Colorado State University

Finance & Real Estate | Personal and Professional Business Explorations in Finance and Real Estate

Financial Risk Management

MSBA in Financial Risk Management

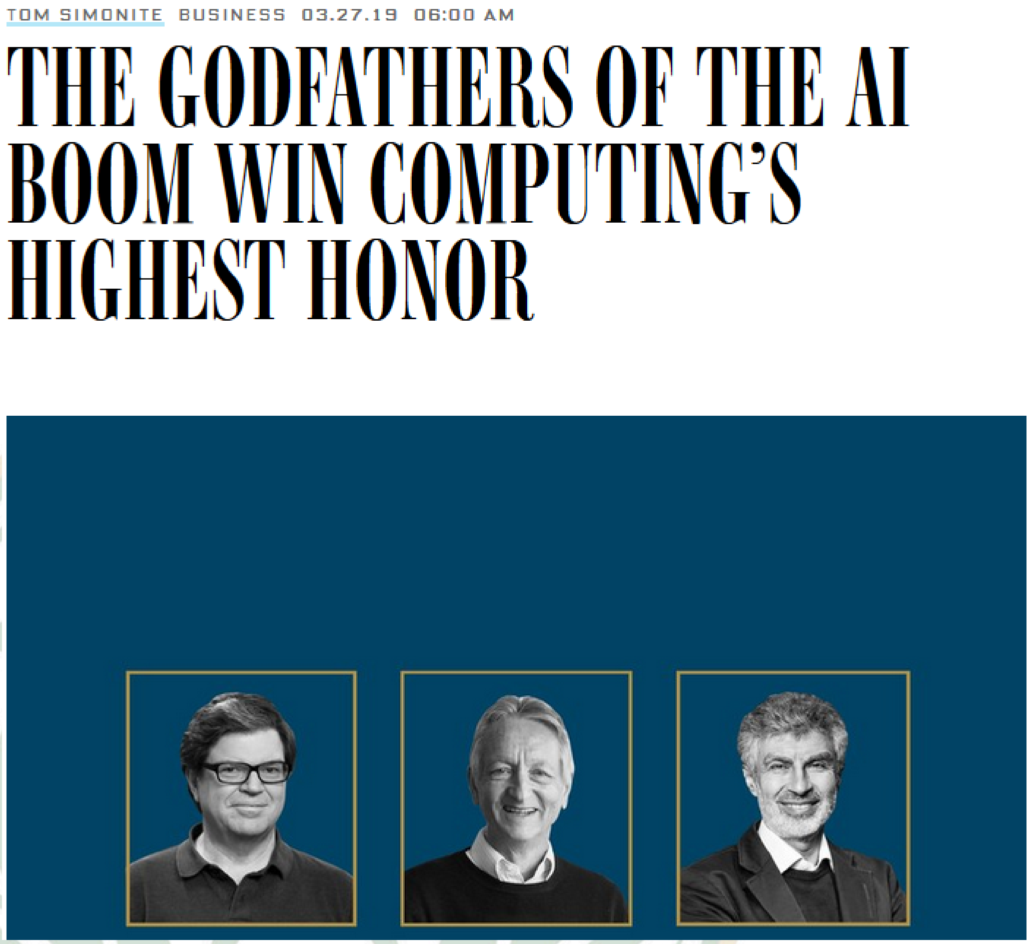
Tianyang Wang CSU

7—Learning Vector Quantization

7—Learning Vector Quantization

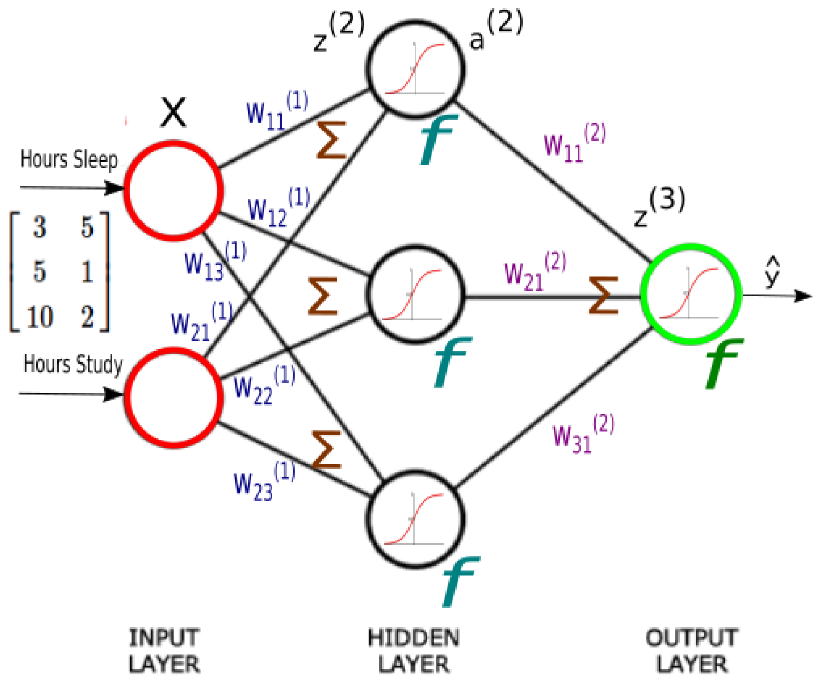
- A downside of K-Nearest Neighbors is that you need to hang on to your entire training dataset.
- The Learning Vector Quantization algorithm (or LVQ for short) is an artificial neural network algorithm that allows you to choose how many training instances to hang onto and learns exactly what those instances should look like.
- If you discover that KNN gives good results on your dataset try using LVQ to reduce the memory requirements of storing the entire training dataset.

