

Financial Econometrics and Machine Learning

1111AIFQA07

MBA, IM, NTPU (M6132) (Fall 2022)

Tue 2, 3, 4 (9:10-12:00) (B8F40)



<https://meet.google.com/paj-zhji-mya>



Min-Yuh Day, Ph.D,
Associate Professor

Institute of Information Management, National Taipei University

<https://web.ntpu.edu.tw/~myday>



Syllabus

Week	Date	Subject/Topics
1	2022/09/13	Introduction to Artificial Intelligence in Finance and Quantitative Analysis
2	2022/09/20	AI in FinTech: Metaverse, Web3, DeFi, NFT, Financial Services Innovation and Applications
3	2022/09/27	Investing Psychology and Behavioral Finance
4	2022/10/04	Event Studies in Finance
5	2022/10/11	Case Study on AI in Finance and Quantitative Analysis I
6	2022/10/18	Finance Theory

Syllabus

Week	Date	Subject/Topics
7	2022/10/25	Data-Driven Finance
8	2022/11/01	Midterm Project Report
9	2022/11/08	Financial Econometrics and Machine Learning
10	2022/11/15	AI-First Finance
11	2022/11/22	Industry Practices of AI in Finance and Quantitative Analysis
12	2022/11/29	Case Study on AI in Finance and Quantitative Analysis II

Syllabus

Week	Date	Subject/Topics
13	2022/12/06	Deep Learning in Finance; Reinforcement Learning in Finance
14	2022/12/13	Algorithmic Trading; Risk Management; Trading Bot and Event-Based Backtesting
15	2022/12/20	Final Project Report I
16	2022/12/27	Final Project Report II
17	2023/01/03	Self-learning
18	2023/01/10	Self-learning

**Financial
Econometrics
and
Machine Learning**

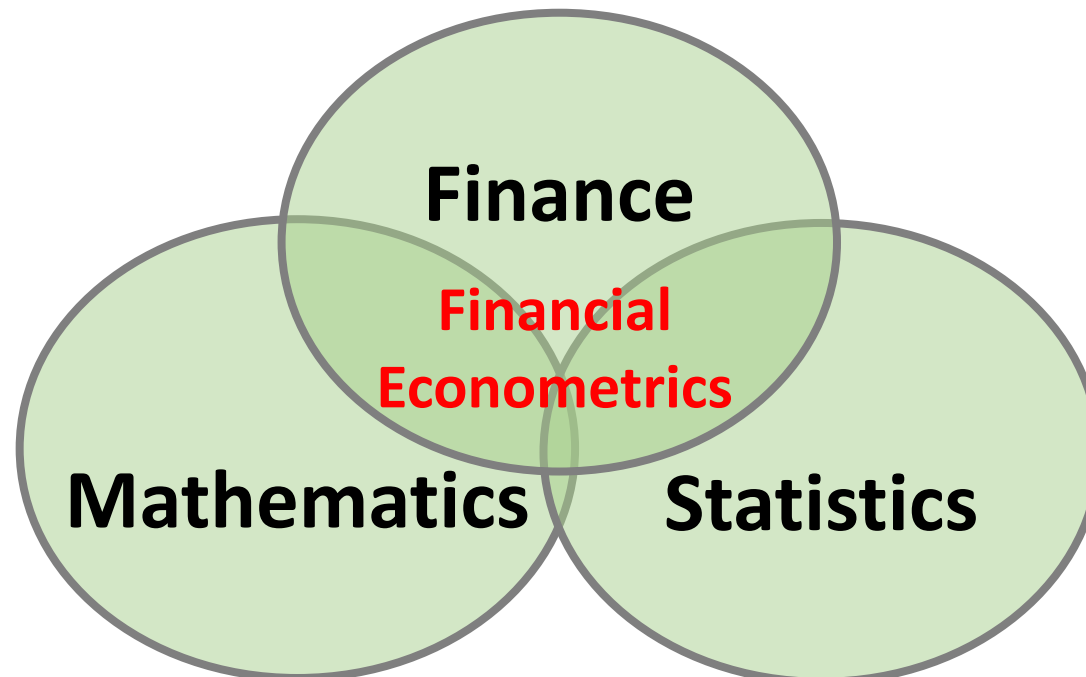
Outline

- **Financial Econometrics**
 - **Financial Theories**
 - **OLS Regression**
- **Machine Learning**
 - **Learning**
 - **Evaluation**
 - **Bias and variance**
 - **Cross-validation**

Financial Econometrics

Financial Econometrics

- The discipline at the intersection of **mathematics, statistics, and finance** that applies such methods to **financial market data** is typically called **financial econometrics**.



Financial Econometrics

(Chris Brooks, 2019)

- **Financial econometrics**
 - the application of **statistical techniques** to problems in **finance**
- Financial econometrics can be useful for testing theories in finance, determining asset prices or returns, testing hypotheses concerning the relationships between variables, examining the effect on financial markets of changes in economic conditions, forecasting future values of financial variables and for financial decision-making.

Financial Econometrics

- **[Financial] econometrics** is the **quantitative application of statistical and mathematical models** using **[financial] data** to develop financial theories or test existing hypotheses in finance and to forecast future trends from historical data.
- It subjects real-world **[financial] data** to statistical trials and then compares and contrasts the results against the **[financial] theory or theories** being tested.

Topics of Financial Econometrics

(Oliver Linton, 2019)

- 1. Econometric**
- 2. Return Predictability and the Efficient Markets Hypothesis**
- 3. Robust Tests and Tests of Nonlinear Predictability of Returns**
- 4. Empirical Market Microstructure**
- 5. Event Study Analysis**
- 6. Portfolio Choice and Testing the Capital Asset Pricing Model**
- 7. Multifactor Pricing Models**

Topics of Financial Econometrics

(Oliver Linton, 2019)

- 8. Present Value Relations**
- 9. Intertemporal Equilibrium Pricing**
- 10. Volatility**
- 11. Continuous Time Processes**
- 12. Yield Curve**
- 13. Risk Management and Tail Estimation**

Applications of Financial Econometrics

(Chris Brooks, 2019)

- 1. Testing whether financial markets are weak-form informationally efficient**
- 2. Testing whether the capital asset pricing model (CAPM) or arbitrage pricing theory (APT) represent superior models for the determination of returns on risky assets**
- 3. Measuring and forecasting the volatility of bond returns**
- 4. Explaining the determinants of bond credit ratings used by the ratings agencies**
- 5. Modelling long-term relationships between prices and exchange rates**

Applications of Financial Econometrics

(Chris Brooks, 2019)

- 6. Determining the optimal hedge ratio for a spot position in oil**
- 7. Testing technical trading rules to determine which makes the most money**
- 8. Testing the hypothesis that earnings or dividend announcements have no effect on stock prices**
- 9. Testing whether spot or futures markets react more rapidly to news**
- 10. Forecasting the correlation between the stock indices of two countries**

Machine Learning and Financial Econometrics

- **ML and DL methods** are able to **discover statistical inefficiencies and even economic inefficiencies** that are not discoverable by **traditional econometric methods**, such as multivariate OLS regression.

Normative Financial Theories

- **Normative financial theories** mostly rely on assumptions and axioms in combination with deduction as the major analytical method to arrive at their central results.
 - **Expected utility theory (EUT)** assumes that agents have the same utility function no matter what state of the world unfolds and that they maximize expected utility under conditions of uncertainty.
 - **Mean-variance portfolio (MVP)** theory describes how investors should invest under conditions of uncertainty assuming that only the expected return and the expected volatility of a portfolio over one period count.

Normative Financial Theories

- The **capital asset pricing model (CAPM)** assumes that only the nondiversifiable market risk explains the expected return and the expected volatility of a stock over one period.
- **Arbitrage pricing theory (APT)** assumes that a number of identifiable risk factors explains the expected return and the expected volatility of a stock over time; admittedly, compared to the other theories, the formulation of APT is rather broad and allows for wide-ranging interpretations.

Financial Econometrics and Regression

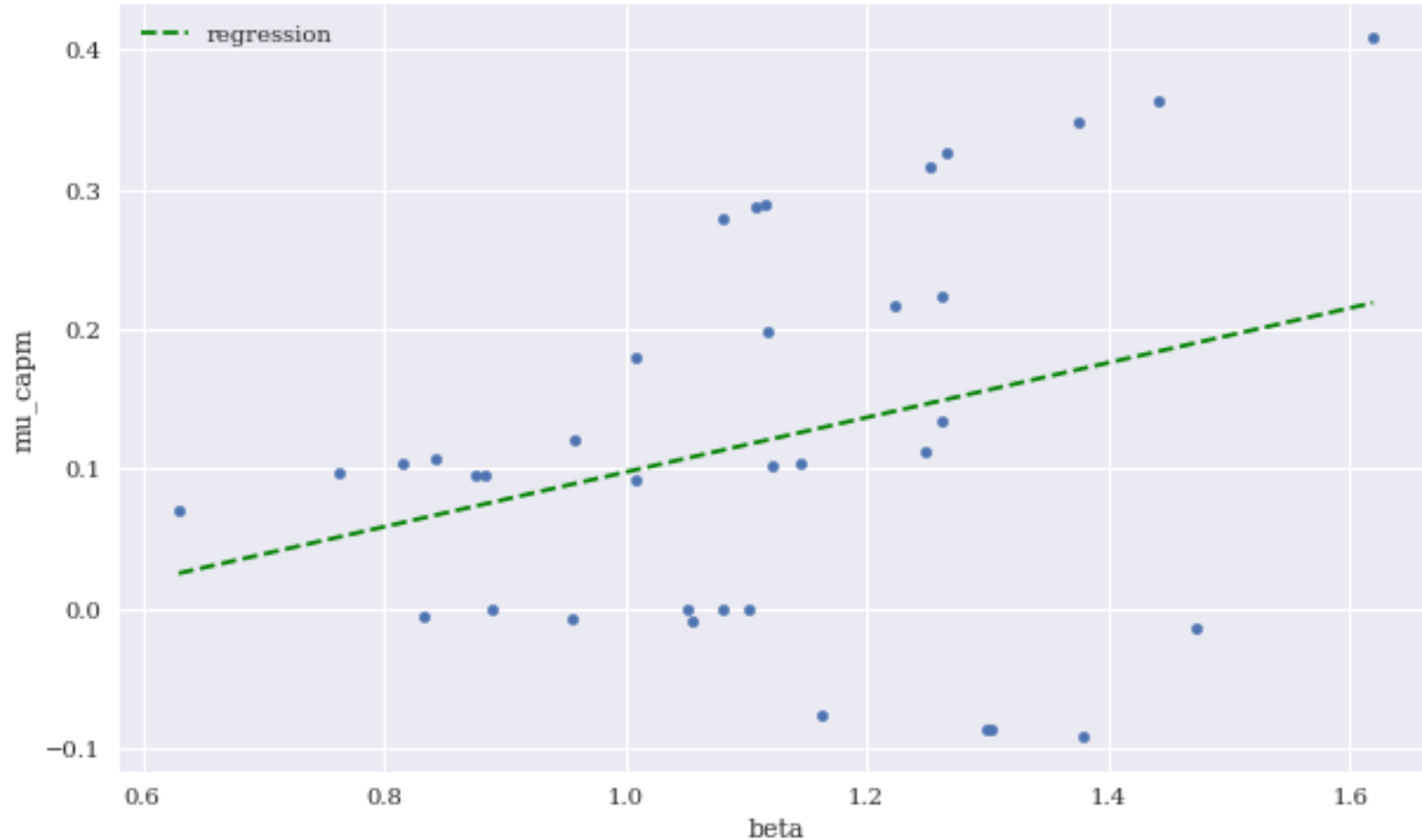
- One of the major tools in **financial econometrics** is **regression**, in both its univariate and multivariate forms
 - $y = \alpha + \beta x$
 - $y = \alpha + \beta_1 x_1 + \beta_2 x_2$
 - $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$
- **Regression** is also a central tool in **statistical learning** in general

CAPM and APT

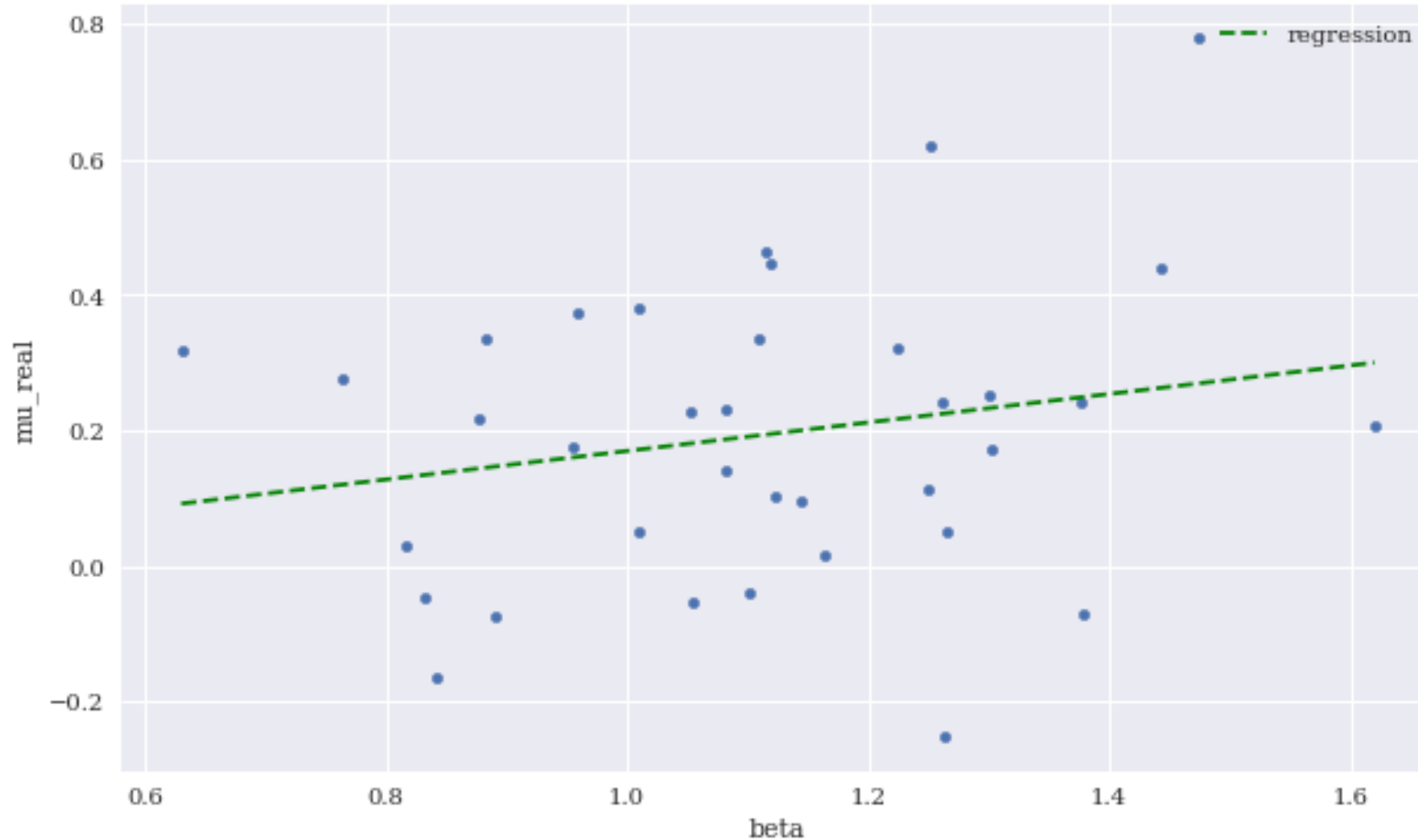
OLS regression

- Both the **CAPM** and the **APT** relate the **output variables** with the relevant **input factors** in **linear** fashion.
- From an econometric point of view, both models are implemented based on linear **ordinary least-squares (OLS) regression**.
- **CAPM**: univariate linear OLS regression
- **APT**: multivariate OLS regression

Expected CAPM return versus beta (including linear regression)



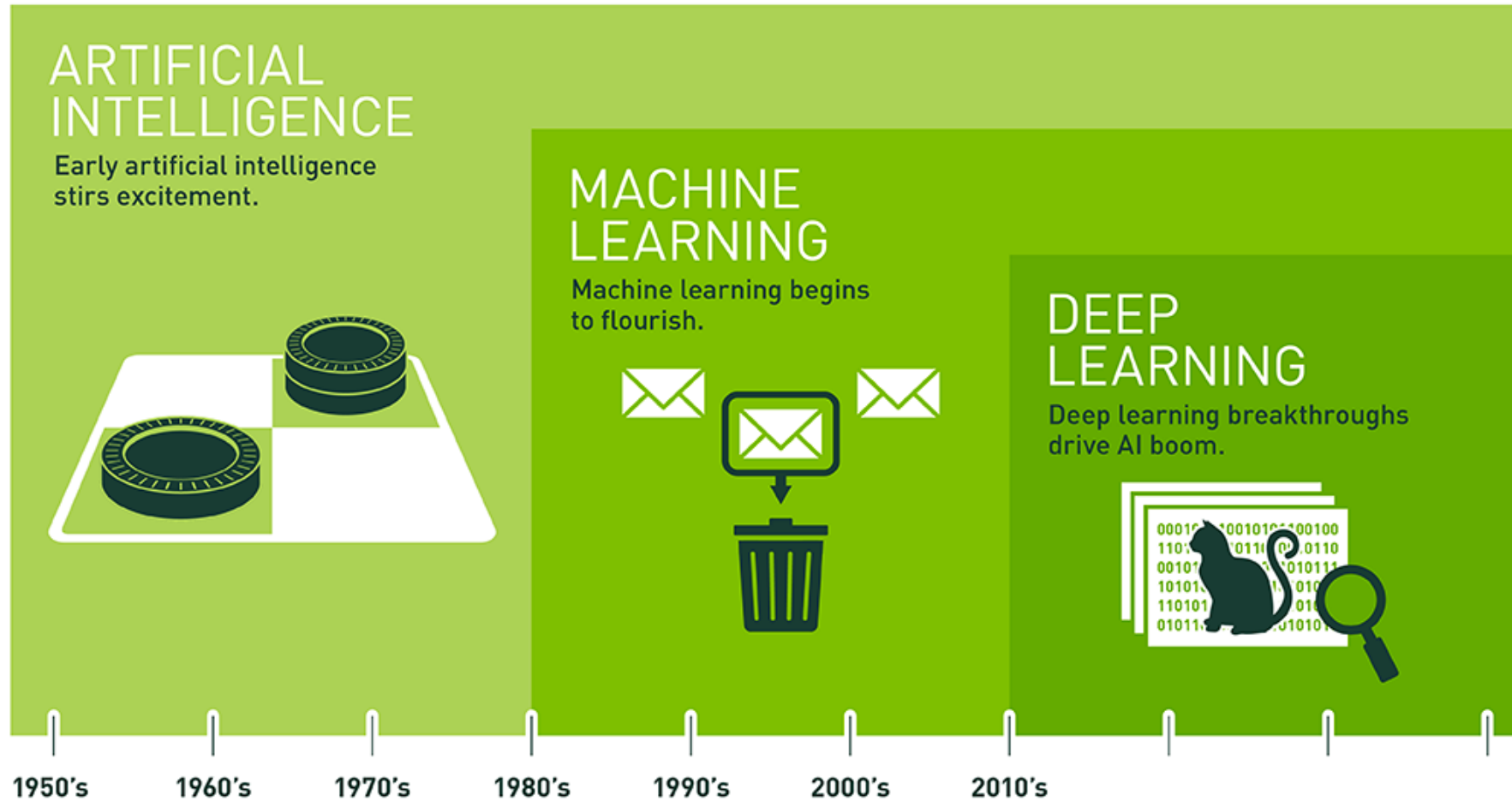
Expected CAPM return versus beta (including linear regression)



