

인공지능과제

딥러닝 오픈소스 기반 협력 추천





인공지능 과제(20%)

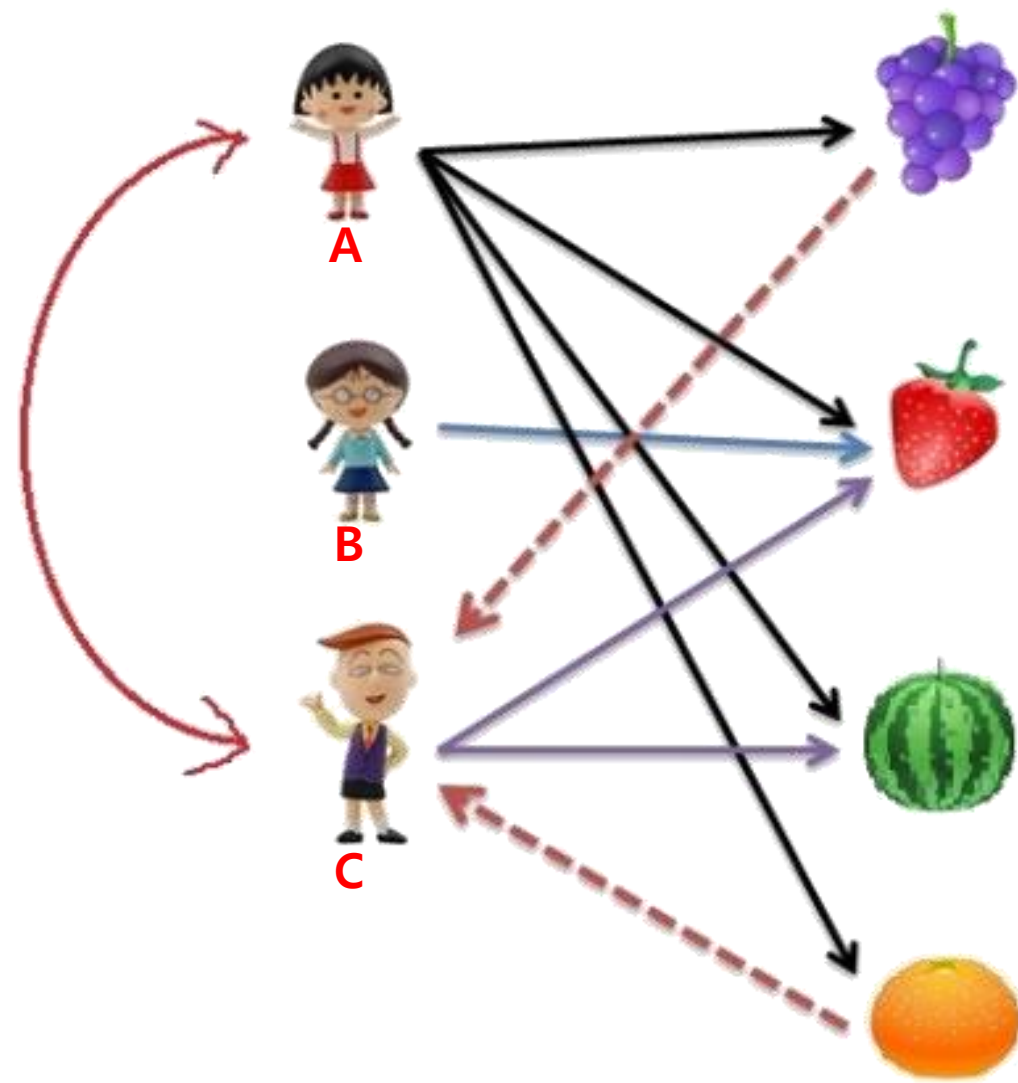
- ▶ 프로젝트 관련 문의 (평일 10:00 ~ 18:00)
 - ▶ 담당 조교 : 공성언
 - ▶ E-Mail : 2ndggong@hanyang.ac.kr
 - ▶ 연구실 : IT/BT관 604-2호(2220-4139)
 - ▶ 010-5703-7587, Kakao : a1ggoma
 - ▶ 담당 조교 : 조수필 (인공지능 멘토)
 - ▶ E-Mail : chosuphil@naver.com
 - ▶ 연구실 : 산학기술관 508호
 - ▶ 010-6424-4083, Kakao : jessay



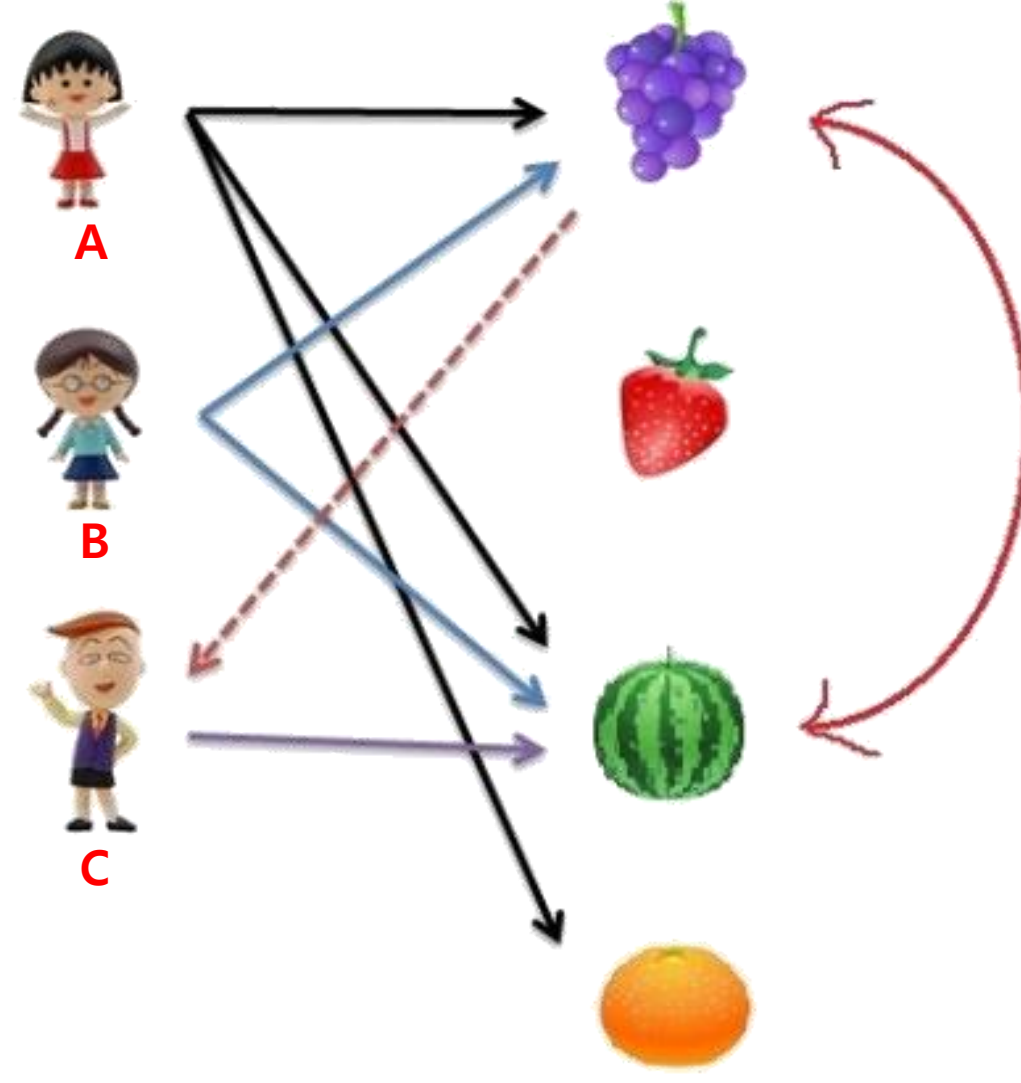
인공지능 과제(20%)

- ▶ 프로젝트 내용
 - ▶ 딥러닝 오픈 소스를 활용한 영화 협력 추천 모델 구상
 - ▶ 영화 협력 추천 모델의 정확도(Precision)를 기준으로 Contest 형식으로 진행
- ▶ 딥러닝 오픈 소스 API : DL4J (DeepLearning for Java)
- ▶ 영화 평점 데이터 : Epinions Dataset

협력 추천



User-based filtering



Item-based filtering

- ▶ 어떠한 아이템, 콘텐츠인지 몰라도 그냥 사용자와 아이템간의 구매 혹은 매핑 정보만을 활용하여 추천해주는 방식
- ▶ 우리는 이것을 영화 평점에 적용하여 영화 추천 진행

DL4J (Deep Learning for Java)

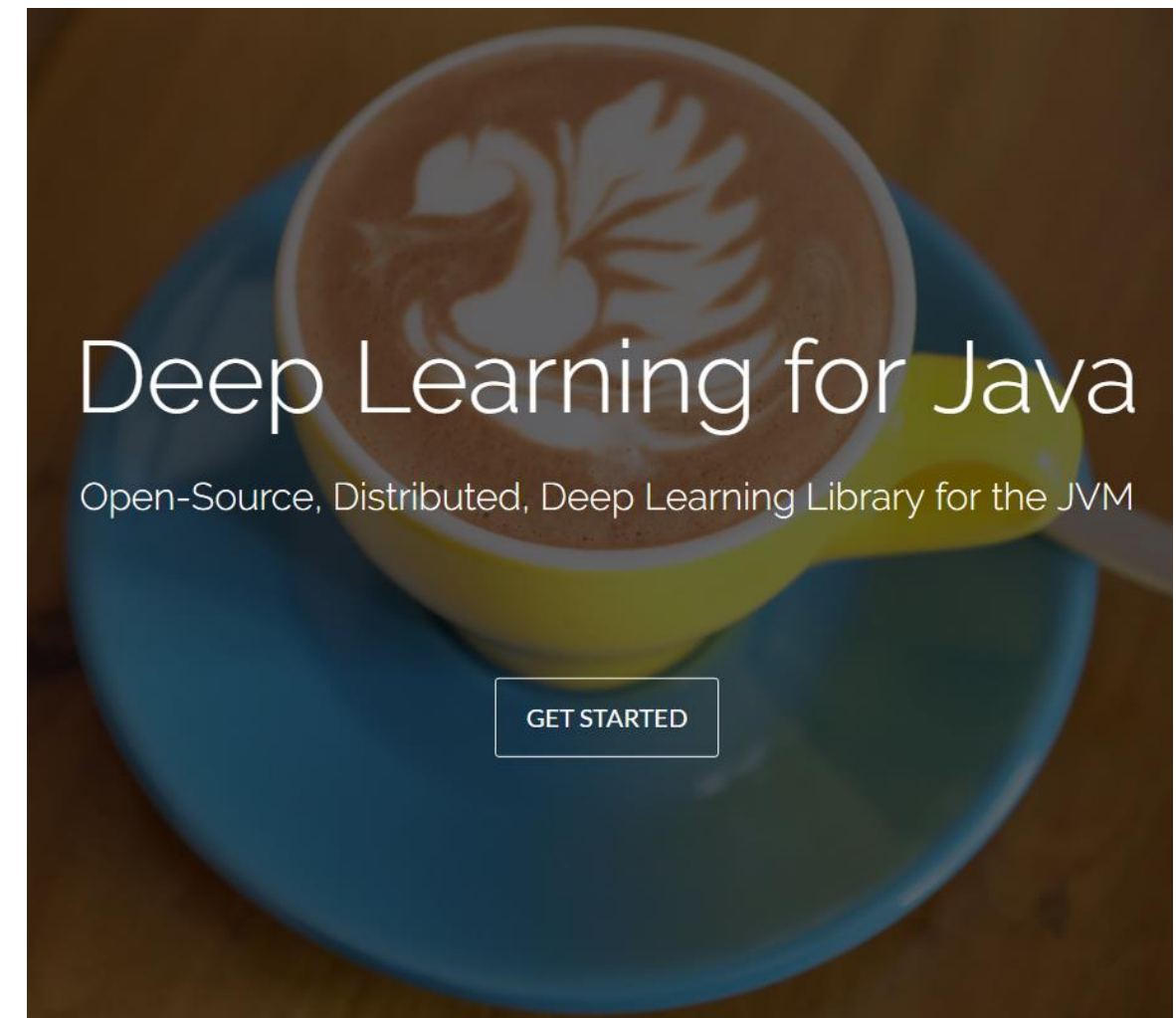
▶ DL4J란?

