Data Mining Lab.
University Of Seoul

2016. 10. 14.
데이터야놀자
MARU180

Spark & Zeppelin을 활용한

머신러닝 실전 적용기

김태준(Jun Kim)
i2r.jun@gmail.com

안녕하세요, **김태준** 입니다!

- 서울시립대 데이터마이닝 연구실 소속
- “화재 예측을 위한 빅데이터 분석” 프로젝트 참여
- Spark와 Zeppelin을
  - Data Warehouse 구축
  - 데이터 분석
  - **머신러닝**
  - 논문 작성

  등에 사용하고 있습니다~

오늘의 주제는 ML!

발표를 준비한 제 마음가짐

최대한 삽질 덜하고 빠른 방법으로 Spark, 머신러닝을 익히셨으면!

실제로 Spark & Zeppelin으로 머신러닝할때 발생하는 문제점과 해결책 공유

데이터 소리에 대한 감을 잡으시길 바라며

기타 등등 꿀팁 공유

오늘의 발표는?

- Spark와 Zeppelin을 활용한
- 엔지니어를 위한
- 텍스트 분류 예를 통해보는
- 논문 작성 경험에 근거한
- 삽질 경험에 근거한
- 진짜로 데이터 소리를 들어보는(?)
- 조금 현실적이고 실질적인(?)

머신러닝 실전 적용기

데이터셋
Zeppelin 노트북

발표 마지막에

모두 공개!

GitHub에 올렸습니당

때는 바야흐로 1년전...

Web과 IoT 개발을 하던 중생의

Spark와의 첫만남

빅데이터라니... 넘 멋정

IT 기술 학습 Process

설치

Hello, World!

예제 돌리기

Guide 문서 정독

Spark에 적용해보자!

우당타다다다닫다앙

Ambari 설치...OK!

설치

Hello, World!

예제 돌리기

Guide 문서 정독

설치

콰광쾅콰과과과과광

Hello, World!

spark-shell...OK!

예제 돌리기

Guide 문서 정독

설치

Hello, World!

예제 돌리기

Guide 문서 정독

LogisticRegression 예제

```scala
import org.apache.spark.ml.classification.LogisticRegression

// Load training data
val training = spark.read.format("libsvm").load("data/mllib/sample_libsvm_data.txt")

val lr = new LogisticRegression()
  .setMaxIter(10)
  .setRegParam(0.3)
  .setElasticNetParam(0.8)

// Fit the model
val lrModel = lr.fit(training)

// Print the coefficients and intercept for logistic regression
println(s"Coefficients: ${lrModel.coefficients} Intercept: ${lrModel.intercept}")
```

??????

실행도되고

Scala 몰라도 다 이해가는 문법인데?

지금 무슨 데이터로 뭘 출력한거니???

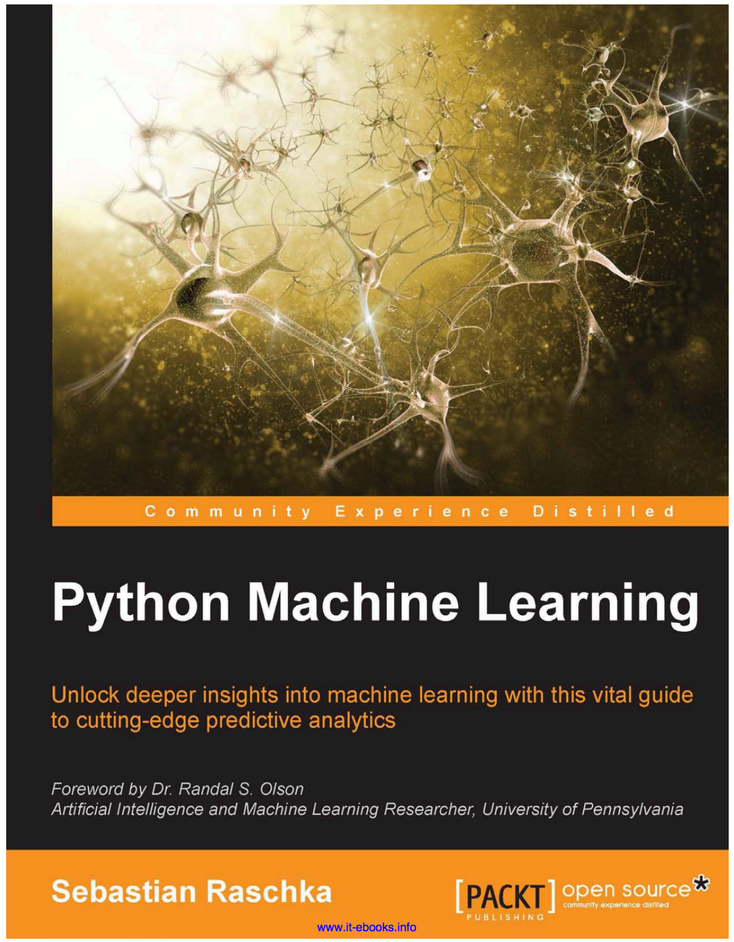
java.lang.NullPointerException

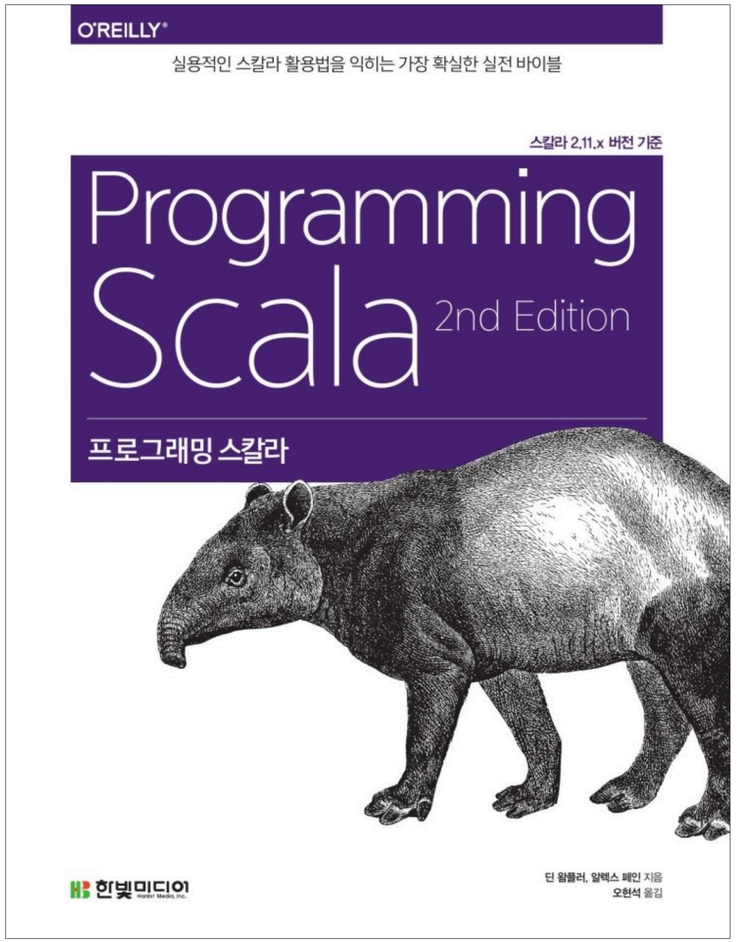
Spark ML, 왜 어려울까?