Machine Learning

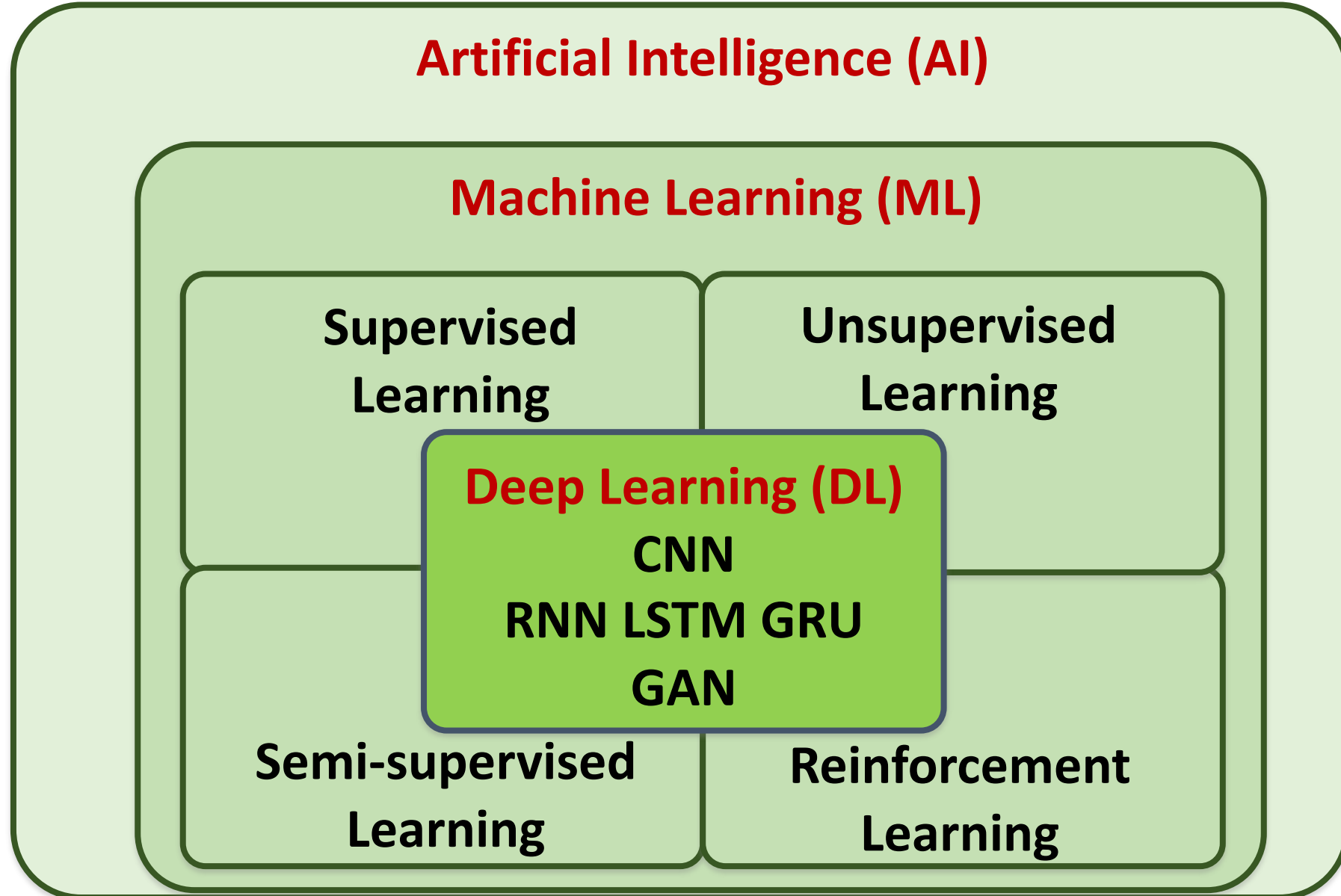
Artificial Intelligence

Machine Learning & Deep Learning

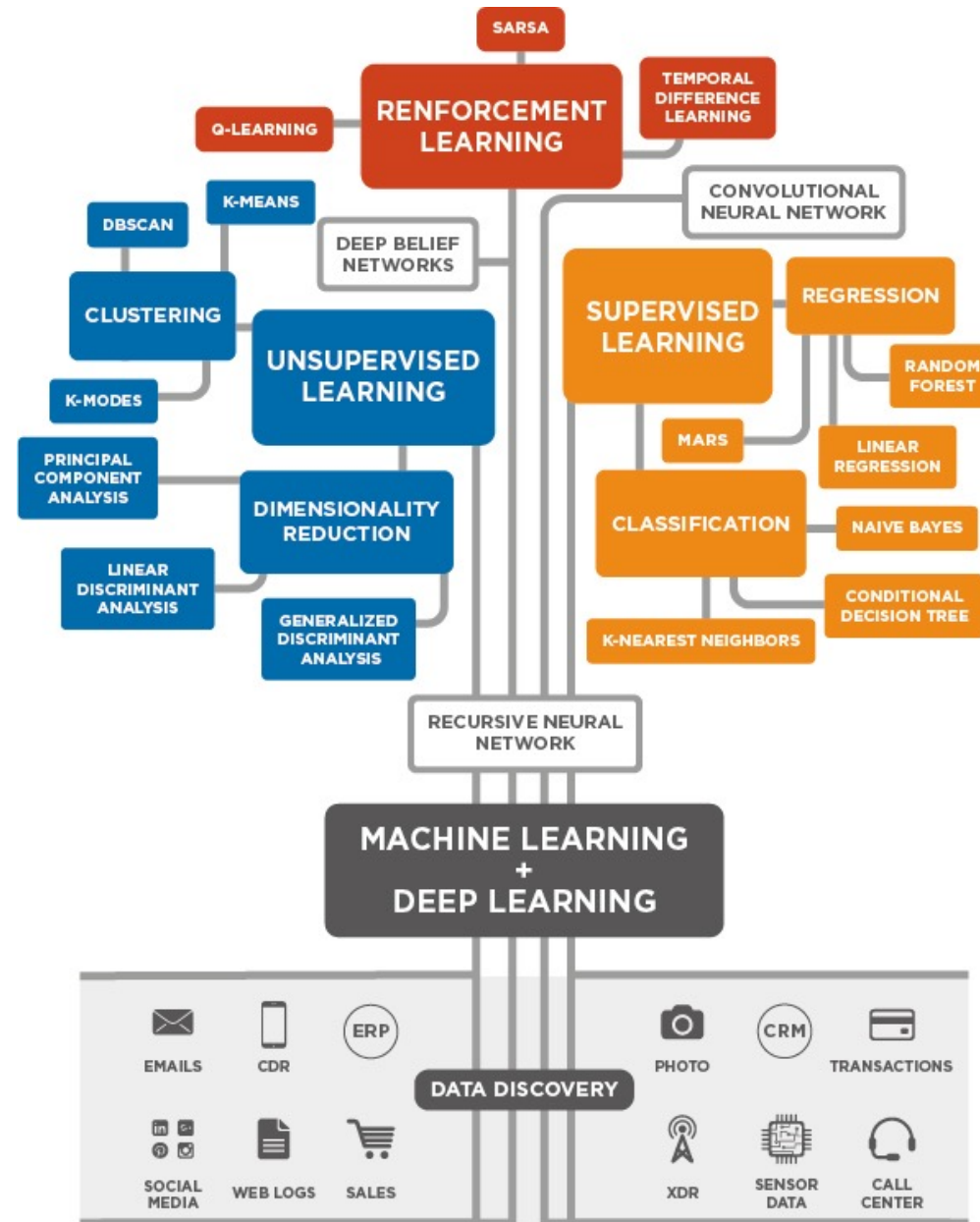


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

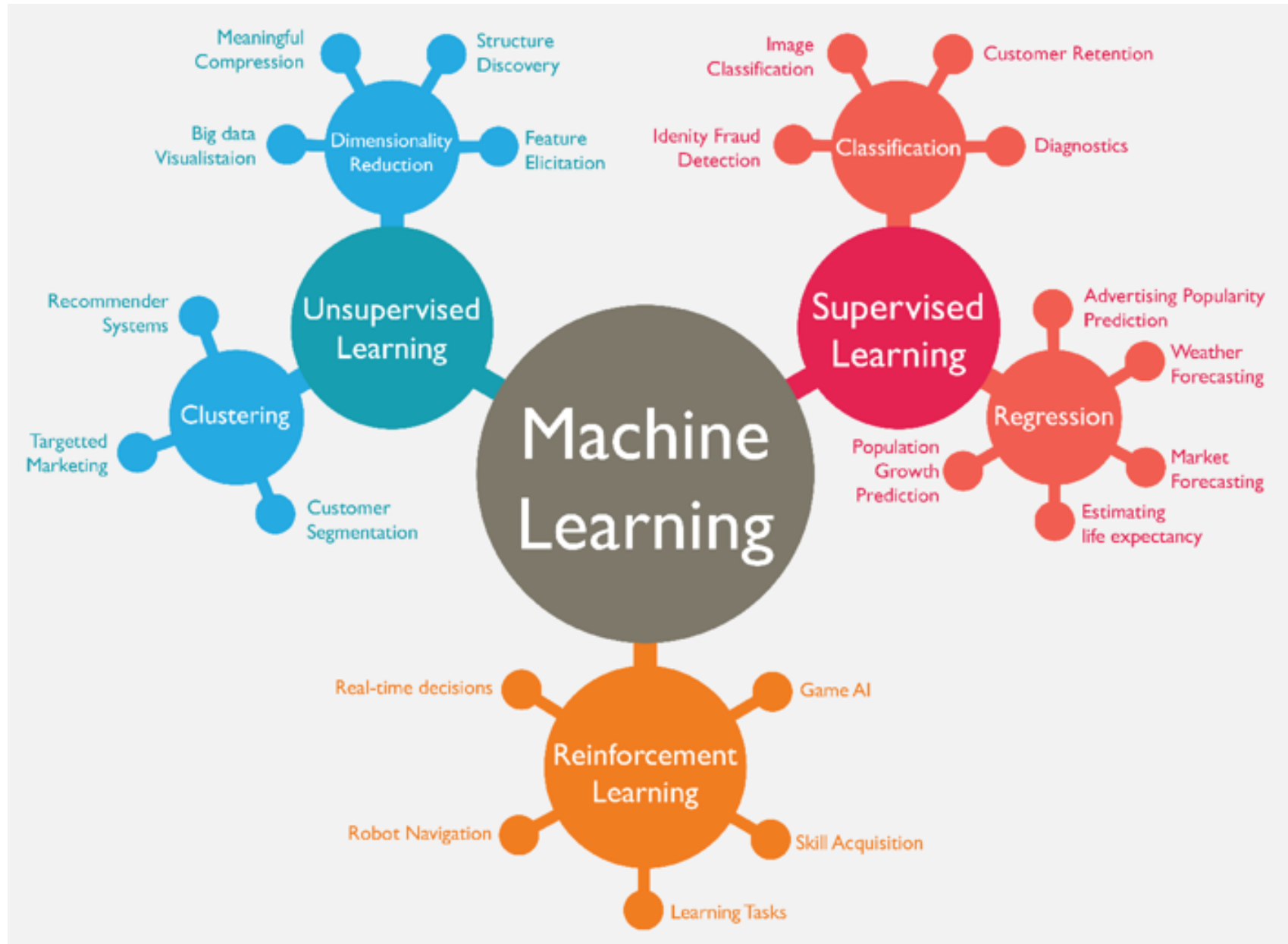
AI, ML, DL



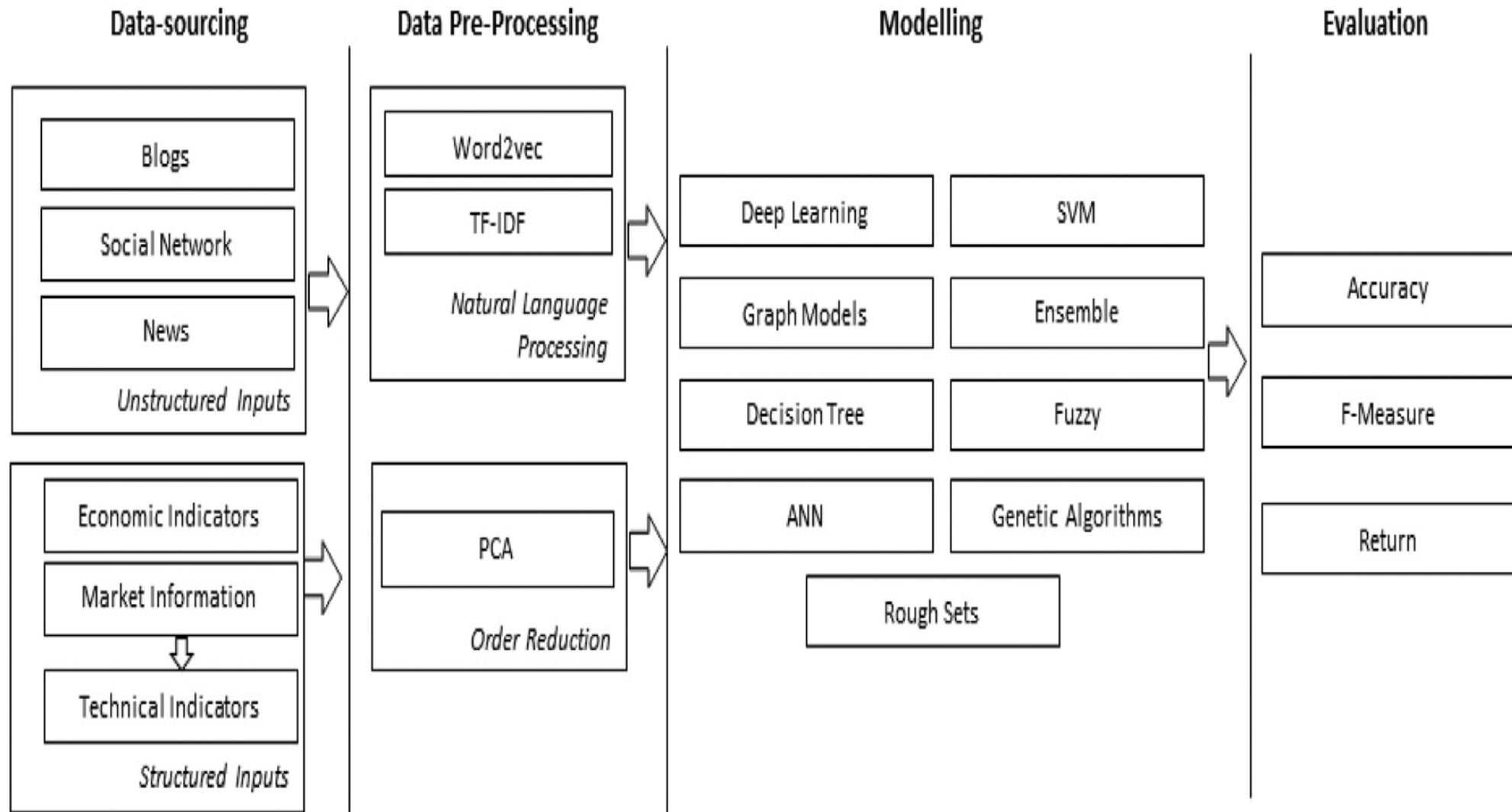
3 Machine Learning Algorithms



Machine Learning (ML)



Stock Market Movement Forecast: ML Phases of the stock market modeling



Machine Learning

- **Learning**
- **Data: Features, Labels**
- **Success (Loss Function): MSE**
- **Capacity (Model Fit)**
- **Evaluation**
- **Bias and variance**
- **Cross-validation**

Learning (Mitchell, 1997)

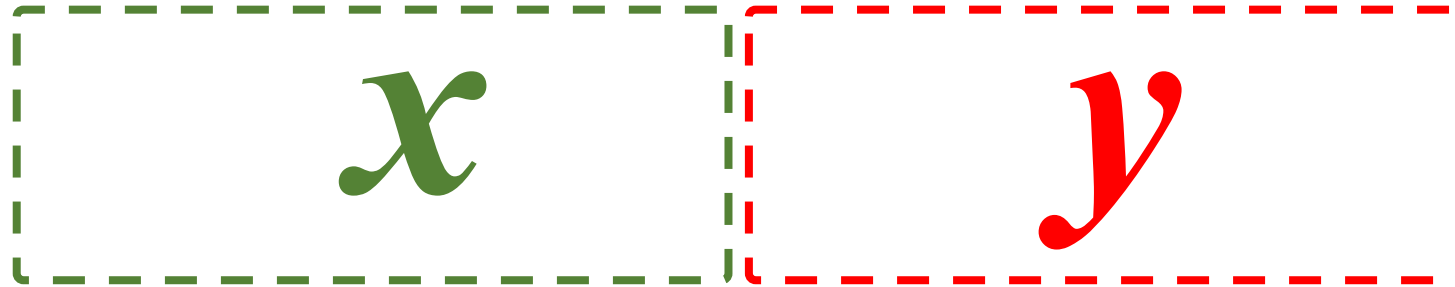
- A computer program is said to **learn** from **experience E** with respect to some class of tasks **T** and **performance measure P**, if its performance at tasks in **T**, as measured by **P**, improves with **experience E**.

Machine Learning

Supervised Learning (Classification)

Learning from Examples

$$y = f(x)$$

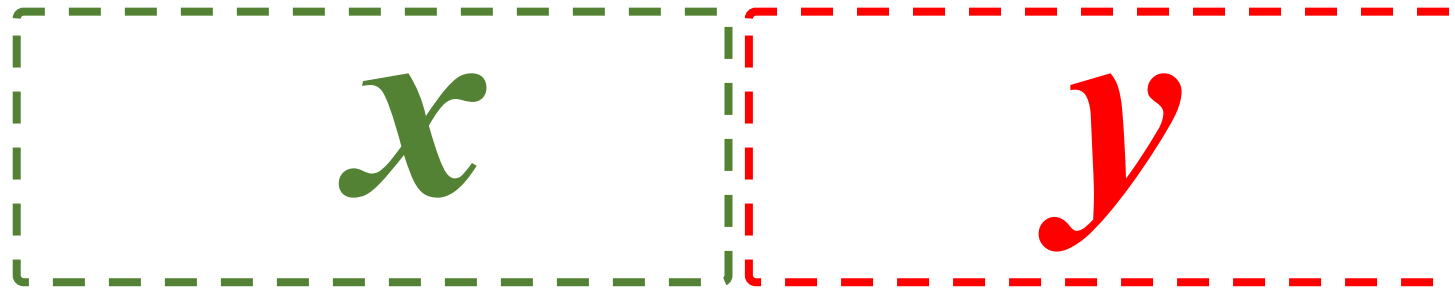


Machine Learning

Supervised Learning (Classification)

Learning from Examples

$$y = f(x)$$



input

Output
label

Machine Learning

Supervised Learning (Classification)

Learning from Examples

$$y = f(x)$$

```
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
7.0, 3.2, 4.7, 1.4, Iris-versicolor
6.4, 3.2, 4.5, 1.5, Iris-versicolor
6.9, 3.1, 4.9, 1.5, Iris-versicolor
6.3, 3.3, 6.0, 2.5, Iris-virginica
5.8, 2.7, 5.1, 1.9, Iris-virginica
7.1, 3.0, 5.9, 2.1, Iris-virginica
```


Machine Learning

Supervised Learning (Classification)

Learning from Examples

$$y = f(x)$$

<i>Example</i>	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
	7.0, 3.2, 4.7, 1.4, Iris-versicolor
	6.4, 3.2, 4.5, 1.5, Iris-versicolor
	6.9, 3.1, 4.9, 1.5, Iris-versicolor
	6.3, 3.3, 6.0, 2.5, Iris-virginica
	5.8, 2.7, 5.1, 1.9, Iris-virginica
	7.1, 3.0, 5.9, 2.1, Iris-virginica

Machine Learning

Supervised Learning (Classification)

Learning from Examples

$$y = f(x)$$

Example

x

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
7.0, 3.2, 4.7, 1.4	Iris-versicolor
6.4, 3.2, 4.5, 1.5	Iris-versicolor
6.9, 3.1, 4.9, 1.5	Iris-versicolor
6.3, 3.3, 6.0, 2.5	Iris-virginica
5.8, 2.7, 5.1, 1.9	Iris-virginica
7.1, 3.0, 5.9, 2.1	Iris-virginica

y

Time Series Data

[10, 20, 30, 40, 50, 60, 70, 80, 90]

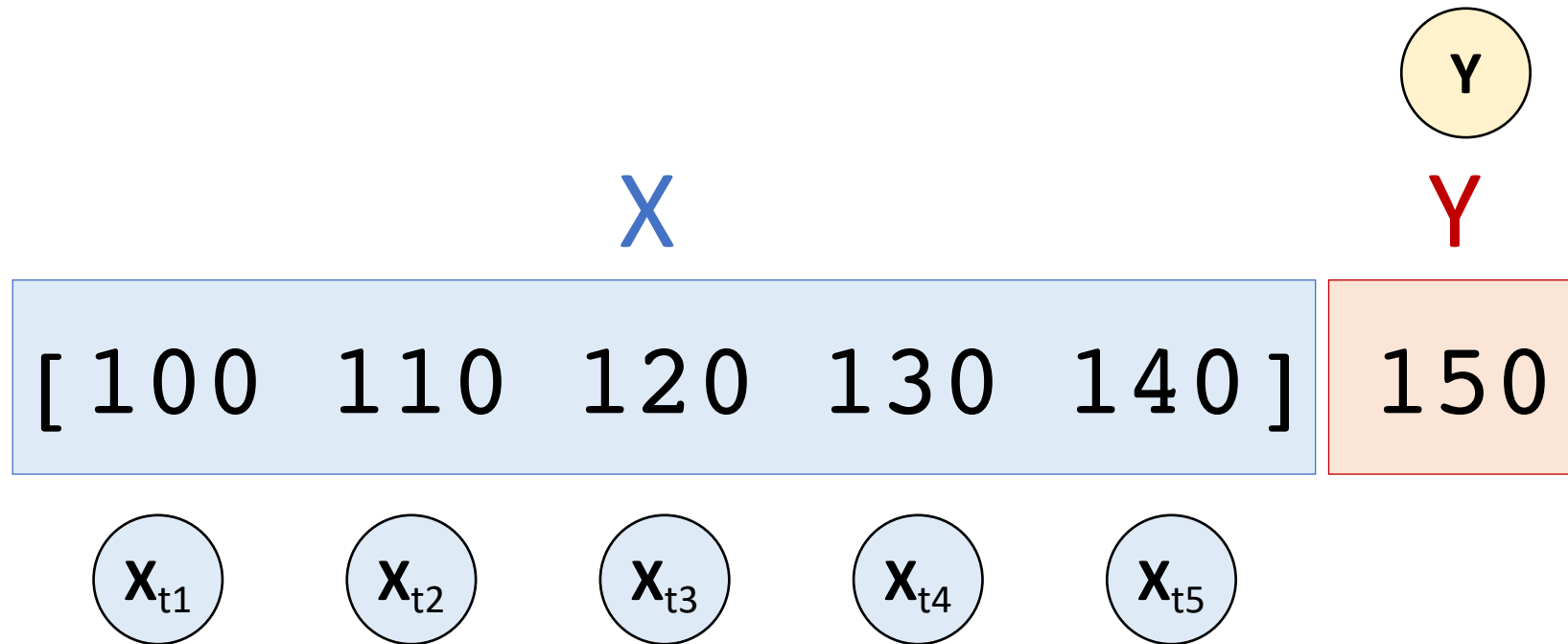
X

Y

[10	20	30]	40
[20	30	40]	50
[30	40	50]	60
[40	50	60]	70
[50	60	70]	80
[60	70	80]	90

Time Series Data

[100, 110, 120, 130, 140, 150]



The Theory of Learning

- How can we be sure that our **learned hypothesis** will **predict** well for previously unseen inputs?
 - How do we know that the **hypothesis h** is close to the **target function f** if we don't know what is?
- How many **examples** do we need to get a good **h** ?
- What **hypothesis space** should we use?
- If the hypothesis space is very complex, can we even find the best **h** or do we have to settle for a **local maximum**?
- How **complex** should **h** be?
- How do we avoid **overfitting**?

