Reticulate + Shiny + Pyspark 사용하기

이상열

학습목표

- **문제의식을 공유한다.**
  - 문제를 발견한다
  - 문제를 기록한다.
  - 문제를 해결한다

- **왜 이걸 하는지?**
  - IT 기반 서비스 회사에서는 빅데이터 프레임워크가 필수
  - HDFS(하둡 파일 시스템)나 S3(AWS 제공하는 온라인 스토리지) 같은 빅데이터 시스템에 MapReduce, Hive, Spark 등 분산처리 프레임워크 개발을 사용하고 있음
  - 기존에 RDB(Relational Database) 하드웨어가 싸므로 Scale Up이 아닌 Scale Out 할 수 있는 분석 환경이 구성되고 있음

학습목표

- 알아야 될 용어
  - Apache Hadoop : 야후 Doug Cutting (Nutch serach engine)
    - HDFS (분산 파일 시스템)
    - MapReduce (데이터를 병렬 처리 (배치 연산)하게 할 수 있는 모듈)
  - Spark : RDD라는 데이터 형태로 단계별 iterate (read-write)의 반복 작업을 개선하기 위해 인메모리 대용량 데이터 고속처리 엔진

- Spark + R
  - SparkR (https://spark.apache.org/docs/latest/sparkr.html)
  - sparklyr (https://spark.rstudio.com/) – dplyr 형식 데이터 처리
  - Spark 진영 vs Rstudio 진영에서 각자 관리

데이터 분석가들이 기존보다 할 수 있는 일이 다양해졌음

- 데이터 전처리
- 통계분석 및 데이터마이닝
- 머신러닝 / 딥러닝 개발
- 리포트 (pdf, word, ppt, shiny report, 메일링 등 모든 것)
- ETL
- 대시보드
- 크롤링
- 네트워크

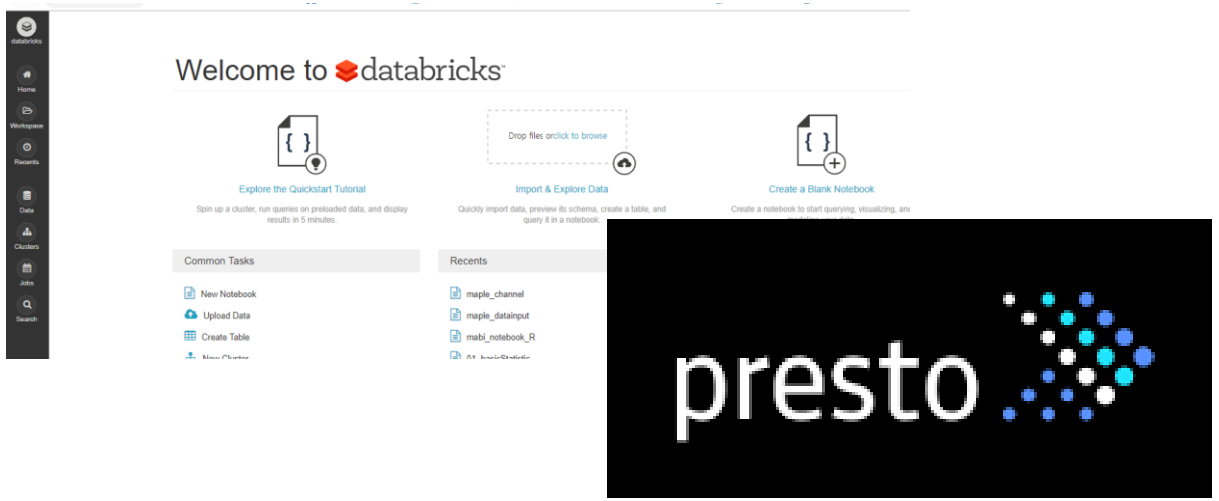
- 기존 RDB에서 Spark 등으로 데이터를 다루는 환경이 재구성되고 있음
- Product 관점에서 분석 시스템은 더 넓어지고 확장 될 가능성이 높음
- **분석 시스템의 확장은 “문제 상황” 동시에 “기회”**