- 이전에 IT 기술에 **접근**하던 **방법**과 다르게 접근해야함 DT(Data Technology)니까!
- 머신러닝 **이론 부족**
- Spark **문서 부실** 머신러닝을 Spark로 입문하기엔 정말 어려움
- ML 과정을 실제 데이터를 사용하여 처음부터 끝까지 다룬 **예제**가 **없음**

그래서 제가 데이터셋과 노트북을 공유합니다 ㅎ ㅎ

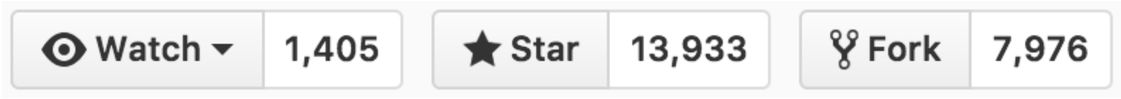
책추천


Community Experience Distilled
Python Machine Learning
Unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics
Foreword by Dr. Randal S. Olson
Artificial Intelligence and Machine Learning Researcher, University of Pennsylvania
Sebastian Raschka
[PACKT] open source
PUBLISHING
www.it-ebooks.info


O'REILLY
실용적인 스칼라 활용법을 익히는 가장 확실한 실전 바이블
스칼라 2.11.x 버전 기준
Programming Scala 2nd Edition
프로그래밍 스칼라
한빛미디어
딘 왐플러, 알렉스 페인 지음
오현석 옮김

Python 책 추천 이유

- **scikit-learn(since 2007)**
  - Python 머신러닝 모듈
  - 기능이 Spark에 비해 많아 ML 공부에 좋다



Spark는 별이 만개 정도

- Spark의 ML Pipeline 개념은 scikit-learn에서 가져온것

Main concepts in Pipelines Spark ML 문서 중

MLlib standardizes APIs for machine learning algorithms to make it easier to combine multiple algorithms into a single pipeline, or workflow. This section covers the key concepts introduced by the Pipelines API, where the pipeline concept is mostly inspired by the scikit-learn project.

- **DataFrame**: Thi ata types. E.g., a DataFrame cou
- **Transformer**: A `Transformer` is an algorithm which can transform one `DataFrame` into another `DataFrame`. E.g., an ML model is a `Transformer` which transforms a `DataFrame` with features into a `DataFrame` with predictions.
- **Estimator**: An `Estimator` is an algorithm which can be fit on a `DataFrame` to produce a `Transformer`. E.g., a learning algorithm is an `Estimator` which trains on a `DataFrame` and produces a model.
- **Pipeline**: A `Pipeline` chains multiple `Transformer`s and `Estimator`s together to specify an ML workflow.

inspired by the scikit-learn project.

처음에는 머신러닝 알고리즘이 정말 많아보이지만,
핵심적인 몇몇 알고리즘들을 공부하고나면 거기서 거기

더 중요한 것은 따로 있다!

데이터의 소리 듣는 법!

데이터 소리...?

있어 보이는 말이긴 한데...

- 데이터는 **질문**을 해야 답한다 무슨 질문을 해야되지?
- 숫자로 답해줘서 정확히 **이해**하기 **힘들다** 내가 제대로 이해한게 맞나?

Text Classification 예제를 통한

데이터 소리 듣는 법

커밍쑨!

원래 하려던 일...

Original Mission

화재 뉴스 기사 분석

다음 뉴스에 “화재”를 검색하여

뉴스 기사 수집!

Problem

수집된 뉴스 기사

14만건

그런데,

화재 뉴스 기사가 아닌
뉴스 기사들 또한 존재

화재 뉴스 기사만을 걸러내야함!

필터링~

Solution

뉴스 기사들 중

일부는 수작업으로 분류하고

나머지는

노가다...

Spark ML에게 맡기자!

ML Mission

뉴스 기사들을

화재 / 비화재

두 가지로 분류하라!

즉, Binary Classification

오늘의 데이터셋 소개! 핸드메이드

화재 뉴스 기사 데이터

Schema

칼럼	설명
label	화재 뉴스 기사 = 1, 아니면 0
title	뉴스 기사 제목
body	뉴스 기사 본문
date	날짜
host	언론
link	뉴스 기사 링크
query	다음 뉴스 검색어

오늘의 데이터셋 소개! 핸드메이드

화재 뉴스 기사 데이터

데이터 분포

	Training	Test	총계	비율
화재	380	158	538	41%
비화재	534	254	788	59%
총계	914	412	1326	
비율	69%	31%		

오늘의 데이터셋 소개! 핸드메이드

화재 뉴스 기사 데이터

데이터 분포

	Training	Test	총계	비율
화재	380	158	538	41%
비화재	534	254	788	59%
총계	914	412	1326	
비율	69%	31%		

오늘의 데이터셋 소개! 핸드메이드

화재 뉴스 기사 데이터

데이터 분포

	Training	Test	총계	비율
화재	380	158	538	41%
비화재	534	254	788	59%
총계	914	412	1326	
비율	69%	31%		

오늘의 데이터셋 소개! 핸드메이드

화재 뉴스 기사 데이터

데이터 분포

	Training	Test	총계	비율
화재	380	158	538	41%
비화재	534	254	788	59%
총계	914	412	1326	
비율	69%	31%		