- ▶ 인공 신경망을 구성 및 학습이 가능한 자바 기반 오픈 소스 API
- ▶ <https://deeplearning4j.org>

▶ DL4J 설치

- ▶ Java 최신버전 (64 bit version)
- ▶ Eclipse (or IntelliJ)
- ▶ Apache Maven
- ▶ Git & GitLab

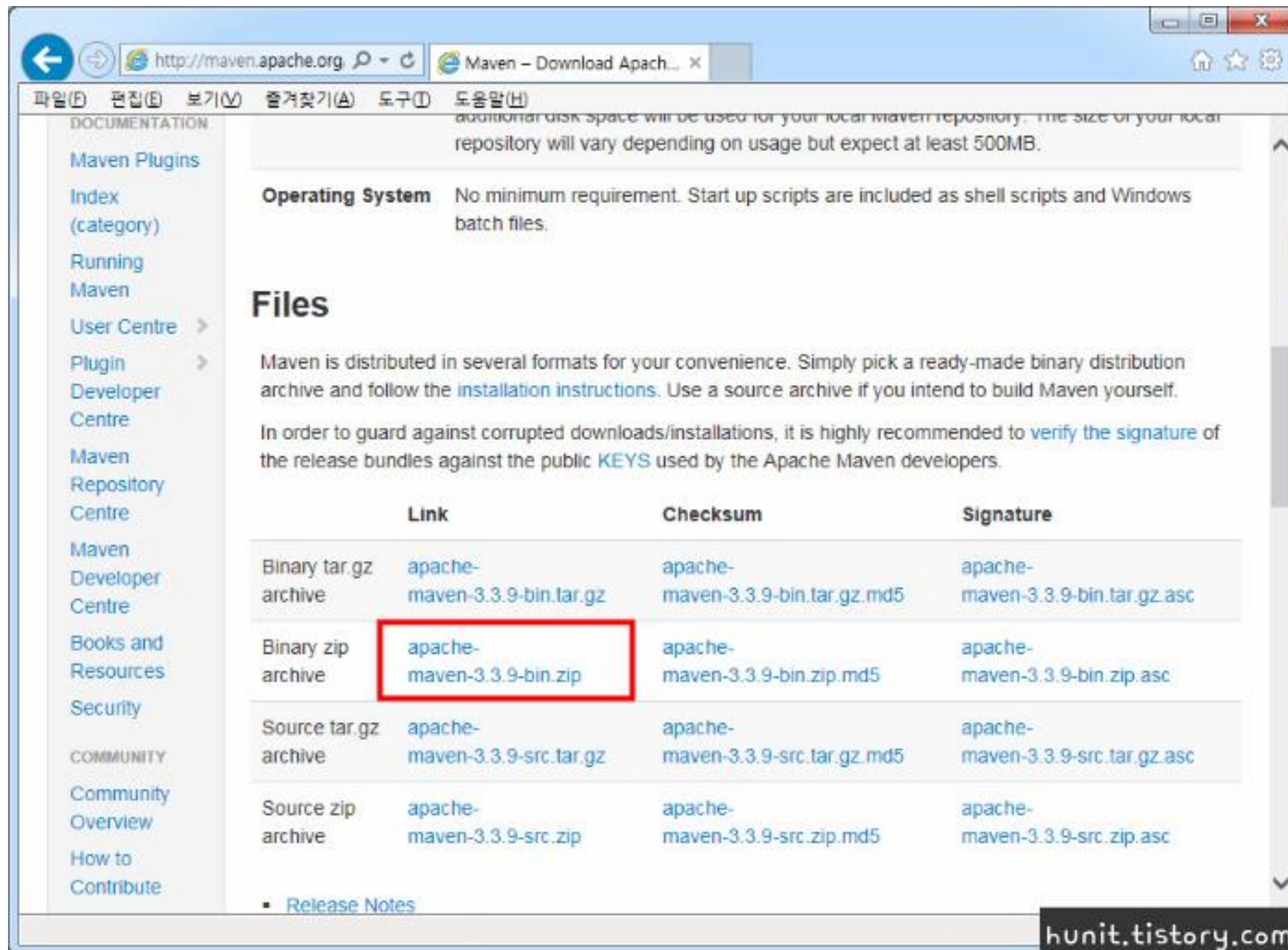
- ▶ 퀵 스타트 가이드 : <https://deeplearning4j.org/kr/quickstart>



DL4J (Deep Learning for Java)

▶ 메이븐(Maven) 다운로드

1) <http://maven.apache.org/download.cgi> 에서 메이븐 3.3.9 zip 파일 다운



The screenshot shows the Maven download page with the following content:

Operating System No minimum requirement. Start up scripts are included as shell scripts and Windows batch files.

Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

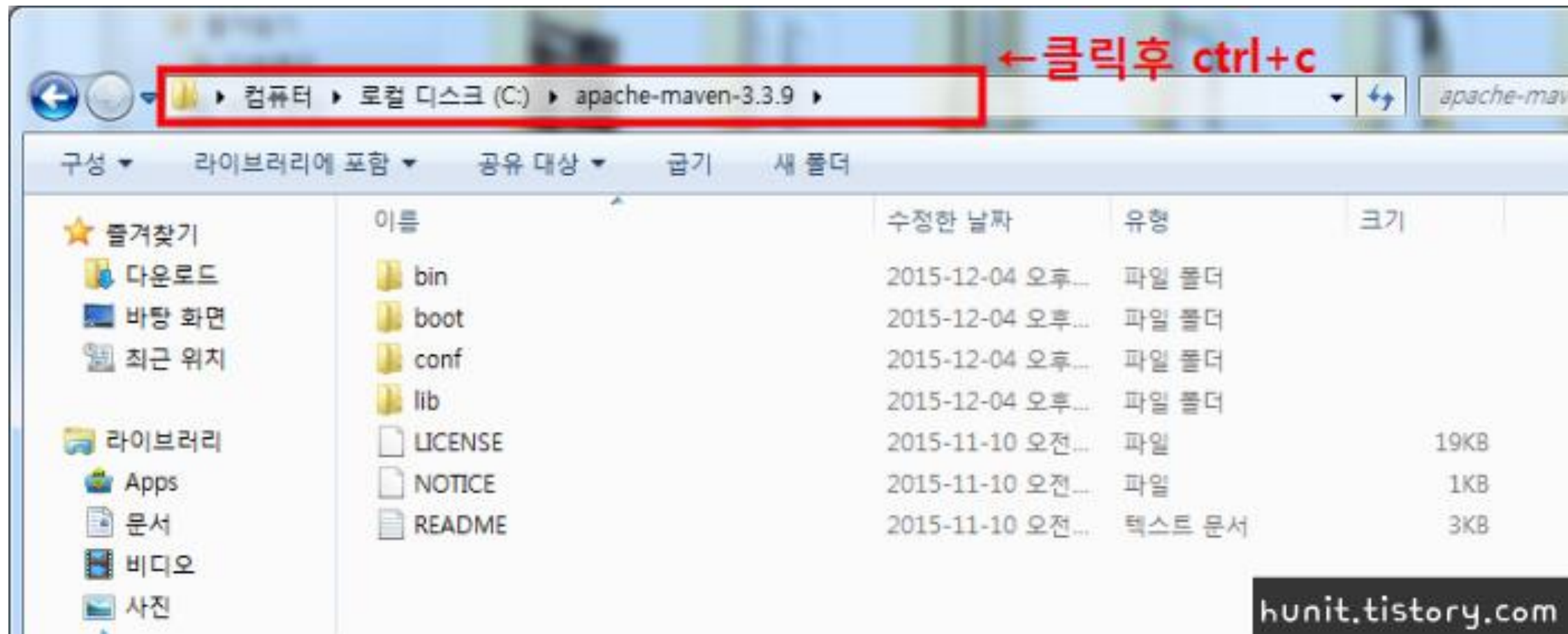
	Link	Checksum	Signature
Binary tar.gz archive	apache-maven-3.3.9-bin.tar.gz	apache-maven-3.3.9-bin.tar.gz.md5	apache-maven-3.3.9-bin.tar.gz.asc
Binary zip archive	apache-maven-3.3.9-bin.zip	apache-maven-3.3.9-bin.zip.md5	apache-maven-3.3.9-bin.zip.asc
Source tar.gz archive	apache-maven-3.3.9-src.tar.gz	apache-maven-3.3.9-src.tar.gz.md5	apache-maven-3.3.9-src.tar.gz.asc
Source zip archive	apache-maven-3.3.9-src.zip	apache-maven-3.3.9-src.zip.md5	apache-maven-3.3.9-src.zip.asc

▪ [Release Notes](#)

hunit.tistory.com

DL4J (Deep Learning for Java)