문제

- 연산(데이터 처리)이 오래 걸림
  - 조회하는 데이터에 따라서 **분석 환경은 자유롭게 조절 가능해야 함**
  - R, Python에서 처리하기엔 너무 무거운 데이터
    - 버킷, 병렬 처리(foreach, multiprocessing...)
  - RDB(Relational Database)로 처리하기에도 너무 조회량이 많다.
    - Spark Functions, Spark SQL, Spark Dataframe, Spark SQL 부터 시작

그 외 좋은 대안들

Databricks Notebook
Presto



Reticulate

- R에서 제공하는 패키지, Reticulate는 R Markdown, Python 스크립트 및 모듈 가져오기, R 세션 내에서 대화식으로 파이썬 사용 등 다양한 방법으로 R에서 파이썬을 호출 가능
- R과 Python 객체 간 변환 (예 : R 및 Pandas 데이터 프레임 간 또는 R 행렬과 NumPy 배열 간 변환)
- 가상 환경 및 Conda 환경을 포함하여 다른 버전의 Python에 유연하게 결합
- Rstudio/tensorflow, Rstudio/keras

Reticulate

install.packages("reticulate")
reticulate::py_config()

bashrc (리눅스 기준)나 Rprofile.site 에서 path 잘 보도록

```
python:         /usr/bin/python3
libpython:      /usr/lib/python3.5/config-3.5m-x86_64-linux-gnu/libpython3.5.so
pythonhome:     /usr:/usr
version:        3.5.2 (default, Nov 12 2018, 13:43:14)  [GCC 5.4.0 20160609]
numpy:          /usr/local/lib/python3.5/dist-packages/numpy
numpy_version:  1.15.1

python versions found:
 /usr/bin/python
 /usr/bin/python3
 /home/syleeie/anaconda3/bin/python
 /home/syleeie/anaconda3/envs/r-mlflow/bin/python
```

Reticulate

Type conversions

When calling into Python, R data types are automatically converted to their equivalent Python types. When values are returned from Python to R they are converted back to R types. Types are converted as follows:

R	Python	Examples
Single-element vector	Scalar	`1`, `1L`, `TRUE`, `"foo"`
Multi-element vector	List	`c(1.0, 2.0, 3.0)`, `c(1L, 2L, 3L)`
List of multiple types	Tuple	`list(1L, TRUE, "foo")`
Named list	Dict	`list(a = 1L, b = 2.0)`, `dict(x = x_data)`
Matrix/Array	NumPy ndarray	`matrix(c(1,2,3,4), nrow = 2, ncol = 2)`
Data Frame	Pandas DataFrame	`data.frame(x = c(1,2,3), y = c("a", "b", "c"))`
Function	Python function	`function(x) x + 1`
NULL, TRUE, FALSE	None, True, False	`NULL`, `TRUE`, `FALSE`

Reticulate

```
def add(x,y):
    return x + y

source_python('add.py')
add(5,10)

[1] 15
```

Python 스크립트를 소스로 만들어서
R 환경 내에서 소스로 사용할 수 있음

```
np <- import("numpy", convert = FALSE)
a <- np$array(c(1:4))
sum <- a$cumsum()

# convert to R explicitly at the end
py_to_r(sum)
```