Linear function

$$y = f(x)$$

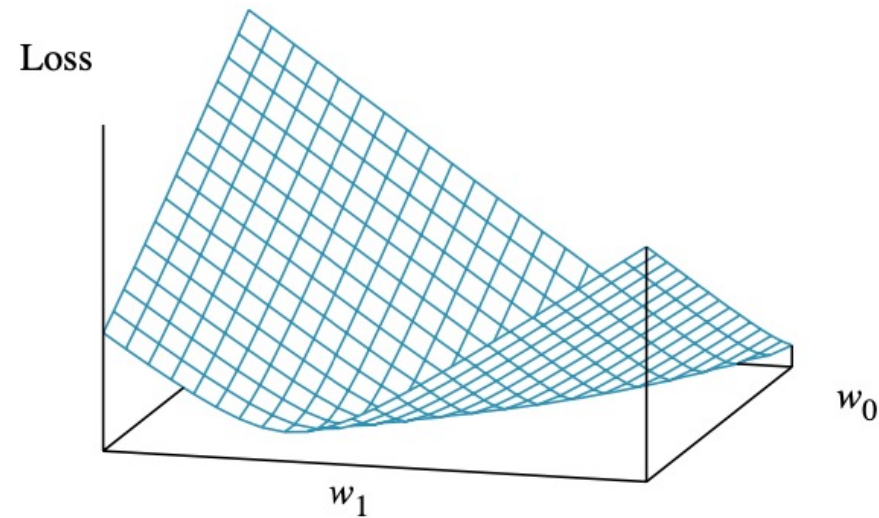
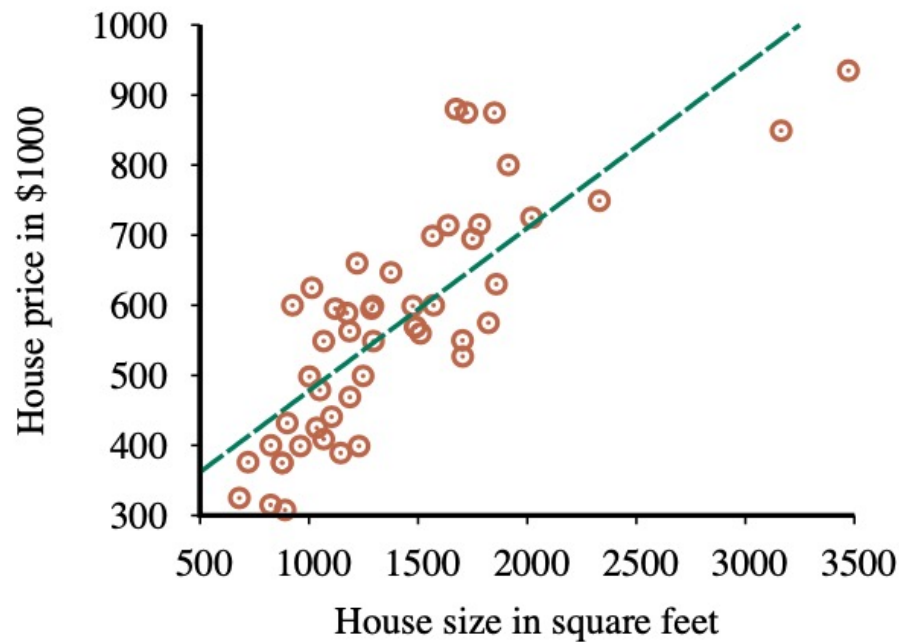
$$y = w_1 x + w_0$$

$$h_w(x) = w_1 x + w_0$$

Linear Regression Weight Space

$$h_w(x) = w_1 x + w_0$$

$$w^* = \operatorname{argmin}_w \operatorname{Loss}(h_w)$$



$$y = 0.232 x + 246$$

Loss function for Weights (w_1, w_0)

Performance Measure

- **The measure of success for estimation problems**
 - **mean-squared error (MSE)**
- **Classification problems**
 - **accuracy ratio**

Evaluation

(Accuracy of Classification Model)

Assessing the Classification Model

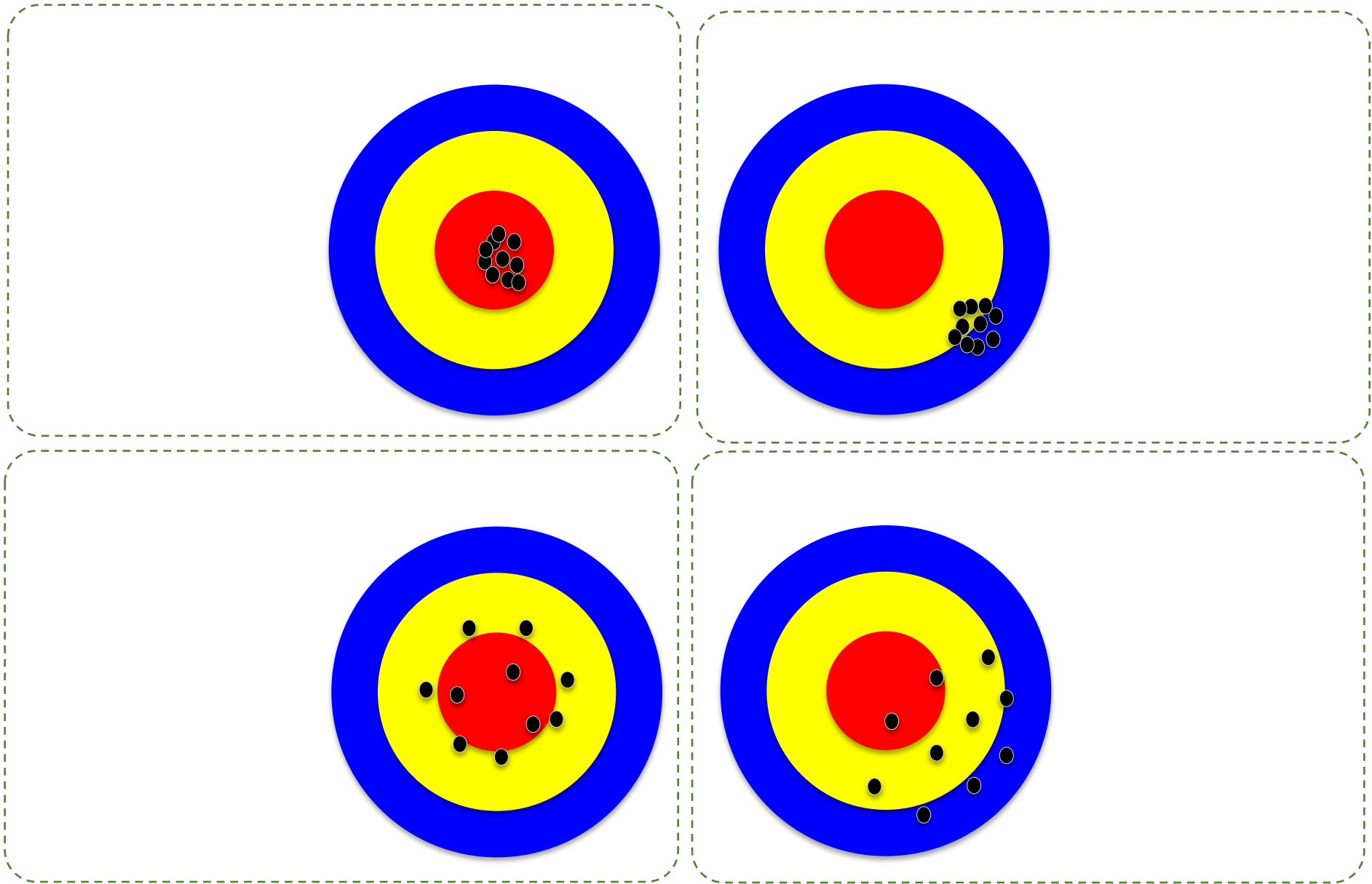
- **Predictive accuracy**
 - **Hit rate**
- **Speed**
 - **Model building; predicting**
- **Robustness**
- **Scalability**
- **Interpretability**
 - **Transparency, explainability**

Accuracy

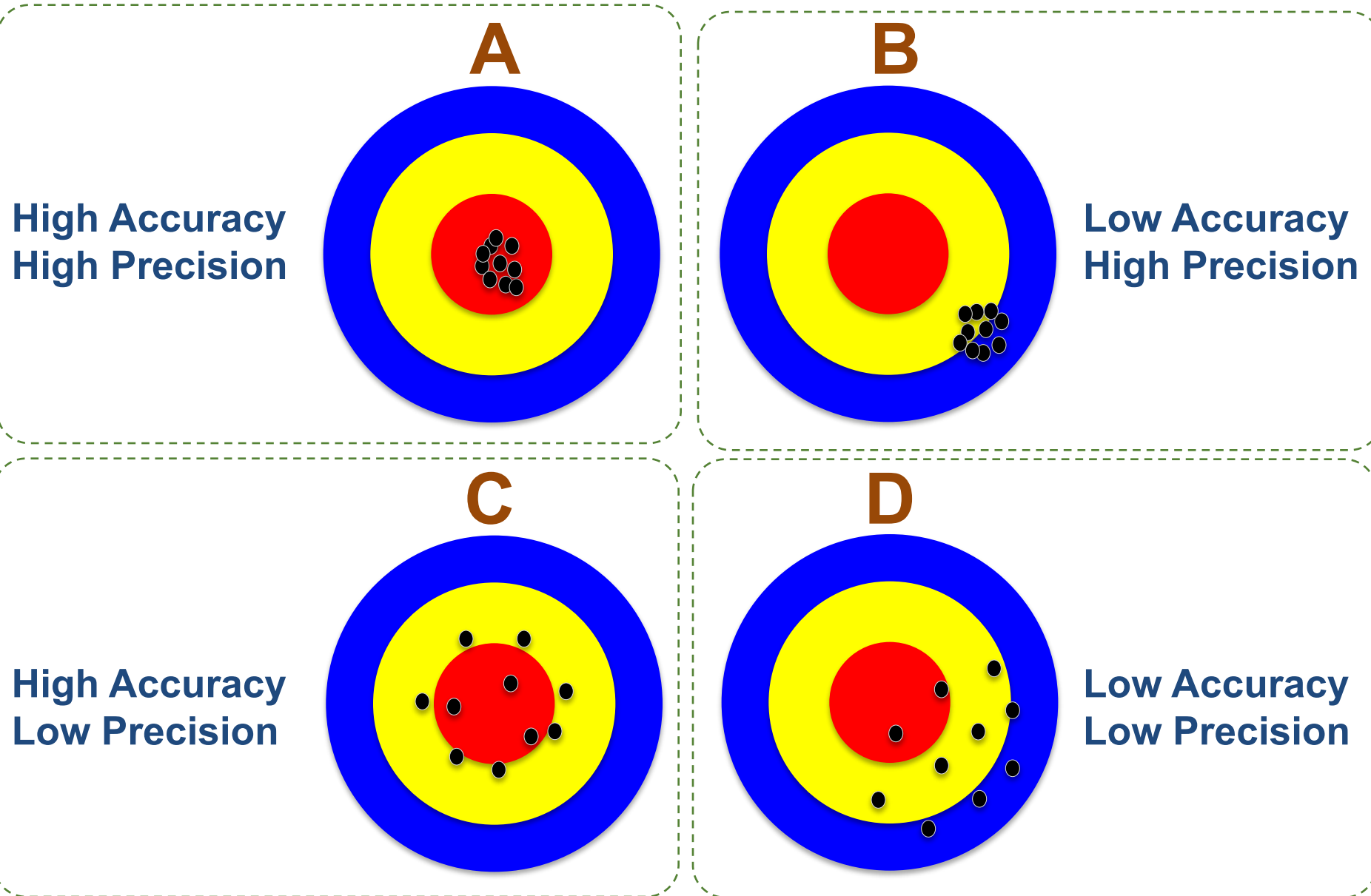
Validity

Precision

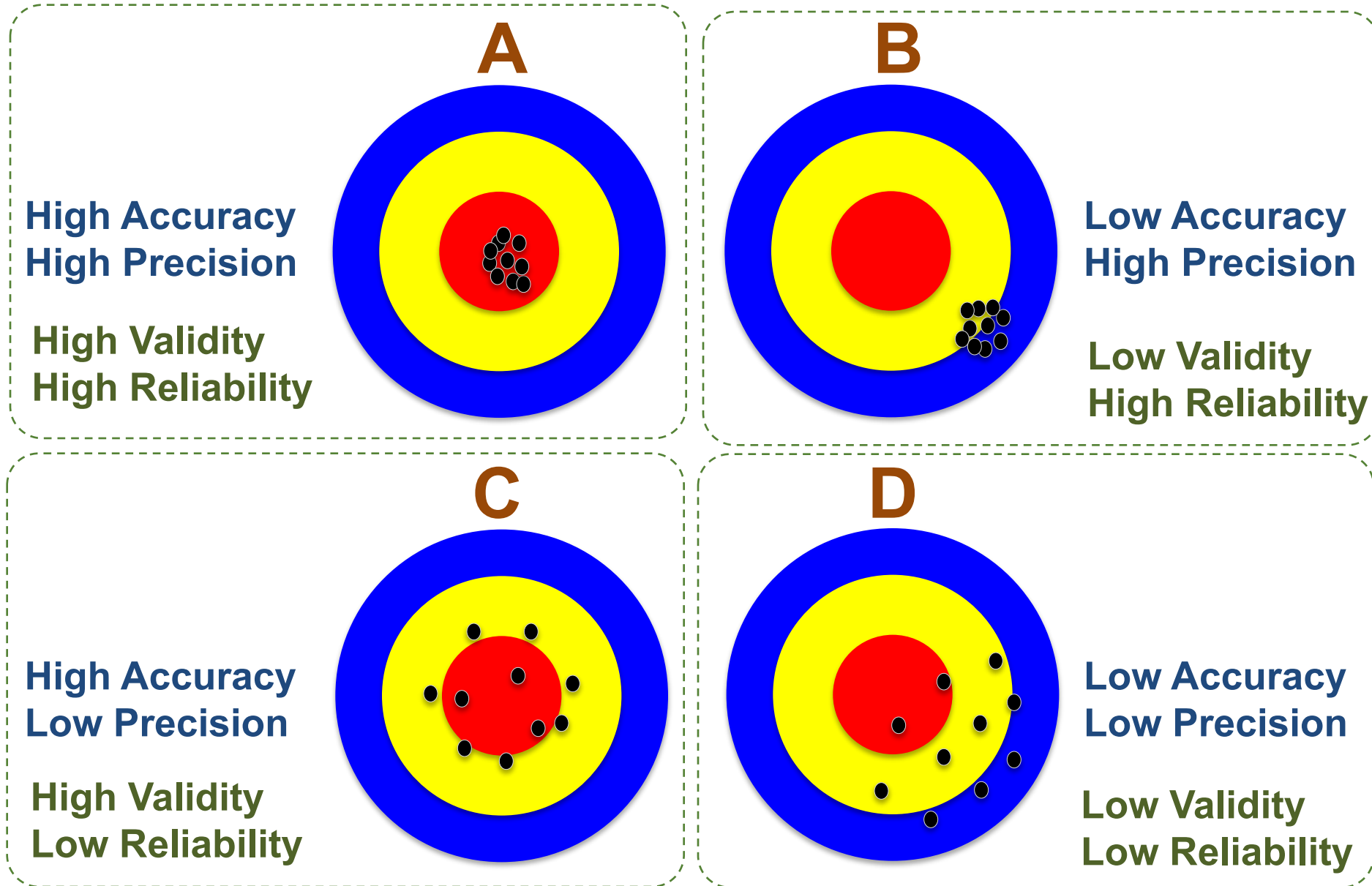
Reliability



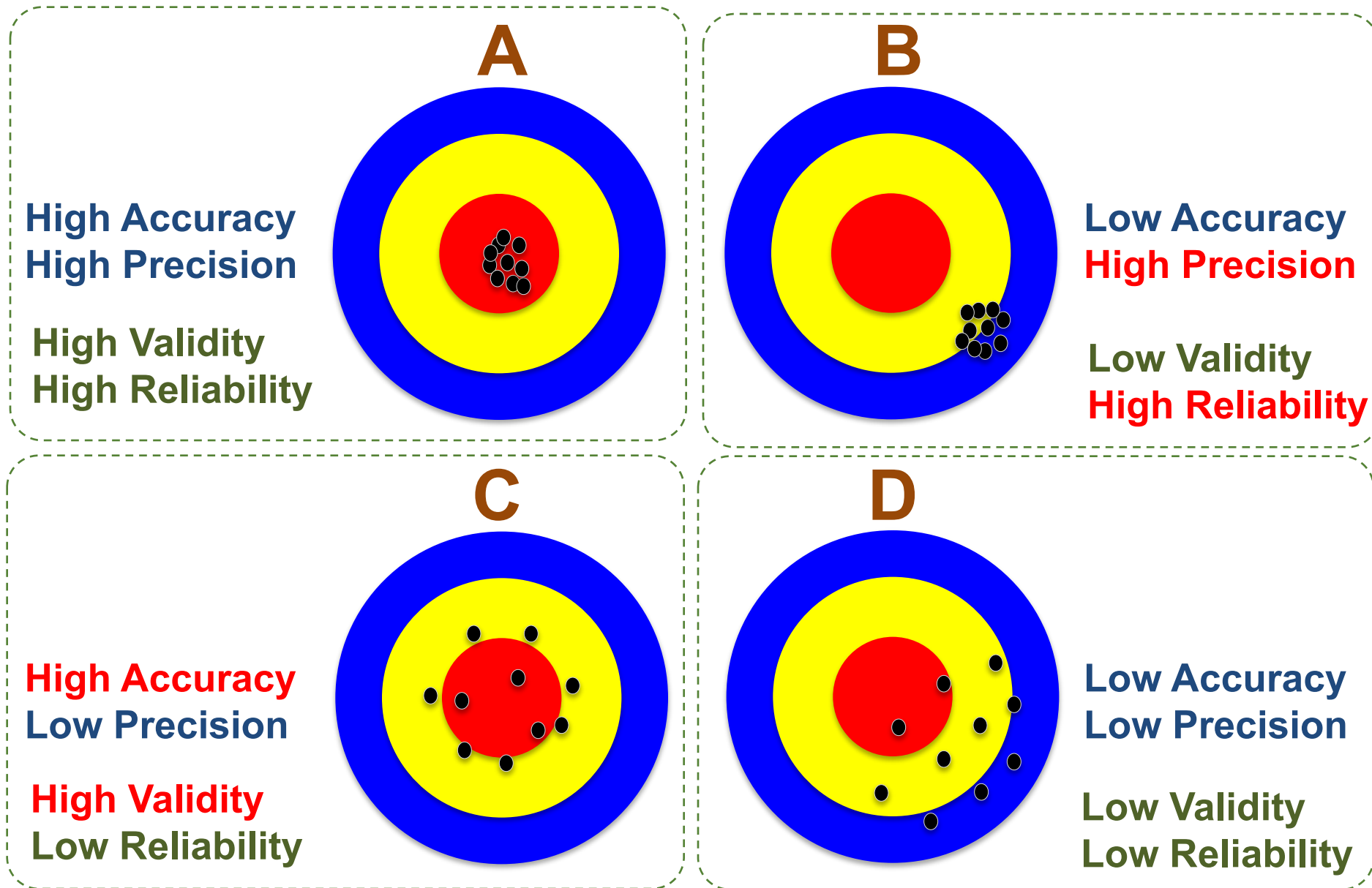
Accuracy vs. Precision



Accuracy vs. Precision



Accuracy vs. Precision



Confusion Matrix

for Tabulation of Two-Class Classification Results

		True/Observed Class	
		Positive	Negative
Predicted Class	Positive	True Positive Count (TP)	False Positive Count (FP)
	Negative	False Negative Count (FN)	True Negative Count (TN)