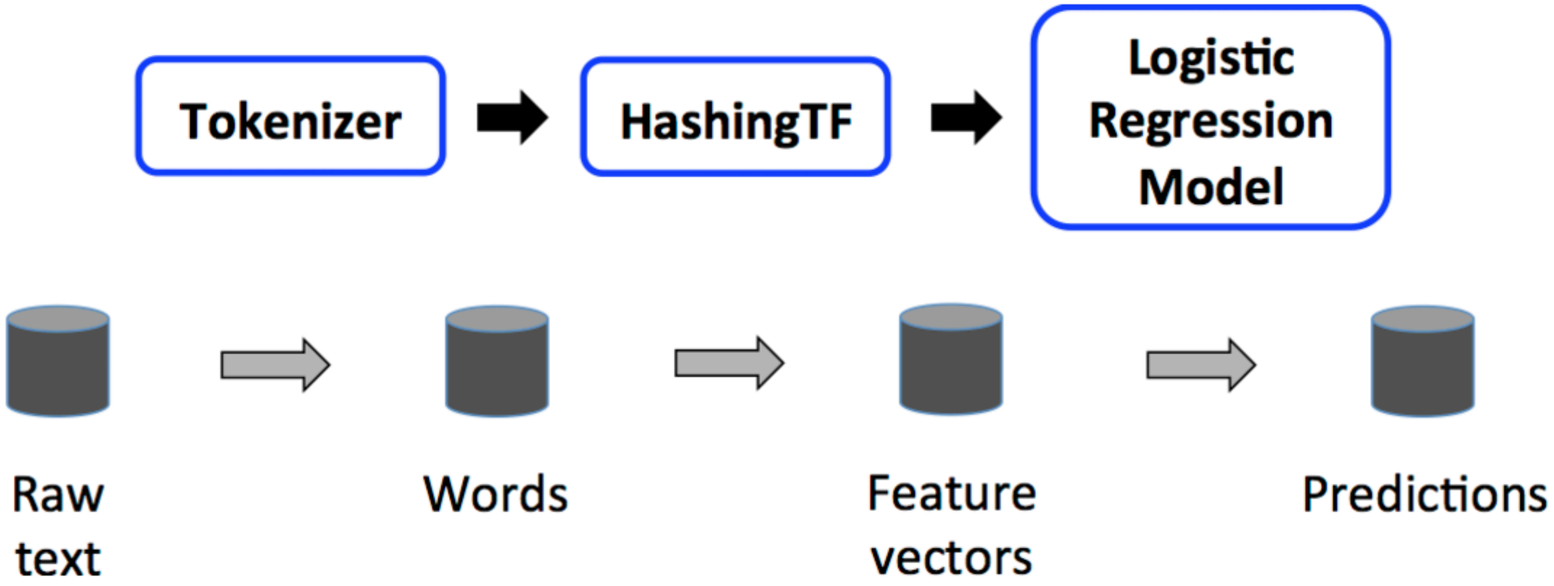
7:3

Text Classification,
어떤 과정이 필요할까?