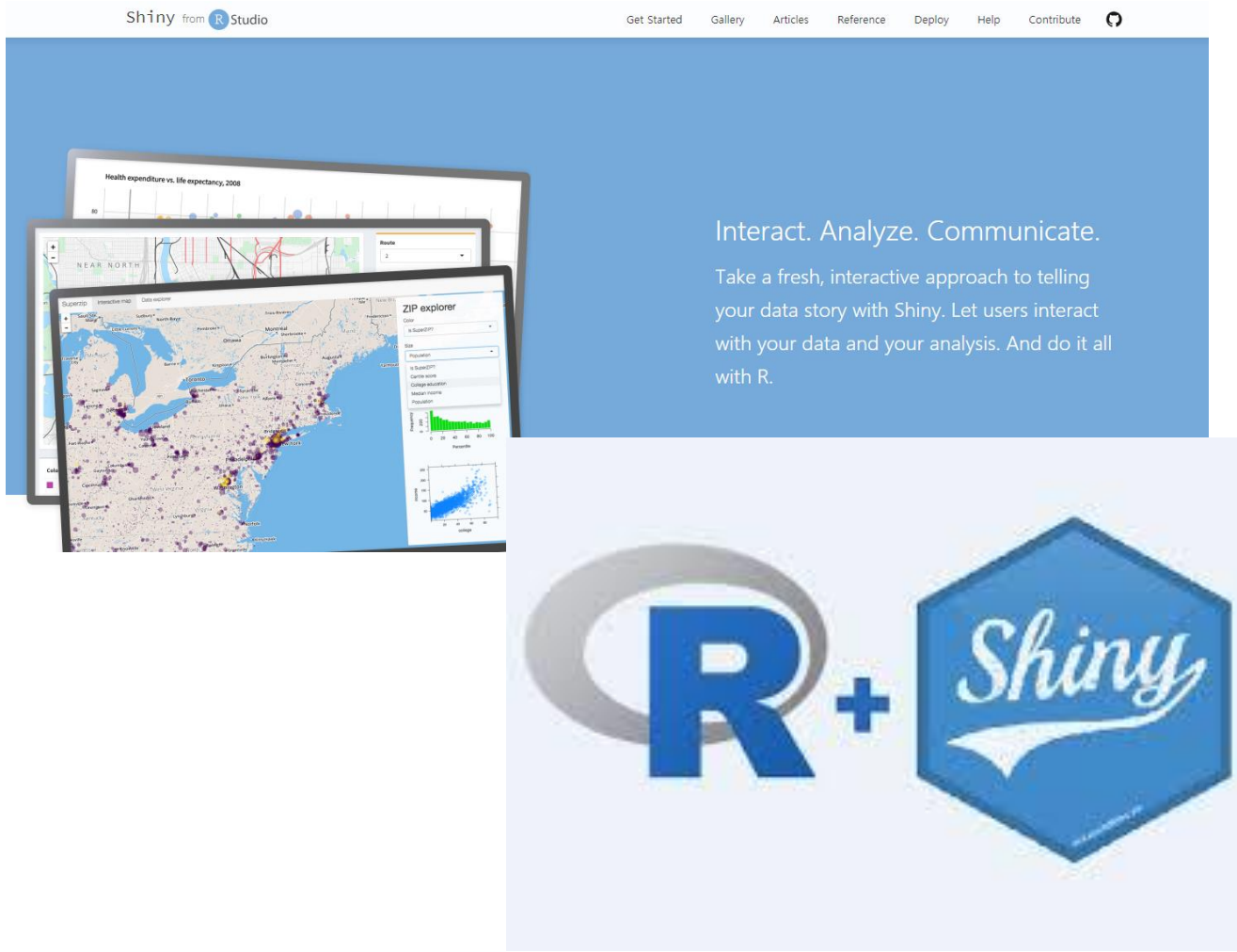
기본적으로 Python 객체가 R로 반환되면 R 객체이지만
Python 객체를 처리하는 경우에는, convert=False

Reticulate

해당 문서를 참고하시면 좋습니다.