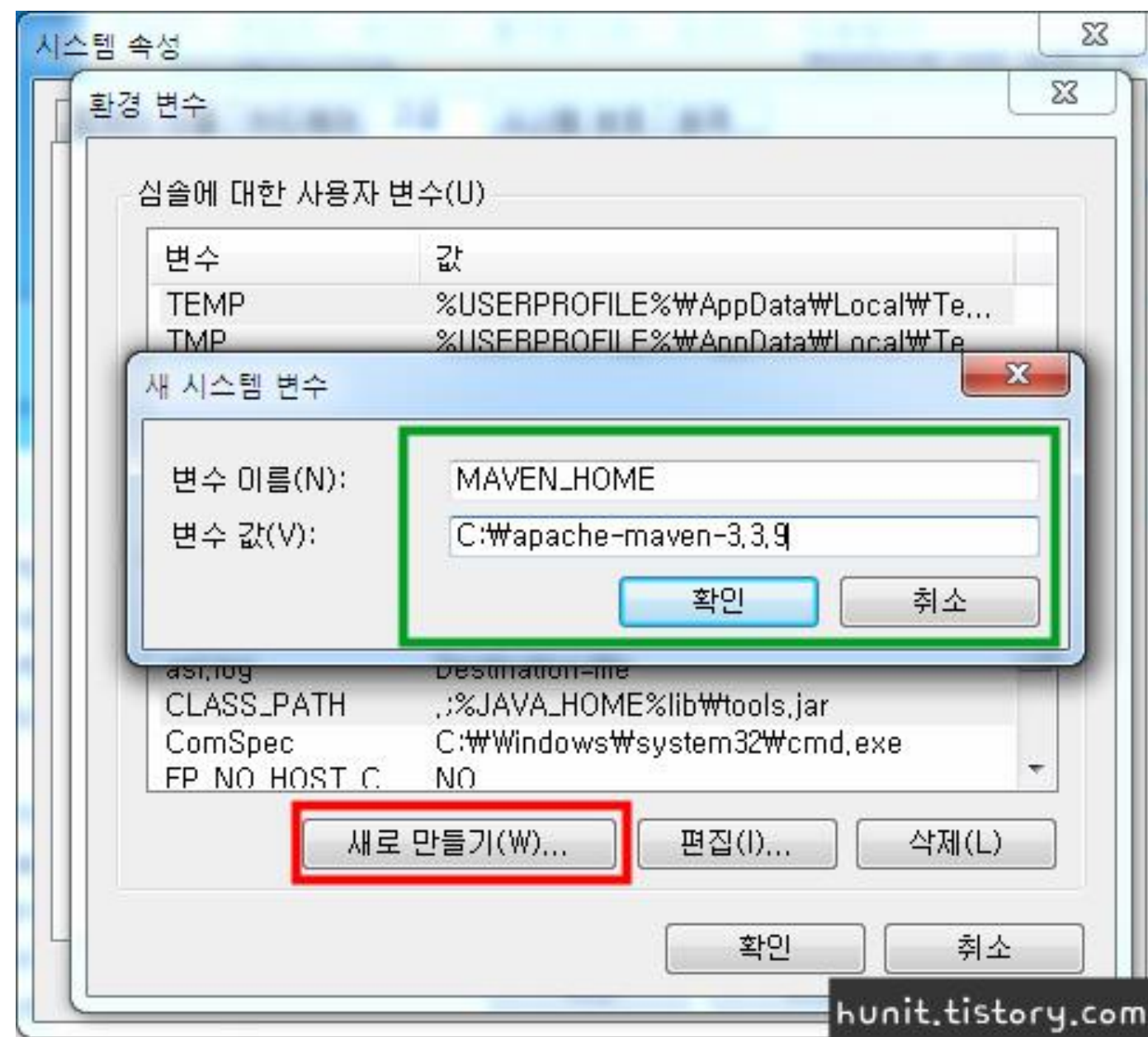
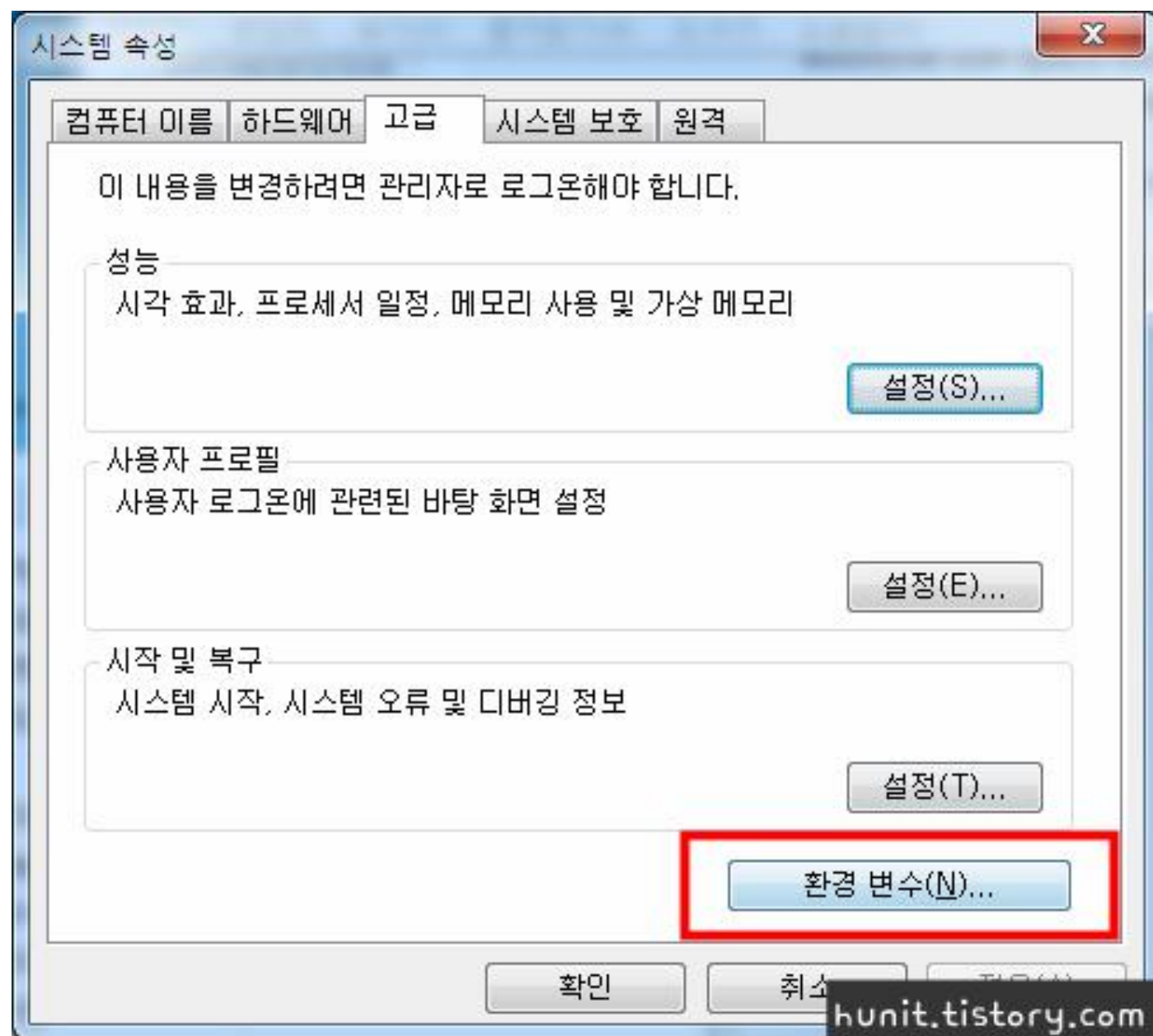
- ▶ 메이븐(Maven) 다운로드
 - 2) 간단한 경로에 압축 해제 후, 폴더 위치 복사



DL4J (Deep Learning for Java)

▶ 메이븐(Maven) 환경 변수 설정

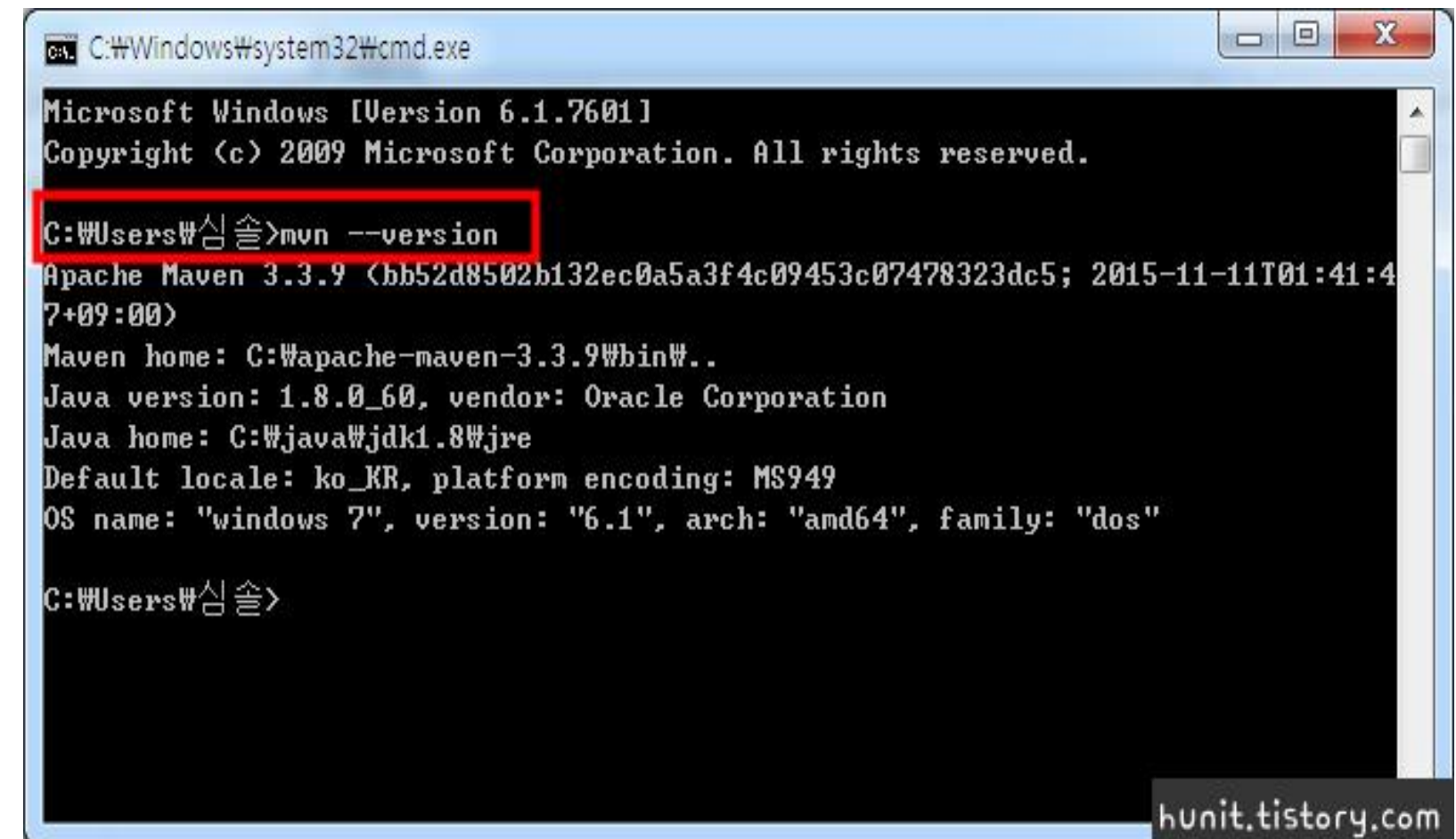
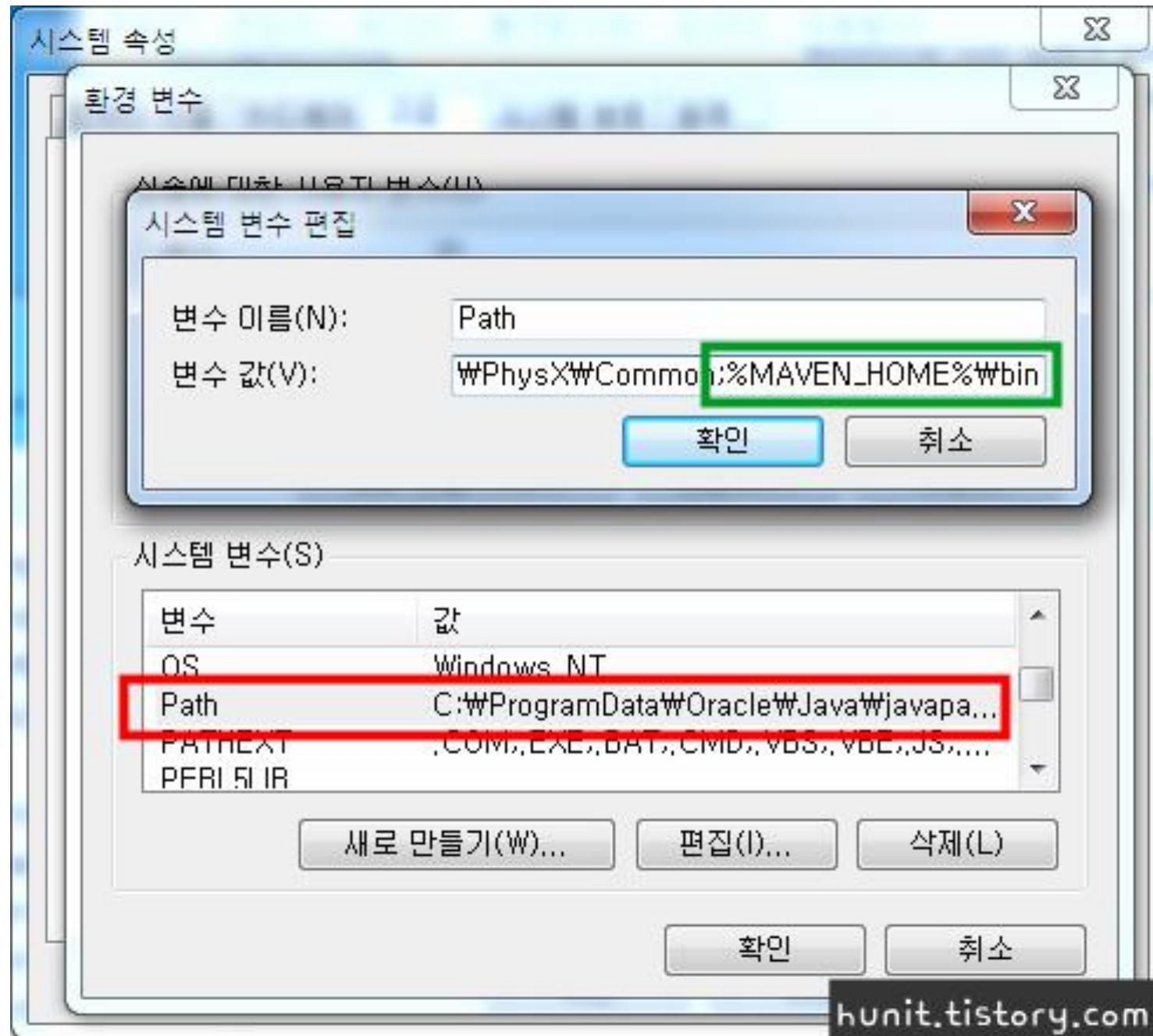
3) 내컴퓨터-속성-환경변수 에서 "새로만들기 " 를 눌러 **초록 네모**와 같이 변수값 이름과 값을 입력