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

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

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

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

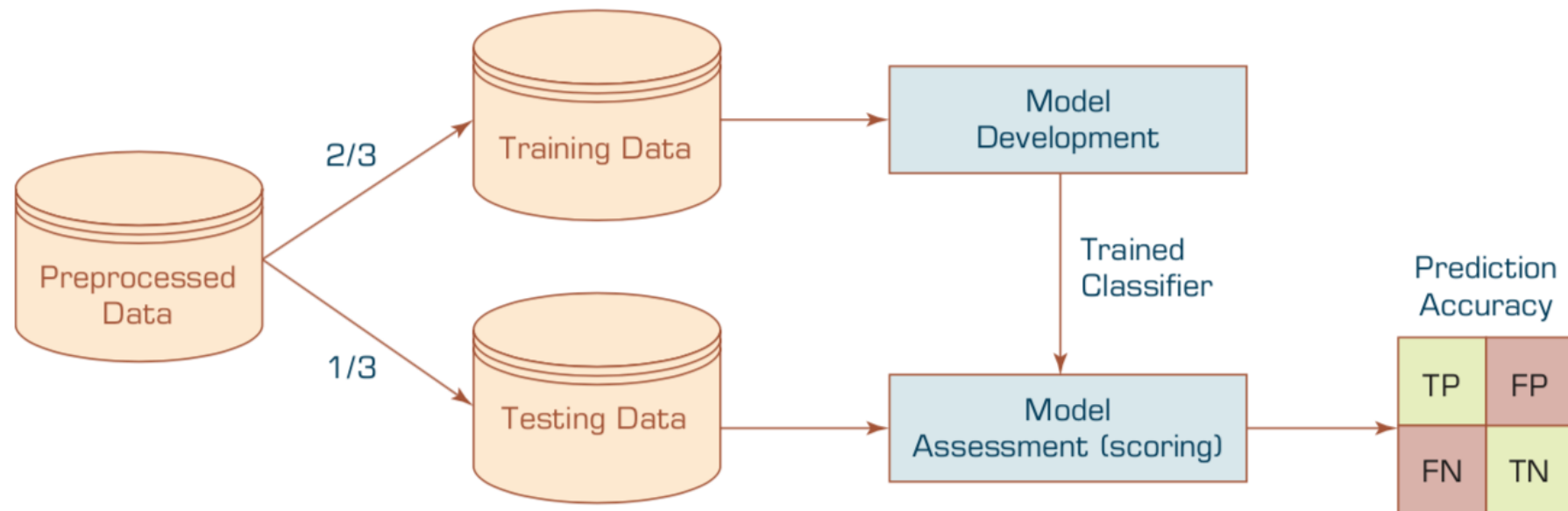
$$Recall = \frac{TP}{TP + FN}$$

Sensitivity = True Positive Rate

Specificity = True Negative Rate

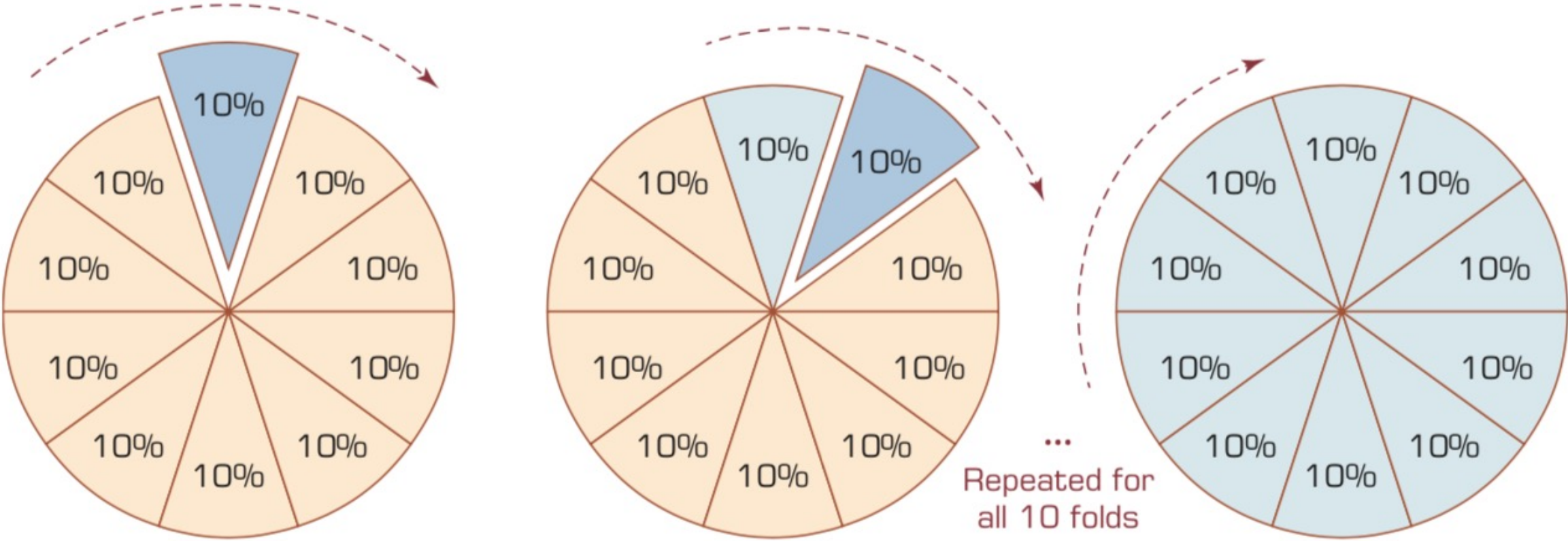
Estimation Methodologies for Classification

- **Simple split** (or holdout or test sample estimation)
 - Split the data into 2 mutually exclusive sets training (~70%) and testing (30%)



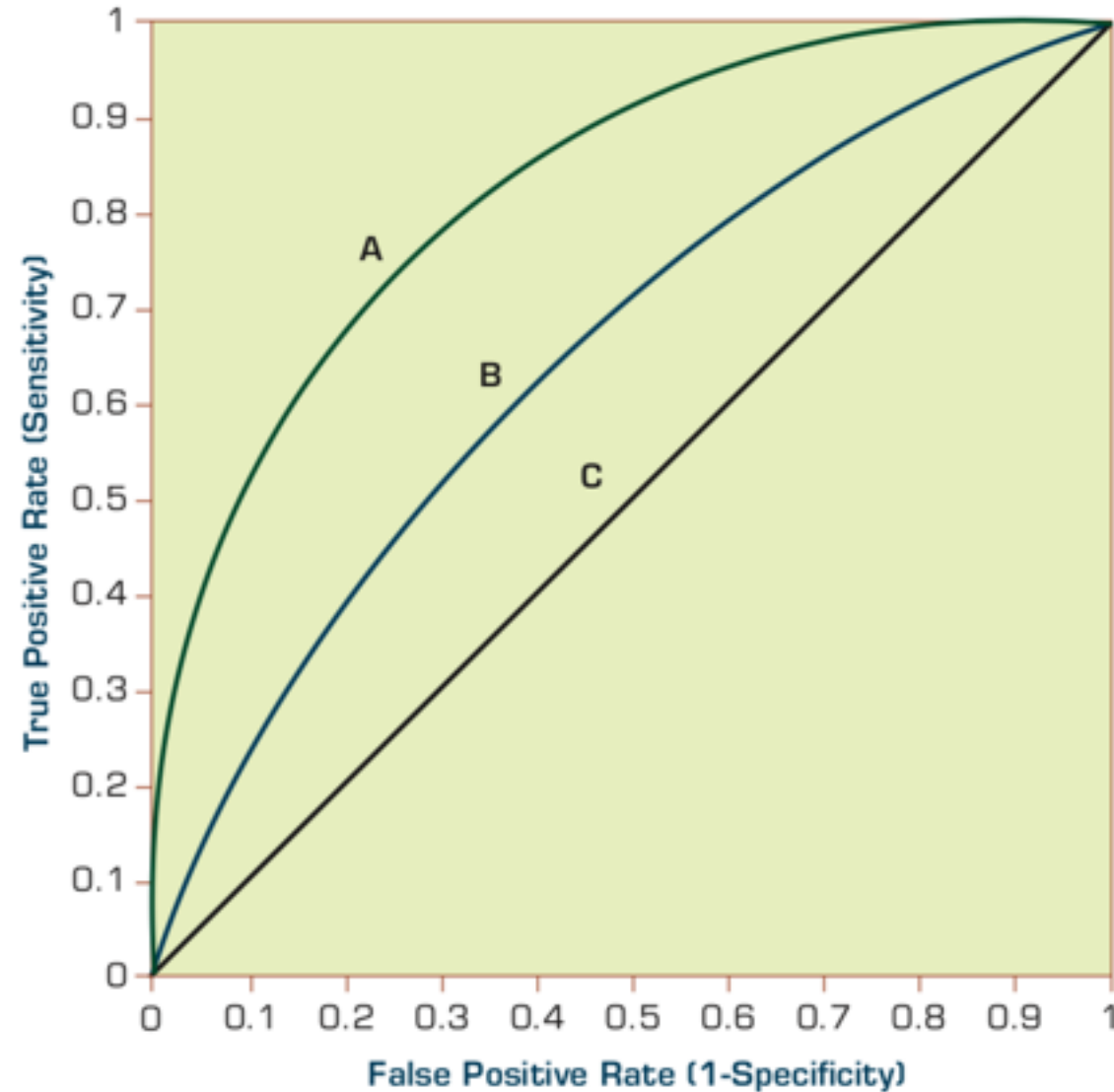
- For ANN, the data is split into three sub-sets (training [~60%], validation [~20%], testing [~20%])

k-Fold Cross-Validation



Estimation Methodologies for Classification

Area under the ROC curve



		True Class (actual value)		total
		Positive	Negative	
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P'
	Negative	False Negative (FN)	True Negative (TN)	N'
total		P	N	

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

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

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

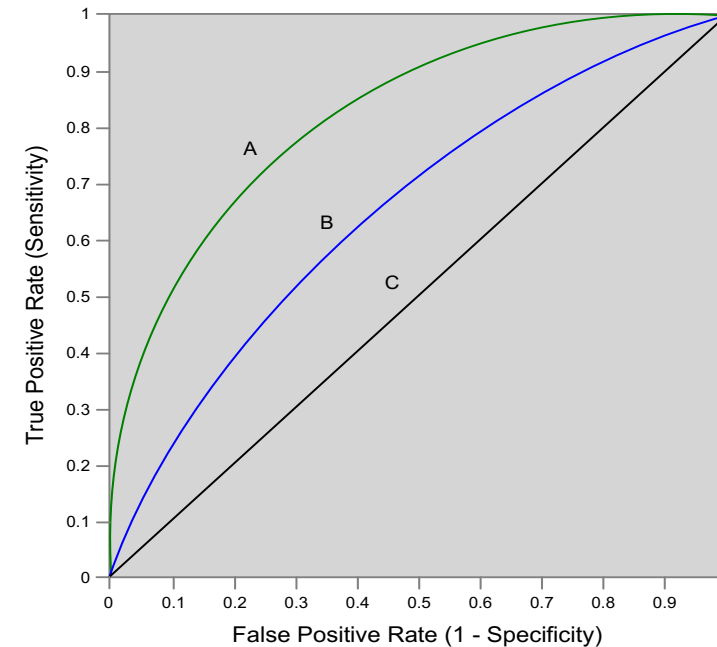
$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

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

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

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

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



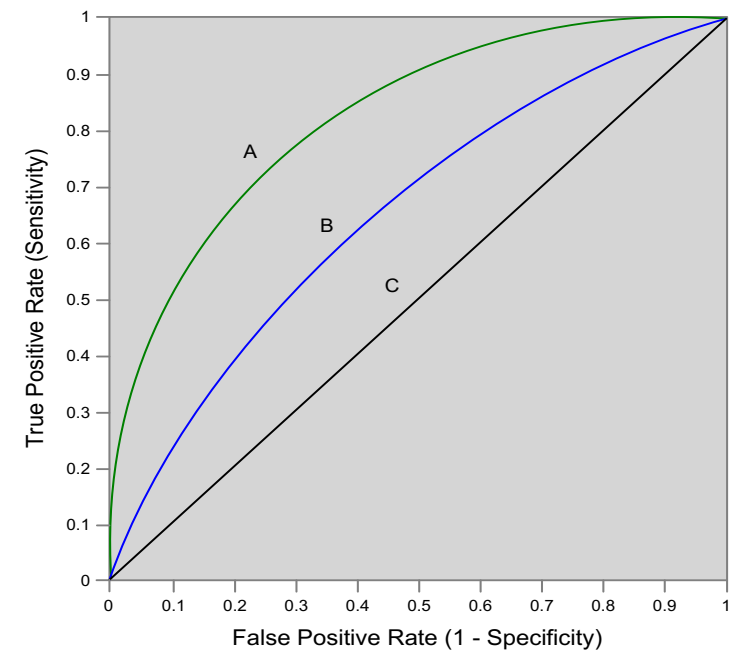
		True Class (actual value)		total
		Positive	Negative	
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P'
	Negative	False Negative (FN)	True Negative (TN)	
total		P	N	

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

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

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

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



		True Class (actual value)		total
		Positive	Negative	
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P'
	Negative	False Negative (FN)	True Negative (TN)	N'
total		P	N	

Specificity

= True Negative Rate

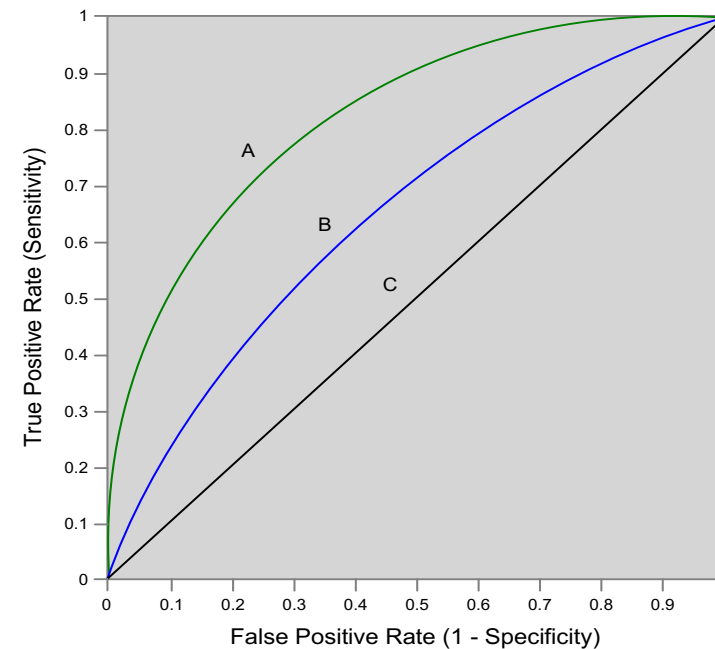
= TN / N

= $TN / (TN + FP)$

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

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

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



		True Class (actual value)		total
		Positive	Negative	
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P'
	Negative	False Negative (FN)	True Negative (TN)	N'
total		P	N	

Precision

= Positive Predictive Value (PPV)

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

Recall

= True Positive Rate (TPR)

= Sensitivity

= Hit Rate

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

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

is the harmonic mean of precision and recall

$$= 2TP / (P + P')$$

$$= 2TP / (2TP + FP + FN)$$

$$F = 2 * \frac{precision * recall}{precision + recall}$$

A		
63 (TP)	28 (FP)	91
37 (FN)	72 (TN)	109
100	100	200

Recall

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

Specificity

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

TPR = 0.63

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

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

FPR = 0.28

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

PPV = 0.69

$$= 63 / (63 + 28)$$

$$= 63 / 91$$

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

Precision

= Positive Predictive Value (PPV)

F1 = 0.66

$$= 2 * (0.63 * 0.69) / (0.63 + 0.69)$$

$$= (2 * 63) / (100 + 91)$$

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

$$F = 2 * \frac{precision * recall}{precision + recall}$$

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

is the harmonic mean of
 precision and recall

$$= 2TP / (P + P')$$

$$= 2TP / (2TP + FP + FN)$$

ACC = 0.68

$$= (63 + 72) / 200$$

$$= 135 / 200 = 67.5$$

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

A

63 (TP)	28 (FP)	91
37 (FN)	72 (TN)	109
100	100	200

$$\text{TPR} = 0.63$$

$$\text{FPR} = 0.28$$

$$\begin{aligned} \text{PPV} &= 0.69 \\ &= 63 / (63 + 28) \\ &= 63 / 91 \end{aligned}$$

$$\begin{aligned} \text{F1} &= 0.66 \\ &= 2 * (0.63 * 0.69) / (0.63 + 0.69) \\ &= (2 * 63) / (100 + 91) \\ &= (0.63 + 0.69) / 2 = 1.32 / 2 = 0.66 \end{aligned}$$

$$\begin{aligned} \text{ACC} &= 0.68 \\ &= (63 + 72) / 200 \\ &= 135 / 200 = 67.5 \end{aligned}$$

B

77 (TP)	77 (FP)	154
23 (FN)	23 (TN)	46
100	100	200

$$\text{TPR} = 0.77$$

$$\text{FPR} = 0.77$$

$$\text{PPV} = 0.50$$

$$\text{F1} = 0.61$$

$$\text{ACC} = 0.50$$

Recall

= True Positive Rate (TPR)

= Sensitivity

= Hit Rate

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

Precision

= Positive Predictive Value (PPV)

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

C

24 (TP)	88 (FP)	112
76 (FN)	12 (TN)	88
100	100	200

$$\text{TPR} = 0.24$$

$$\text{FPR} = 0.88$$

$$\text{PPV} = 0.21$$

$$\text{F1} = 0.22$$

$$\text{ACC} = 0.18$$

C'

76 (TP)	12 (FP)	88
24 (FN)	88 (TN)	112
100	100	200

$$\text{TPR} = 0.76$$

$$\text{FPR} = 0.12$$

$$\text{PPV} = 0.86$$

$$\text{F1} = 0.81$$

$$\text{ACC} = 0.82$$

Recall

= True Positive Rate (TPR)

