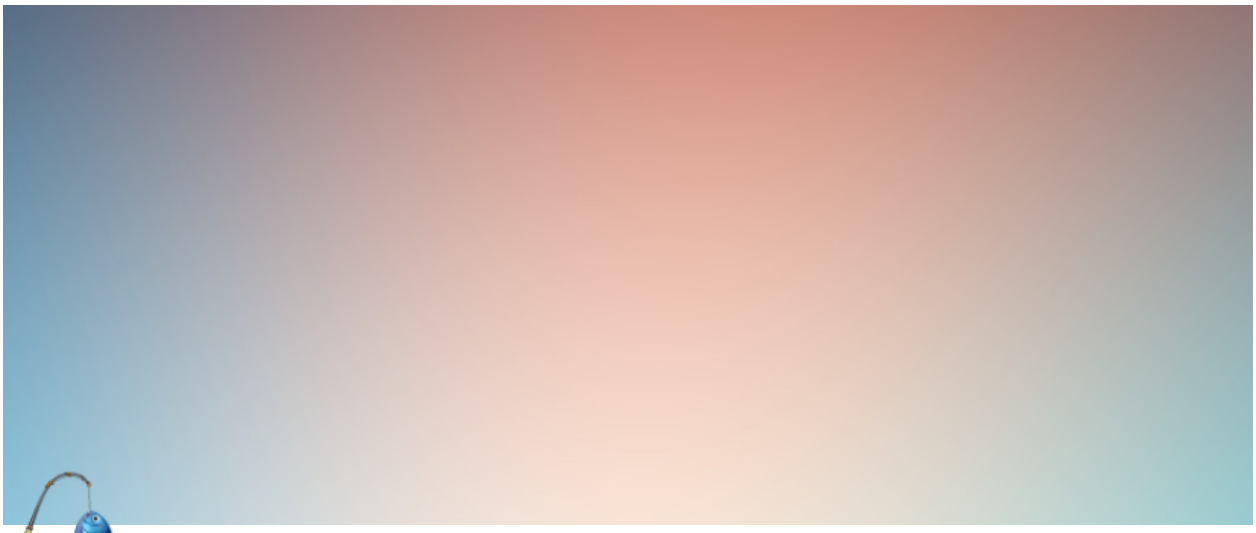


24-1 데이터마이닝 입문

> **피싱 사이트 검출 모델**
>
> - 목적: 피싱 피해 줄이기
> - **기본 가정: 피싱url(Y, 1), 정상url(N, 0)**
> - **False negative** 가 더 중요할 것으로 예상. (잘못된 주소를 피싱도메인이 아니라고 판정하면 안되니깐)
> - Phishing URL

📋 Progress

🎣 데마입 중간발표

🎣 데마입 기말발표

Define

1. **OUTPUT** : test URL이 피싱 사이트의 주
2. **PROCESS** : Predictive task → Decision Tree
3. **INPUT** : training URL → 각 URL의 특징을 training data로 사용

Variables for URL

Aa 변수	변수이름	종류	설명	사용여부	완료여부	+
length	url 길이	Nomial(명목) factor	url 길이 (54자를 기준으로 길면 피싱 (1), 짧으면 정상 (0))	사용O	완료	
special_chars	특수문자 개수	Ratio(비율) int	특수문자 개수	사용O	완료	
at_chars	'@' 개수	Ratio(비율) int	'@' 개수	사용O	완료	
double_slash_chars	'//' 개수	Ratio(비율) int	'//' 개수	사용O	완료	
dash_chars	'-' 개수	Ratio(비율) int	'-' 개수	사용O	완료	
numbers	숫자 개수	Ratio(비율) int	숫자 개수	사용O	완료	

Aa 변수	변수이름	종류	설명	사용여부	완료여부	+
protocol	https 사용 여부	Nomial(명목) factor	https 사용 여부	사용O	완료	
축약형URL	단축서비스여부	Nomial(명목) factor	url 축약형인지 여부	사용O	완료	→ 단축어 서비스 전부를 반영하지 못한 부분을 감안해야 한다
sample 피싱 url과의 편집 거리	편집거리	int	정상도메인 2000개를 추출하여 정상도메인과 피싱도메인의 최소 편집거리 비교	사용O	완료	
도메인 수명		Nomial(명목) factor	whois에 도메인 수명(1년 이상 남았으면 1,등록안되어있거나 1년 이하 -1,에러0)	보류	진행중	코드 실행 시간 초과로 인해 잘 작동안함
코잇 사이트 등록 여부		Nomial(명목)		보류	진행중	
subdomain	서브도메인	Nomial(명목) factor	서브 도메인 개수 - 2개 이상 피싱 (1), 2개 의심 (0), 그외 정상 (-1)	사용O	완료	