7—Learning Vector Quantization



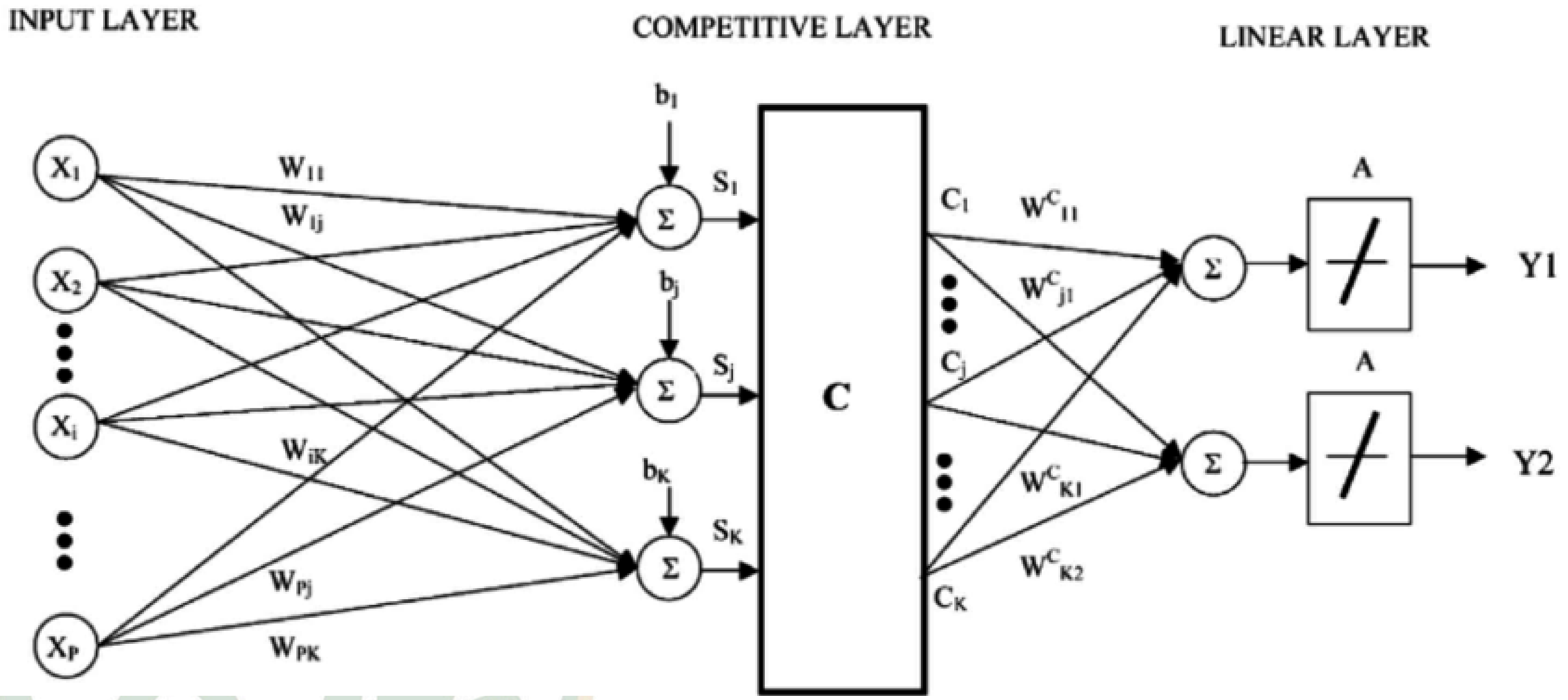
TOM SIMONITE BUSINESS 03.27.19 06:00 AM

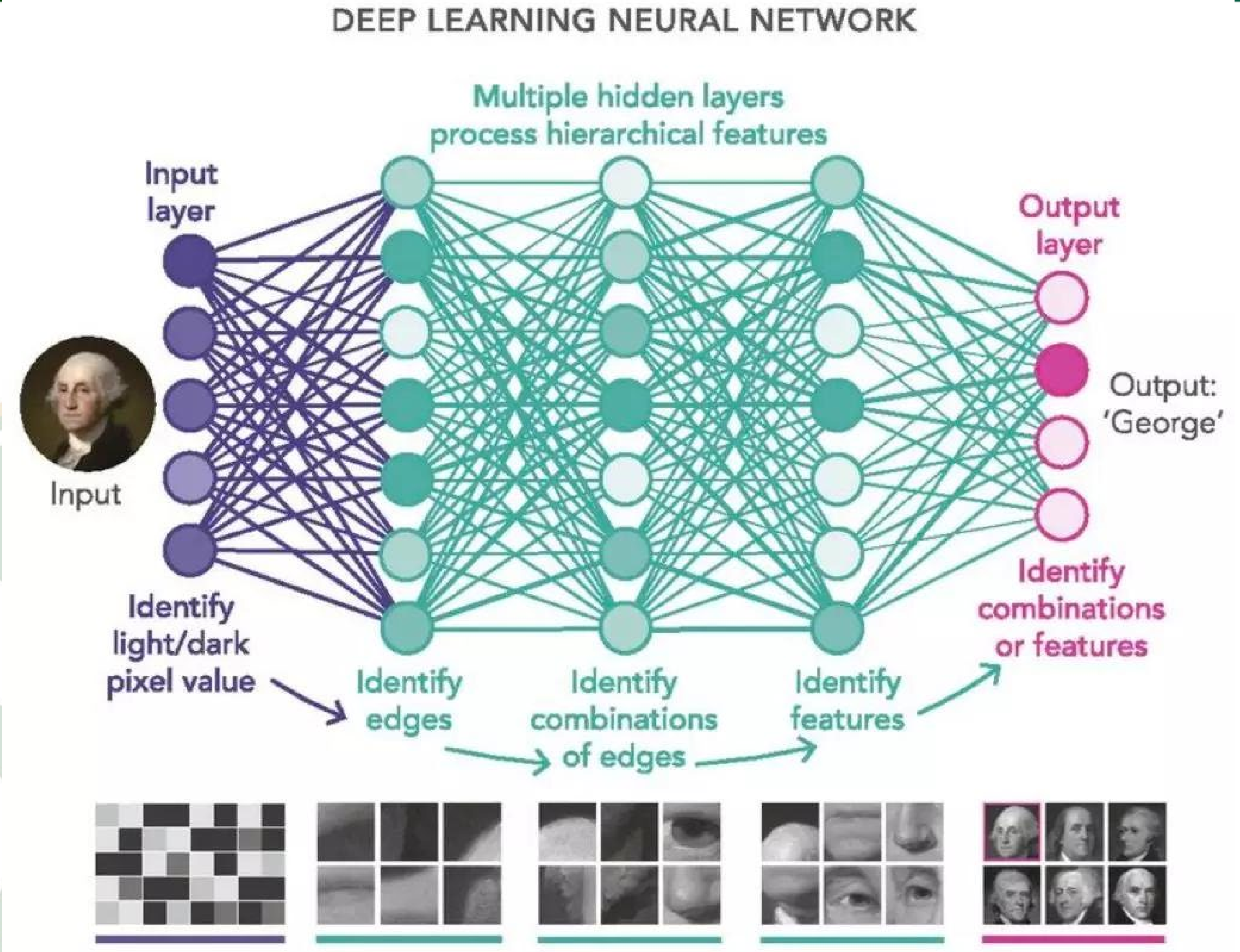
THE GODFATHERS OF THE AI BOOM WIN COMPUTING'S HIGHEST HONOR



Turing Award winners (from left to right) Yann LeCun, Geoff Hinton, and Yoshua Bengio reoriented artificial intelligence around neural networks.