DL4J (Deep Learning for Java)

▶ 메이븐(Maven) 환경 변수 설정

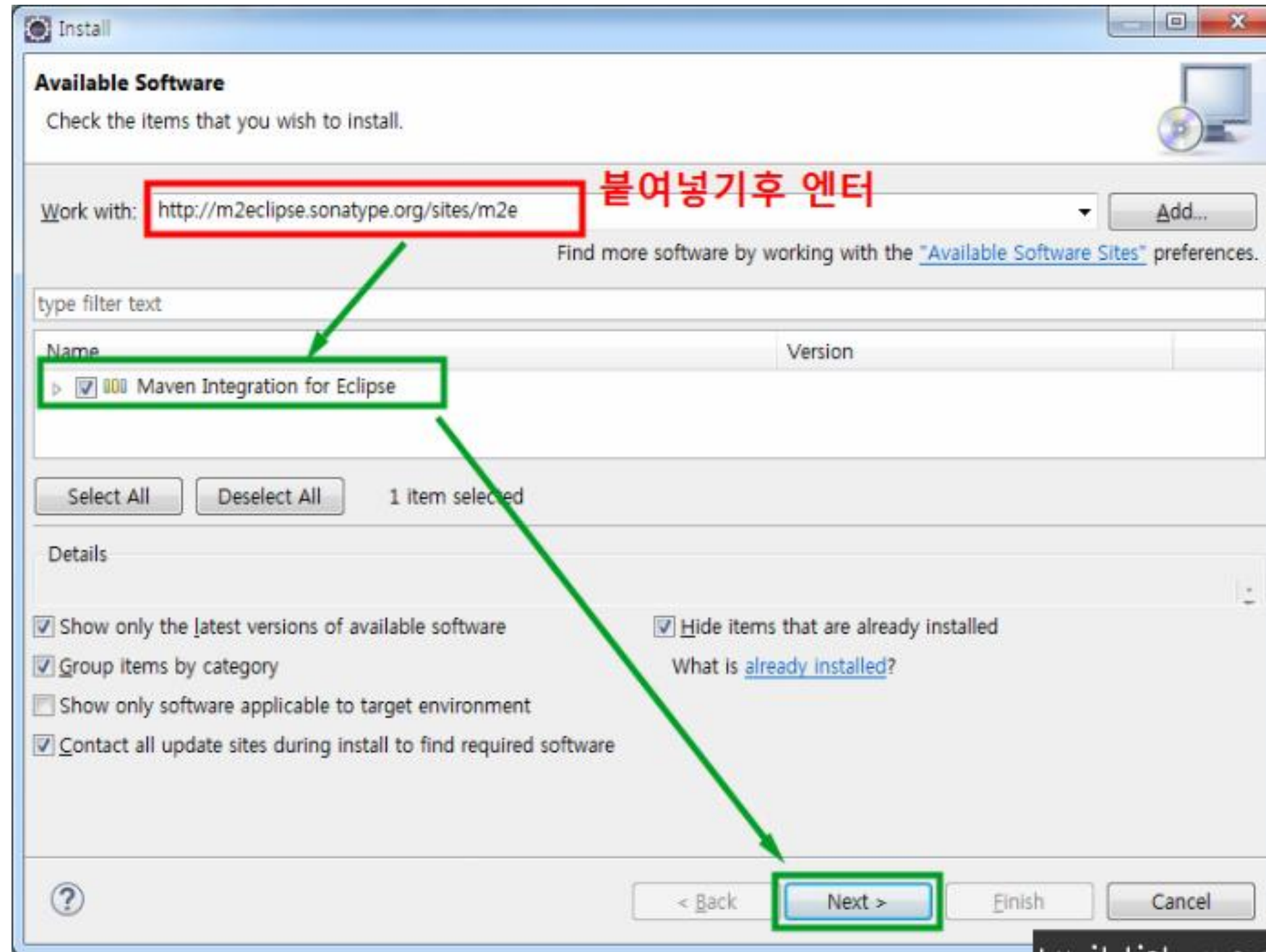
4) 시스템 변수 중 Path를 찾아 편집을 누른 후, 맨 끝에 ;%MAVEN_HOME%\bin 입력



DL4J (Deep Learning for Java)

▶ 이클립스와 메이븐 연동 (필수!)

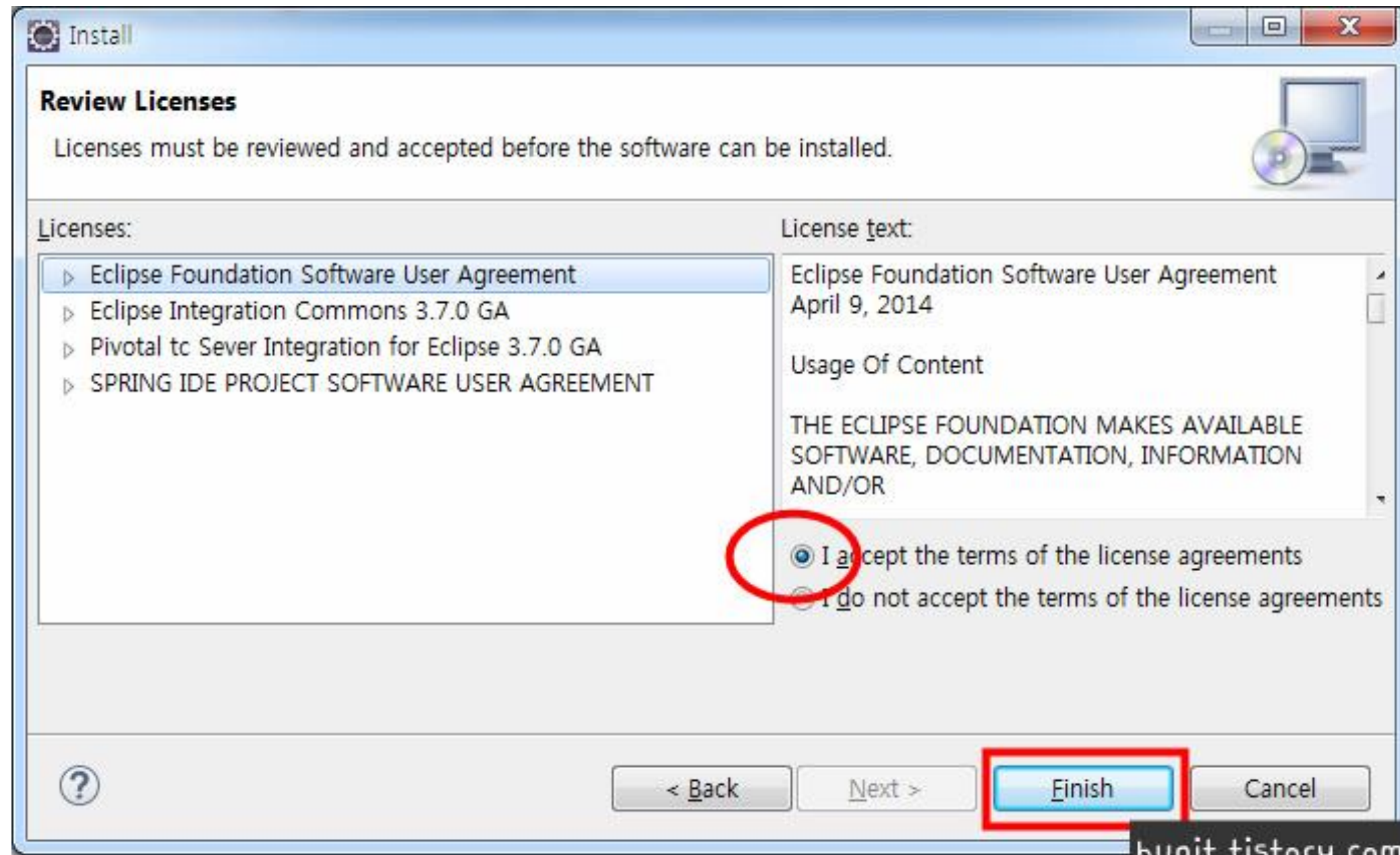
1) 이클립스 HELP – INSTALL NEW SOFTWARE를 클릭하여,
<http://m2eclipse.sonatype.org/sites/m2e>를 복사 후 빨간색 네모칸에 입력



DL4J (Deep Learning for Java)

▶ 이클립스와 메이븐 연동 (필수!)

2) Next 계속 누르고 라이선스 동의하고 Finish 누르면 연동 완료

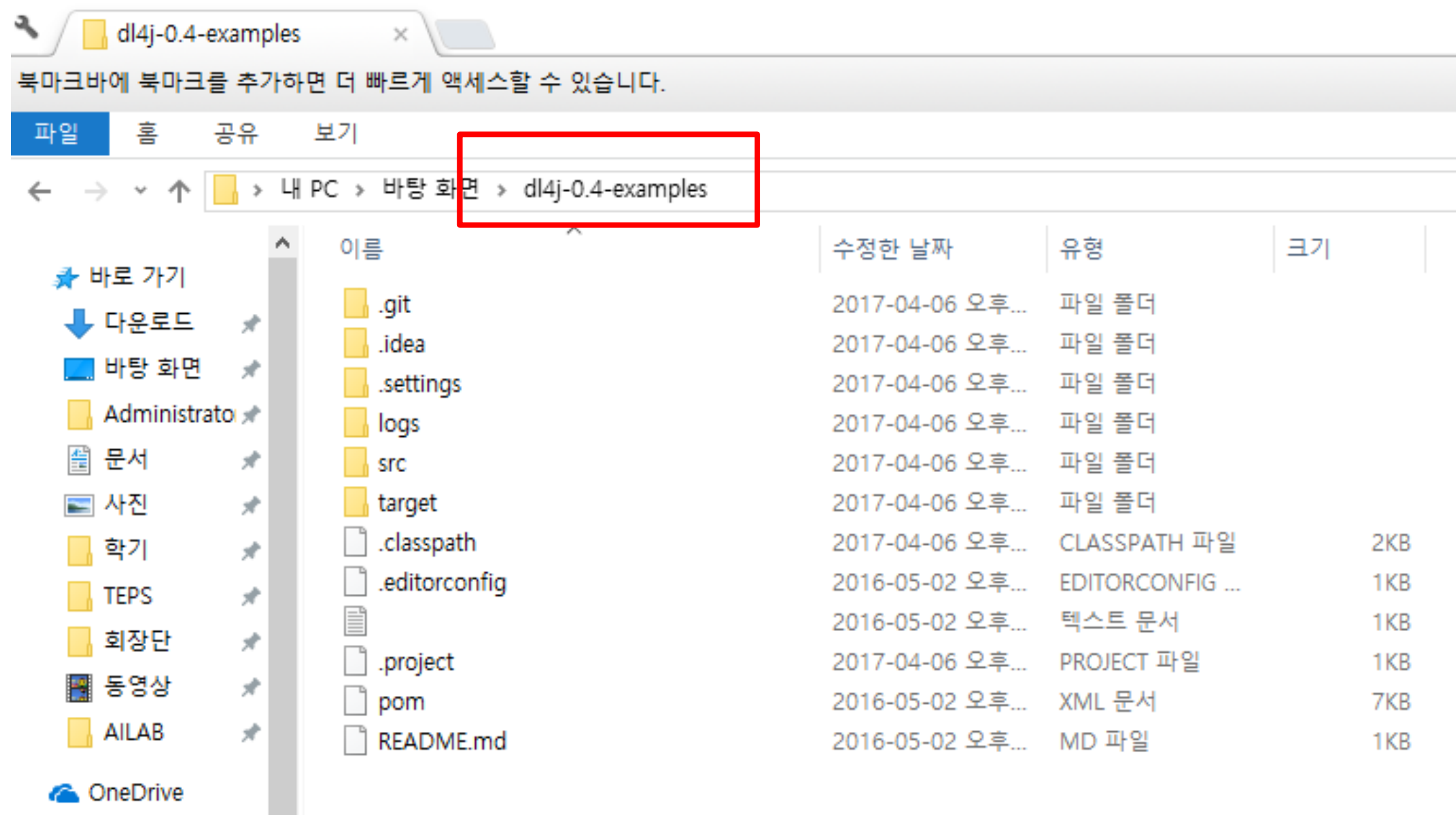


3) 이클립스 재시작

DL4J (Deep Learning for Java)