Aa 변수	변수이름	종류	설명	사용여부	완료여부	+
전체 길이 중 숫자 비율	숫자비율	int	전체 url 길이 대비 숫자의 비율	사용O	완료	
전체 길이 중 특수문자 비율	특수문자비율	int	전체 url 길이 대비 특수문자의 비율	사용O	완료	→ 정상 데이터가 평균적으로 더 작으나, 이상치가 많아 추가 처리 필요할듯
path 깊이	path 깊이	int	웹 서버의 리소스에 대한 경로 (path 개수만큼 라벨링)	사용O	완료	→ 피싱 사이트보다 정상 사이트 path 길이가 더 깊
대문자 비율	대문자 비율	int	url에서 대문자의 비율	보류	완료	
http 위치				사용X	완료	
파일 확장자 여부	확장자 포함여부	Nomial(명목) factor	.txt .exe .js 같은 파일 확장자가 url 내에 있는지 확인 (확장자 있으면 1, 없으면 0)	보류	완료	→ 정상 사이트에서 확장자가 더 많이 발견된 점에 더불어 피싱 사이트와의 큰 개수 차이가 없어서 별다른 의미 없어 보임
[도메인] 특수문자 개수	도메인_특수문자개수			사용O	완료	

Aa 변수	변수이름	종류	설명	사용여부	완료여부	+
[도메인] . 개수	도메인_점개수			사용O	완료	
[도메인] 숫자 개수	도메인_숫자개수			사용O	완료	
[도메인] - 개수	도메인_대시개수			사용O	완료	
query length	쿼리길이		쿼리 길이가 길면 피싱, 아니면 정상	보류	완료	
피싱 단어 빈도		Nomial(명목) factor	피싱, 정상 URL에서 빈출되는 단어가 포함되는지 아닌지 여부	보류	완료	

▼ Dataset (raw)

phishing_url.csv

출처: 공공데이터포털 - 한국인터넷진흥원이 탐지한 피싱 사이트 URL
- 총 1,986

normal_url.csv

출처: 공공데이터포털 - 행정안전부_공공기관 웹,모바일웹 사이트 정보
- 총 22,341

top-1m.csv

출처: https://www.kaggle.com/datasets/cheedcheed/top1m (alexa) - Alexa Top 1 Million Sites
- 총 100,0000
* 도메인만 있음 → http, https 추가 처리 필요
* 일일 방문자 수와 페이지 뷰를 통해 순위를 계산한 상위 백만 개의 도메인 → 100% 정상 도메인은 아닐수도

urldata.csv

출처: https://www.kaggle.com/datasets/siddharthkumar25/malicious-and-benign-urls
- 총 450,176
- 77% benign 33% malicious

malicious_phish.csv

출처 : https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset
- 총 641,119
- 66% benign 15% defacement 19% phishing&malware

Dataset (가공 중)

- **기본 가정: 피싱url(Y), 정상url(N)**

phishing_data.csv

url2000.xlsx

url.csv

R 코드

▼ 기본 변수 설정, 데이터 처리

```
setwd("/Users/chaseoyun/Documents/data_analysis/teamproject")
phishing_url <- read.csv("phishing_url.csv")

phishing_url <- phishing_url[-1]
isPhishing <- "Y"

phishing_total <- data.frame(phishing_url, isPhishing)
names(total_url) <- c("url", "isPhishing")

write.csv(phishing_total, "phishing_data.csv")
```

```
setwd("/Users/chaseoyun/Documents/data_analysis/teamproject")
normal_url <- read.csv("normal_url.csv", fileEncoding="euc-kr

isPhishing <- "N"

normal_total <- data.frame(normal_url, isPhishing)
names(total_url) <- c("url", "isPhishing")
```

```
write.csv(normal_total, "normal_data.csv")
```

```
setwd("/Users/chaseoyun/Documents/data_analysis/teamproject")
phishing_data <- read.csv("phishing_data.csv", stringsAsFacto
phishing_data <- phishing_data[-1]
phishing_data$isPhishing <- factor(phishing_data$isPhishing,

library(stringr)
phishing_data$url <- str_extract(phishing_data$url, "^[^ ]+")
write.csv(phishing_data, "phishing_data.csv")
```