7—Learning Vector Quantization




DEEP LEARNING NEURAL NETWORK
Multiple hidden layers process hierarchical features
Input layer
Output layer
Input
Output: 'George'
Identify light/dark pixel value
Identify edges
Identify combinations of edges
Identify features
Identify combinations or features

7—Learning Vector Quantization

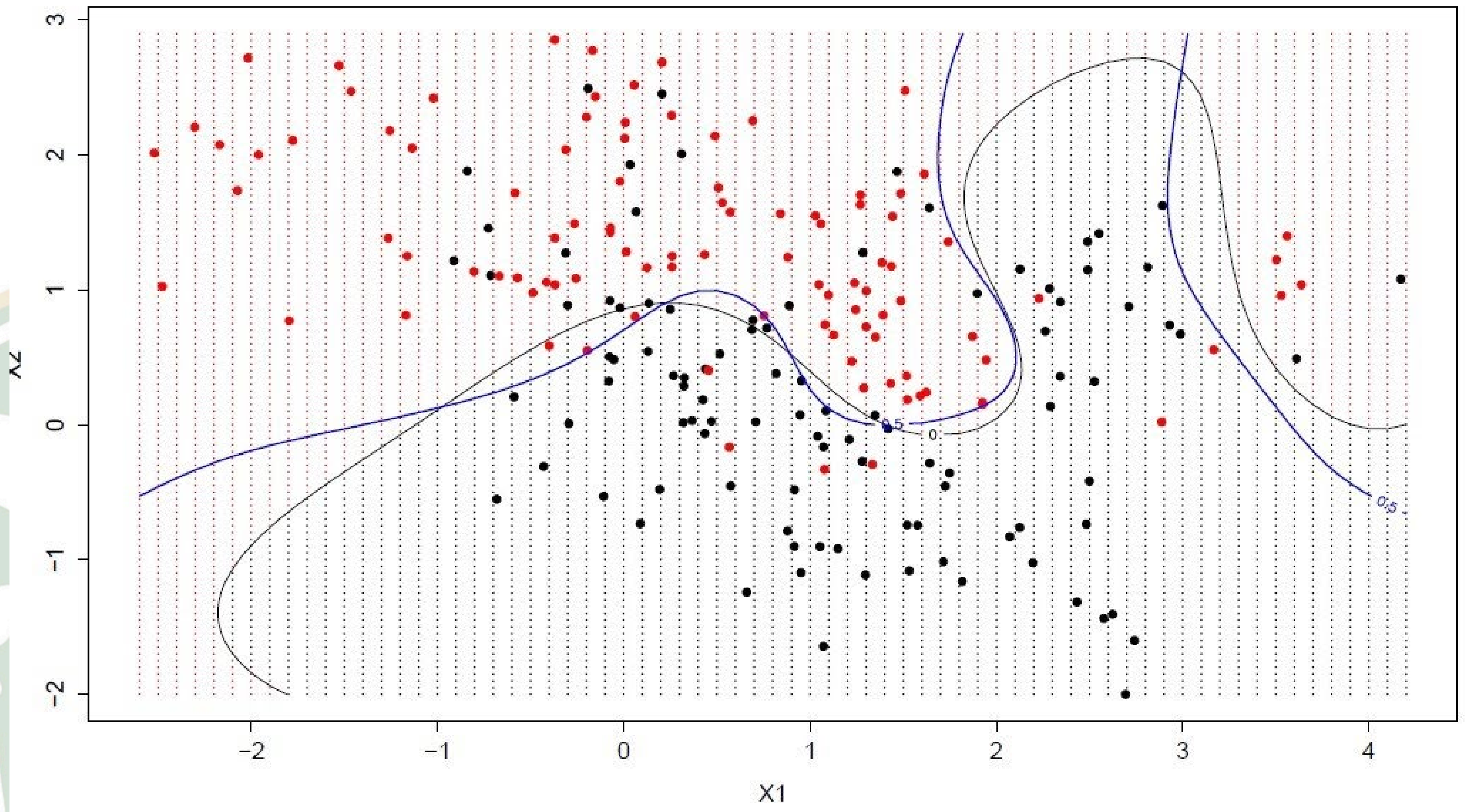
- The representation for LVQ is a collection of codebook vectors. These are selected randomly in the beginning and adapted to best summarize the training dataset over a number of iterations of the learning algorithm. After learned, the codebook vectors can be used to make predictions just like K-Nearest Neighbors.
- The most similar neighbor (best matching codebook vector) is found by calculating the distance between each codebook vector and the new data instance. The class value or (real value in the case of regression) for the best matching unit is then returned as the prediction. Best results are achieved if you rescale your data to have the same range, such as between 0 and 1.

8—Support Vector Machines

8—Support Vector Machines

- Support Vector Machines are perhaps one of the most popular and talked about machine learning algorithms.
- A hyperplane is a line that splits the input variable space.
- In SVM, a hyperplane is selected to best separate the points in the input variable space by their class, either class 0 or class 1.
- In two-dimensions, you can visualize this as a line and let's assume that all of our input points can be completely separated by this line. The SVM learning algorithm finds the coefficients that results in the best separation of the classes by the hyperplane.

8—Support Vector Machines



8—Support Vector Machines

- The distance between the hyperplane and the closest data points is referred to as the margin.
- The best or optimal hyperplane that can separate the two classes is the line that has the largest margin.
- Only these points are relevant in defining the hyperplane and in the construction of the classifier. These points are called the support vectors. They support or define the hyperplane. In practice, an optimization algorithm is used to find the values for the coefficients that maximizes the margin.
- SVM might be one of the most powerful out-of-the-box classifiers and worth trying on your dataset.