https://rstudio.github.io/reticulate/articles/calling_python.html

Shiny



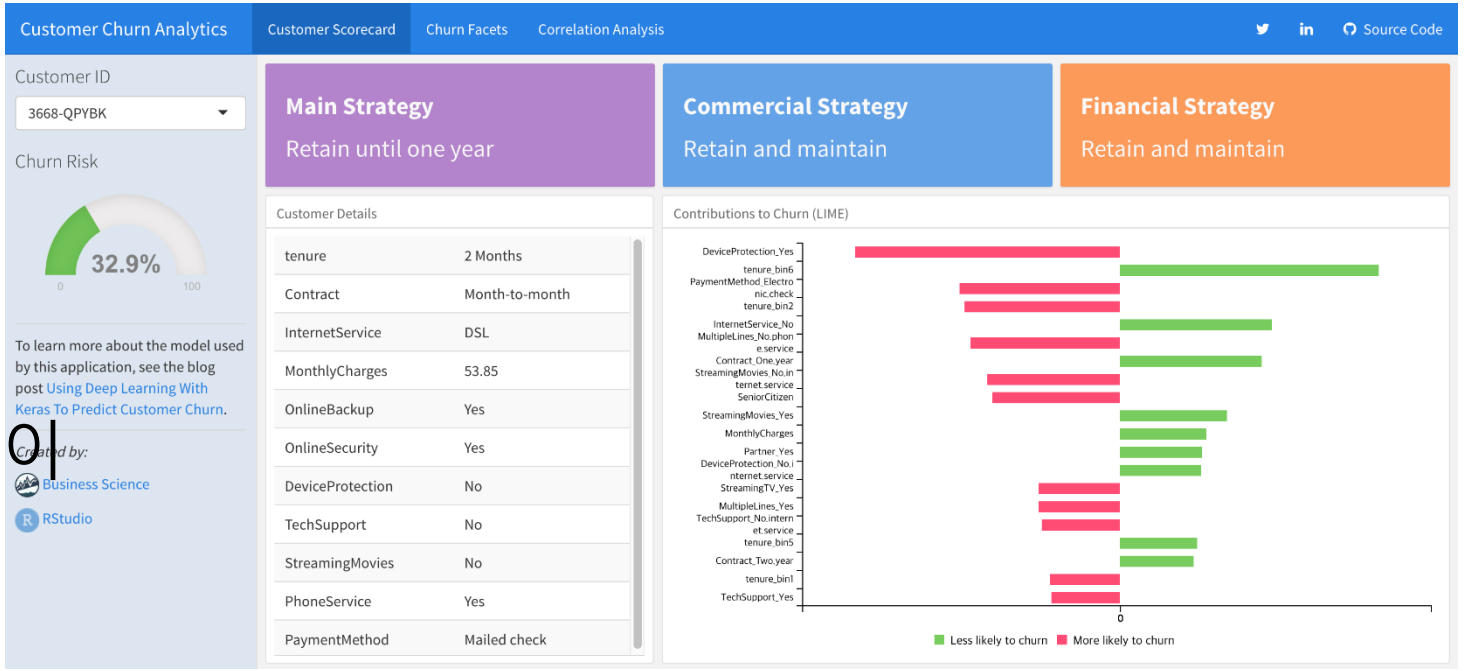
- 프로토타입이나 대시보드 용도 빠르게 분석가가 만들 수 있는 웹 어플리케이션 제작 라이브러리
- R programming (html, css, 자바스크립트 몰라도 가능)
- 반응성 앱(Reactive)이라 다이나믹한 결과 제공

Shiny + Machine Learning

Shiny에서 머신러닝 모델을 설명하려는 시도는 많았습니다.
Rstudio 블로그에서 제공한 예제, keras 라이브러리 이용하여 이탈모델 결과를 샤이니 어플리케이션으로 제작
(유저 아이디 별로 인터렉티브 하게 결과 확인 가능, Lime 같은 방법론도 사용)