ML Pipeline에 대한 고찰

Spark 문서에 따른

Text Classification Pipeline



텍스트를 단어로 쪼갬

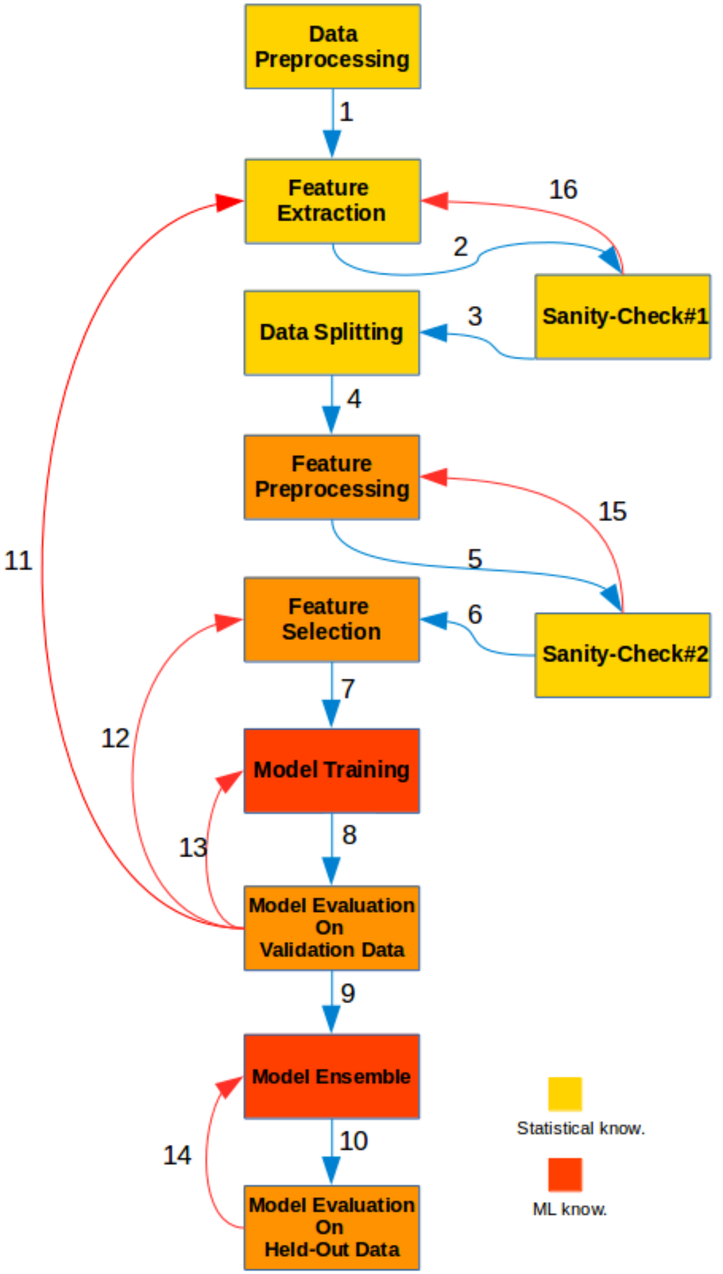
단어를 벡터로 변환

학습, 예측

껍이네~

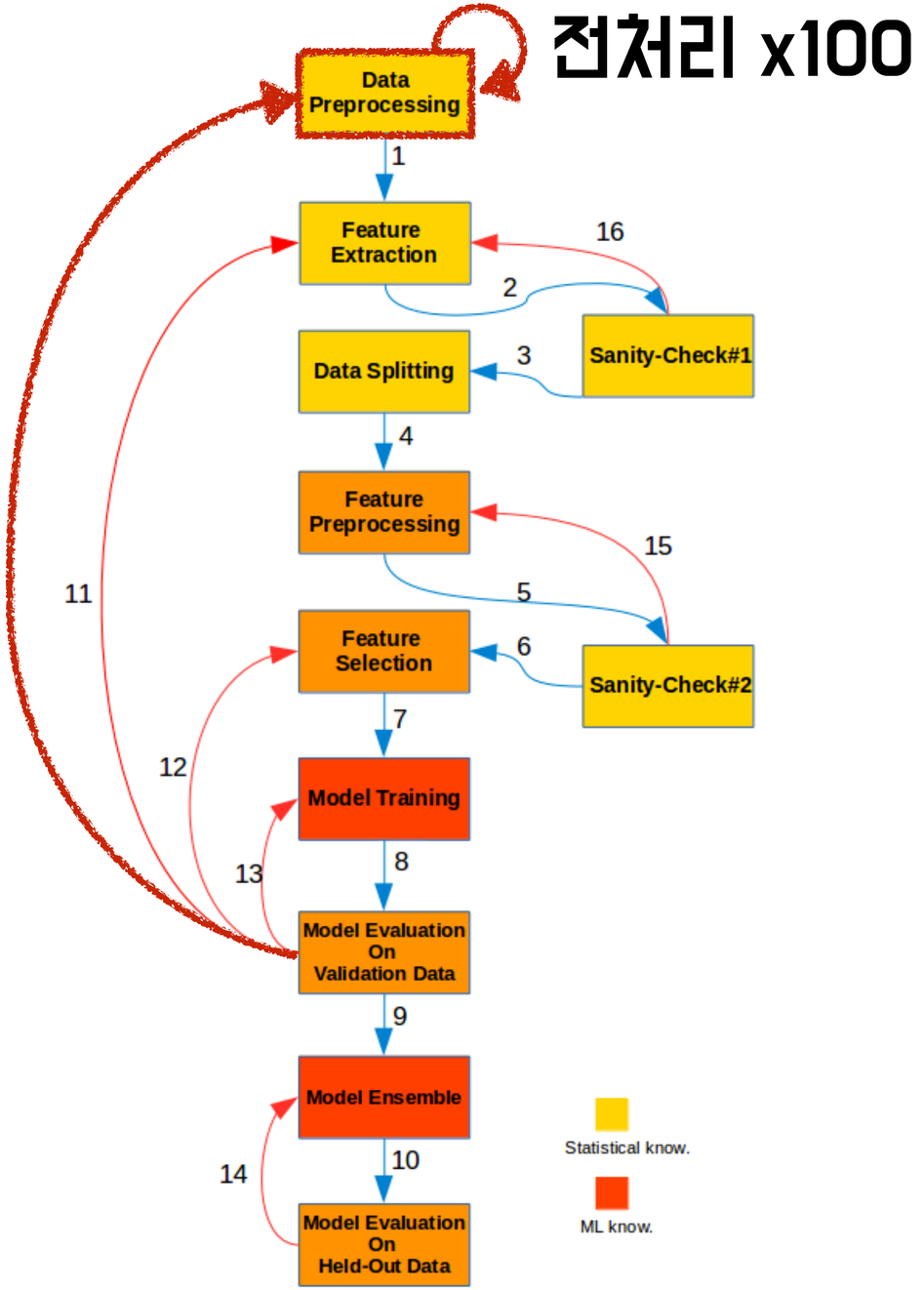
ML Pipeline

현실 세계



ML Pipeline

진짜 현실 세계



ML Pipeline의 현실

- 머신러닝은 하루 이틀에 끝나지 않는다
- **Stage**는 **순서**대로 **진행**되지 **않는다** 인간은 실수를 저지르기 마련...
- 전처리 코드는 정말 길고, 학습 코드는 매우 짧다
- 코드가 길어져 여러 파일에 나누어 저장할 필요가 생김 즉, 분석 코드의 모듈화
- 하지만, **분석 코드**를 **파일**로 **저장**하는 것은 굉장히 **비효율적**
  - **결과**를 다시 보려면 **재실행**하거나, 주석으로 결과를 저장해야함
  - 팀원간의 분석 코드, 결과 **공유** 또한 **힘듬**
  - 또한 **무한 ctrl+{c, v}**는 우리의 삶을 힘들게 한다
  - 실수로 터미널 끄면 다 날라감

이 모든 문제점들을 해결해주는

ML을 위한 Zeppelin

이외에도 장점이 매우 많지만, ML에서의 장점들 입니다

- 깔끔한 **코드 정리**
  - Stage 별로 폴더 생성
  - 한 stage에 여러 노트 생성

긴 전처리 코드를 나누어 저장하면 좋겠죠?

- **시각화**
- 분석 코드와 **결과**가 같이 **저장**됨

결과 보려고 다시 돌리는 짓 안해도됨...