▶ DL4J Example 프로젝트 Import

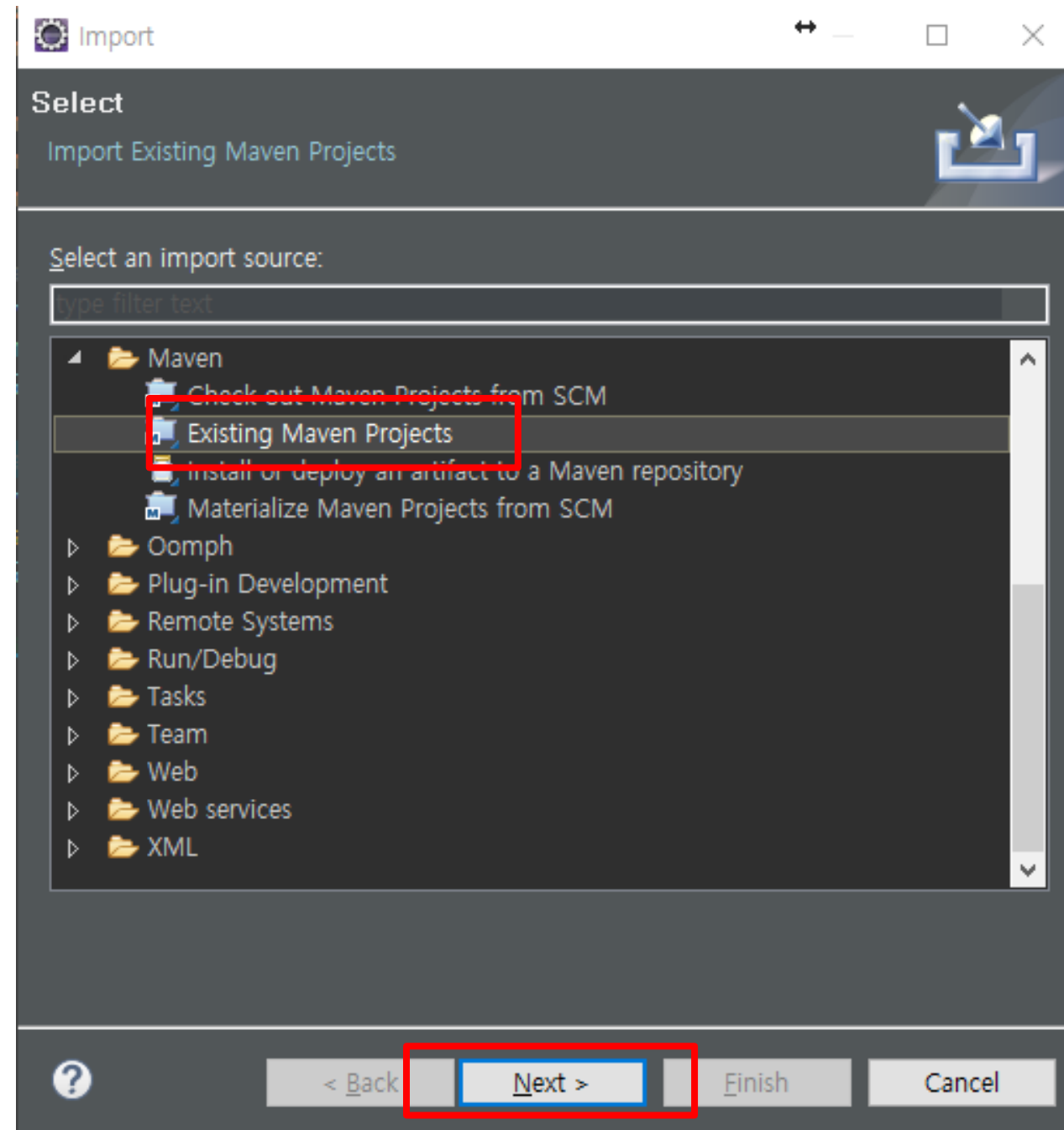
1) 한양인(HY-In) – 인공지능 – 학습자료 에 업로드된 “dl4j-0.4-examples.zip” 파일 다운로드 후 압축 해제



DL4J (Deep Learning for Java)

▶ DL4J Example 프로젝트 Import

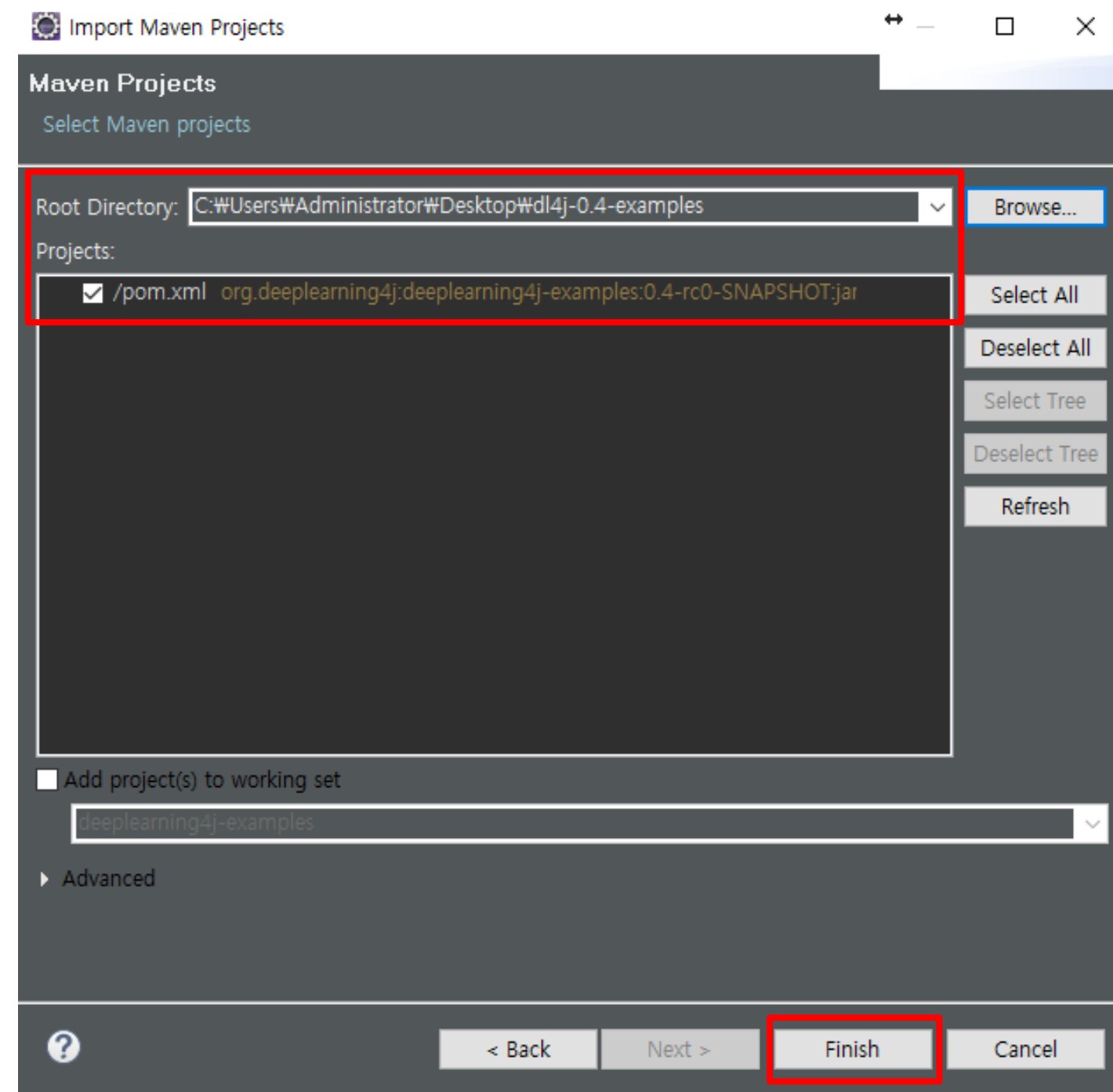
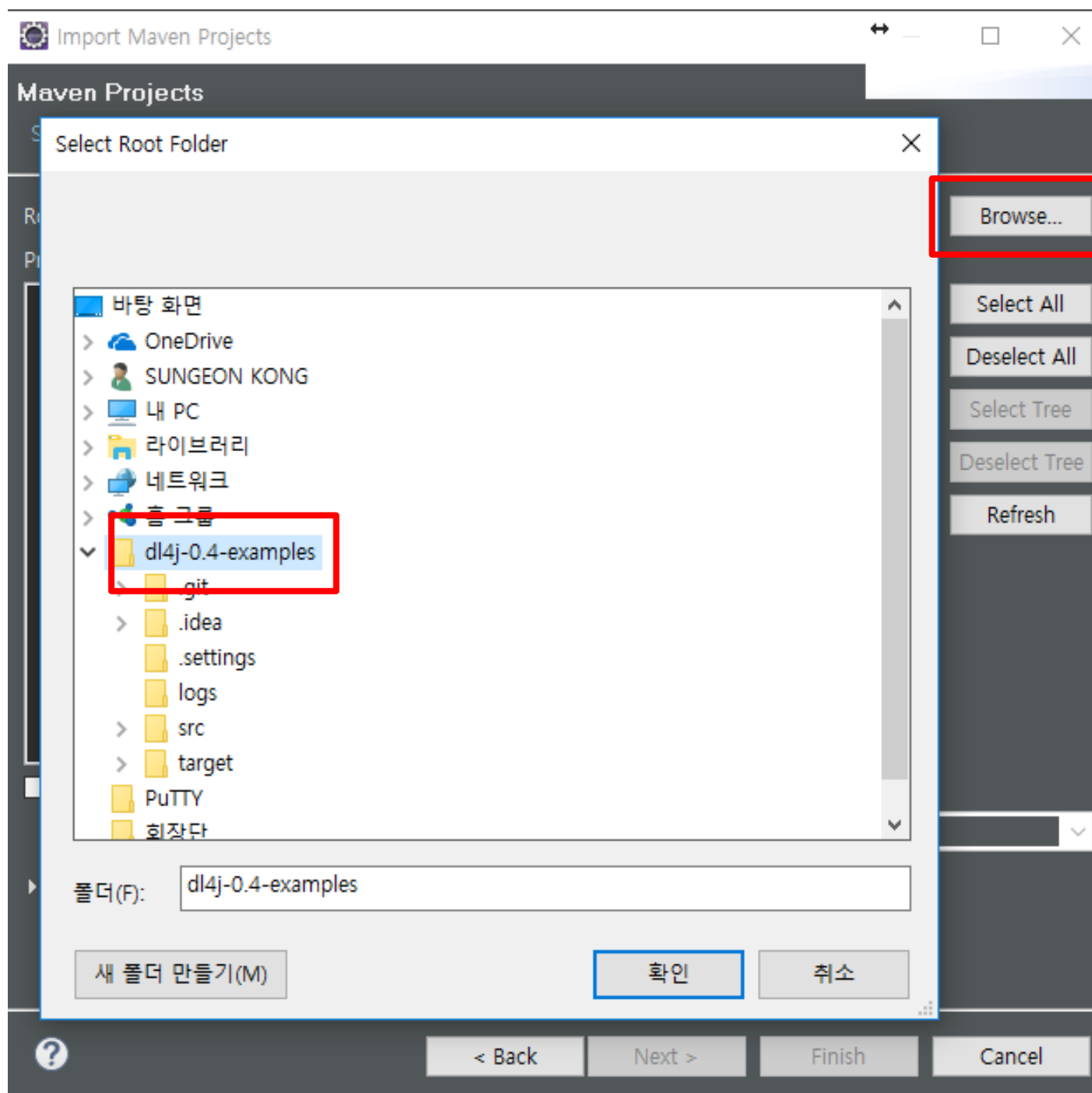
2) 이클립스 FILE – Import를 누르고, Maven – Existing Maven Projects를 누름