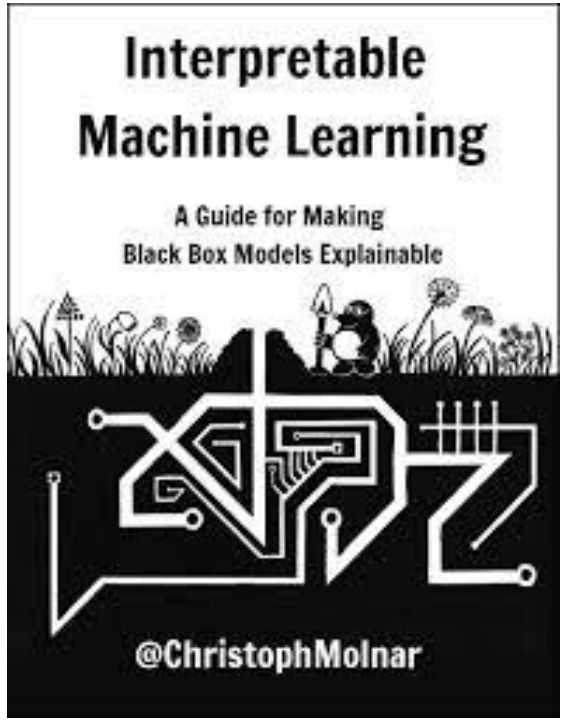
https://jjallaire.shinyapps.io/keras-customer-churn/

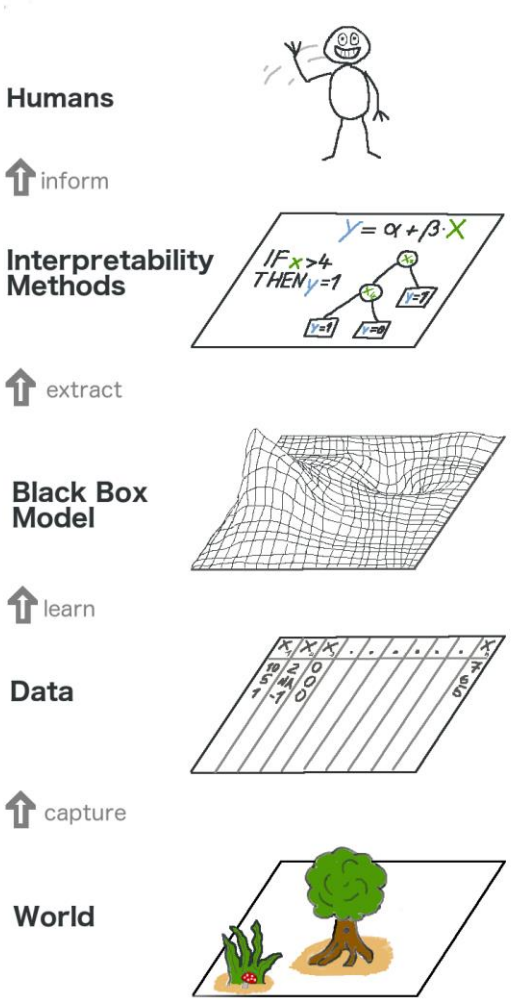
Shiny 통해서
Interactive Plot을 제공하고
Reactive 하면
모델을 시각화 하여 해석하기 용이