9—Bagging and Random Forest

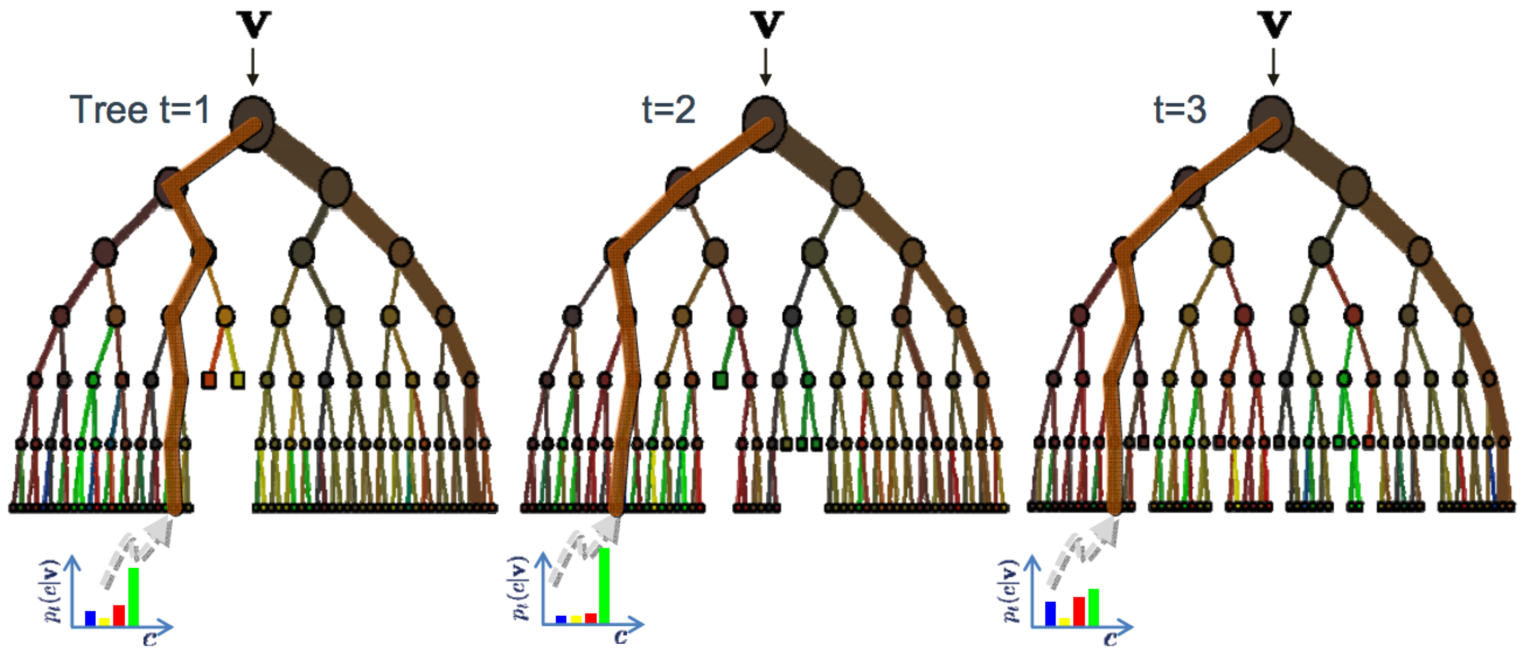
9—Bagging and Random Forest

- Random Forest is one of the most popular and most powerful machine learning algorithms. It is a type of ensemble machine learning algorithm called Bootstrap Aggregation or bagging.
- The bootstrap is a powerful statistical method for estimating a quantity from a data sample. Such as a mean. You take lots of samples of your data, calculate the mean, then average all of your mean values to give you a better estimation of the true mean value.

9—Bagging and Random Forest

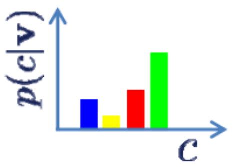
- In bagging, the same approach is used, but instead for estimating entire statistical models, most commonly in decision trees.
- Multiple samples of your training data are taken then models are constructed for each data sample. When you need to make a prediction for new data, each model makes a prediction and the predictions are averaged to give a better estimate of the true output value.

9—Bagging and Random Forest



The ensemble model

Forest output probability $p(c|\mathbf{v}) = \frac{1}{T}\sum_{t}^{T} p_t(c|\mathbf{v})$



9—Bagging and Random Forest

- Random forest is a tweak on this approach where decision trees are created so that rather than selecting optimal split points, suboptimal splits are made by introducing randomness.
- The models created for each sample of the data are therefore more different than they otherwise would be, but still accurate in their unique and different ways. Combining their predictions results in a better estimate of the true underlying output value.
- If you get good results with an algorithm with high variance (like decision trees), you can often get better results by bagging that algorithm.