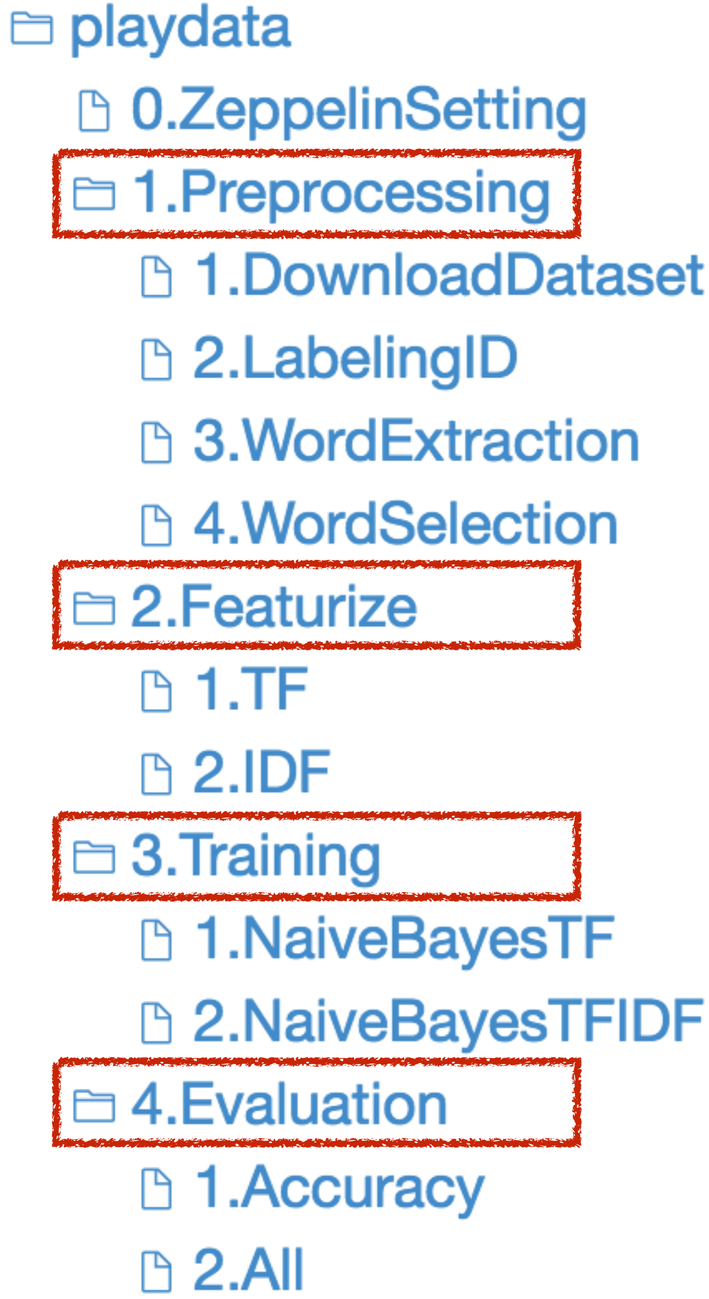
- 팀원에게 링크만 쏴주면 **공유** 끝
- **Markdown**으로 노트 중간중간 **설명** 작성 가능

- playdata
  - 0.ZeppelinSetting
  - 1.Preprocessing
    - 1.DownloadDataset
    - 2.LabelingID
    - 3.WordExtraction
    - 4.WordSelection
  - 2.Featurize
    - 1.TF
    - 2.IDF
  - 3.Training
    - 1.NaiveBayesTF
    - 2.NaiveBayesTFIDF
  - 4.Evaluation
    - 1.Accuracy
    - 2.All

제가 추천하는

Zeppelin ML 노트북 구조

여러분께 드릴 노트북 구조입니당



Stage 별로 폴더 생성!

다음 stage에게 output 전달

- playdata
  - 0.ZeppelinSetting
  - 1.Preprocessing
    - 1.DownloadDataset
    - 2.LabelingID
    - 3.WordExtraction
    - 4.WordSelection
  - 2.Featurize
    - 1.TF
    - 2.IDF
  - 3.Training
    - 1.NaiveBayesTF
    - 2.NaiveBayesTFIDF
  - 4.Evaluation
    - 1.Accuracy
    - 2.All

Stage 내에서도 다음
노트에게 output 전달

하지만,

노트 간 변수 공유는 죄악입니다...!

전역변수 쓰지마!

노트 간 변수 공유의 문제점

- 노트 간 **종속성(dependency)** 발생

  즉, 어떤 노트를 실행하려면 이전에 다른 노트를 실행해야함

- **알 수 없는 변수**의 출처

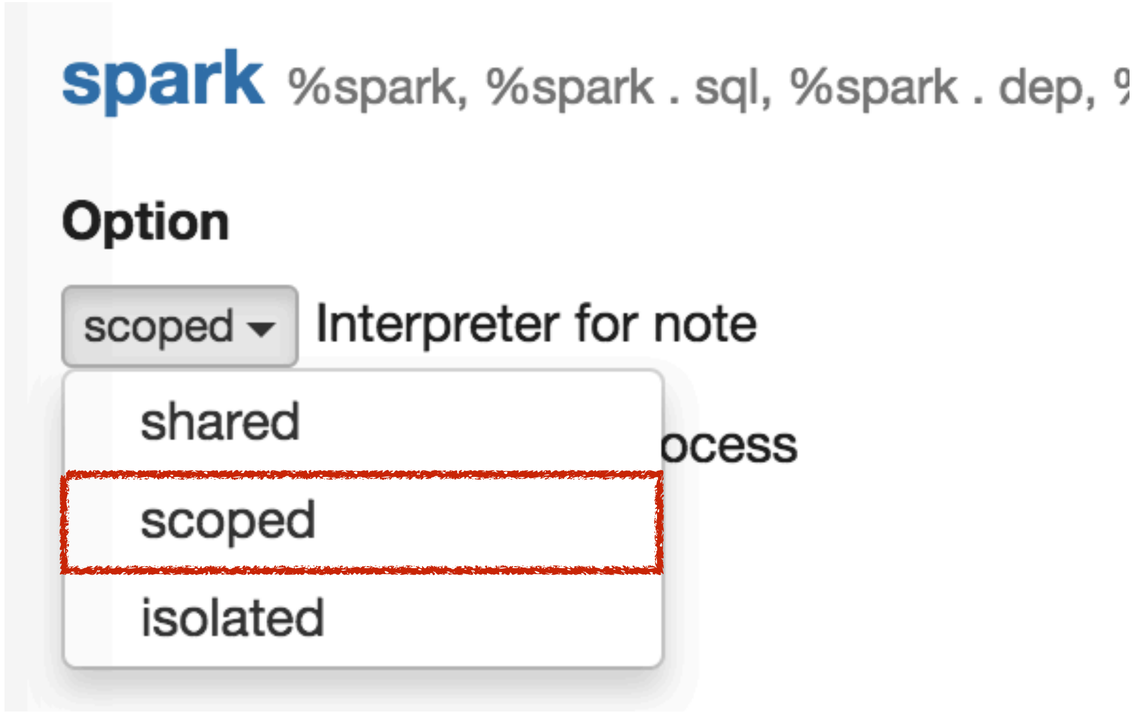
  모르는 변수가 갑자기 튀어나옴...;

- 다른 노트와 변수 **이름 중복**

  제 경험... 디버깅 한참함...