▼ Decision Tree 모델 코드

train, test set

C5.0 패키지

tree 패키지

rpart 패키지

randomforest

xgboost

모델 성능 비교

전체 데이터 셋 개수 8000개

훈련 셋 6400개, 테스트 셋 1600개

- C5.0 패키지
  - tree size : 74
  - 훈련 셋 정확도: 0.96 (6115/6400)
  - FN : 170 (2.625%)

    - 테스트 셋 정확도: 0.92 (1484/1600)
    - FN : 58 (3.625%)
- 성능 향상 (boost10)
    - tree size : 62.5
    - 훈련 셋 정확도: 0.97 (6242/6400)
    - FN : 105 (2.34%)
    - 테스트 셋 정확도: 0.94 (1513/1600)
    - FN : 49 (3.06%)
- 고비용 실수

  → FN : 피싱사이트인데 아니라고 분류했을 경우, FP : 피싱사이트가 아닌데 피싱사이트라고 분류했을 경우

  → FN를 잘못 예측하면 FP를 잘못 예측했을 때보다 4배의 손해를 본다고 가정
    - 훈련 셋 정확도: 0.92 (5916/6400)
    - FN : 49 (0.7%)
    - 테스트 셋 정확도: 0.91 (1456/1600)
    - FN : 25 (1.56%)
- trees 패키지
    - tree size = 10
    - 테스트 셋 정확도 : 0.88 (1410/1600)
    - FN : 41 (2.5%)
- rpart 패키지
    - 테스트 셋 정확도: 0.88 (1410/1600)
    - FN : 101 (6.3%)
- 부스팅 10번, costs 4배 적용
    - 훈련 셋 정확도: 0.97 (6239/6400)
    - FN : 92 (1.4%)

    - 테스트 셋 정확도: 0.93 (1496/1600)
    - FN : 73 (4.5%)
- 랜덤 포레스트
    - 훈련 셋 정확도: 0.94 (6079/6400)
    - FN : 209 (3.2%)
    - 테스트 셋 정확도: 0.93 (1493/1600)
    - FN : 61 (3.8%)
- XGBoost
    - 테스트 셋 정확도: 0.91
    - FN : 57 (3.5%)

▼ 특수문자 처리

```
# csv파일에서 피싱url 데이터 열기
phishing_data <- read.csv("phishing_data.csv", stringsAsFacto
# 1행은 인덱스이므로 제외
phishing_data <- phishing_data[-1]

# 피싱여부 데이터 factor화 (레벨: Y/N)
phishing_data$isPhishing <- factor(phishing_data$isPhishing,
levels(phishing_data$isPhishing) <- c("Y","N")

# 피싱데이터에서 알파벳, 숫자 제외한 특수문자 개수 셈 >> special_chars열
phishing_data$special_chars <- str_count(phishing_data$url, "

# 피싱데이터에서 "@" 개수 셈 >> at_chars열에 저장
phishing_data$at_chars <- str_count(phishing_data$url, "@")
# 피싱데이터에서 "//" 개수 셈 >> double_slash_chars열에 저장
phishing_data$double_slash_chars <- str_count(phishing_data$u
# 피싱데이터에서 "-" 개수 셈 >> dash_chars열에 저장
phishing_data$dash_chars <- str_count(phishing_data$url, "-")
```

```
normal_data <- read.csv("normal_data.csv", stringsAsFactors =
normal_data <- normal_data[-1]

normal_data$isPhishing <- factor(normal_data$isPhishing, leve
levels(normal_data$isPhishing) <- c("Y", "N")

normal_data$special_chars <- str_count(normal_data$url, "[^A-

normal_data$at_chars <- str_count(normal_data$url, "@")
normal_data$double_slash_chars <- str_count(normal_data$url,
normal_data$dash_chars <- str_count(normal_data$url, "-")
```

도메인 추출

```
phishing_domain <- phishing_data$url
phishing_domain <- data.frame(phishing_domain)
names(phishing_domain) <- c("domain")
phishing_domain <- apply(phishing_domain, MARGIN = 2, extract
phishing_domain <- as.data.frame(phishing_domain)
```

```
extract_domain <- function(x) {
    domain <- gsub("https?://", "", x)
    return(domain)
}
```

▼ 숫자 개수

```
normal_data$numbers <- str_count(normal_data$url, "[0-9]")
phishing_data$numbers <- str_count(phishing_data$url, "[0-9]"
```