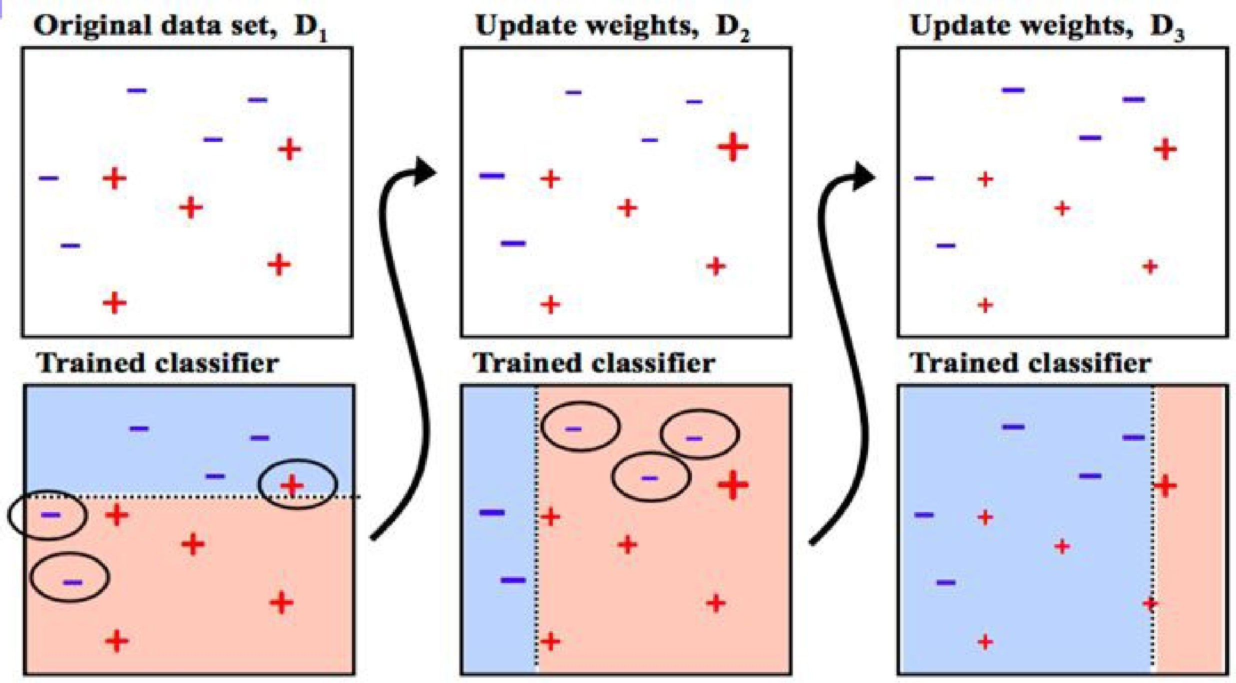
10—Boosting and AdaBoost

10—Boosting and AdaBoost

- Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added.
- AdaBoost was the first really successful boosting algorithm developed for binary classification. It is the best starting point for understanding boosting. Modern boosting methods build on AdaBoost, most notably stochastic gradient boosting machines.

10—Boosting and AdaBoost

Algorithm Adaboost - Example

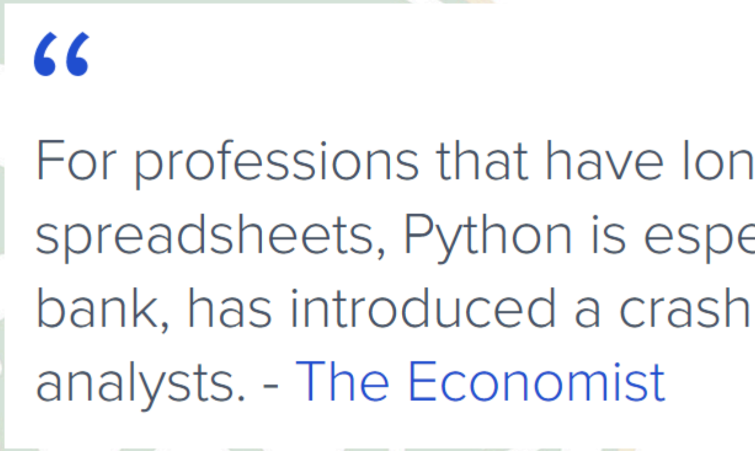


courtesy to Alexander Ihler http://sli.ics.uci.edu/Classes/2012F-273a?action=download&upname=10-ensembles.pdf

10—Boosting and AdaBoost

- AdaBoost is used with short decision trees. After the first tree is created, the performance of the tree on each training instance is used to weight how much attention the next tree that is created should pay attention to each training instance.
- Training data that is hard to predict is given more weight, whereas easy to predict instances are given less weight. Models are created sequentially one after the other, each updating the weights on the training instances that affect the learning performed by the next tree in the sequence. After all the trees are built, predictions are made for new data, and the performance of each tree is weighted by how accurate it was on training data.
- Because so much attention is put on correcting mistakes by the algorithm it is important that you have clean data with outliers removed.

Python in Finance



"

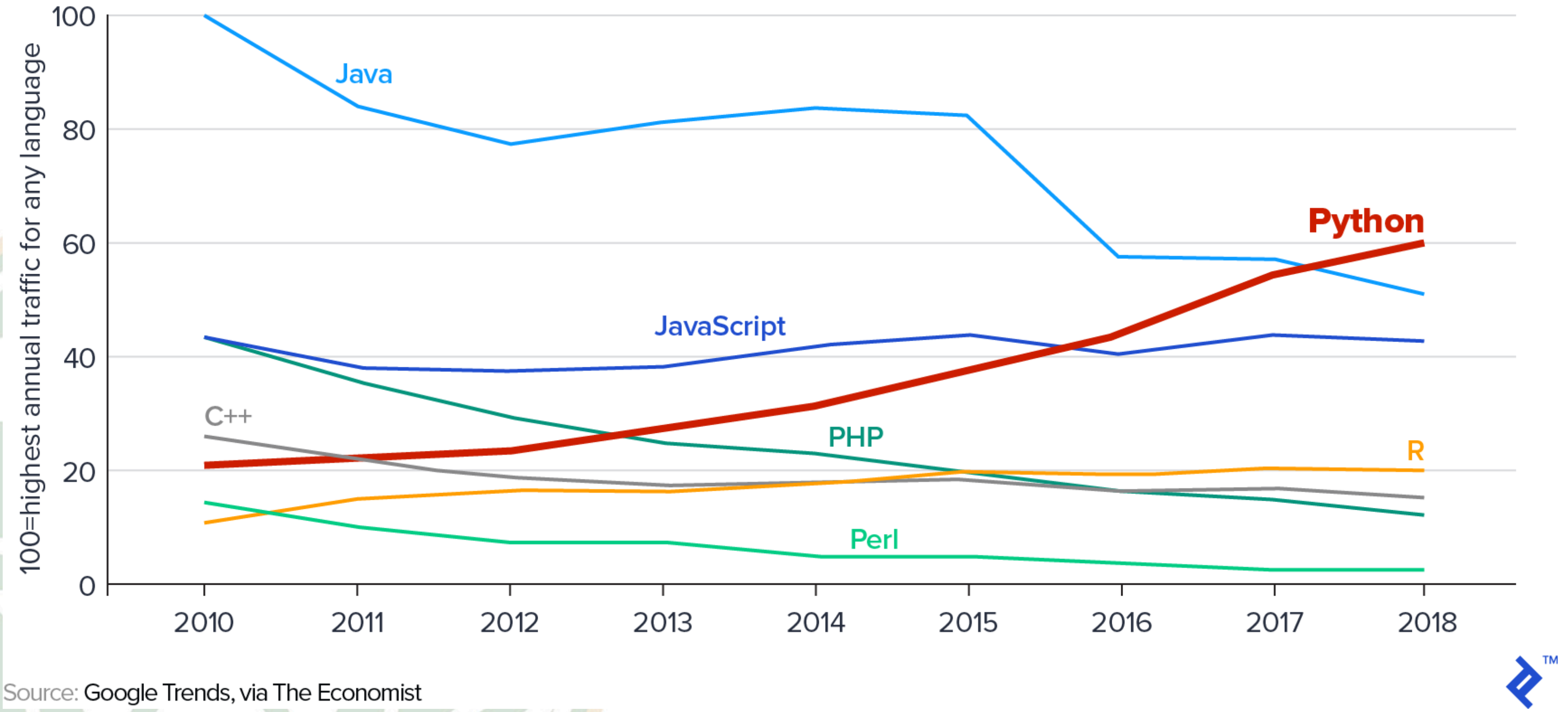
For professions that have long relied on trawling through spreadsheets, Python is especially valuable. Citigroup, an American bank, has introduced a crash course in Python for its trainee analysts. - The Economist