Solution! Zeppelin의 똑똑한 기능

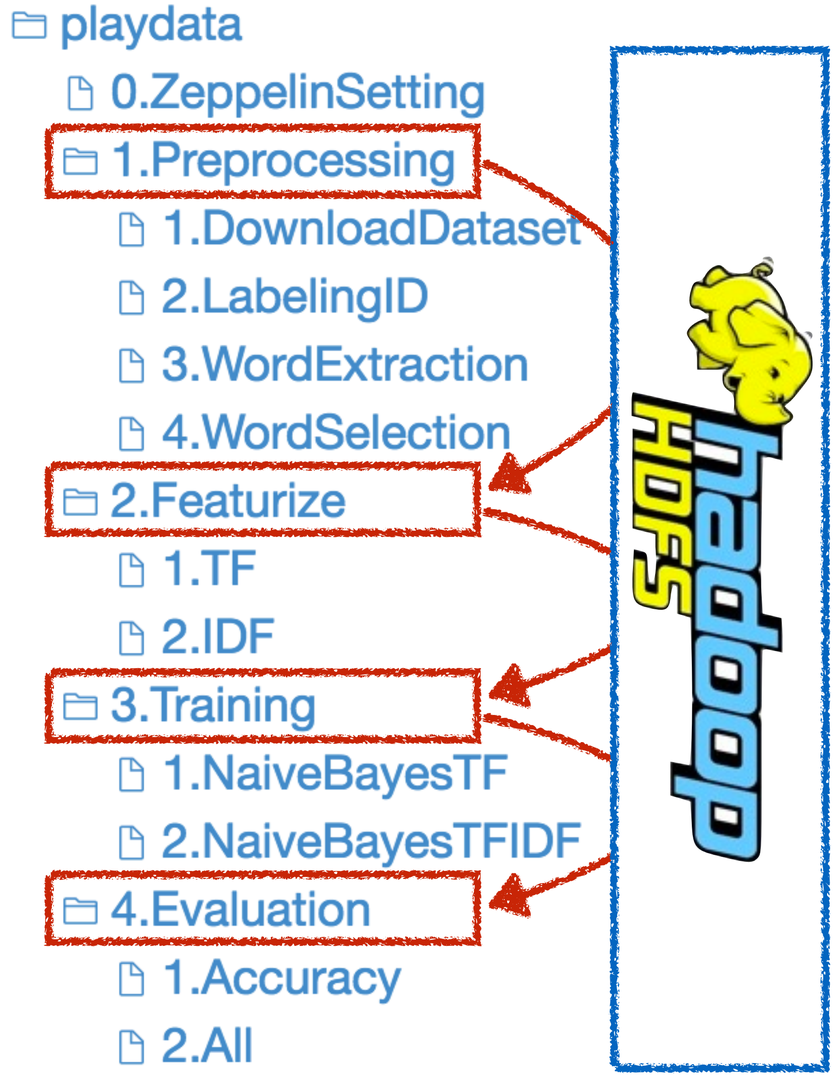
변수 공유 끄기



Zeppelin > interpreter 설정 > Spark interpreter

"scoped"

FS(File System)를 통한 데이터 공유



Stage Output 저장 구조

- data
  - news-with-id.parquet
  - news.parquet
  - words-nv.parquet
  - words.parquet
- feature
  - tf-idf.parquet
  - tf.parquet
- model
  - cntVec
  - idf
  - nb-tf
  - nb-tf-idf

정돈된 삶

- 변수 공유 문제점 **모두 해결**
- 모든 **노트 구조 통일**: **input > processing > output**
  - 노트의 input, output을 한눈에
- **모든 과정**을 **기록**하기 때문에, 후에 어떤 stage에 **문제**가 있었는지 **검토 가능**

화재 뉴스 기사
ML Pipeline

- playdata
  - 0.ZeppelinSetting — 0. Spark, 형태소분석기 동작 확인
  - 1.Preprocessing — 1. 데이터 다운로드 및 전처리: 단어 추출
  - 2.Featurize — 2. 단어 > 벡터
  - 3.Training — 3. 벡터 > 학습 > 모델
  - 4.Evaluation — 4. 모델 평가

0.ZeppelinSetting

형태소 분석기

"seunjeon" 추가!

Dependencies

artifact	exclude
org.bitbucket.eunjeon:seunjeon_2.11:1.1.0	org.scala-lang:scala-library,org.scala-lang:scala-reflect

Zeppelin > Interpreter 설정 > Spark interpreter
dependency에 위 그림과 같이 추가

GitHub에도 README.md 파일에 정리해놨습니당

1. Pre-processing(전처리)

데이터를 머신러닝 알고리즘이 학습할 수 있는 형태로 변환하는 과정

1. 데이터 **다운로드**, 포맷 변환

2. 각 뉴스 기사에게 **식별자** 부여

3. 형태소 분석, **단어 추출**

4. 단어 **선택**

1-1. DownloadDataset

GitHub에 있습니다

- JSON 데이터셋을 다운받아 Parquet로 저장

- Parquet:
  - Hadoop 생태계의 어떤 프로젝트에서든 사용가능한 columnar 저장 형식
  - Raw data format과 비교했을때 속도 차이가 매우 큼

그냥 Spark DataFrame 저장 형식이라 생각하셔도...?

1-2. LabelingID

- 각각의 뉴스 기사들을 식별하기 위해 unique ID 생성

계속 진행하기 전에!

생각해봅시다

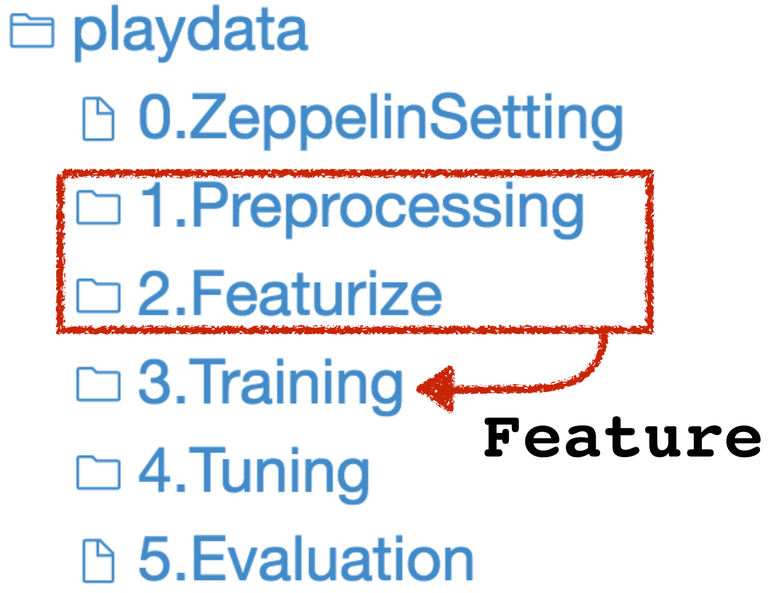
목적을 갖고 전처리해야죠!