<결과>

```
> round(prop.table(table(phishing_data$numbers)),3)

    0     1     2     3     4     5     6    11    12
0.510 0.298 0.088 0.095 0.008 0.001 0.001 0.001 0.001
> round(prop.table(table(normal_data$numbers)),3)

    0     1     2     3     4     5     6     7     8     9    10    11    12
0.931 0.034 0.011 0.015 0.006 0.000 0.000 0.001 0.001 0.000 0.000 0.000 0.000
   14    15    18    20
0.000 0.000 0.000 0.000
```

▼ URL 길이

```
install.packages("readxl")
library(readxl)
library(readr)

df$'url length' <- nchar(df$'url')

classify_address <- function(length) {
  if (length >= 54) {
    return('1') # 피싱
  } else {
    return('0') # 정상
  }
}

df$'url_length' <- sapply(df$'url length', classify_address)
```

```
> df$'url_length' <- sapply(df$'url length', classify_address)
> head(df)
                      url url length isphishing url_length
1 http://masf.krhes.boston         24          Y          0
2   con08.nu8w.love 외11건         20          Y          0
3            coz.ul3u.show         13          Y          0
4   https://donate.do/h73t         22          Y          0
5           oabe.fyy8.plus         14          Y          0
6            mp.fnb4.media         13          Y          0
>
```

```
> head(df)
      category                                                  url name
1 중앙행정기관 가습기살균제사건과4.16세월호참사특별조사위원회
2 중앙행정기관                                                감사원
3 중앙행정기관                                                감사원
4 중앙행정기관                    감사원 어린이 청소년 홈페이지
5 중앙행정기관                                              공공감사
6 중앙행정기관                                감사원 (영어)(BAI)
                                    url url length isphishing url_length
1 http://socialdisasterscommission.go.kr/         39      FALSE          0
2    http://www.bai.go.kr/mobile/index.do         36      FALSE          0
3                  http://www.bai.go.kr           20      FALSE          0
4     http://www.bai.go.kr/child/index.do         35      FALSE          0
5                 http://www.pasa.go.kr           21      FALSE          0
6               http://english.bai.go.kr          24      FALSE          0
```

- Phishing URL

```
# 작업 디렉토리 세팅
setwd("C:/DataAnalysis/project")

# 라이브러리 설치
# readr - 파일에서 데이터 읽기, 데이터를 파일로 내보내기, 문자열 구문 분
# R 표준 함수에 비해 처리 속도가 빠르며 직관적이고 사용하기 쉬움
library(readr)

# 파일 읽어오기
df <- read.csv("phishing_data.csv", fileEncoding = "euc-kr")

# 각 홈페이지 주소의 글자 수를 계산하여 새로운 열 추가
df$'url length' <- nchar(df$'url')

# 파일 내 열 순서 변경
df <- df[c('url', 'url length', 'isphishing')]

# 수정된 DataFrame을 새로운 CSV 파일로 저장
write_csv(df, 'phishing_data_length.csv')
```

<결과>

```
> summary(df2)
     url              url length     isphishing
 Length:1986        Min.   :10.00   Length:1986
 Class :character   1st Qu.:20.00   Class :character
 Mode  :character   Median :22.00   Mode  :character
                    Mean   :20.99
                    3rd Qu.:23.00
                    Max.   :51.00
```

- Normal URL

```
# 작업 디렉토리 세팅
setwd("C:/DataAnalysis/project")

# 라이브러리 설치
# readr - 파일에서 데이터 읽기, 데이터를 파일로 내보내기, 문자열 구문 분
# R 표준 함수에 비해 처리 속도가 빠르며 직관적이고 사용하기 쉬움
library(readr)

# 파일 읽어오기
df <- read.csv("normal_data.csv", fileEncoding = "euc-kr")

# 열 이름 변경
colnames(df) <- c("category", "url name", "url", "isphishing"

# 각 홈페이지 주소의 글자 수를 계산하여 새로운 열 추가
df$'url length' <- nchar(df$'url')

# 파일 내 열 순서 변경
df <- df[c('category', 'url name', 'url', 'url length', 'isph

# 수정된 DataFrame을 새로운 CSV 파일로 저장
write_csv(df, 'normal_data_length.csv')
```