DL4J (Deep Learning for Java)

▶ DL4J Example 프로젝트 Import

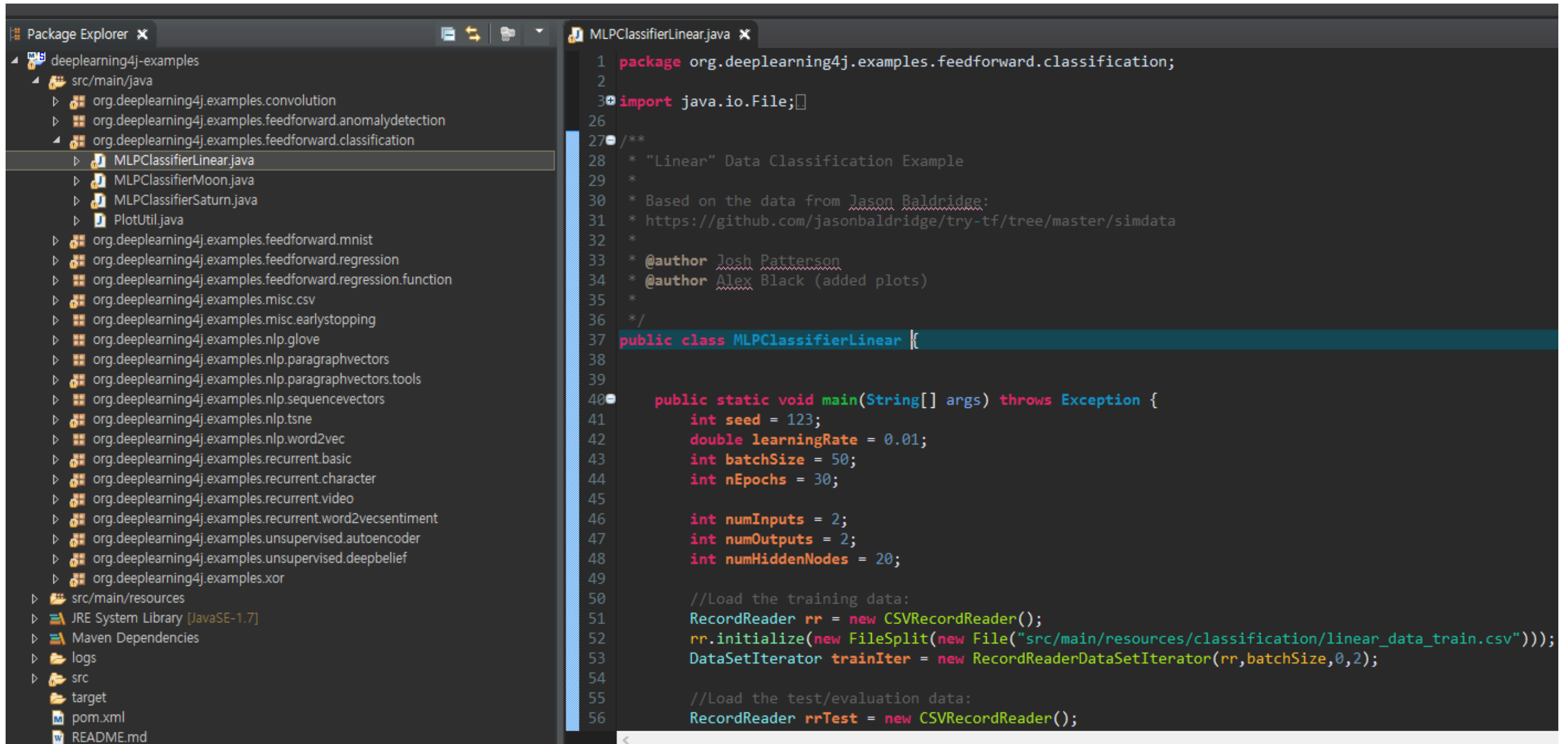
3) Browse 버튼을 눌러 압출 해제된 "dl4j-0.4-examples" 폴더를 선택한 후, Finish를 눌러준다.



DL4J (Deep Learning for Java)

▶ DL4J Example 프로젝트 Import

4) Import 결과



The screenshot displays an IDE interface with two main panels. The left panel, titled 'Package Explorer', shows the project structure for 'deeplearning4j-examples'. It includes a 'src/main/java' directory with several sub-packages under 'org.deeplearning4j.examples'. The 'org.deeplearning4j.examples.feedforward.classification' package is expanded, showing files like 'MLPClassifierLinear.java', 'MLPClassifierMoon.java', 'MLPClassifierSaturn.java', and 'PlotUtil.java'. Other packages include 'mnist', 'regression', 'misc', 'nlp', 'recurrent', 'unsupervised', and 'xor'. The right panel shows the code for 'MLPClassifierLinear.java'. The code starts with a package declaration and an import for 'java.io.File'. It includes a multi-line comment describing the example as a 'Linear' Data Classification Example based on data from Jason Baldridge. The code defines a 'public class MLPClassifierLinear' with a 'main' method that initializes training and test data using 'CSVRecordReader' and 'RecordReaderDataSetIterator'.

```
1 package org.deeplearning4j.examples.feedforward.classification;
2
3 import java.io.File;
4
5 /**
6  * "Linear" Data Classification Example
7  *
8  * Based on the data from Jason Baldridge:
9  * https://github.com/jasonbaldridge/try-tf/tree/master/simdata
10  *
11  * @author Josh Patterson
12  * @author Alex Black (added plots)
13  */
14 public class MLPClassifierLinear {
15
16     public static void main(String[] args) throws Exception {
17         int seed = 123;
18         double learningRate = 0.01;
19         int batchSize = 50;
20         int nEpochs = 30;
21
22         int numInputs = 2;
23         int numOutputs = 2;
24         int numHiddenNodes = 20;
25
26         //Load the training data:
27         RecordReader rr = new CSVRecordReader();
28         rr.initialize(new FileSplit(new File("src/main/resources/classification/linear_data_train.csv")));
29         DataSetIterator trainIter = new RecordReaderDataSetIterator(rr, batchSize, 0, 2);
30
31         //Load the test/evaluation data:
32         RecordReader rrTest = new CSVRecordReader();
```

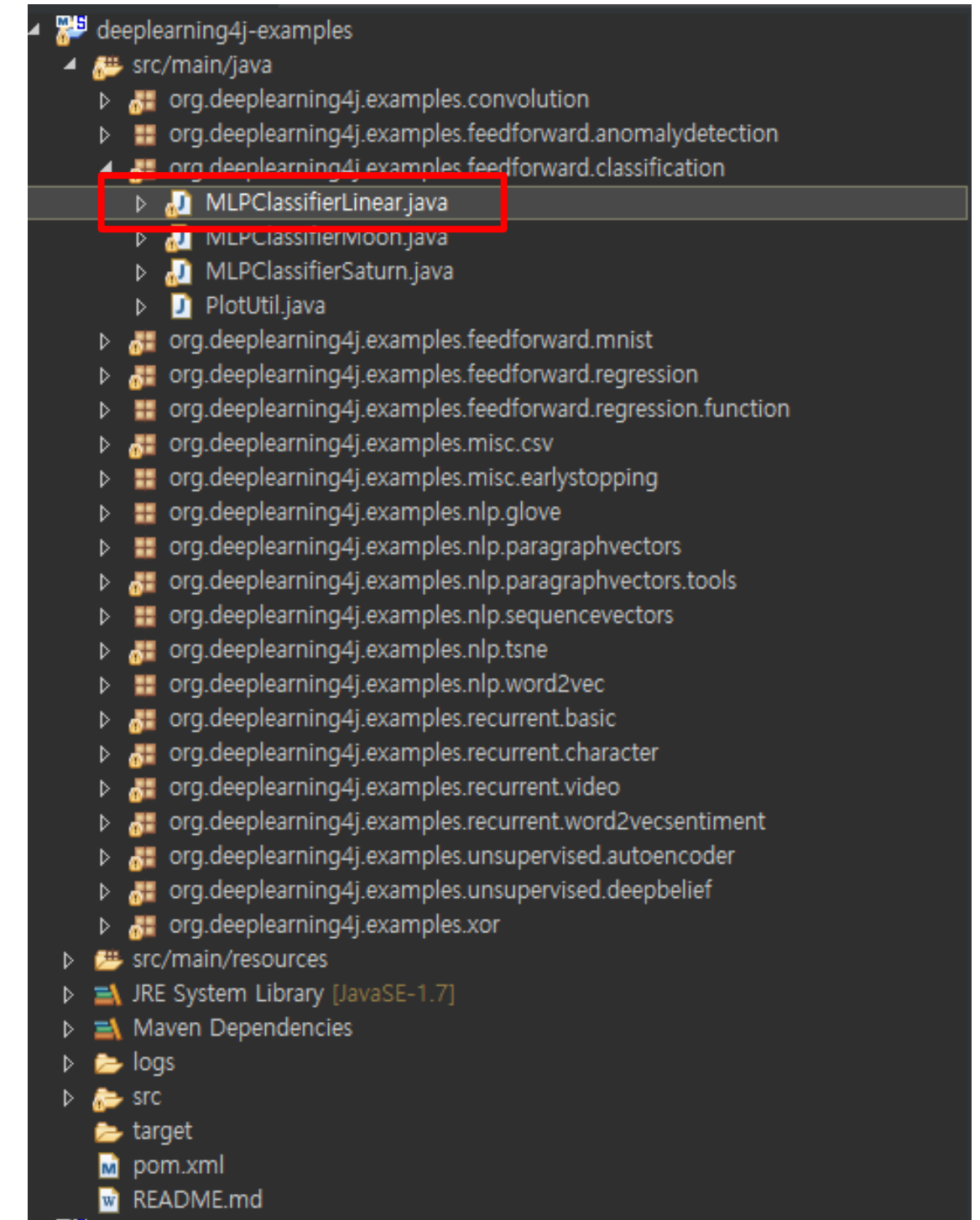
DL4J (Deep Learning for Java)

▶ DL4J Example 코드

- ▶ MLP (Multi Layer Perceptron)
- ▶ FF (Feedforward Network)
- ▶ RNN (Recurrent NN)
- ▶ CNN (Convolution NN)
- ▶ Autoencoder
- ▶ RBM
- ▶ DBN (Deep Belief Network)
- ▶ 등등... 딥러닝 관련 코드 겁나 많죠

▶ 이 중 **MLP, FF** 정도로 간단히만 구현 하셔도 무방합니다.