데이터를 어떤 형태로 머신러닝 알고리즘에게 전달해야할까?

즉, 어떤 feature를 전달해야할까?

Feature

- 각 사건을 **정량적**으로 측정한 **속성**(사건을 수치로 나타냄)
- 각 사건을 잘 **설명**하고, 다른 사건들과 **구분**짓는 속성
- 각 뉴스 기사의 특성을 잘 나타내어 **주제**가 **화재**인지, **아닌지** 분류할 수 있는 feature를 전달해야함!



여러분은 문서의 **주제**를 어떻게 **분류**하시나요?

문서의 **중요**
명사나
동사로
판단할 수 있을까요?

예

주제	단어
정치	국감, 탈북
경제	금리, 전망
IT	스마트폰, 폭발

화재 뉴스 기사 Feature

II

뉴스 기사의 중요 명사, 동사

1-3. WordExtraction

- 형태소 분석기를 이용해 각 문서의 모든 단어들을 추출

1-4. WordSelection

- 단어들 중 문서를 가장 잘 대표할 수 있는 체언(명사), 용언(동사)만을 선택
- Feature의 길이(차원)가 너무 길어지면, 차원의 저주(Curse of Dimensionality) 발생

말이 너무 많으면 오히려 주제 파악하기 힘들잖아요?
핵심만 말합시다

2. Featurize

데이터를
기계학습 알고리즘이
이해할 수 있는
숫자로 변환하자!

추출한 단어를 TF 벡터로!

2-1. TF(Term Frequency)

- **TF**: 특정 단어가 문서에 **몇번 등장**했는지 나타내는 값
- Spark의 **CountVectorizer**를 이용해 **TF vector**를 구합니다

예

뉴스 ID	화재 [0]	공장 [1]	날씨 [2]	TF Vector
뉴스 1	4	1	0	[4, 1, 0]
뉴스 2	1	0	3	[1, 0, 3]
뉴스 3	2	0	1	[2, 0, 1]

2-2. IDF(Inverse Document Frequency)

TF
- “어떤 단어가 한 **문서에 자주** 나온다면, 그 단어는 해당 문서를 대표한다”

TF-IDF
- “하지만, **다른 문서에도 자주** 나오는 단어라면 **아니다**”
- **IDF**를 통해 문서 전반적으로 많이 나오는 단어의 TF 값을 낮춰줍니다
- TF-IDF는 문서의 **중요한 단어**를 나타내는 통계적 수치

아이디어를 수치로 나타내는 것!

3. Training

텍스트 데이터에
성능이 좋다고 알려진
Naive Bayes 사용

4. Evaluation

어떤 모델의 성능이 더 좋을까?

4-1. Accuracy(정확도)

- 전체 test 데이터셋 중 모델이 맞춘 것의 비율

학습 결과

모델	Accuracy
Naive Bayes TF	0.913
Naive Bayes TF-IDF	0.886

하지만,
과연 TF가
더 좋은 것일까요?

Wikipedia만 해도 모델 평가 지표가 이렇게 많은데?!

		Predicted condition			
	Total population	Predicted Condition positive	Predicted Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
True condition	condition positive	**True positive**	**False Negative** (Type II error)	True positive rate (TPR), Sensitivity, Recall $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$
	condition negative	**False Positive** (Type I error)	**True negative**	False positive rate (FPR), Fall-out $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$
Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$		Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

세상은 넓고 모델 평가 지표는 수 없이 많다

혹시 accuracy를 그저
"높으면 좋은 값"이라고
생각하고 계시진 않나요?

Accuracy가 높으면 좋은 모델일까요?

Accuracy라는 데이터 소리,
정확히 들어봅시다!

예) 카드 도용 감지 시스템

		예측 결과	
		도둑 이야!	도둑 아니야
진짜 사실	진짜 도둑	1	99
	도둑 아님	9	10000

Accuracy = 98.9%

뭔가 잘못되었다...

당연히 도둑 아닌 경우가 훨씬 많음

예) 카드 도용 감지 시스템

		예측 결과	
		도둑 이야!	도둑 아니야
진짜 사실	진짜 도둑	1	99
	도둑 아님	9	10000

아닌 것을 아니라하는 것은 별 의미가 없을 때가 있다

예) 카드 도용 감지 시스템

		예측 결과	
		도둑 이야!	도둑 아니야
진짜 사실	진짜 도둑	1	오류 2
	도둑 아님	오류 1	10000

당신의 오류는 어느 것이 더 심각하십니까?

예) 카드 도용 감지 시스템

오류1 심각

		예측 결과	
		도둑 이야!	도둑 아니야
진짜 사실	진짜 도둑	1	99
	도둑 아님	9	10000

도둑이라 한 자들이 진짜 도둑일 확률

`1/(1 + 9) = 10%`

Precision(정밀도)

예) 카드 도용 감지 시스템

오류2 심각

		예측 결과	
		도둑 이야!	도둑 아니야
진짜 사실	진짜 도둑	1	99
	도둑 아님	9	10000

진짜 도둑을 검거할 확률

`1/(1 + 99) = 1%`

Recall(재현율)

예) 카드 도용 감지 시스템

오류1 & **오류2** 심각

		예측 결과	
		도둑 이야!	도둑 아니야
진짜 사실	진짜 도둑	1	99
	도둑 아님	9	10000

Precision(P) & Recall(R) **조화평균**

2(PxR)/(P+R) = 1.8%

F1-measure

<조화평균>
분자가 같고
분모가 다를때
평균 내는 법

예) 카드 도용 감지 시스템

지표	의미	값
Accuracy	맞춘 비율(도둑 아닌 사람 포함)	98.9%
Precision	도둑이라한 사람이 진짜 도둑일 확률	10%
Recall	진짜 도둑을 잡아낼 확률	1%
F1-measure	precision과 recall 평균	1.8%

화재 뉴스 기사 분류는?