<결과>

```
  category          url name            url             url length
Length:22341       Length:22341       Length:22341       Min.   : 13.00
Class :character   Class :character   Class :character   1st Qu.: 22.00
Mode  :character   Mode  :character   Mode  :character   Median : 24.00
                                                         Mean   : 25.22
                                                         3rd Qu.: 27.00
                                                         Max.   :101.00

 isphishing
Length:22341
Class :character
Mode  :character
```

url2000

```
install.packages("readxl")
library(readxl)
library(readr)

setwd("C:/DataAnalysis/project")

df <- read_excel("url2000.xlsx")

df$'url length' <- nchar(df$'url')

classify_address <- function(length) {
  if (length >= 54) {
    return('1') # 피싱
  } else {
    return('0') # 정상
  }
}

df$'url_length' <- sapply(df$'url length', classify_address)
```

```
# A tibble: 2,000 × 5
    ...1 url                                                    label `url length` url_length
   <dbl> <chr>                                                  <chr>         <int> <chr>
 1     1 https://www.en.wikipedia.org/wiki/Surf_Canyon          beni…            45 0
 2     2 https://www.publicampaign.org/blog-tags/loni-hancock   beni…            52 0
 3     3 https://www.consequenceofsound.net/2011/04/former-unicorns-vo… beni…    90 1
 4     4 https://www.ticketloot.com/sidney-harman-hall-tickets  beni…            53 0
 5     5 https://www.lonelyplanet.com/india/kolkata-calcutta    beni…            51 0
 6     6 https://www.ebay.com/itm/1934-Ad-Wrigleys-Double-Mint-Gum-Mar… beni…   113 1
 7     7 https://www.tuneheaven.com/Sheet-Music-Tabs/American_Dream.ht… beni…    63 1
 8     8 https://www.amazon.com/Man-Year-Widescreen-Robin-Williams/dp/… beni…    71 1
 9     9 https://www.bourque.org/notes.html                     beni…            34 0
10    10 https://www.occq-qcco.com/main.cfm?p=421&l=en&CategorieID=361… beni…   83 1
# ⓘ 1,990 more rows
# ⓘ Use `print(n = ...)` to see more rows
```

▼ 서브도메인 개수

- Phishing URL

```
# null 값 생기는 오류 코드

library(stringr)

setwd("C:/DataAnalysis/project")

 classify_url <- function(url) {
  # "www." 제거
  url <- gsub("www\\.", "", url)

  # 프로토콜 제거
  url <- gsub("^https?://", "", url)

  # 도메인 부분 추출
  domain_parts <- unlist(strsplit(url, "\\."))

  # ccTLD 제거 (마지막 최상위 도메인 부분 제거)
  if (length(domain_parts) > 1) {
    domain_parts <- domain_parts[-length(domain_parts)]
  }

  # 서브도메인 개수 세기
  subdomain_count <- length(domain_parts)

  # 서브도메인의 개수에 따라 분류
```

```
  if (subdomain_count > 2) {
    return("1") # 피싱
  } else if (subdomain_count == 2) {
    return("0") # 의심
  } else {
    return("-1") # 정상
  }
}

# URL에서 도메인을 추출하는 함수
extract_domain <- function(url) {
parsed_url <- httr::parse_url(url)
return(parsed_url$hostname)}

# 도메인 열 추가
df$domain <- sapply(df$url, extract_domain)

# 'sub domain' 열에 classify_url 함수 적용
df$sub_domain <- sapply(df$url, classify_url)

# 결과 확인
head(df)
```

	url	isphishing	domain	sub_domain
1	http://masf.krhes.boston	Y	masf.krhes.boston	의심
2	con08.nu8w.love 외11건	Y	NULL	정상
3	coz.ul3u.show	Y	NULL	정상
4	https://donate.do/h73t	Y	donate.do	정상
5	oabe.fyy8.plus	Y	NULL	정상
6	mp.fnb4.media	Y	NULL	정상

```
# null 값 수정

classify_url <- function(url) {
  # 프로토콜 제거
  url <- gsub("^https?://", "", url)
```

```
  # "www." 제거
  url <- gsub("www\\.", "", url)

  # 도메인 부분 추출
  domain_parts <- unlist(strsplit(url, "\\."))

  # ccTLD 제거 (마지막 최상위 도메인 부분 제거)
  if (length(domain_parts) > 1) {
    domain_parts <- domain_parts[-length(domain_parts)]
  }

  # 서브도메인 개수 세기
  subdomain_count <- length(domain_parts)

  # 서브도메인의 개수에 따라 분류
  if (subdomain_count > 2) {
    return("1") # 피싱
  } else if (subdomain_count == 2) {
    return("0") # 의심
  } else {
    return("-1") # 정상
  }
}

df$sub_domain <- sapply(df$url, classify_url)
```