Interpretable Machine Learning

머신러닝 (블랙박스) 모델을 해석하고자 하는 노력





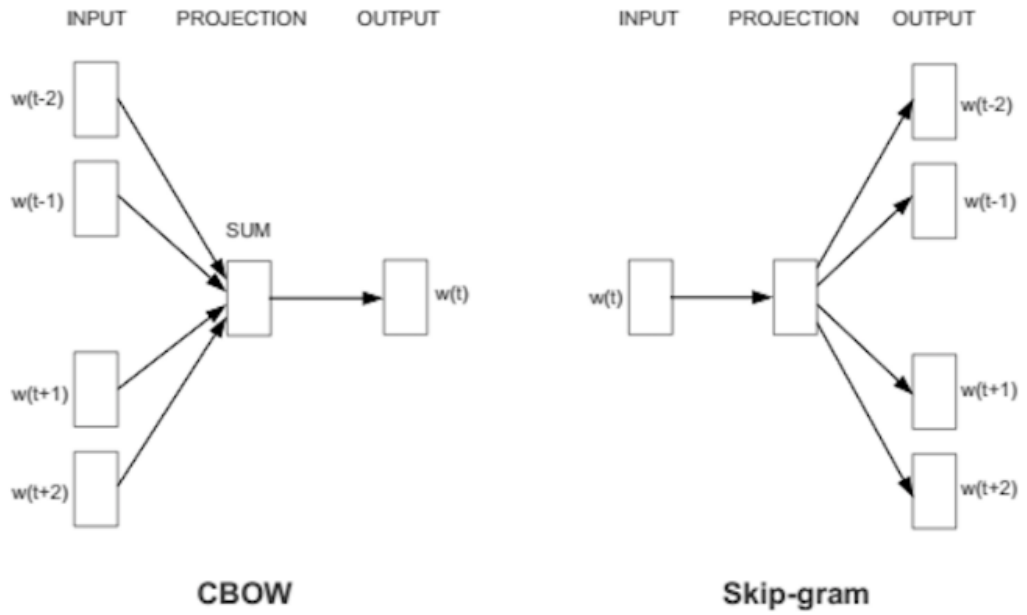
Interpretable Machine Learning

- LIME(Local Interpretable Model-agnostic Explanations)
  - Local Fieldity, Model Agnostic, Global Perspective
- Shapely Value
  - 게임이론 + SHAP(https://github.com/slundberg/shap)
- DALEX
  - PDP 이용하여 Conditional responses, 변수 별 기여도 파악
- Grad CAM : (Gradient-weighted Class Activation Mapping, arxiv)
  - CNN 모형에 적용할 수 있는 모형 해석 방법론
  - 쉽게 설명하면 CNN 층(마지막)에 들어가는 그래디언트를 가지고 자료의 어느 부분에 가중

사용 예제

- 추천 로직에서 자주 사용되는 방법론, 컨텐츠 정보를 벡터로 표현하기 위해서 **word2vec** 사용 (neural network)
- **word2vec**
  - 비슷한 분포를 가진 단어들은 비슷한 의미를 가질 것이라는 가정 (단어들이 같은 문맥에서 등장한다는 것)
- Example
  - The king loves the queen
  - The queen loves the king
  - The dwarf hates the king
  - The queen hates the dwarf
  - The dwarf poisons the king
  - The dwarf poisons the queen

- The king _____ the dwarf
  (연속적인 여러 단어로부터 한 단어를 추측하는 것.
  CBOW : Continuous bag of words model)
- _____ _____ dwarf _____ _____
  (한 단어로부터 여러 단어를 추측하는 것. Skip gram Model)



〈Word2vec 학습방법〉

사용 예제

예제 (특정 게임 네이버 공식카페)