Python and Finance

- Python is a high-level programming language, meaning that it abstracts away and handles many of the technical aspects of programming, such as memory management, that must be explicitly handled in other languages. This makes Python easy to use for those without a technical background.
- Because the language was designed with readability and ease-of-use in mind, it is one of the easiest languages to learn. Python code is concise and close to plain English.
- Python is ideal for prototyping and rapid, iterative development. Its interactive interpreter tools provide environments where you can write and execute each line of code in isolation and see the results immediately.
- At the same time, Python is robust and performant, making it a viable choice also for core systems and larger applications.
- In addition to its large standard library of useful tools, Python has great third-party libraries for financial analysis and computing, such as the Pandas and NumPy libraries.

A Great Choice for Finance Professionals

Google Searches for Coding Languages in the USA



Source: Google Trends, via The Economist

Python Is a High-Level Programming Language

- A high-level programming language is one that abstracts away many of the details of the inner workings of the computer. A good example is memory management. Lower-level programming languages require a detailed understanding of the complexities of how the computer's memory is laid out, allocated and released, in addition to the time spent and lines of code required to handle tasks. Python abstracts away and handles many of these details automatically, leaving you to focus on what you want to accomplish.

It Is Concise

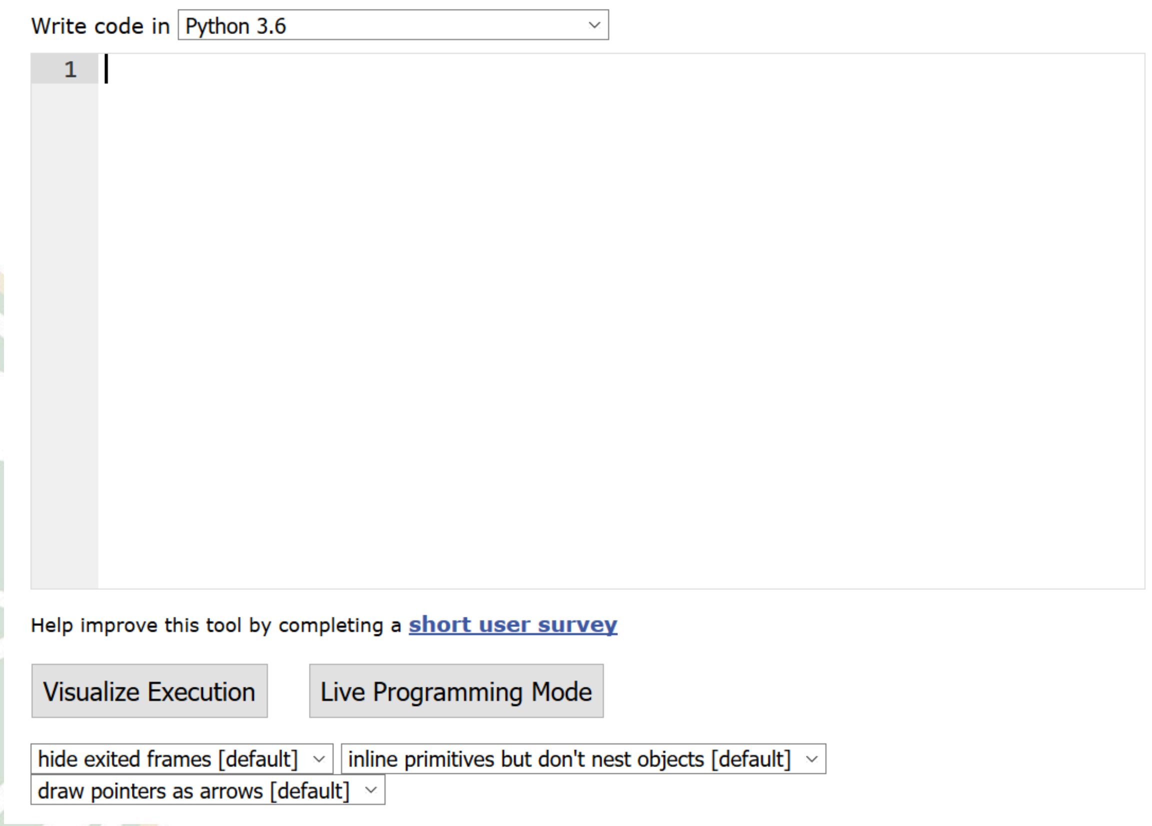
- Because Python is a high-level programming language, the code is more concise and almost entirely focused on the business logic of what you want to achieve, rather than technical implementation details. Language design choices contribute to this: as an example, Python doesn’t require the use of curly braces or semicolons to delineate functions, loops, and lines the way many other languages do, which makes it more concise and, as some argue, improves readability.
- Easy to Learn and Understand
- One observation that has influenced language design choices in Python is that programs are read more often than they are written. Python excels here as it’s code looks very close to plain English, especially if you name the different components of your script or program in a sensible manner.

- Suitable for Rapid, Iterative Development
- Comes with "Batteries Included:" The Python Standard Library
- Everything needed for basic operations is built right into the language, but in addition to that, the Python standard library has tools for working with files, media, networking, date and time information, and much more. This allows you to accomplish a wide variety of tasks without having to look for third-party packages.
- Great Third-party Libraries for Financial Analysis
- For finance professionals, Pandas with its *DataFrame* and *Series* objects, and Numpy with its *ndarray*are the workhorses of financial analysis with Python. Combined with matplotlib and other visualization libraries, you have great tools at your disposal to assist productivity.
- Python Is Free!
- Python is developed under an open source license making it free also for commercial use.