<결과>

```
                          url isphishing sub_domain
1 http://masf.krhes.boston           Y          0
2   con08.nu8w.love 외11건           Y          0
3            coz.ul3u.show           Y          0
4   https://donate.do/h73t           Y         -1
5           oabe.fyy8.plus           Y          0
6            mp.fnb4.media           Y          0

> table(df$sub_domain)

  -1    0    1
 262 1707   17
```

- Normal URL

```
# 공공기관 중복 도메인 제
install.packages("httr")
library(httr)

setwd("C:/DataAnalysis/project")

df <- read.csv("normal_data_length.csv", fileEncoding = "utf-

# URL에서 도메인을 추출하는 함수
extract_domain <- function(url) {
  parsed_url <- httr::parse_url(url)
  return(parsed_url$hostname)}

# 도메인 열 추가
df$domain <- sapply(df$url, extract_domain)

# 중복 제거된 DataFrame 생성
unique_df <- df[!duplicated(df$domain), ]

# 중복 제거된 DataFrame 출력
unique_df
```

```
> nrow(unique_df)
[1] 19914
```

```r
# null 값 수정

classify_url <- function(url) {
  # 프로토콜 제거
  url <- gsub("^https?://", "", url)

  # www. 제거
  url <- gsub("www\\.", "", url)

  # 도메인 부분 추출
  domain_parts <- unlist(strsplit(url, "\\."))

  # ccTLD 제거 (마지막 최상위 도메인 부분 제거)
  if (length(domain_parts) > 1) {
    domain_parts <- domain_parts[-length(domain_parts)]
  }

  # 서브도메인 개수 세기
  subdomain_count <- length(domain_parts)

  # 서브도메인의 개수에 따라 분류
  if (subdomain_count > 2) {
    return("1") # 피싱
  } else if (subdomain_count == 2) {
    return("0") # 의심
  } else {
    return("-1") # 정상
  }
}

df$sub_domain <- sapply(df$url, classify_url)
```

<결과>

```
> table(df$sub_domain)

  -1     0     1
 778 12857  8706
```

	category	url.name
1	중앙행정기관	가습기살균제사건과4.16세월호참사특별조사위원회
2	중앙행정기관	감사원
3	중앙행정기관	감사원
4	중앙행정기관	감사원 어린이 청소년 홈페이지
5	중앙행정기관	공공감사
6	중앙행정기관	감사원 (영어)(BAI)

	url	url.length	isphishing	sub_domain
1	http://socialdisasterscommission.go.kr/	39	FALSE	0
2	http://www.bai.go.kr/mobile/index.do	36	FALSE	1
3	http://www.bai.go.kr	20	FALSE	1
4	http://www.bai.go.kr/child/index.do	35	FALSE	1
5	http://www.pasa.go.kr	21	FALSE	1
6	http://english.bai.go.kr	24	FALSE	1

url2000 data

- 서브 도메인

```
# readxl 패키지 설치 및 불러오기
install.packages("readxl")
library(readxl)

# Excel 파일 불러오기

setwd("C:/DataAnalysis/project")

df <- read_excel("url2000.xlsx")

df$sub_domain <- sapply(df$url, classify_url)
```

```
# A tibble: 2,000 × 4
    ...1 url                                                                                                  label sub_domain
   <dbl> <chr>                                                                                                <chr> <chr>
 1     1 https://www.en.wikipedia.org/wiki/Surf_Canyon                                                        Y     0
 2     2 https://www.publicampaign.org/blog-tags/loni-hancock                                                 Y     -1
 3     3 https://www.consequenceofsound.net/2011/04/former-unicorns-vocalist-alden-penner-drops-ep/           Y     -1
 4     4 https://www.ticketloot.com/sidney-harman-hall-tickets                                                Y     -1
 5     5 https://www.lonelyplanet.com/india/kolkata-calcutta                                                  Y     -1
 6     6 https://www.ebay.com/itm/1934-Ad-Wrigleys-Double-Mint-Gum-Marguerite-Churchill-ORIGINAL-ADVERTISING-/300606142914 Y     -1
 7     7 https://www.tuneheaven.com/Sheet-Music-Tabs/American_Dream.html                                      Y     0
 8     8 https://www.amazon.com/Man-Year-Widescreen-Robin-Williams/dp/B000LC5B7O                              Y     -1
 9     9 https://www.bourque.org/notes.html                                                                   Y     0
10    10 https://www.occq-qcco.com/main.cfm?p=421&l=en&CategorieID=3613&MembreID=3556&page=1                  Y     0
# ⓘ 1,990 more rows
# ⓘ Use `print(n = ...)` to see more rows
```

```
> table(df$sub_domain)

 -1   0   1
961 837 202
>
```

