8 Hours)
  - [https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/pix2pix/pix2pix\\_eager.ipynb](https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/pix2pix/pix2pix_eager.ipynb)
- M5. NMT with Attention (Option) (20-30 minutes)
  - [https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/nmt\\_with\\_attention/nmt\\_with\\_attention.ipynb](https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/nmt_with_attention/nmt_with_attention.ipynb)

# Basic Classification

## Fashion MNIST Image Classification

<https://colab.research.google.com/drive/19PJOJi1vn1kjcultzNHjRSLbeVI4kd5z>

The image shows a Google Colab notebook interface. At the top, the notebook title is 'tf01\_basic\_classification.ipynb'. Below the title, there are navigation options: 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right side, there are icons for 'COMMENT', 'SHARE', and a user profile icon. Below the navigation bar, there are tabs for '+ CODE', '+ TEXT', and 'CELL' (with up and down arrows). On the far right, there are 'CONNECT' and 'EDITING' options. The main content area is divided into a 'Table of contents' on the left and a code editor on the right. The table of contents includes: 'Copyright 2018 The TensorFlow Authors.', 'Licensed under the Apache License, Version 2.0 (the "License");', 'MIT License', 'Train your first neural network: basic classification', 'Import the Fashion MNIST dataset', 'Explore the data', 'Preprocess the data', 'Build the model', 'Setup the layers', 'Compile the model', 'Train the model', 'Evaluate accuracy', and 'Make predictions'. The code editor shows a code cell with the following content:

```
▶ Copyright 2018 The TensorFlow Authors.
↳ 2 cells hidden

▶ Train your first neural network: basic classification

View on TensorFlow.org Run in Google Colab View source on GitHub

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.

1 # memory footprint support libraries/code
2 !ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUutil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
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 {2:3.0f}% | Total {3:.0f}MB".format
16     printm()
```

# Text Classification

## IMDB Movie Reviews

[https://colab.research.google.com/drive/1x16h1GhHsLlrLYtPCvCHaoO1W-i\\_gror](https://colab.research.google.com/drive/1x16h1GhHsLlrLYtPCvCHaoO1W-i_gror)

The screenshot shows a Google Colab notebook titled "tf02\_basic-text-classification.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". On the right, there are buttons for "COMMENT", "SHARE", and a user profile icon. Below the menu, there are controls for "CODE", "TEXT", "CELL", and "CONNECT". A sidebar on the left contains a "Table of contents" with the following items: "Copyright 2018 The TensorFlow Authors.", "Licensed under the Apache License, Version 2.0 (the 'License');", "MIT License", "Text classification with movie reviews" (highlighted), "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", and "Evaluate the model". The main content area shows a copyright notice, a section titled "Text classification with movie reviews" with links to TensorFlow.org, Google Colab, and GitHub, and a code cell with the following Python code:

```
1 # memory footprint support libraries/code
2 !ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUtil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
12 def printm():
13     process = psutil.Process(os.getpid())
```

Source: [https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic\\_text\\_classification.ipynb](https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb)



# Basic Regression

## Predict House Prices

[https://colab.research.google.com/drive/1v4c8ZHTnRtgd2\\_25K\\_AURjR6SCVBRdj](https://colab.research.google.com/drive/1v4c8ZHTnRtgd2_25K_AURjR6SCVBRdj)

tf03\_basic-regression.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CONNECT EDITING

Table of contents Code snippets Files X

▶ Copyright 2018 The TensorFlow Authors.

↳ 2 cells hidden

### ▼ Predict house prices: regression

 [View on TensorFlow.org](#)  [Run in Google Colab](#)  [View source on GitHub](#)

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.

```
1 # memory footprint support libraries/code
2 !ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUtil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
12 def printm():
13     process = psutil.Process(os.getpid())
14     print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Proc size: "
15     print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total {3:.0f}MB".format(gpu.memo
```

Source: [https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic\\_regression.ipynb](https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_regression.ipynb)

# AI+VDI POC

## ISAC+TKU Test

- AI+VDI POC Folder (3+1 ipynb) (v3.0.20181120)
  - <https://drive.google.com/open?id=1qHOemktbEmUz-ot8eFxlKbGwJvXlrjtc>
- run3models.ipynb
  - [https://colab.research.google.com/drive/1HQ1GrlqQUUPCct7\\_AVgoMwMrh0UqMm0f](https://colab.research.google.com/drive/1HQ1GrlqQUUPCct7_AVgoMwMrh0UqMm0f)
- AI+VDI POC ISAC+TKU Test Report
  - <https://docs.google.com/spreadsheets/d/1meMwqn15PSuTk6d5TgendDpdDX6L3OfHM4E0Slkq1Zk/edit?usp=sharing>

# Summary

- Prescriptive Analytics
- Model-Based Decision Making
- Structure of Mathematical Models for Decision Support
- Certainty, Uncertainty, and Risk
- Mathematical Programming Optimization
- Simulation

# References

- Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson.
- Jake VanderPlas (2016), Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media.