Colorado State University

Step-by-step Tutorial of Using Python and Finance Together

- http://www.pythontutor.com/visualize.html#mode=edit


Write code in Python 3.6

Help improve this tool by completing a short user survey

Visualize Execution
Live Programming Mode
hide exited frames [default]
inline primitives but don't nest objects [default]
draw pointers as arrows [default]

- Jupyter Notebook is an application that allows learners to create documents with Python code. The course team has leveraged the "Jupyter Notebook" feature in Coursera, and created Jupyter Notebook documents for every lecture video. Learners are able to practice the Python coding with the explanatory texts given in the document after going through the lecture videos.
- You may refer to the basic tutorial in https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook , which explains how to use Jupyter Notebooks.
- For learners who wish to install Jupyter Notebook locally in your computer, you may also follow the same tutorial for installation instructions.

Colorado State University

Packages

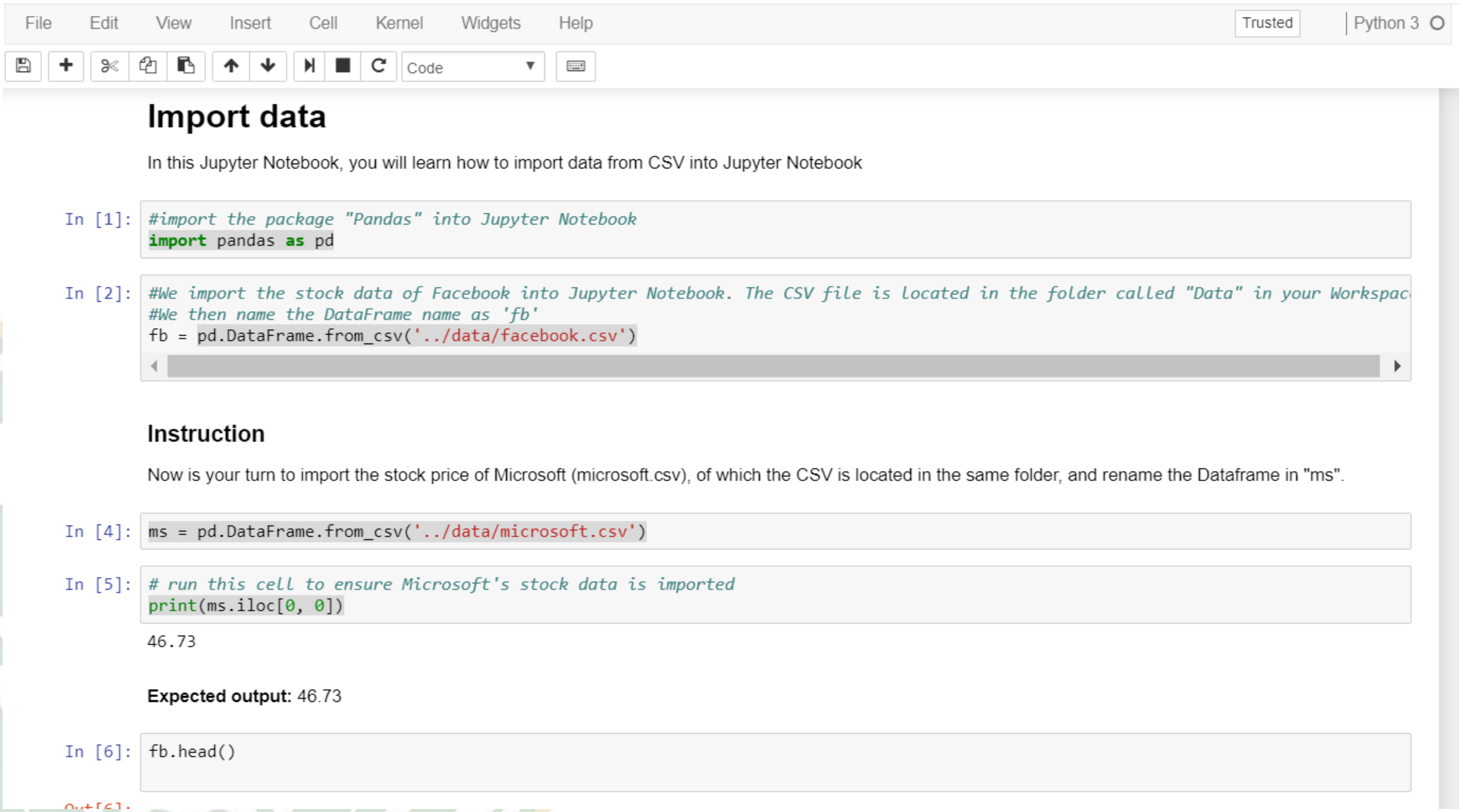
- Pandas is a python package, providing fast, flexible, and expressive data structures.
- It aims to be the fundamental high-level building blocks, for doing practical real-world data analysis.
- For example, DataFrame and the series from Pandas, are excellent data structures to store table and time series data. With DataFrame, we can easily pre-process data. For example, handling missing value, computing pairwise correlation.
- NumPy is a fundamental package for numerical computing of array and matrix. It is also very convenient tool for generating random numbers, which could be helpful if we want to shuffle data, or generate a dataset with normal distribution.
- Matplotlib is a plotting package, which produces high-quality figures with all kinds of customization.
- Statsmodels is a powerful library for statistician. It contains modules for regression and time series analysis.

Data Scientists in Finance

- Alongside all this, you'll need a good understanding of optimization (underpinned by solid linear algebra and calculus learnt in school), of statistical inference, simulation, multivariate analysis and proper data visualization.
- If you possess such training, then understanding techniques such as support vector machines, neural networks, random forests and gradient boosting are merely a hop, skip and a jump away. I might just throw in NLP as well.
- With all this, your data science career will be underway. Good luck!