▼ url 랜덤추출

```
#정상 url추출(4천개 train,test에 사용, 2천개 편집거리 비교에 사용)
# 데이터를 불러오기
norm_data <- read.csv("c:/data/url_all.csv")

# 데이터프레임에서 랜덤으로 6000개의 샘플을 추출
norm_sampled_data <- norm_data[sample(nrow(norm_data), 6000),

# 새로운 CSV 파일 생성
write.csv(norm_sampled_data, file = "C://data/normal_url.csv'

#-------------------------------------------------------------
#피싱 url추출
# 데이터를 불러오기
phishing_data <- read.csv("c:/data/201910.csv")

# 데이터프레임에서 랜덤으로 500개의 샘플을 추출
phishing_sampled_data <- phishing_data[sample(nrow(phishing_d

# 새로운 CSV 파일 생성
write.csv(phishing_sampled_data, file = "C://data/phishing_ur
#--------------
```

```
# 데이터를 불러오기(기존 데이터에 추가)
phishing_data <- read.csv("C://data/phishing_url.csv")

# 파일 리스트 설정
file_list <- c("c:/data/202004.csv", "c:/data/202111.csv", "c
               "c:/data/202201.csv", "c:/data/202302.csv", "c

# 각 파일에서 데이터 추출하여 행으로 추가
for (file in file_list) {
  data <- read.csv(file)
  if (file == "c:/data/202403.csv") {
    sampled_data <- data[sample(nrow(data), 600), ]
  } else {
    sampled_data <- data[sample(nrow(data), 500), ]
  }
  phishing_data <- rbind(phishing_data, sampled_data)
}

# 새로운 CSV 파일 생성
write.csv(phishing_data, file = "C://data/phishing_url.csv",
```

▼ 도메인 추출

```
#피싱,정상url 8천개에서 도메인 추출
library(httr)

# 도메인 추출 함수
extract_domain <- function(url) {
  parsed_url <- httr::parse_url(url)
  hostname <- parsed_url$hostname
  if (startsWith(hostname, "www.")) {
    hostname <- substring(hostname, 5)
  }
  return(hostname)
```

```
}


# phishing 데이터 불러오기
urls <- read.csv("C://data/url.csv")

# 각 URL에서 도메인 추출하여 새로운 열 추가
urls$domain <- sapply(urls$url, extract_domain)

# 새로운 CSV 파일로 저장
write.csv(urls, file = "C://data/url_domain.csv", row.names =



#-----------------------------
#편집거리 계산을 위한 정상url 2천개 도메인 추출
library(httr)

# 도메인 추출 함수
extract_domain <- function(url) {
  parsed_url <- httr::parse_url(url)
  hostname <- parsed_url$hostname
  if (startsWith(hostname, "www.")) {
    hostname <- substring(hostname, 5)
  }
  return(hostname)
}

# phishing 데이터 불러오기
url_2000 <- read.csv("C://data/norm_url_2000.csv")

# 각 URL에서 도메인 추출하여 새로운 열 추가
url_2000$domain <- sapply(url_2000$url, extract_domain)

# 새로운 CSV 파일로 저장
write.csv(url_2000, file = "C://data/norm_url_2000_domain.csv
```

▼ 대문자 비율 구하기

```
# 데이터 불러오기
urls <- read.csv("C://data/url.csv")

# 대문자 비율 구하기
urls$uppercase_ratio <- sapply(urls$url, function(url) {
  uppercase_count <- sum(utf8ToInt(url) >= 65 & utf8ToInt(url
  total_characters <- nchar(url)
  if (total_characters == 0) {
    return(0) # URL에 문자가 없는 경우를 고려하여 0을 반환
  } else {
    return(uppercase_count / total_characters)
  }
})

# 결과를 새로운 CSV 파일에 저장
output_file <- "C://data/urls_Uppercase_ratio.csv"
write.csv(urls, file = output_file, row.names = FALSE)
```

▼ 정상 도메인과의 편집 거리

처음에 뽑은 정상 url 2000개의 도메인과 피싱,정상 도메인 8천개를 비교하여 각각 편집 거리 구함

```
library(stringdist)

# calculate_min_similarity 함수 정의
calculate_min_similarity <- function(url1, url_list2) {
  min_distance <- min(stringdist::stringdist(url1, url_list2))
  return(min_distance)
}

# 첫 번째 CSV 파일의 링크들 불러오기
norm_2000_domain <- read.csv("C://data/norm_url_2000_domain.c
```