= Sensitivity

= Hit Rate

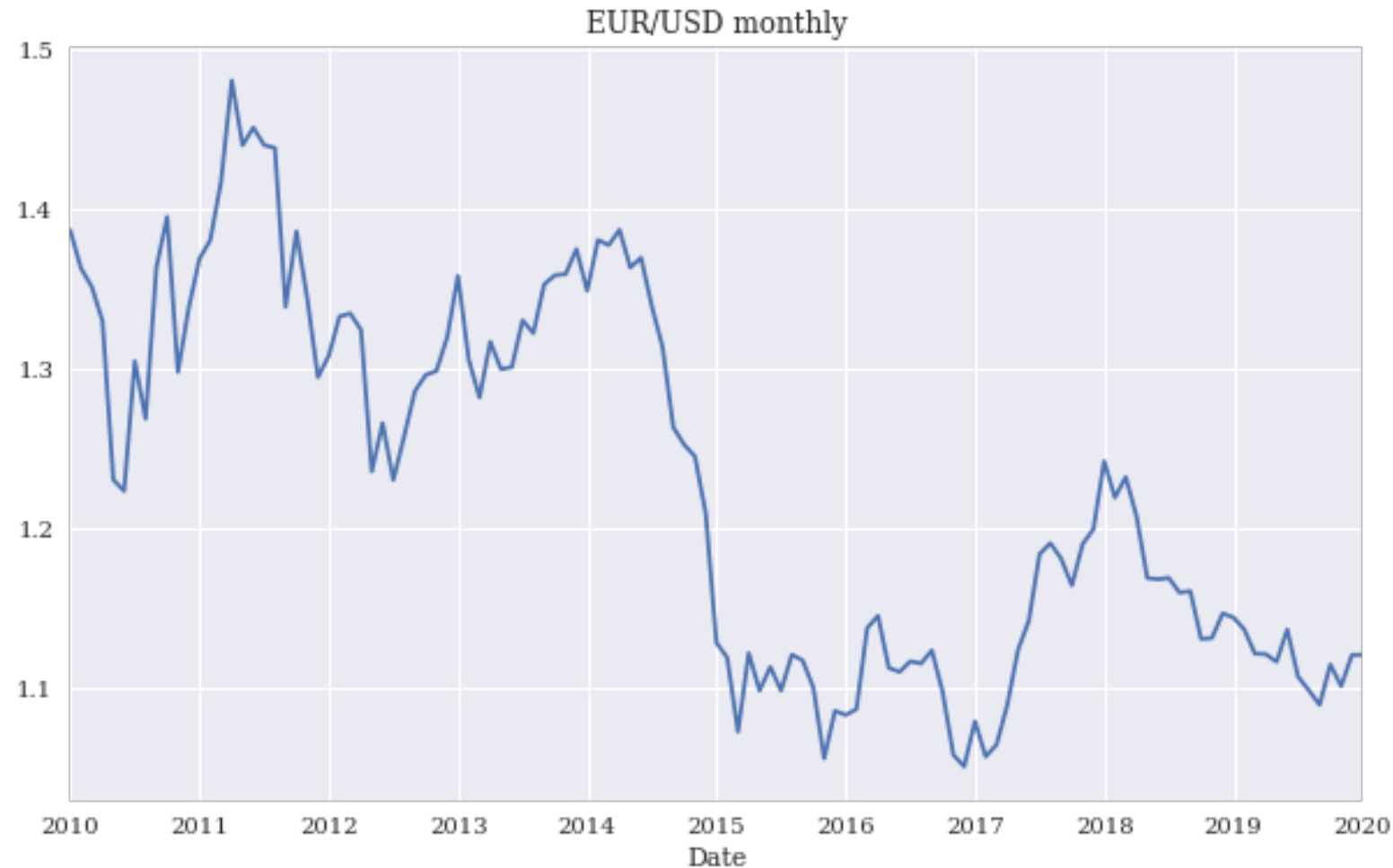
$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision

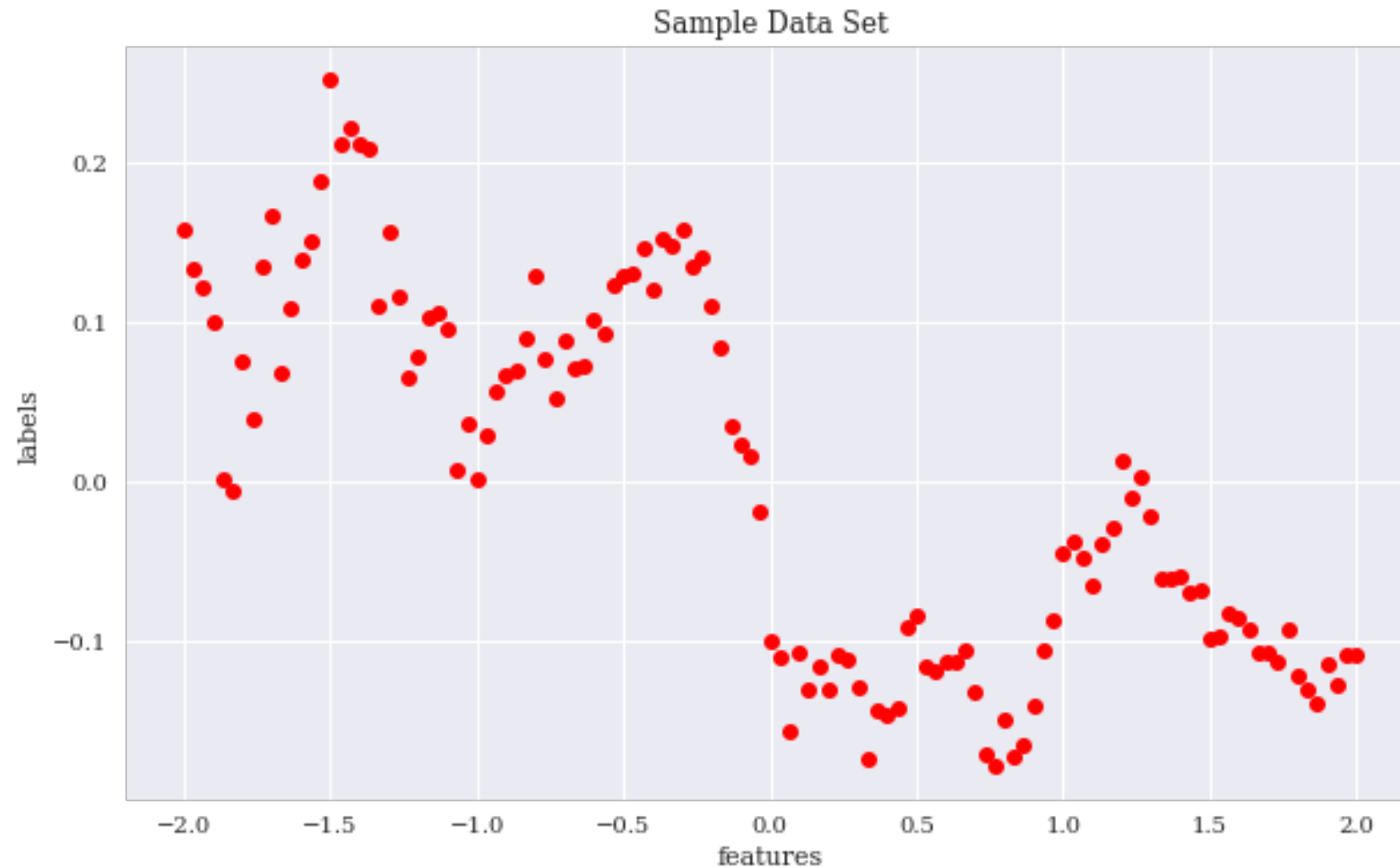
= Positive Predictive Value (PPV)

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

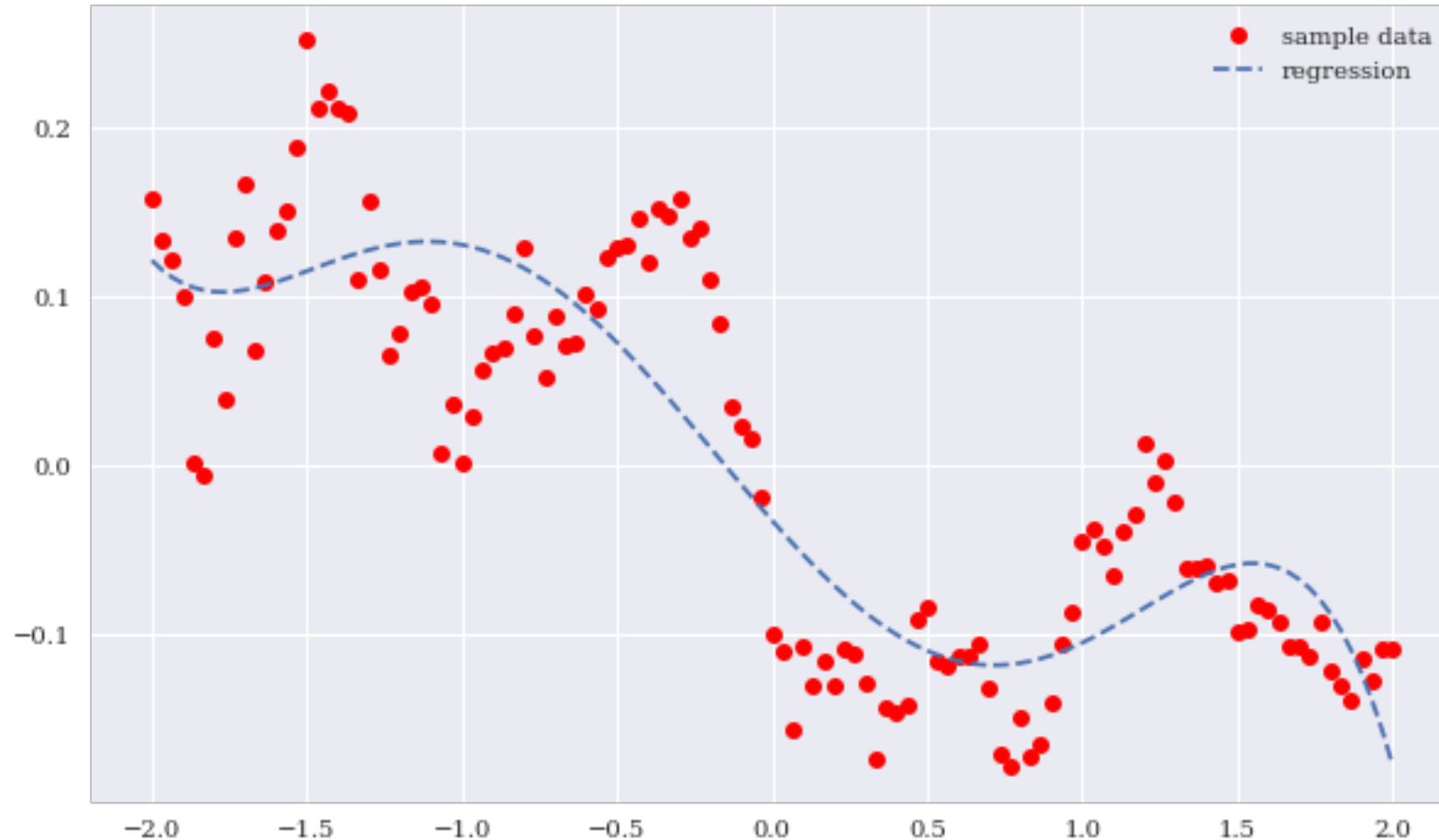
EUR/USD exchange rate as time series (monthly)



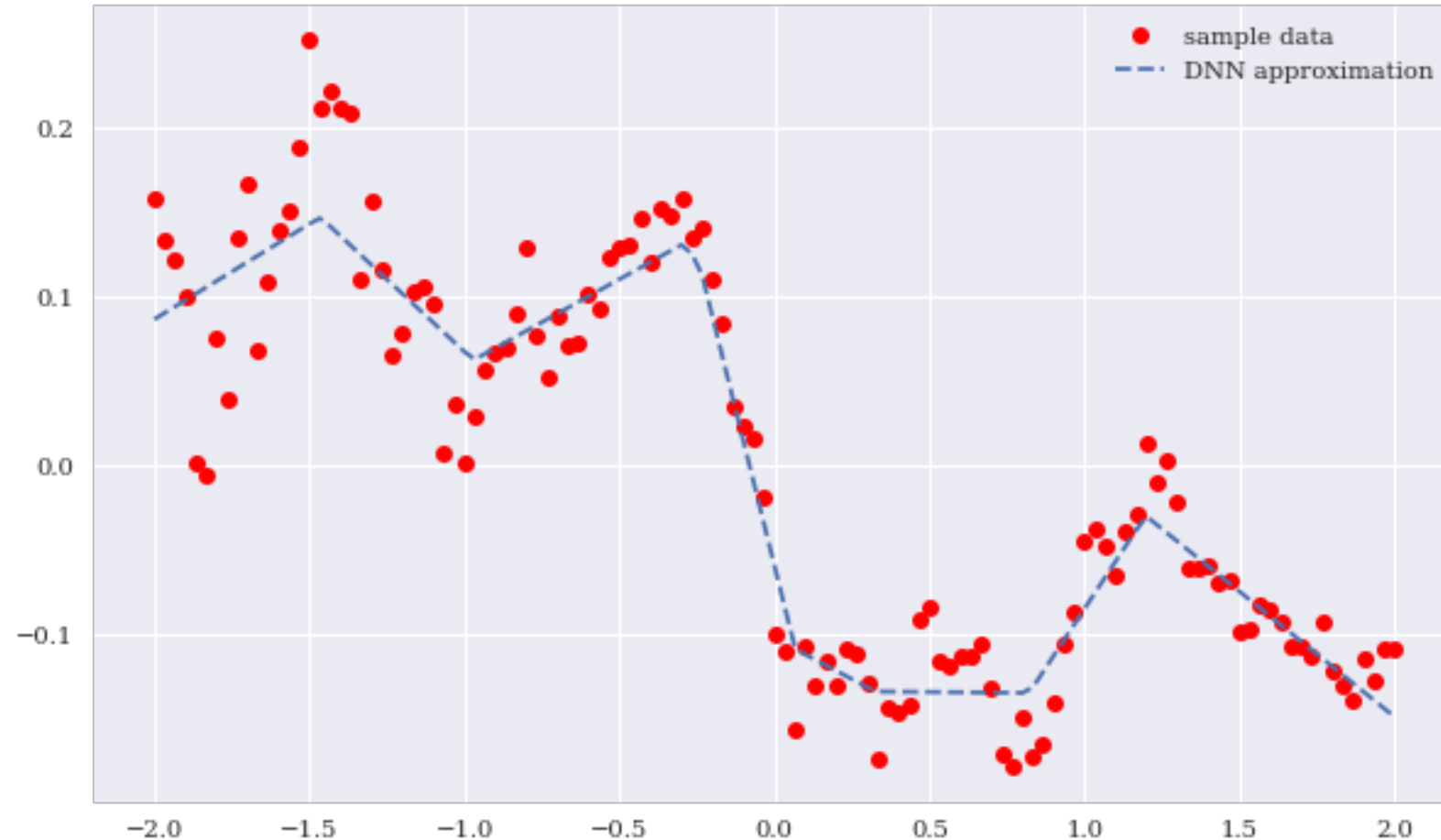
Sample data set



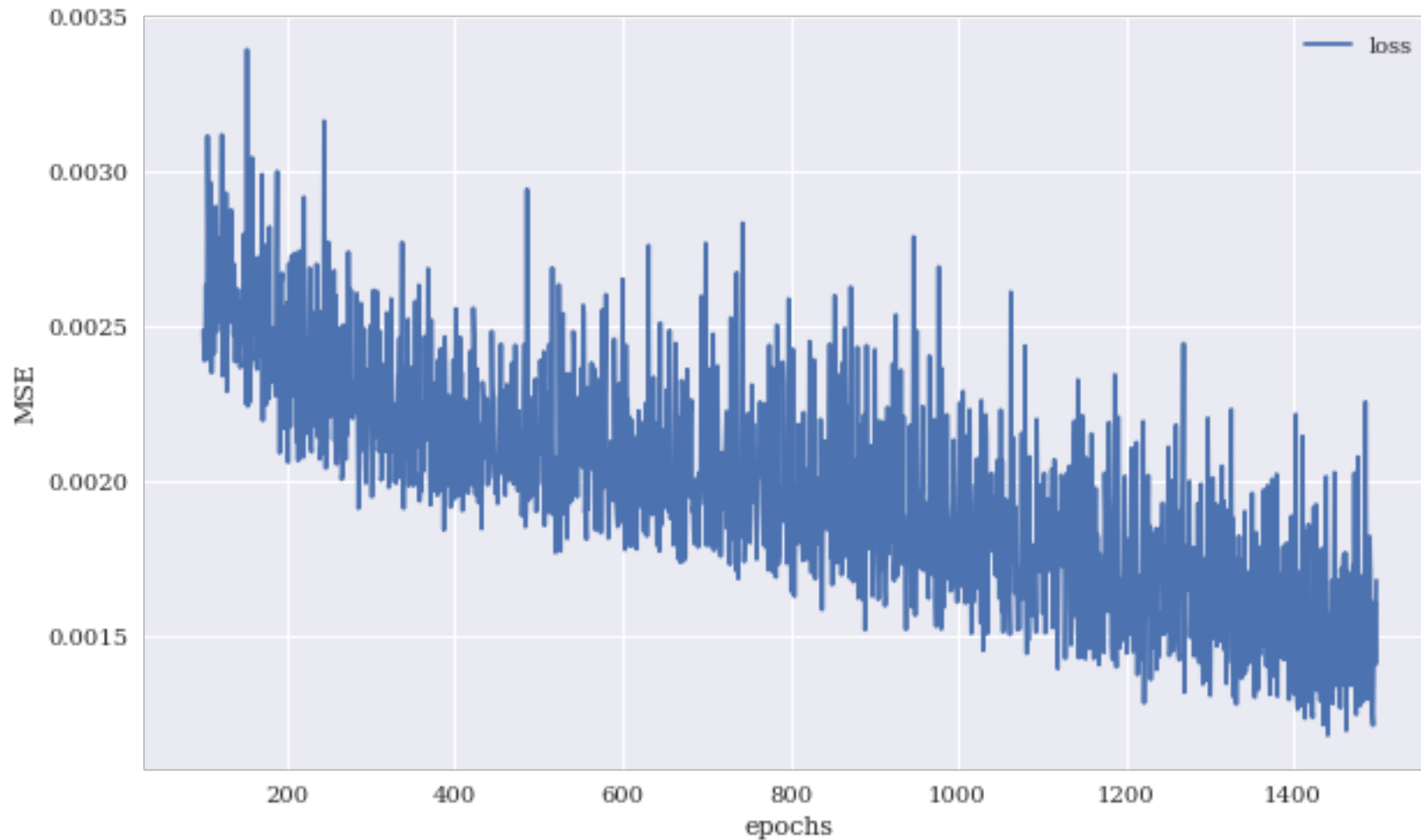
Sample data and cubic regression line



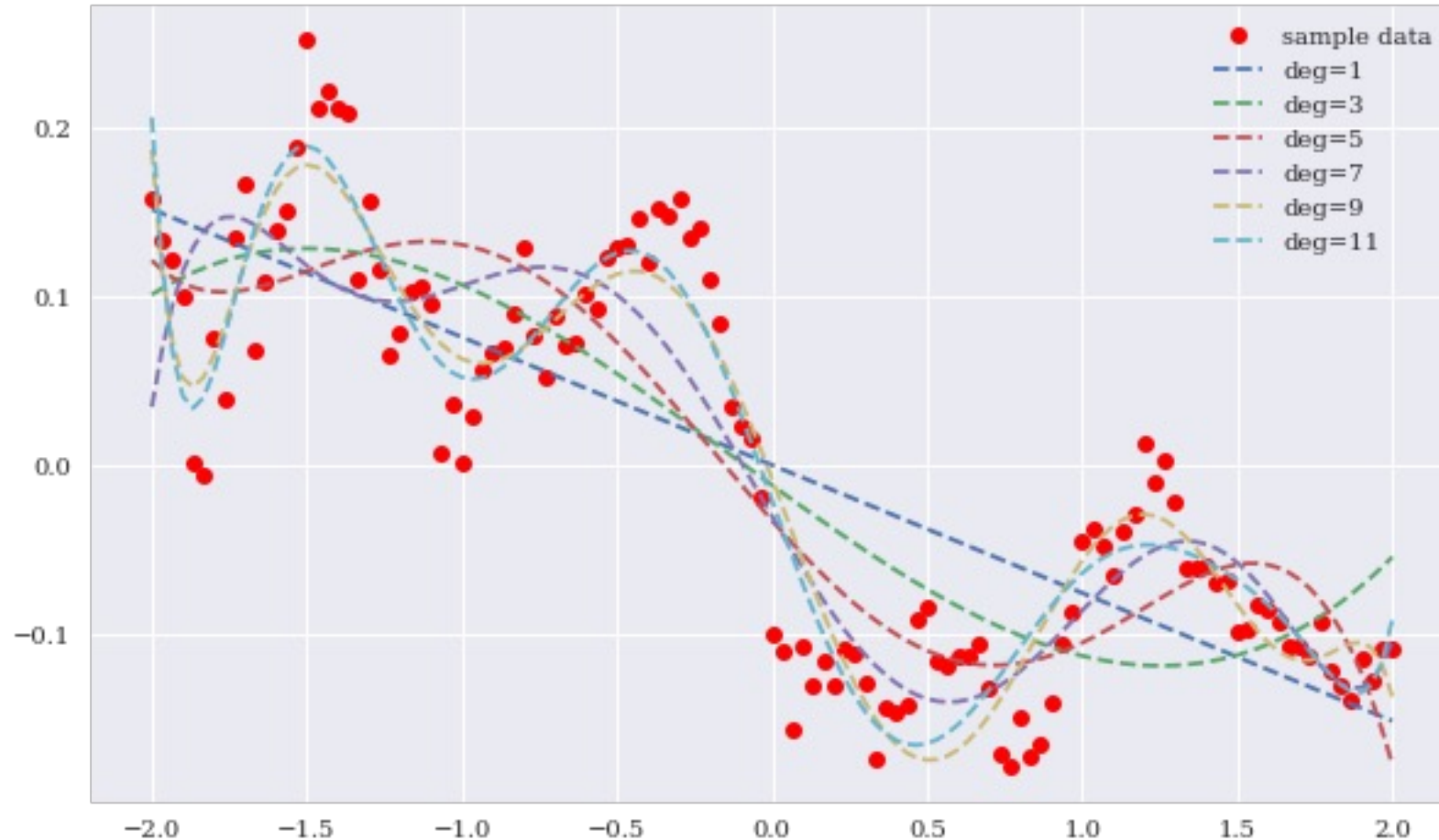
Sample data and neural network approximation