Basics of DataFrame

- In this Jupyter Notebook, you will practice the following basics of DataFrame:
- 1. Import stock data (in csv format) into a new DataFrame
- 2. Display the size of a DataFrame using ".shape"
- 3. Display the summary statistics of a DataFrame using ".describe()"
- 4. Slice row(s) of data of a DataFrame using "Selection by label" - loc and "Selection of position - iloc"
- 5. Plot the data of a DataFrame



Import data

In this Jupyter Notebook, you will learn how to import data from CSV into Jupyter Notebook

```
In [1]: #import the package "Pandas" into Jupyter Notebook
        import pandas as pd
```

```
In [2]: #We import the stock data of Facebook into Jupyter Notebook. The CSV file is located in the folder called "Data" in your Workspac
        #We then name the DataFrame name as 'fb'
        fb = pd.DataFrame.from_csv('../data/facebook.csv')
```

Instruction

Now is your turn to import the stock price of Microsoft (microsoft.csv), of which the CSV is located in the same folder, and rename the Dataframe in "ms".

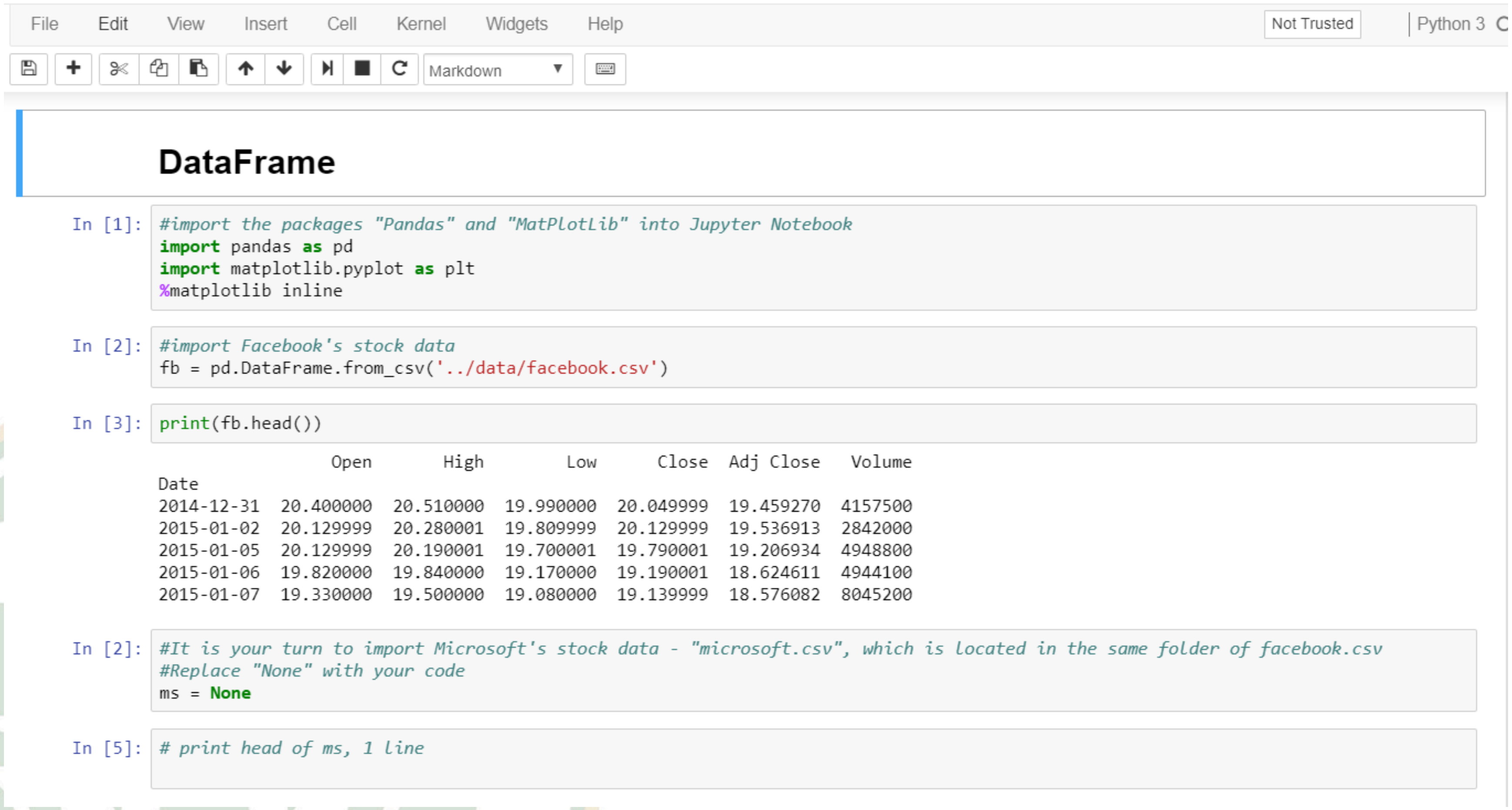
```
In [4]: ms = pd.DataFrame.from_csv('../data/microsoft.csv')
```

```
In [5]: # run this cell to ensure Microsoft's stock data is imported
        print(ms.iloc[0, 0])

46.73
```

Expected output: 46.73

```
In [6]: fb.head()
```



DataFrame

```
In [1]: #import the packages "Pandas" and "MatPlotLib" into Jupyter Notebook
        import pandas as pd
        import matplotlib.pyplot as plt
        %matplotlib inline
```

```
In [2]: #import Facebook's stock data
        fb = pd.DataFrame.from_csv('../data/facebook.csv')
```

```
In [3]: print(fb.head())

                 Open       High        Low      Close  Adj Close   Volume
Date
2014-12-31  20.400000  20.510000  19.990000  20.049999  19.459270  4157500
2015-01-02  20.129999  20.280001  19.809999  20.129999  19.536913  2842000
2015-01-05  20.129999  20.190001  19.700001  19.790001  19.206934  4948800
2015-01-06  19.820000  19.840000  19.170000  19.190001  18.624611  4944100
2015-01-07  19.330000  19.500000  19.080000  19.139999  18.576082  8045200
```

```
In [2]: #It is your turn to import Microsoft's stock data - "microsoft.csv", which is located in the same folder of facebook.csv
        #Replace "None" with your code
        ms = None
```

```
In [5]: # print head of ms, 1 line
```

Create features and columns in DataFrame

- In this Jupyter Notebook, you will practice the following codes to create new features/columns in a DataFrame
- 1. Create new columns in the DataFrame by arithmetic calculation - Price Difference and Daily Return
- 2. Create a new column using List Comprehension - Direction
- 3. Create a new column using Rolling Window Calculation - Any days of Moving Average