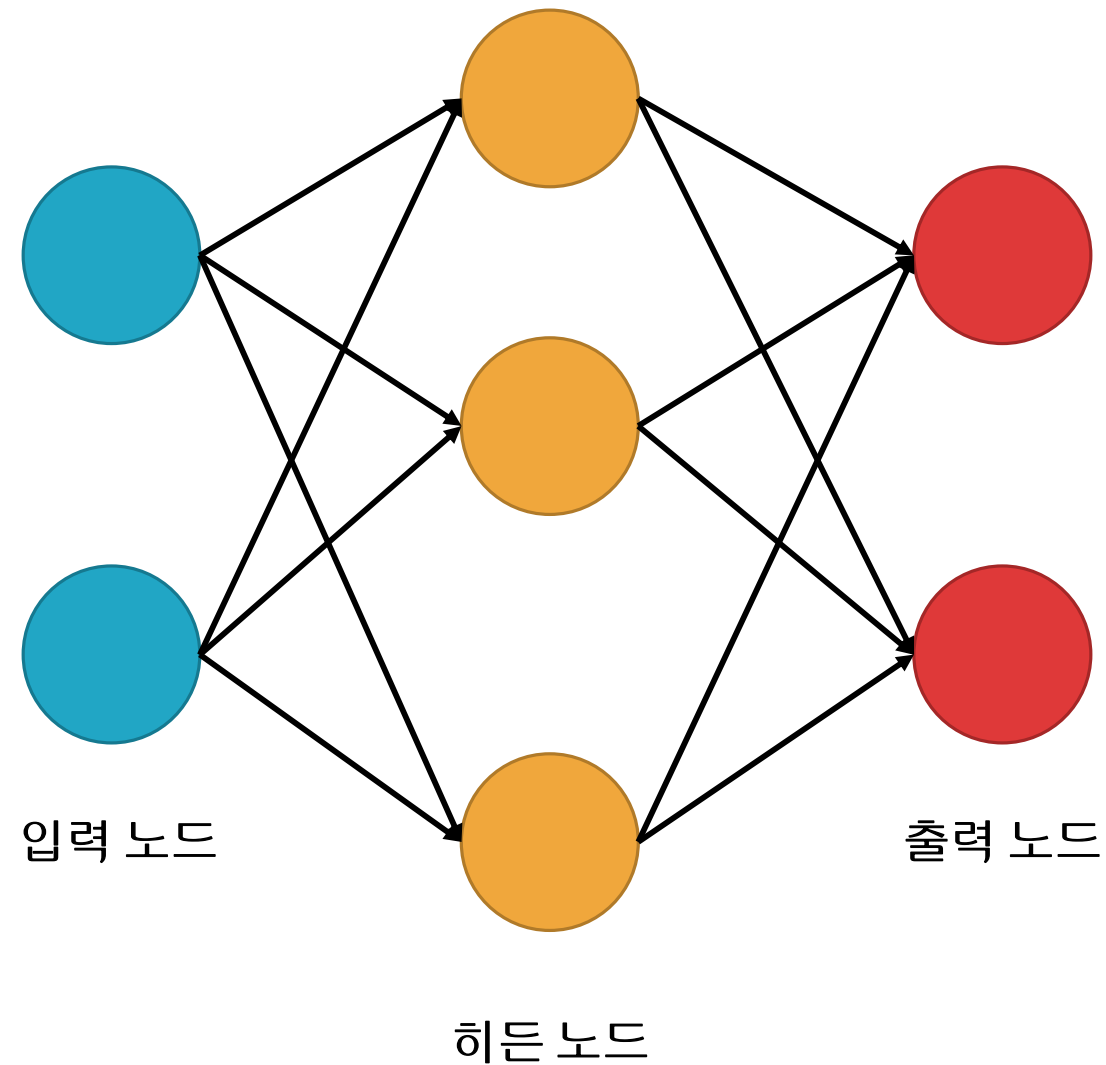
▶ 조금 더 심화된 모델을 구현해보고 싶으신 분들은 **RNN, DBN**을 참조하셔도 됩니다.



- ▶ MLPClassifierLinear.java 파일을 대표 예제로 뜯어서 이해할 것!!!

DL4J (Deep Learning for Java)

▶ DL4J 활용한 신경망 구성 방법(기초) – MLP



- ▶ 입력 노드 : 2개
- ▶ 히든 노드 : 3개
- ▶ 출력 노드 : 2개
- ▶ 학습률 : 0.01
- ▶ 기타 : 학습 최적화 방법을 Stochastic Gradient Descent로 쓰고... 활성화 함수로 Relu를 쓰고... 학습을 30회나 반복하고... BackPropagation 쓰고... 등등

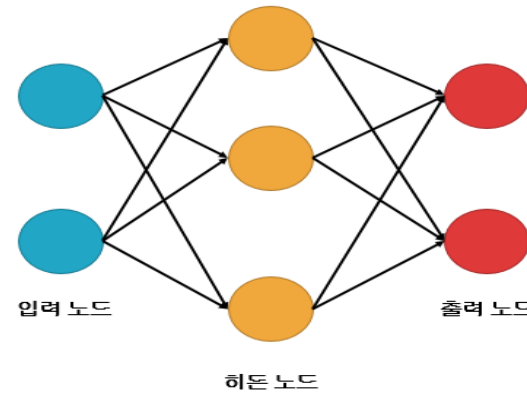
이런 식으로 모델을 구상하였다고 한다면!

DL4J (Deep Learning for Java)

▶ DL4J 활용한 신경망 구성 방법(기초) – MLPClassifierLinear.java

```
int seed = 123;
double learningRate = 0.01;
int batchSize = 50;
int nEpochs = 30;
int numInputs = 2;
int numOutputs = 2;
int numHiddenNodes = 3;

MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(seed)
    .iterations(1)
    .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
    .learningRate(learningRate)
    .updater(Updater.NESTEROVS).momentum(0.9)
    .list(2)
    .layer(0, new DenseLayer.Builder().nIn(numInputs).nOut(numHiddenNodes)
        .weightInit(WeightInit.XAVIER)
        .activation("relu")
        .build())
    .layer(1, new OutputLayer.Builder(LossFunction.NEGATIVELOGLIKELIHOOD)
        .weightInit(WeightInit.XAVIER)
        .activation("softmax").weightInit(WeightInit.XAVIER)
        .nIn(numHiddenNodes).nOut(numOutputs).build())
    .pretrain(false).backprop(true).build();
```



▶ 신경망 구성 파라미터

- ▶ seed : 난수 발생 시드
- ▶ **learningRate** : 가중치 학습률
- ▶ batchSize : data batch size
- ▶ **nEpochs** : 모델 학습 횟수
- ▶ **numInputs** : 입력 노드 수
- ▶ **numOutputs** : 출력 노드 수
- ▶ **numHiddenNodes** : 히든 노드 수

▶ 신경망 구성 함수

- ▶ **layer** : 레이어의 수
- ▶ **optimizationAlgo** : 가중치 학습 최적화 방법
- ▶ updater : 가중치 수정 방법
- ▶ weightInit : 가중치 초기화 방법
- ▶ **activation** : 활성 함수 설정
 - ▶ tanh, sigmoid, Relu, Softmax, softplus 등
- ▶ pretrain, backprop

▶ 빨간색 파라미터, 함수가 신경망 구성의 중요한 요소

DL4J (Deep Learning for Java)

▶ DL4J 활용한 신경망 구성 방법(기초) – MLPClassifierLinear.java

- ▶ 구성 완료한 인공 신경망 conf 를 이제 model로 구축해주고 초기화 해줌

```
MultiLayerNetwork model = new MultiLayerNetwork(conf);
model.init();
model.setListeners(Collections.singletonList((IterationListener) new ScoreIterationListener(1)));
```

- ▶ nEpoch 회전 수 만큼 model을 fitting해줌 (=Train), 너무 많이 돌리면 Overfitting 발생

```
for ( int n = 0; n < nEpochs; n++) {
    model.fit( trainIter );
}
```

- ▶ 분류 및 예측 결과치를 출력하는 부분

```
System.out.println("Evaluate model....");
Evaluation eval = new Evaluation(numOutputs);
while(testIter.hasNext()){
    DataSet t = testIter.next();
    INDArray features = t.getFeatureMatrix();
    INDArray labels = t.getLabels();
    INDArray predicted = model.output(features, false);
    eval.eval(labels, predicted);
}

//Print the evaluation statistics
System.out.println(eval.stats());
```

Evaluate model....

Examples labeled as 0 classified by model as 0: 100 times
Examples labeled as 1 classified by model as 0: 1 times
Examples labeled as 1 classified by model as 1: 99 times

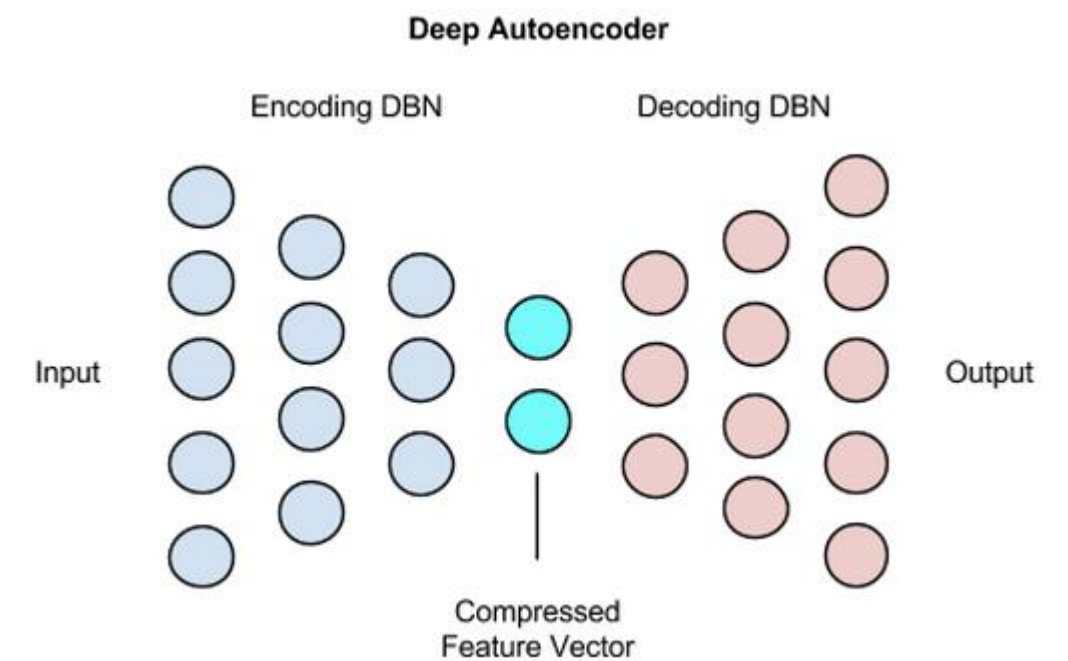
```
=====Scores=====+=====
Accuracy:  0.995
Precision: 0.995
Recall:    0.995
F1 Score:  0.995
=====
*****
```

DL4J (Deep Learning for Java)

▶ DL4J 활용한 신경망 구성 방법(심화 예제)

▶ Deep Belief Network (DBN) 예제

```
logInfo("Building model...");
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(seed)
    .iterations(iterations)
    .optimizationAlgorithm(OptimizationAlgorithm.LINE_GRADIENT_DESCENT)
    .list(10)
    .layer(0, new RBM.Builder().nIn(numRows * numColumns).nOut(1000).lossFunction(LossFunctions.LossFunction.RMSE_XENT).build())
    .layer(1, new RBM.Builder().nIn(1000).nOut(500).lossFunction(LossFunctions.LossFunction.RMSE_XENT).build())
    .layer(2, new RBM.Builder().nIn(500).nOut(250).lossFunction(LossFunctions.LossFunction.RMSE_XENT).build())
    .layer(3, new RBM.Builder().nIn(250).nOut(100).lossFunction(LossFunctions.LossFunction.RMSE_XENT).build())
    .layer(4, new RBM.Builder().nIn(100).nOut(30).lossFunction(LossFunctions.LossFunction.RMSE_XENT).build()) //encoding stops
    .layer(5, new RBM.Builder().nIn(30).nOut(100).lossFunction(LossFunctions.LossFunction.RMSE_XENT).build()) //decoding starts
    .layer(6, new RBM.Builder().nIn(100).nOut(250).lossFunction(LossFunctions.LossFunction.RMSE_XENT).build())
    .layer(7, new RBM.Builder().nIn(250).nOut(500).lossFunction(LossFunctions.LossFunction.RMSE_XENT).build())
    .layer(8, new RBM.Builder().nIn(500).nOut(1000).lossFunction(LossFunctions.LossFunction.RMSE_XENT).build())
    .layer(9, new OutputLayer.Builder(LossFunctions.LossFunction.RMSE_XENT).nIn(1000).nOut(numRows*numColumns).build())
    .pretrain(true).backprop(true)
    .build();
```



- ▶ 그림으로는 피곤해서 못그렸는데,
- ▶ 레이어가 10개, 첫번째 레이어부터 입력이 이제 몇천개부터 시작해서 노드 수가 몇천개 - 1000 - 500 - 250 - 100 - 30 - 100 - 250 - 500 - 1000 - 몇천개 로 구성된 신경망

Epinions Movie Dataset - 학습

- ▶ 영화 데이터 셋
 - ▶ 사용자/영화 : 10,000명/10,000개

User ID	User Name	Movie ID	Movie Name	Rating	Rating Date
1	petra	65	Hannibal	3	Aug/13/`01
2	deaser26	1	Matrix	5	May/25/`01

⋮

- ▶ 학습(Training) : 40,000개의 사용자-영화 평점 데이터
 - ▶ User ID 기준의 정렬이 아닌 랜덤하게 정렬된 상태
 - ▶ **"EpinionDataset_for_Training.csv"** 파일 참조

Epinions Movie Dataset - 평가

▶ 영화 데이터 셋

▶ 사용자/영화 : 10,000명/10,000개

정답
맞추기!