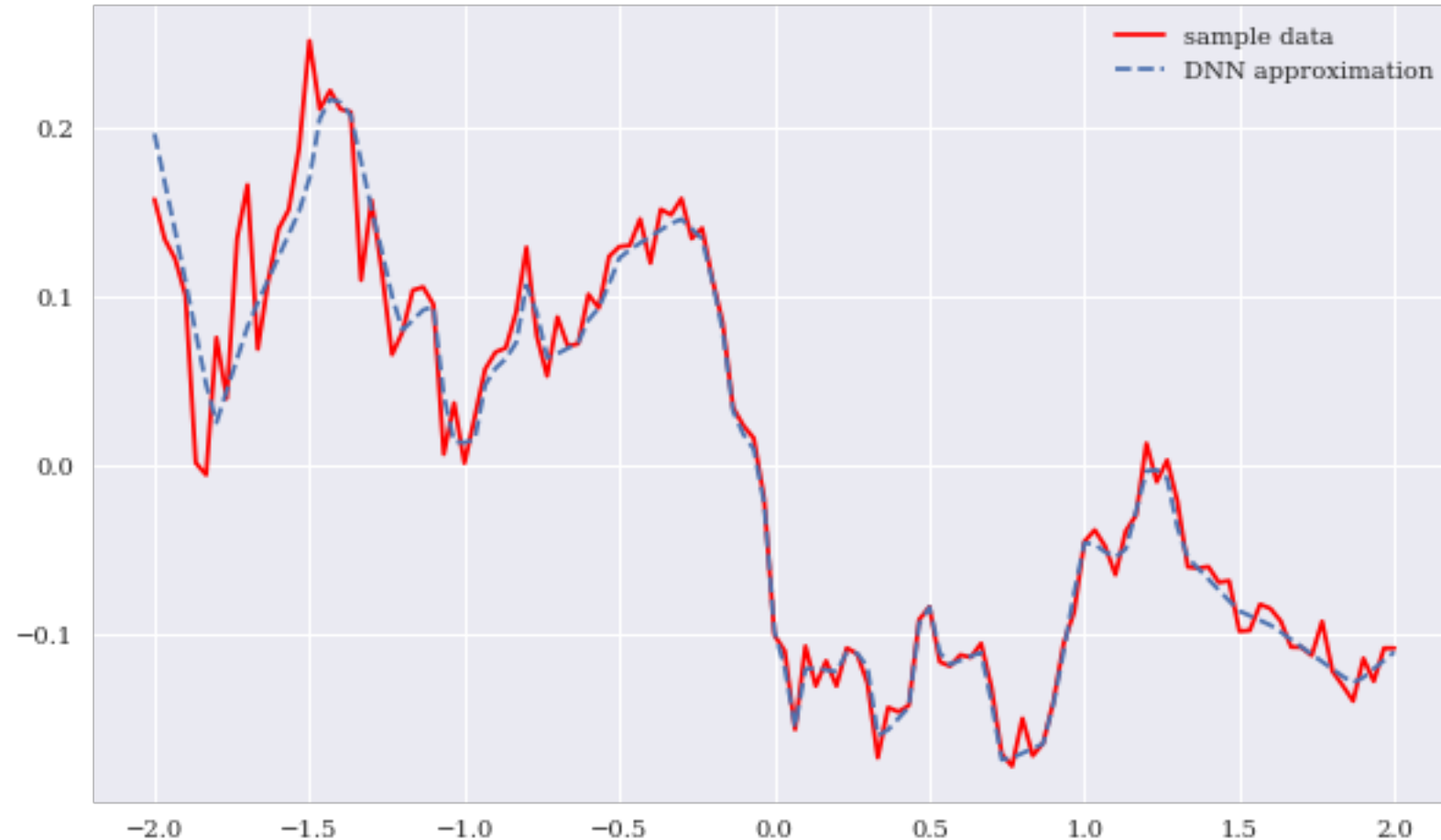
MSE values against number of training epochs



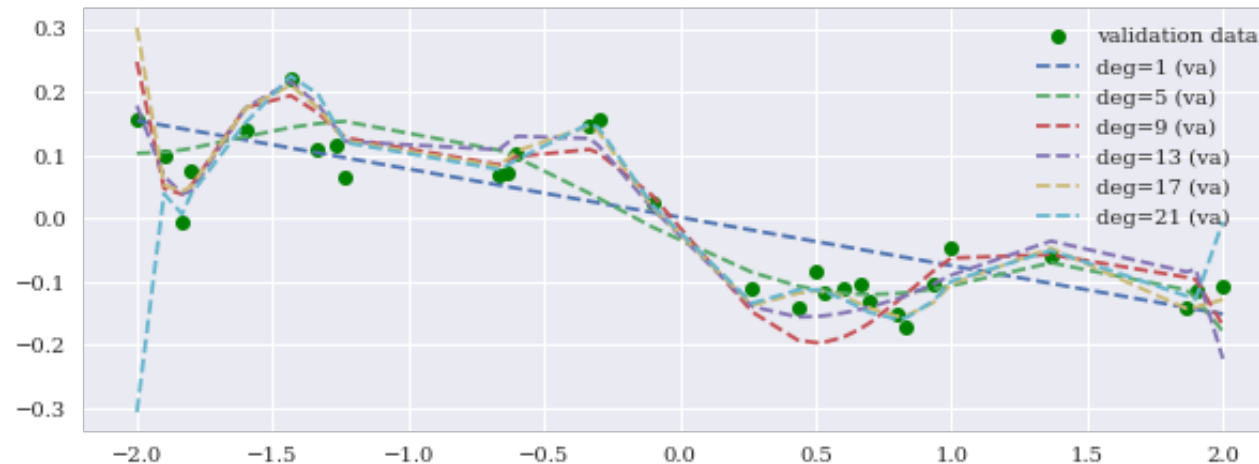
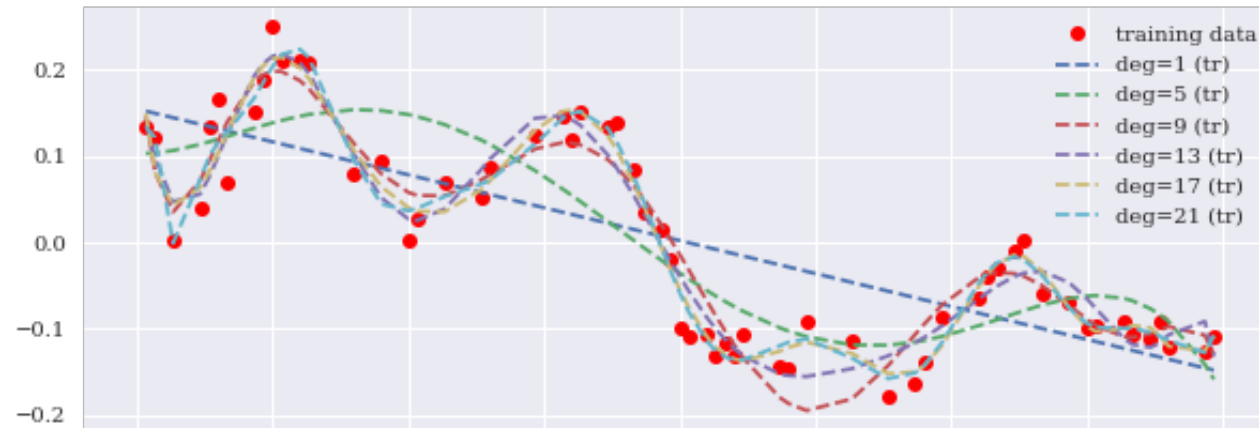
Regression lines for different highest degrees



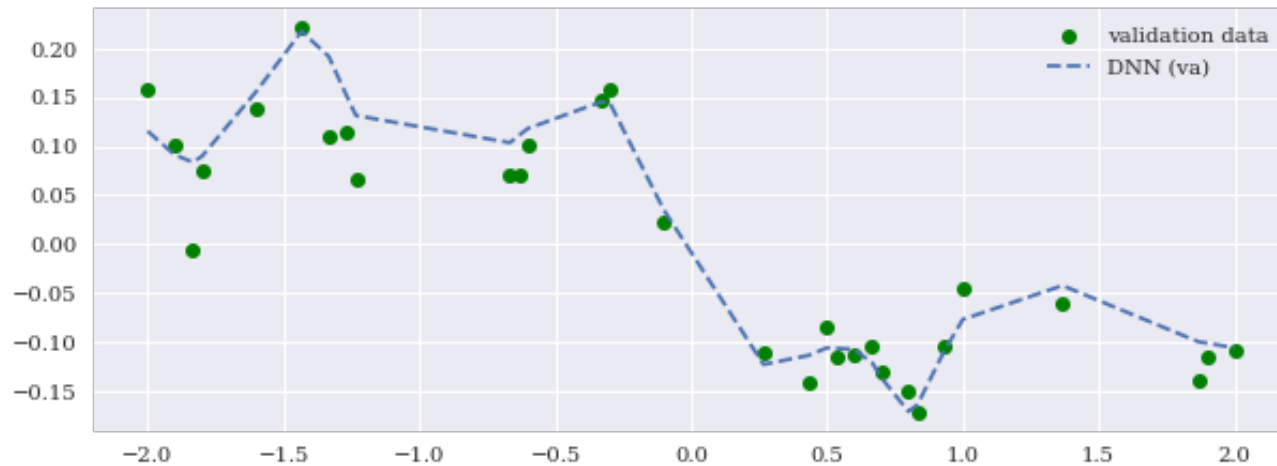
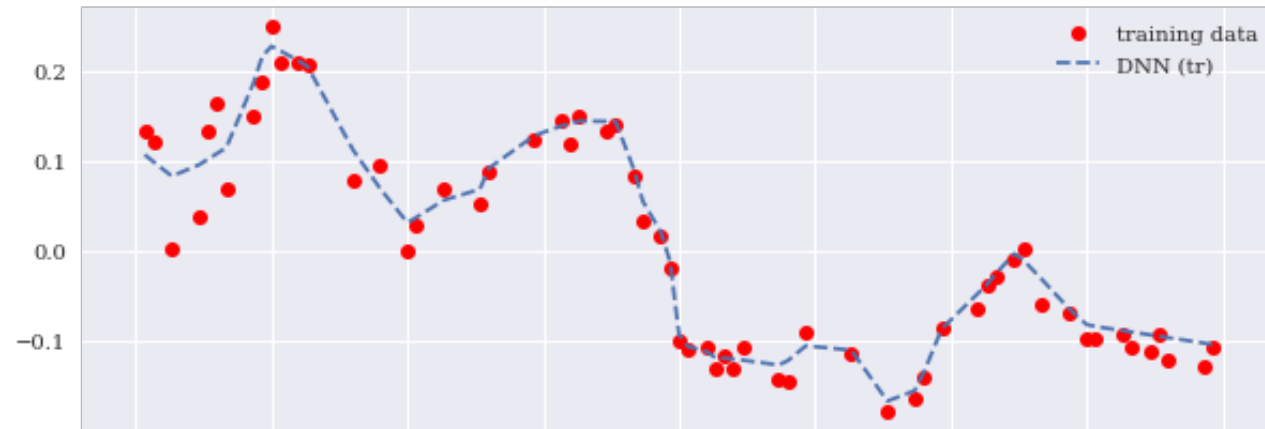
Sample data and DNN approximation (higher capacity)



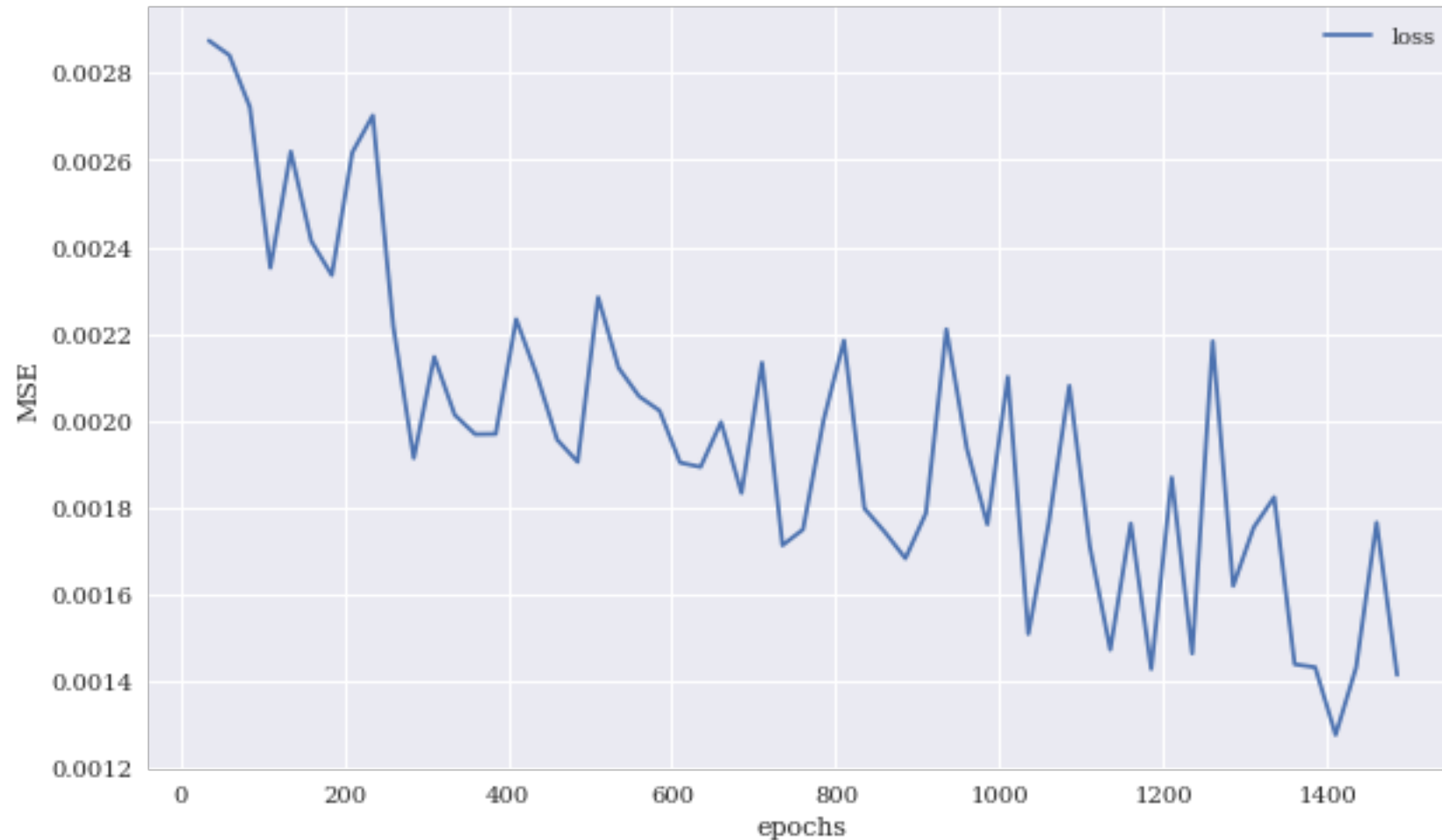
Training and validation data including regression fits



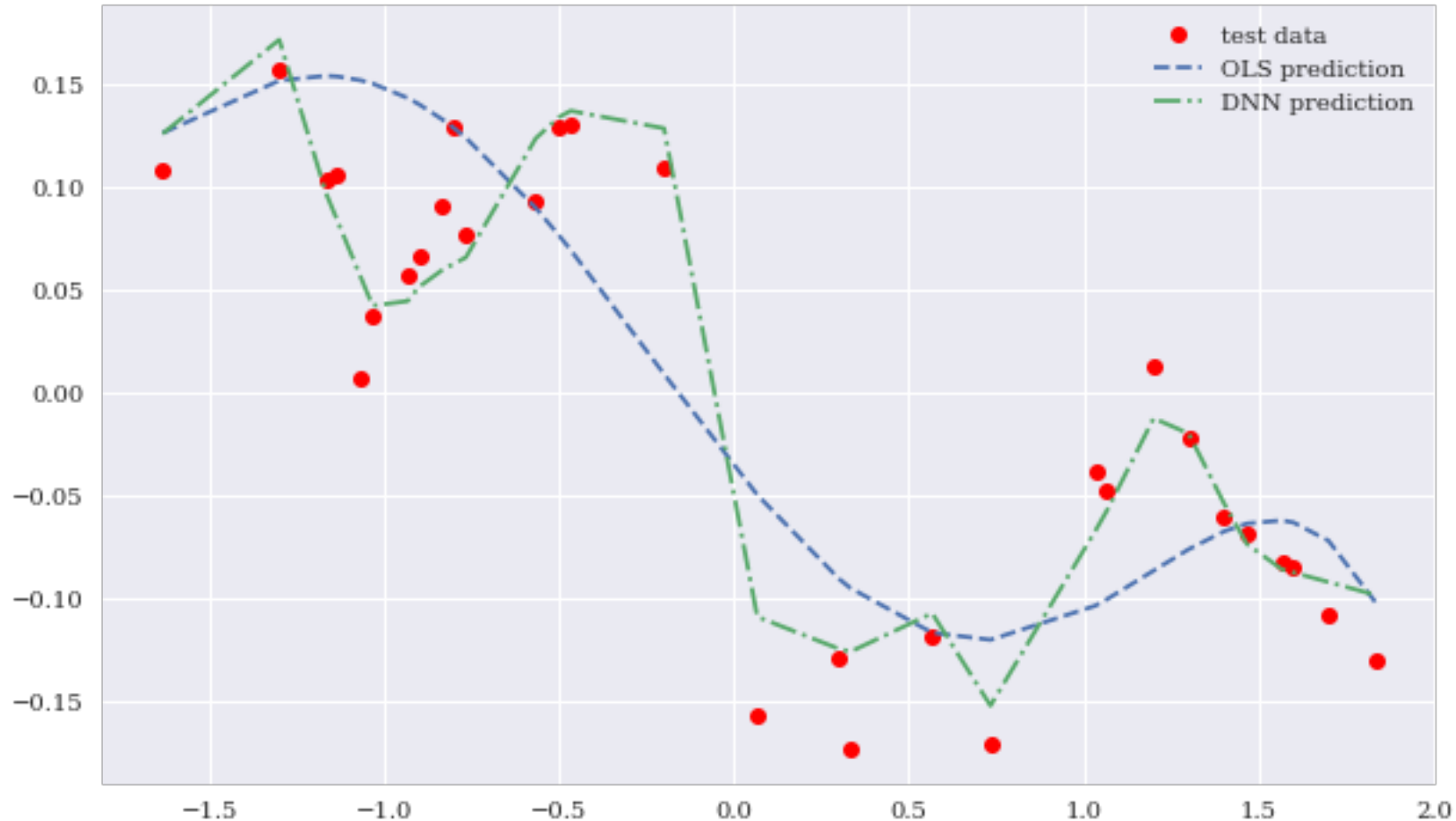
Training and validation data including DNN predictions