		예측 결과	
		화재	비화재
사실	화재	-	오류 2
	비화재	오류 1	-

당신의 오류는 어느 것이 더 심각하십니까?

화재 뉴스 기사

		예측 결과	
		화재	비화재
사실	화재	-	오류 2
	비화재	오류 1	-

필터링 결과에 비화재 문서가 섞여 있으면 안되죠!

Precision(정밀도)

화재 뉴스 기사

		예측 결과	
		화재	비화재
사실	화재	-	오류 2
	비화재	오류 1	-

그렇다고, 필터링 결과에 너무 적은 양의
화재 뉴스 기사만 남아 있으면 안됩니다

Recall(재현율)

화재 뉴스 기사

		예측 결과	
		화재	비화재
사실	화재	-	-
	비화재	-	-

둘 다 고려합시다!

F1-measure

4-2. All

- 모든 평가 지표를 고려합시다!

학습 결과

모델	Accuracy	F1-measure	Precision	Recall
Naive Bayes TF	0.913	0.895	0.828	0.975
Naive Bayes TF-IDF	0.886	0.868	0.779	0.981

TF 승리!

힝 속앗지롱 ㅎ ㅎ

TF-IDF는 문서의 중요 단어를 나타내는 좋은 통계적 수치이나,
그것이 화재/비화재를 분류하는데 도움이 될지는 모르는 것

여기서 알게 된 중요한 사실 세 가지,

1. 온갖 기법 다 쓴다고 성능 좋아지는 것 아니다

2. 상황에 맞는 feature를 사용해야한다

3. 모델의 좋고 나쁨은 어떻게 보느냐에 따라 다르다

인간의 욕심은 끝이 없기에...

화재 뉴스 기사 분류 성능 향상 기법

제가 논문 쓴 내용입니다 ㅎ ㅎ

교수님의 가르침,
머신러닝 **성능**은

양질의 데이터 > 좋은 Feature > 알고리즘

교수님의 가르침,
머신러닝 **성능**은

양질의 데이터 > 좋은 Feature > 알고리즘

양질의 데이터는 어찌할 수 없는 경우가 많다

"화재 / 비화재"를

구분 짓는 특성은

무엇이 있을까?

5.ExploratoryDataAnanlysis

- Exploratory Data Ananlysis(EDA): **데이터 탐색**
- 머신러닝 전, 어떤 feature를 학습에 사용할지 탐색하는 **중요한 단계**

문득 생긴 호기심

"뉴스 기사의 말투가 다르지 않을까?"

말투 Feature

Feature 이름	벡터 길이
{ 내용, 제목 } 글자 수	2
{ 내용, 제목 } { 인명, 지명 } 단어 수	4
{ 내용, 제목 } { X | 품사 16개 } 구성 비율	32
총계	38

말투 Feature 학습 결과

성능이 크게 떨어지지 않는다!

모델	Accuracy	F1-measure	Precision	Recall
Logistic Regression	0.827	0.782	0.760	0.804
Random Forest	0.871	0.823	0.872	0.778
Naive Bayes TF	0.922	0.904	0.858	0.956

*이전보다 성능이 더 높은것은 parameter 튜닝을 마친 후이기 때문입니다

"TF 벡터에도 말투를 반영할순 없을까?"

화재/비화재 뉴스 기사, 제목/내용
많이 나오는 단어가 다를 것이다!

제목 TF 벡터 내용 TF 벡터

따로 만들어서 이어 붙임

벡터의 길이가 길어져 차원의 저주 위험이 있음!

말투 TF 벡터 학습 결과

모델	Accuracy	F1-measure	Precision	Recall
Naive Bayes TF	0.922	0.904	0.858	0.956
Naive Bayes 말투 TF	0.922	0.903	0.856	0.955
SVM TF	0.903	0.882	0.824	0.949
SVM 말투 TF	0.932	0.913	0.886	0.942

"말투 Feature와 말투 TF 모두 사용해보자!"

- 방법1. 말투 feature와 말투 TF를 **이어붙인다**.
- 방법2. **모델**을 **결합**한다: 앙상블(Ensemble)

  → 확률 기반 Majority Voting 기법 사용

6.TunningResult

- 모든 모델 검증 결과
- 성능 표 및 곡선은 CSV로 저장 가능!

ML Pipeline 끝!

데이터에게 다양한 질문을 던지고,

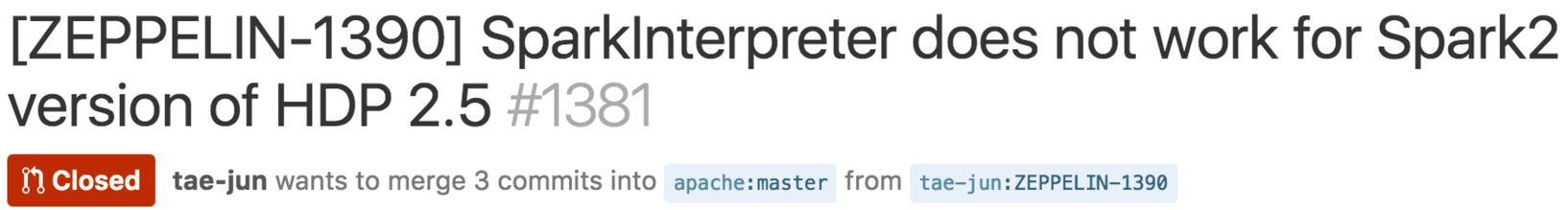
아이디어를 수치에 적용하고,

데이터의 소리를 정확히 들으세요!

Spark & Zeppelin은 여러분의 강력한 도구가 됩니다

마지막으로 드리고 싶은 말씀,

Zeppelin으로
Apache Contributor가 되세요!



- 이렇게 친절한 레포지토리 또 없습니다... 코멘트 보고 감동 ㅠㅠ
- 아직 초기 프로젝트라 발전 가능성이 많고, 그것은 곧 기여할 수 있는 것이 많다는 것!
- Apache 프로젝트의 contributor가 될 기회!

감사합니다

김태준(Jun Kim)
i2r.jun@gmail.com

Zeppelin Notebook, Dataset

datamining.uos.ac.kr

혹은 GitHub 주소

github.com/uosdmlab/playdata-zeppelin-notebook

후에 Facebook 커뮤니티 스사모, 제플린과 친구들에도 글 올리겠습니다