User ID	User Name	Movie ID	Movie Name	Rating	Rating Date
1	petra	8572	Ailien	?	Jun/06/`04
2356	garethvk	235	Spider-man 2	?	Jun/29/`04

⋮

▶ 평가(Testing) : 10,000개의 사용자-영화 평점 데이터

▶ Rating 필드가 공란으로 되어있음

▶ 역시, 랜덤하게 정렬된 상태

▶ **"EpinionDataset_for_Testing.csv"** 파일 참조

Epinions Movie Dataset - 추가 정보

- ▶ 영화 메타데이터 셋
 - ▶ 영화 10,000개에 대한 메타데이터 정보

MovieID	Movie Name	Genre	Subgenre	Date	Lang	Director	Actors	Stars
1	Matrix	Science-Fiction/ Fantasy	Science-Fiction	1999	English	Larry Wachowski	Anthony Ray Parker	Keanu Reeves
2	Sixth Sense	Horror/ Suspense	Suspense	1999	English	M. Night Shyamalan	Toni Collette	Bruce Willis

⋮

- ▶ 협력 추천이나 인공지능망의 Input 혹은 Weight 설정 등에 활용될 수 있음
- ▶ **“EpinionDataset_movie_metadata.csv”** 파일 참조

쉽게 요약하면

40,000개의
Training Data

User ID	User Name	Movie ID	Movie Name	Rating	Rating Date
1	petra	65	Hannibal	3	Aug/13/`01
2	deaser26	1	Matrix	5	May/25/`01

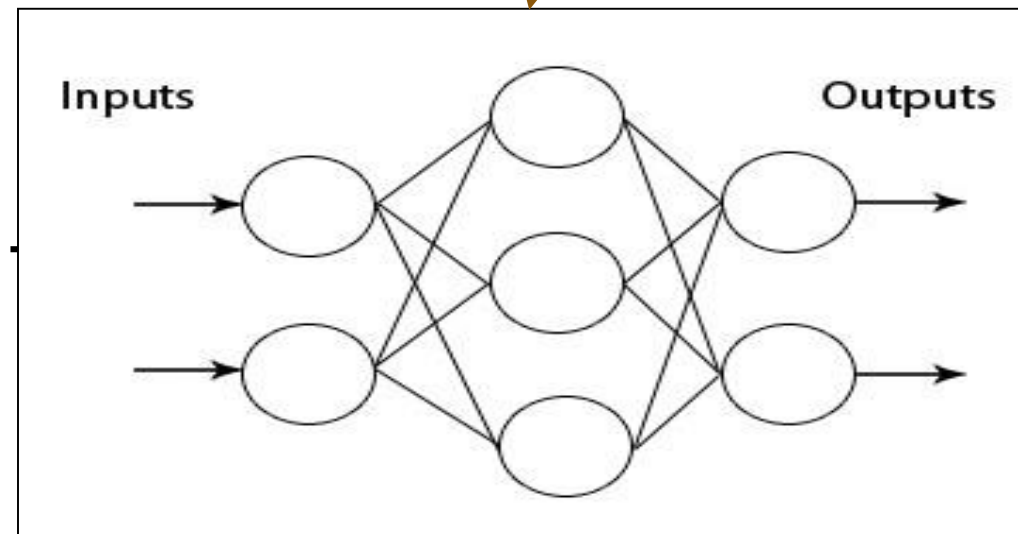
⋮

학습(Training)

<학습 과정>

<예측 과정>

인공 신경망
모델



10,000개의
Testing Data

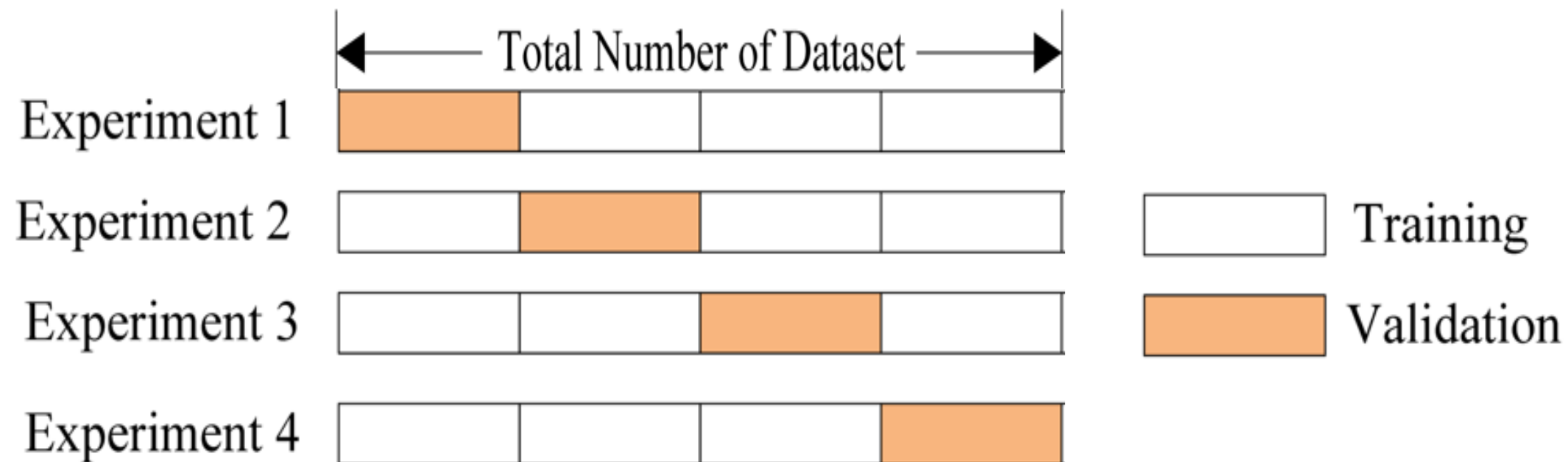
User ID	User Name	Movie ID	Movie Name	Rating	Rating Date
1	petra	8572	Ailien	?	Jun/06/`04
2356	garethvk	235	Spider-man 2	?	Jun/29/`04

⋮

정답
맞추기!

모델의 정확도 평가 방법은 어떻게?

- ▶ k-fold cross-validation (if, $k=4$)
 - ▶ 사용자-영화 평점 40,000개의 Training Dataset에서
 - ▶ 30,000개의 Training을 학습
 - ▶ 10,000개의 Validation Data로 활용하여 4번의 사전평가 진행



- ▶ 데이터가 동일한 특성을 가지므로 효율적인 검증 방법



프로젝트 구현

- ▶ 구글링해보시면 관련 논문 및 코드 많이 나옵니다.
- ▶ 제공된 40,000개의 Training Dataset을 활용하여 각자의 신경망 모델 학습
- ▶ 신경망 모델을 학습한 후,
 - ▶ 모델 평가 : 10,000개의 사용자-영화 평점(1~5점, 소수점 가능) 예측
 - ▶ 반드시 “**EpinionDataset_for_Testing_학번.csv**” 파일로 출력
 - ▶ 이 파일 생성해서 안찍어주시면 감점하겠습니다.



문서화

- ▶ 내용
 - ▶ 각자 구상한 평점 예측 방식에 대하여 상세히 설명
 - ▶ 사전 성능 평가 결과 (있어도 되고 없어도 됩니다)
- ▶ 형식 : 자유롭게 .doc나 .hwp로 작성
- ▶ 분량 : 1~3 page 내외

평가 방법

- ▶ “**EpinionDataset_for_Testing_학번.csv**” 파일에 출력한 예측 평점값과 실제 평점값을 비교
 - ▶ 실제 평점값은 오로지 저만 가지고 있습니다!
 - ▶ 대략 Precision 70(%)만 나와도 1등이라고 예상 중입니다!
- ▶ RMSE(Root Mean Squared Error) 계산 후, Precision 측정

- ▶
$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^N (r_i - r_i^*)^2}{N}}$$

- ▶ r_i : 예측된 Rating 값
- ▶ r_i^* : 실제 Rating 값

- ▶
$$\text{Precision} = 1 - \frac{\text{RMSE}}{\text{MaxError}}$$
 얼마나 정답과 가까운지에 대한 정확도 개념!

- ▶ (MaxError는 가능한 오차 범위 최대치. Rating이 1~5점 사이므로 MaxError는 4)

유의 사항 (중요!)

- ▶ 반드시 평점 예측 결과가 **“EpinionDataset_for_Testing_학번.csv”** 파일에 출력되도록 하여야 합니다.
- ▶ 모델링한 방식에 따라 몇몇 영화의 평점 예측이 불가능할 경우, 예외처리 방식 각자 구상하셔야 합니다.
 - ▶ 학습한 적 없는 영화의 평점을 예측해야되는 경우 등
- ▶ Training Dataset 내에서 다양한 샘플링 방법을 통해 유의미한 데이터를 추출하시거나 복제하시는 거 허용됩니다.

유의 사항

- ▶ 영화의 실제 평점은 1~5점 사이의 정수이지만, 예측한 평점은 1~5점 사이의 소수 점수 표현 가능합니다.
 - ▶ 예) Matrix : 4.5점, Sixth Sense : 3.25점
- ▶ 만일, 사전 성능 평가(using k-fold cross-validation)로 Precision을 측정해보고 싶으신 분은 본인이 직접 Precision 구하는 함수를 만드셔야 합니다.
 - ▶ DL4J에서 출력해주는 Precision은 정답이냐 아니냐의 Precision이므로 엄청 낮게 나옵니다!!



평가 방법

- ▶ 예측 정확도(70%)
 - ▶ 예측 정확도(Precision)순으로 나열 후, 차등 점수 부여
 - ▶ 1~5등(5명) : A
 - ▶ 6~20등(15명) : B
 - ▶ 21~40등(20명) : C
 - ▶ 41~55등(15명) : D
 - ▶ 56~62등(7명) : E
- ▶ 인공지능경망 구성의 타당성 및 창의성(10%)
- ▶ 오픈 소스 활용성(10%)
- ▶ 문서화 점수(10%)

프로젝트 제출

▶ 제출 기한

▶ GitLab을 통해 제출

▶ GitLab 제출이 잘 안될 경우, 사전에 공지하도록 하겠습니다.

▶ 6/11(일)까지, 제출기한 엄수! 추가제출 없습니다!!

▶ 프로젝트 폴더에 업로드해야 될 파일

1) 프로젝트 파일(소스코드 포함)

2) 문서(.doc or .hwp)

3) 예측 평점 결과 파일

(**"EpinionDataset_for_Testing_학번.csv"**)

+) 외부 라이브러리 쓰신 분은 함께 첨부

Thank You