word2vec 방법론은 NLP(자연어 처리)에서 주로 사용하는 방법론이지만 **다양하게 응용 가능하고 추천시스템에서 많이 사용됨**

graph2vec

event2vec

content2vec

…

```
In [23]: model.doesnt_match("최악 바보 망하다 대박".split())
Out[23]: '대박'

In [59]: model.most_similar(positive=['스톤에이지', '결투장'], negative=['세븐나이츠'], topn=5)
Out[59]: [('투기장', 0.4158957004547119),
          ('골로스', 0.35050666332244873),
          ('결장', 0.3488597273826599),
          ('조련사', 0.3357577323913574),
          ('결투장다', 0.3320501148700714)]

In [60]: model.most_similar(positive=['스톤에이지', '스페셜'], negative=['세븐나이츠'], topn=5)
Out[60]: [('전설', 0.47745075821876526),
          ('투기장', 0.4256047010421753),
          ('펫', 0.37289372086524963),
          ('NAVER', 0.3654584586620331),
          ('회피_수', 0.3583749830722809)]

In [61]: print(model['좋다'])
[  3.13897729e+00  -2.89208126e+00   3.38524270e+00   2.46348977e+00
  -4.32944155e+00  -2.84330893e+00  -8.47756088e-01   3.50296259e+00
   3.85893297e+00   2.25194407e+00  -4.98124552e+00   3.69822979e+00
  -4.22231817e+00   1.00052369e+00  -1.97644189e-01  -1.80142140e+00
   8.05867374e-01   3.44789839e+00  -1.01836848e+00  -5.70359409e-01
   4.89500427e+00   4.41284227e+00  -1.58597851e+00   3.97007942e-01
   3.60424852e+00   2.21780777e+00   1.49146402e+00   2.36920285e+00
  -2.17725587e+00   6.41840458e+00  -1.06625533e+00   1.64387167e-01
  -3.38128605e-03  -1.38111115e+00  -4.46919823e+00  -3.23771381e+00
   8.24807465e-01   2.22643065e+00  -1.12048960e+00  -3.18940568e+00
  -9.51956093e-01   5.95636666e-01   1.62952471e+00   4.51868820e+00
   1.23470676e+00  -1.75023830e+00   3.85837793e+00  -2.59093785e+00
  -1.07106194e-01  -3.51427031e+00  -1.45260644e+00   7.47411668e-01
  -2.15475106e+00  -2.23514843e+00   7.24580288e+00   2.61557430e-01
   7.90766060e-01  -4.44740200e+00   3.76671599e-03  -4.65279913e+00
  -6.98413563e+00  -1.67394304e+00  -1.00013636e-01  -1.23531425e+00
   8.81124115e+00  -2.97741365e+00   3.34249711e+00   2.30745959e+00
   9.65027630e-01   3.65694213e+00   2.37108231e+00   4.26628637e+00
  -1.48842013e+00   5.20411921e+00   1.92380989e+00  -1.22045434e+00
```

사용 예제

```
require(reticulate)
pyspark <- import("pyspark", convert = T)
sc <- pyspark$SparkContext()
word2vec_result <- pyspark$ml.feature$Word2VecModel$load('')

sc$stop()
```

- **스파크 모델을 읽기 전에 SparkContext로 스파크 클러스터를 접근할 수 있어야 함**
  (다만 해당 예제 클러스터는 EMR 형태의 클러스터가 아닌 호스터 한대 짜리의 standalone mode)
- **사용하고 있는 shiny 서버 (ec2)에 spark 설치가 먼저 필요함**
  (https://lovit.github.io/spark/2019/01/29/spark_0_install/)
- **스파크 모델에서 저장되고 있는 형태는 parquet 파일 구조 (data, metadata 폴더)**

- **spark.mlib은 https://spark.apache.org/docs/latest/ml-guide.html 참조하세요!**

사용 예제

```
## server.R 일부 내용
content_sim <- reactive({
    data.frame(matrix(unlist(word2vec_result$findSynonymsArray(content_select()$itemName, 10L)),
ncol = 2, byrow=T))
  })
```

- 스파크를 띄워서 word2vec 모델을 로딩하여 객체로 만듦
- pyspark 함수(findSynonymsArray) 사용하여 input 컨텐츠를 입력하여 가까운 컨텐츠를 뽑아줌
- word2vec 모델의 데이터는 spark.data.frame으로 저장되어 있기에, R의 data.frame으로 변환하여 tsne으로 그려 plot으로 시각화 할 수도 있음

(tsne는 고차원 데이터의 매니폴드(내재한 저차원)를 찾기 위해 사용, 자세한 것은 dos-tacos 블로그 내용을 참고하세요!

Reference

- https://rstudio.github.io/reticulate/
- https://jjallaire.shinyapps.io/keras-customer-churn/
- https://lovit.github.io/spark/2019/01/29/spark_0_install
- https://spark.apache.org/docs/latest/ml-guide.html

들어주셔서 감사합니다.

- Rstudio에서 R + Python + SQL를 Spark 안에서 자유롭게!!