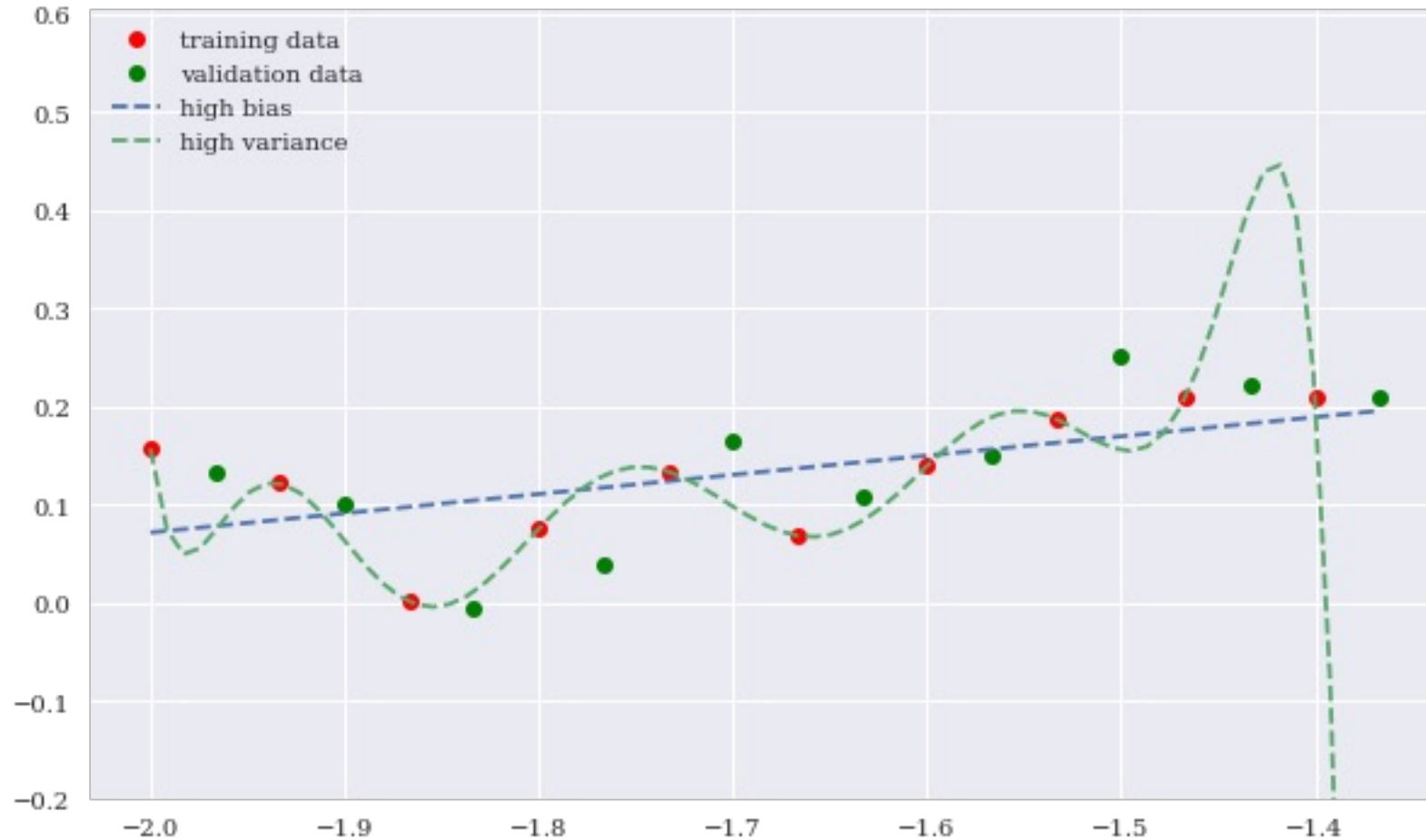
MSE values for DNN model on the training and validation data sets



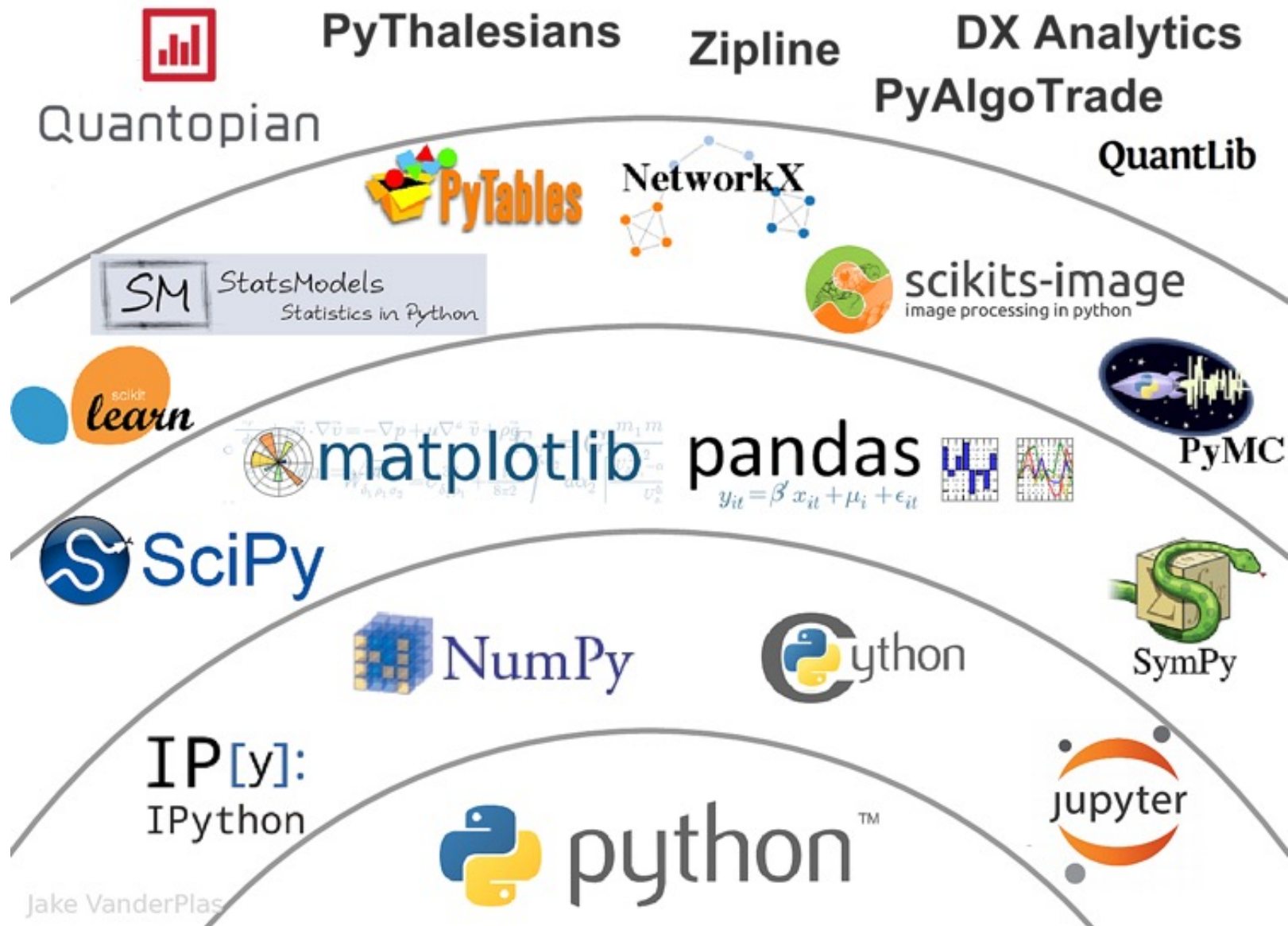
Test data and predictions from OLS regression and the DNN model



High bias and high variance OLS regression fits



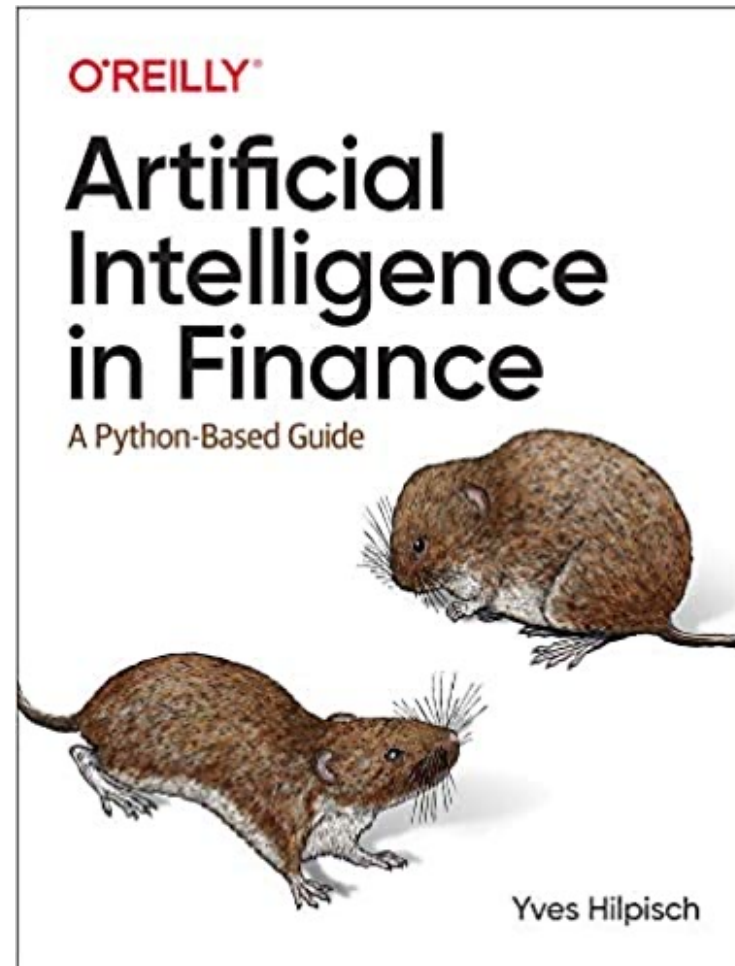
The Quant Finance PyData Stack



Jake VanderPlas

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

Yves Hilpisch (2020),
Artificial Intelligence in Finance:
A Python-Based Guide,
O'Reilly



Yves Hilpisch (2020), **Artificial Intelligence in Finance: A Python-Based Guide**, O'Reilly

yhilpisch / aiif Public <https://github.com/yhilpisch/aiif> Notifications Star 98 Fork 77

Code Issues Pull requests Actions Projects Wiki Security Insights

main 1 branch 0 tags Go to file Code

yves Code updates for TF 2.3. e334251 on Dec 8, 2020 4 commits


code	Code updates for TF 2.3.	11 months ago
.gitignore	Code updates for TF 2.3.	11 months ago
LICENSE.txt	Code updates.	11 months ago
README.md	Code updates.	11 months ago

README.md

Artificial Intelligence in Finance

About this Repository

This repository provides Python code and Jupyter Notebooks accompanying the **Artificial Intelligence in Finance** book published by [O'Reilly](#).



About

Jupyter Notebooks and code for the book **Artificial Intelligence in Finance** (O'Reilly) by Yves Hilpisch.

home.tpq.io/books/aiif

Readme View license

Releases

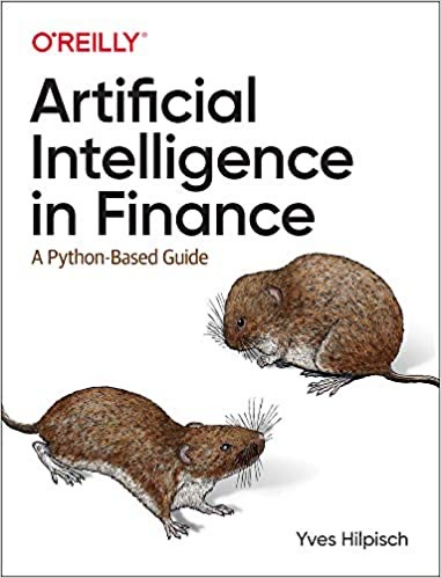
No releases published

Packages

No packages published

Languages

- Jupyter Notebook 97.4%
- Python 2.6%



Yves Hilpisch (2020), **Artificial Intelligence in Finance: A Python-Based Guide**, O'Reilly

yhilpisch / aiif Public

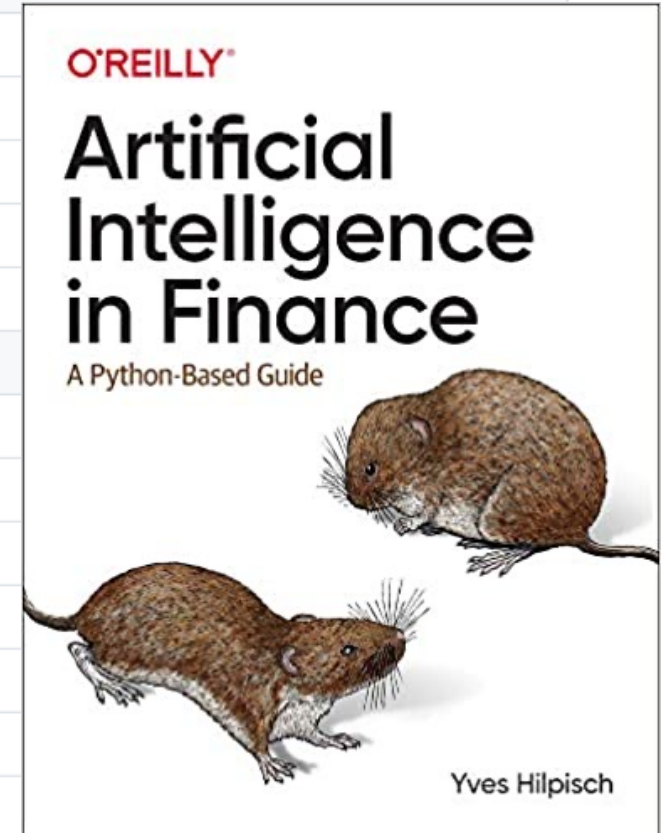
Notifications Star 98 Fork 77

Code Issues Pull requests Actions Projects Wiki Security Insights

main aiif / code / <https://github.com/yhilpisch/aiif/tree/main/code> Go to file

yves Code updates for TF 2.3. e334251 on Dec 8, 2020 History

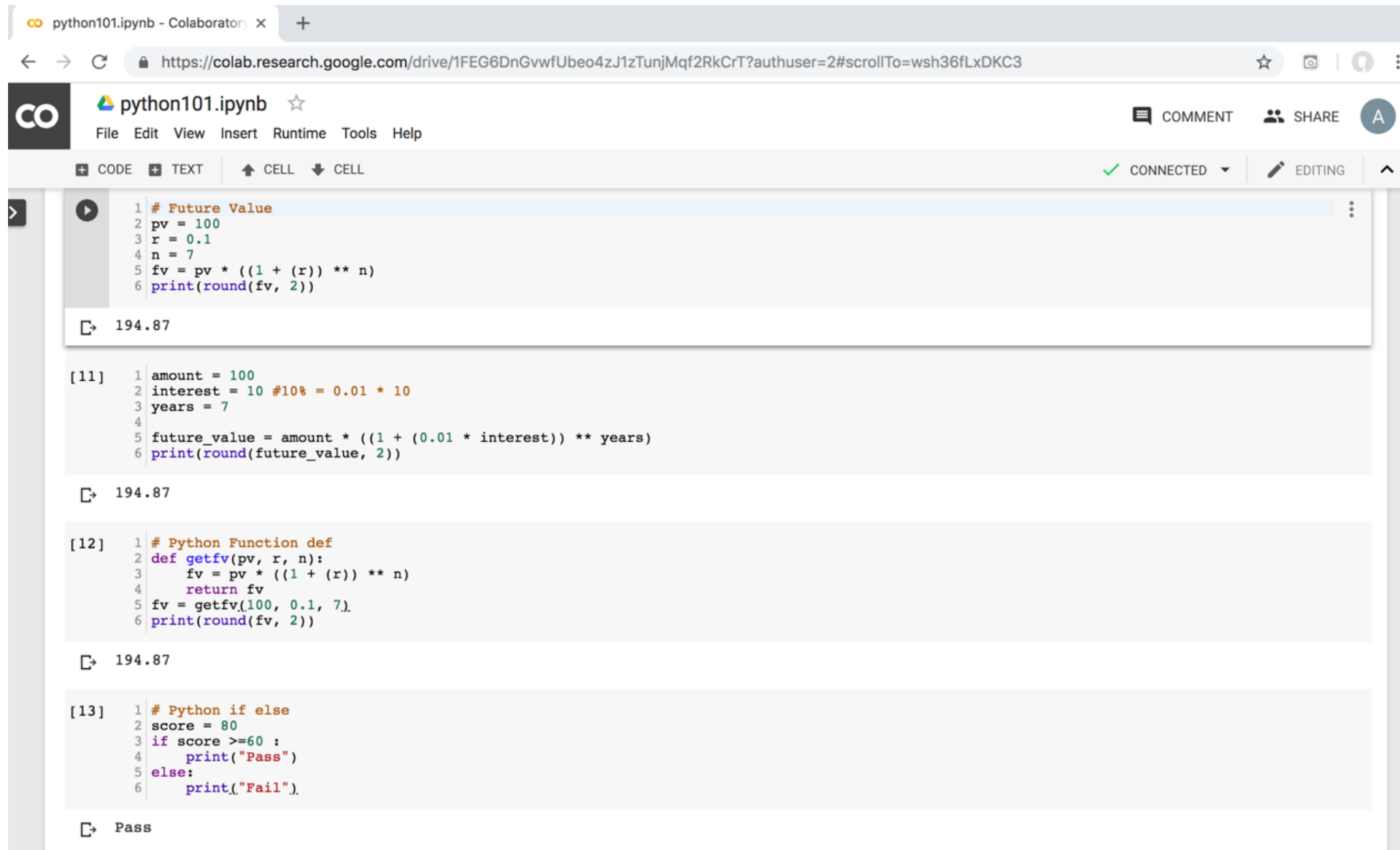
..	
oanda	Code updates for TF 2.3.
01_artificial_intelligence.ipynb	Code updates for TF 2.3.
02_superintelligence.ipynb	Code updates for TF 2.3.
03_normative_finance.ipynb	Code updates for TF 2.3.
04_data_driven_finance_a.ipynb	Initial commit.
04_data_driven_finance_b.ipynb	Initial commit.
05_machine_learning.ipynb	Code updates for TF 2.3.
06_ai_first_finance.ipynb	Code updates for TF 2.3.
07_dense_networks.ipynb	Code updates for TF 2.3.
08_recurrent_networks.ipynb	Code updates for TF 2.3.
09_reinforcement_learning_a.ipynb	Code updates.
09_reinforcement_learning_b.ipynb	Code updates for TF 2.3.



Source: <https://github.com/yhilpisch/aiif/tree/main/code>

Python in Google Colab (Python101)

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



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: <https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT?authuser=2#scrollTo=wsh36fLxDKC3>. The notebook title is "python101.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with options for CODE, TEXT, CELL, and a status indicator showing "CONNECTED" and "EDITING".

The notebook contains four code cells:

- Cell 1:** A code cell with the following Python code:

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

The output is "194.87".
- Cell [11]:** A code cell with the following Python code:

```
1 amount = 100
2 interest = 10 #10% = 0.01 * 10
3 years = 7
4
5 future_value = amount * ((1 + (0.01 * interest)) ** years)
6 print(round(future_value, 2))
```

The output is "194.87".
- Cell [12]:** A code cell with the following Python code:

```
1 # Python Function def
2 def getfv(pv, r, n):
3     fv = pv * ((1 + (r)) ** n)
4     return fv
5 fv = getfv(100, 0.1, 7)
6 print(round(fv, 2))
```

The output is "194.87".
- Cell [13]:** A code cell with the following Python code:

```
1 # Python if else
2 score = 80
3 if score >=60 :
4     print("Pass")
5 else:
6     print("Fail").
```

The output is "Pass".

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", along with a "Comment" button, a "Share" button, and a settings gear. A "Table of contents" sidebar is open on the left, listing various topics under "AI in Finance", with "Uncertainty and Risk" currently selected. The main content area displays a table of contents with expandable sections: "AI in Finance", "Normative Finance and Financial Theories", and "Uncertainty and Risk". Below the table of contents, a code cell is visible, containing Python code that initializes NumPy arrays for stock and bond prices and payoffs.

python101.ipynb ☆
File Edit View Insert Runtime Tools Help [All changes saved](#) Comment Share ⚙️ A

RAM Disk Editing ^

Table of contents

- AI in Finance
 - Normative Finance and Financial Theories
 - Uncertainty and Risk**
 - Expected Utility Theory (EUT)
 - Mean-Variance Portfolio Theory (MVPT)
 - Capital Asset Pricing Model (CAPM)
 - Arbitrage Pricing Theory (APT)
 - Deep Learning for Financial Time Series Forecasting
 - Portfolio Optimization and Algorithmic Trading
 - Investment Portfolio Optimisation with Python
 - Efficient Frontier Portfolio Optimisation in Python
 - Investment Portfolio Optimization

AI in Finance

- Source: Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media.
- Github: <https://github.com/yhilpisch/aiif/>

Normative Finance and Financial Theories

Uncertainty and Risk

```
1 import numpy as np
2
3 #The prices of the stock and bond today.
4 S0 = 10
5 B0 = 10
6 print('S0', S0)
7 print('B0', B0)
8
9 #The uncertain payoff of the stock and bond tomorrow.
10 S1 = np.array((20, 5))
11 B1 = np.array((11, 11))
12 print('S1', S1)
13 print('B1', B1)
14
15 #The market price vector
16 M0 = np.array((S0, B0))
```

<https://tinyurl.com/aintpuython101>

Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share Settings A

RAM Disk Editing ^

Table of contents

- Data Driven Finance
 - Financial Econometrics and Regression**
 - Data Availability
 - Normative Theories Revisited
 - Mean-Variance Portfolio Theory
 - Capital Asset Pricing Model
 - Arbitrage-Pricing Theory
 - Debunking Central Assumptions
 - Normality
 - Sample Data Sets
 - Real Financial Returns
 - Linear Relationships
- Deep Learning for Financial Time Series Forecasting
- Portfolio Optimization and Algorithmic Trading
 - Investment Portfolio Optimisation with Python
 - Efficient Frontier Portfolio Optimisation in Python
 - Investment Portfolio Optimization

Data Driven Finance

Financial Econometrics and Regression

```
[18] 1 import numpy as np
      2
      3 def f(x):
      4     return 2 + 1 / 2 * x
      5
      6 x = np.arange(-4, 5)
      7 x

array([-4, -3, -2, -1, 0, 1, 2, 3, 4])
```

```
1 y = f(x)
2 y

array([ 0.00,  0.50,  1.00,  1.50,  2.00,  2.50,  3.00,  3.50,  4.00])
```

```
1 print('x', x)
2
3 print('y', y)
4
5 beta = np.cov(x, y, ddof=0)[0, 1] / x.var()
6 print('beta', beta)
```

Python in Google Colab (Python101)

The screenshot shows a Google Colab notebook interface. At the top, the notebook is titled 'python101.ipynb' and has a star icon. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', with a status message 'All changes saved'. On the right, there are icons for 'Comment', 'Share', and a user profile 'A'. Below the menu, there are indicators for 'RAM' and 'Disk' usage, and a status 'Editing'. A 'Table of contents' sidebar is on the left, listing various topics. The main area shows a code cell with the following Python code:

```
1 import numpy as np
2 import pandas as pd
3 from pylab import plt, mpl
4 np.random.seed(100)
5 plt.style.use('seaborn')
6 mpl.rcParams['savefig.dpi'] = 300
7 mpl.rcParams['font.family'] = 'serif'
8
9 url = 'http://hilpisch.com/aiif_eikon_eod_data.csv'
10
11 raw = pd.read_csv(url, index_col=0, parse_dates=True)['EUR=']
12 raw.head()
```

The output of the code cell is a DataFrame with the following data:

Date	EUR=
2010-01-01	1.4323
2010-01-04	1.4411
2010-01-05	1.4368
2010-01-06	1.4412
2010-01-07	1.4318

Below the output, there is a code cell with the following code:

```
[2] 1 raw.tail()
```

Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help Saving...

Comment Share

RAM Disk Editing

Table of contents

- Financial Econometrics and Regression
- Data Availability
- Normative Theories Revisited
 - Mean-Variance Portfolio Theory
 - Capital Asset Pricing Model
 - Arbitrage-Pricing Theory
- Debunking Central Assumptions
- Normality
 - Sample Data Sets
 - Real Financial Returns
- Linear Relationships
- Financial Econometrics and Machine Learning
 - Machine Learning
 - Data
 - Success**
 - Capacity
 - Evaluation
 - Bias & Variance
 - Cross-Validation

```
+ Code + Text
```

Success

```
1 def MSE(l, p):
2     return np.mean([(1 - p) ** 2])
```

```
[9] 1 reg = np.polyfit(f, l, deg=5)
     2 reg

array([-0.01910626, -0.0147182 ,  0.10990388,  0.06007211, -0.20833598,
        -0.03275423])
```

```
[10] 1 p = np.polyval(reg, f)
      2 p

array([ 0.12088427,  0.11526131,  0.11080193,  0.10739461,  0.10493286,
         0.10331514,  0.10244475,  0.10222973,  0.10258281,  0.10342126,
         0.10466683,  0.10624564,  0.1080881 ,  0.1101288 ,  0.11230643,
         0.11456366,  0.11684709,  0.11910711,  0.12129784,  0.123377 ,
         0.12530587,  0.12704913,  0.12857481,  0.1298542 ,  0.1308617 ,
         0.1315748 ,  0.13197395,  0.13204243,  0.13176634,  0.13113443,
         0.13013803,  0.12877097,  0.12702948,  0.12491207,  0.12241947,
         0.11955452,  0.11632208,  0.11272891,  0.10878364,  0.1044966 ,
         0.09987977,  0.09494668,  0.0897123 ,  0.08419296,  0.07840627,
         0.07237098,  0.06610693,  0.05963494,  0.05297671,  0.04615473,
         0.03919218,  0.03211286,  0.02494106,  0.01770149,  0.01041918,
         0.00311939, -0.00417251, -0.0114311 , -0.01863101, -0.02574704,
        -0.03275423, -0.03962796, -0.04634406, -0.05287887, -0.05920936,
        -0.06531322, -0.07116897, -0.07675602, -0.08205478, -0.08704677,
        -0.09171147, -0.09601254, -0.10001567, -0.10328882, -0.10651674])
```


Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help Saving...

Comment Share Settings A

RAM Disk Editing

Table of contents

- Financial Econometrics and Regression
- Data Availability
- Normative Theories Revisited
 - Mean-Variance Portfolio Theory
 - Capital Asset Pricing Model
 - Arbitrage-Pricing Theory
- Debunking Central Assumptions
- Normality
 - Sample Data Sets
 - Real Financial Returns
- Linear Relationships
- Financial Econometrics and Machine Learning
 - Machine Learning
 - Data
 - Success**
 - Capacity
 - Evaluation
 - Bias & Variance
 - Cross-Validation

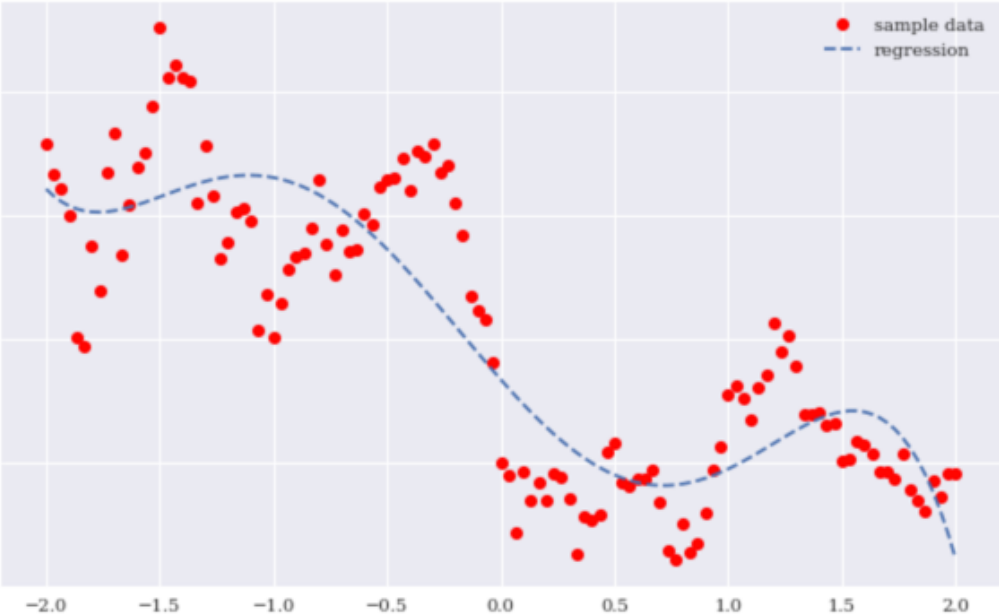
+ Code + Text

```
[11] 1 MSE(l, p)

0.003416642295737103
```

```
1 plt.figure(figsize=(10, 6))
2 plt.plot(f, l, 'ro', label='sample data')
3 plt.plot(f, p, '--', label='regression')
4 plt.legend()
```

<matplotlib.legend.Legend at 0x7f66a41d5750>



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share Settings

RAM Disk Editing

Table of contents

- Financial Econometrics and Regression
- Data Availability
- Normative Theories Revisited
 - Mean-Variance Portfolio Theory
 - Capital Asset Pricing Model
 - Arbitrage-Pricing Theory
- Debunking Central Assumptions
- Normality
 - Sample Data Sets
 - Real Financial Returns
- Linear Relationships
- Financial Econometrics and Machine Learning
 - Machine Learning
 - Data
 - Success
 - Capacity**
 - Evaluation
 - Bias & Variance
 - Cross-Validation

+ Code + Text

```
[21] 1 reg = {}
      2 for d in range(1, 12, 2):
      3     reg[d] = np.polyfit(f, l, deg=d)
      4     p = np.polyval(reg[d], f)
      5     mse = MSE(l, p)
      6     print(f'{d:2d} | MSE={mse}')

1 | MSE=0.005322474034260403
3 | MSE=0.004353110724143185
5 | MSE=0.003416642295737103
7 | MSE=0.002738950177235401
9 | MSE=0.0014119616263308346
11 | MSE=0.0012651237868752398
```

```
1 plt.figure(figsize=(10, 6))
2 plt.plot(f, l, 'ro', label='sample data')
3 for d in reg:
4     p = np.polyval(reg[d], f)
5     plt.plot(f, p, '--', label=f'deg={d}')
6 plt.legend()
```

<matplotlib.legend.Legend at 0x7f6630300310>

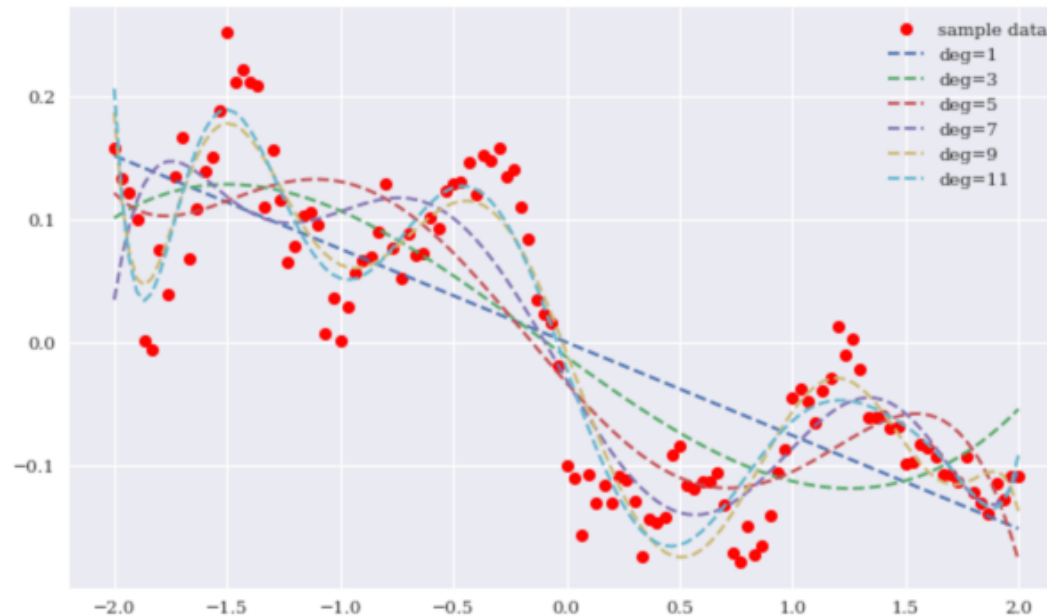
Python in Google Colab (Python101)

- Financial Econometrics and Regression
- Data Availability
- Normative Theories Revisited
 - Mean-Variance Portfolio Theory
 - Capital Asset Pricing Model
 - Arbitrage-Pricing Theory
- Debunking Central Assumptions
- Normality
 - Sample Data Sets
 - Real Financial Returns
- Linear Relationships
- Financial Econometrics and Machine Learning
 - Machine Learning
 - Data
 - Success
 - Capacity**
 - Evaluation
 - Bias & Variance
 - Cross-Validation

```
9 | MSE=0.0014119616263308346  
11 | MSE=0.0012651237868752398
```

```
1 plt.figure(figsize=(10, 6))  
2 plt.plot(f, l, 'ro', label='sample data')  
3 for d in reg:  
4     p = np.polyval(reg[d], f)  
5     plt.plot(f, p, '--', label=f'deg={d}')  
6 plt.legend()
```

<matplotlib.legend.Legend at 0x7f6630300310>



Python in Google Colab (Python101)

```
def create_dnn_model(hl=1, hu=256):  
    ''' Function to create Keras DNN model.  
    Parameters  
    =====  
    hl: int  
    number of hidden layers  
    hu: int  
    number of hidden units (per layer)  
    '''  
    model = Sequential()  
    for _ in range(hl):  
        model.add(Dense(hu, activation='relu', input_dim=1))  
    model.add(Dense(1, activation='linear'))  
    model.compile(loss='mse', optimizer='rmsprop')  
    return model  
  
model = create_dnn_model(3)  
  
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_2 (Dense)	(None, 256)	512
dense_3 (Dense)	(None, 256)	65792
dense_4 (Dense)	(None, 256)	65792
dense_5 (Dense)	(None, 1)	257

=====
Total params: 132,353
Trainable params: 132,353
Non-trainable params: 0

Python in Google Colab (Python101)

python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings A

RAM Disk Editing

Table of contents

- Financial Econometrics and Regression
- Data Availability
- Normative Theories Revisited
 - Mean-Variance Portfolio Theory
 - Capital Asset Pricing Model
 - Arbitrage-Pricing Theory
- Debunking Central Assumptions
- Normality
 - Sample Data Sets
 - Real Financial Returns
- Linear Relationships
- Financial Econometrics and Machine Learning
 - Machine Learning
 - Data
 - Success
 - Capacity**
 - Evaluation
 - Bias & Variance
 - Cross-Validation

```
1 p = model.predict(f).flatten()
2
3 MSE(l, p)
4
5 plt.figure(figsize=(10, 6))
6 plt.plot(f, l, 'r', label='sample data')
7 plt.plot(f, p, '--', label='DNN approximation')
8 plt.legend();
```



Python in Google Colab (Python101)



python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment

Share



A

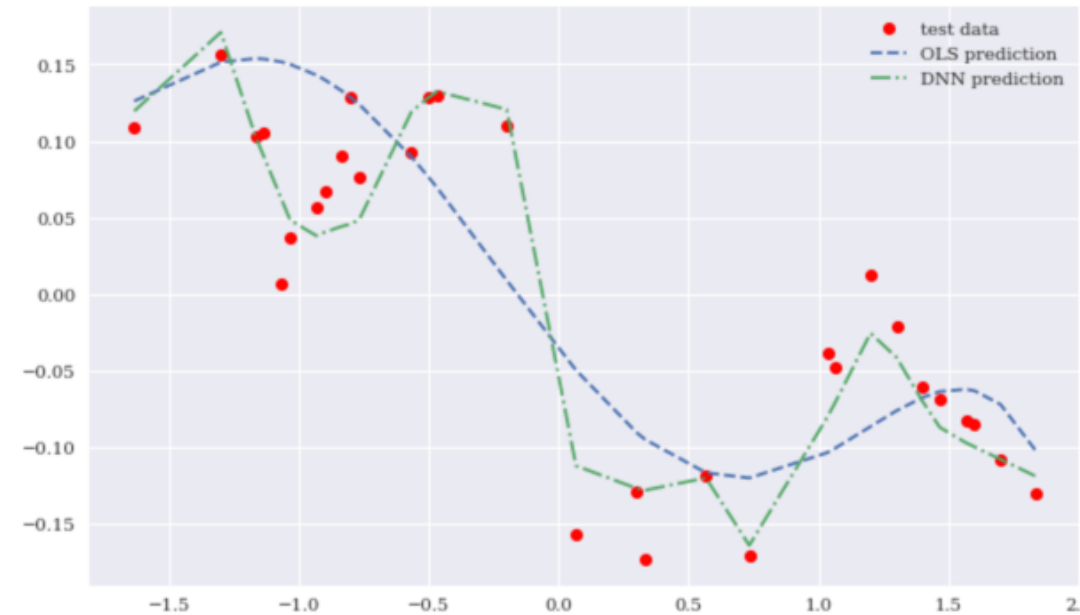
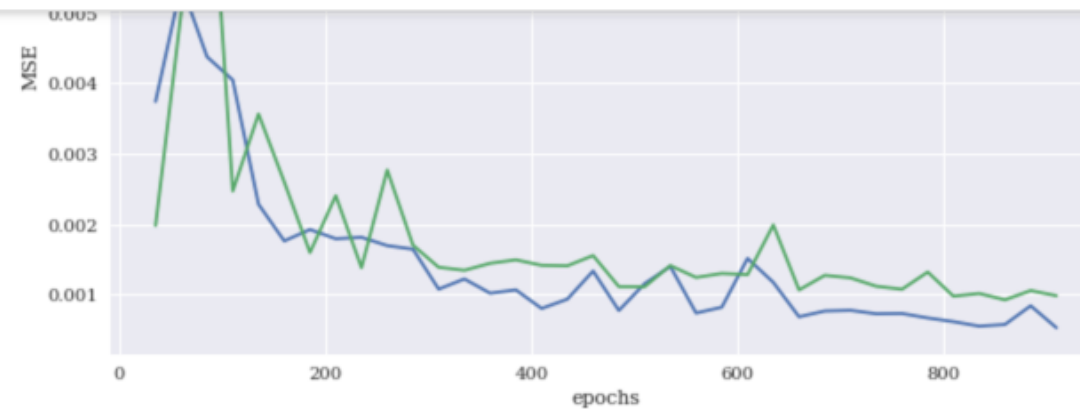
Table of contents

- Financial Econometrics and Regression
- Data Availability
- Normative Theories Revisited
 - Mean-Variance Portfolio Theory
 - Capital Asset Pricing Model
 - Arbitrage-Pricing Theory
- Debunking Central Assumptions
- Normality
 - Sample Data Sets
 - Real Financial Returns
- Linear Relationships
- Financial Econometrics and Machine Learning
 - Machine Learning
 - Data
 - Success
 - Capacity
 - Evaluation**
 - Bias & Variance
 - Cross-Validation

+ Code + Text

RAM
Disk

Editing



<https://tinyurl.com/aintpupython101>

Summary

- **Financial Econometrics**
 - **Financial Theories**
 - **OLS Regression**
- **Machine Learning**
 - **Learning**
 - **Evaluation**
 - **Bias and variance**
 - **Cross-validation**

References

- Yves Hilpisch (2020), Artificial Intelligence in Finance: A Python-Based Guide, O'Reilly Media, <https://github.com/yhilpisch/aiif> .
- Chris Brooks (2019), Introductory Econometrics for Finance, 4th Edition, Cambridge University Press
- Oliver Linton (2019), Financial Econometrics: Models and Methods, Cambridge University Press
- Tom Mitchell (1997), Machine Learning, McGraw-Hill.
- Min-Yuh Day (2022), Python 101, <https://tinyurl.com/aintpupython101>