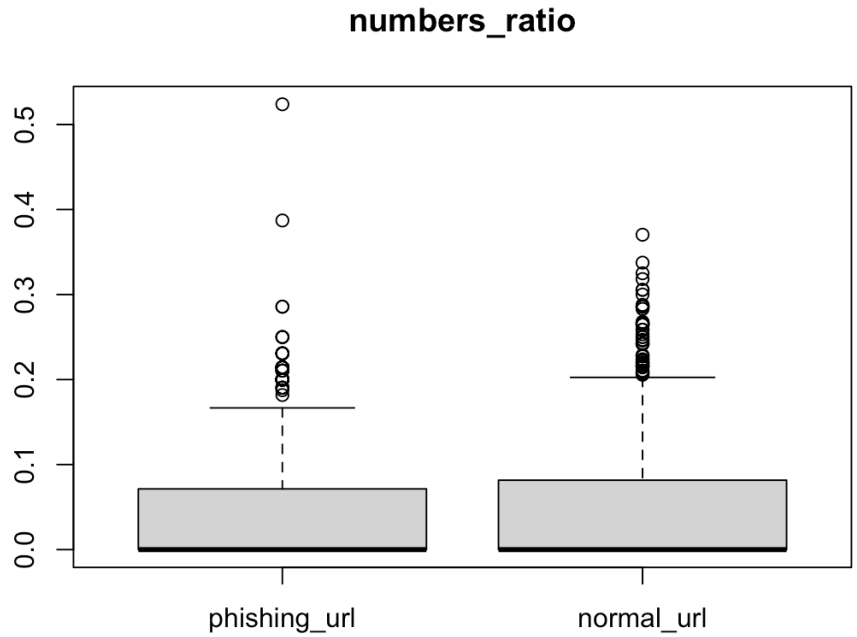
```
# 두 번째 CSV 파일의 링크들 불러오기
url_domain <- read.csv("C://data/url_domain.csv", header = TR

# 각 url 데이터와 정상 2000개 도메인 간의 최소 편집 거리 계산
url_domain$min_edit_distances <- sapply(url_domain$domain, fu
  calculate_min_similarity(url_domain, norm_2000_domain)
})

# 수정된 데이터프레임을 CSV 파일로 저장
write.csv(url_domain, file = "C://data/url_min_edit_distance.
```

▼ 숫자 비율

```
library(stringr)

number_ratio <- function(url) {
        length <- nchar(url)
        numbers <- str_count(url, "[0-9]")
        return(numbers/length)
}

phishing_data$numbers_ratio <- sapply(phishing_data$url, numb
normal_data$numbers_ratio <- sapply(normal_data$url, number_r
```

```
boxplot(phishing_data$numbers_ratio, normal_data$numbers_rati
```

<결과>

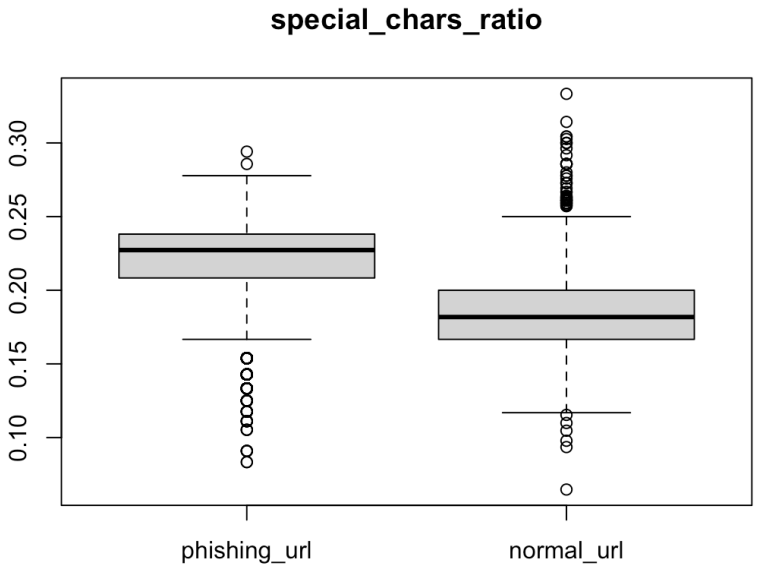


▼ 특수문자 비율

```
special_chars_ratio <- function(url) {
        length <- nchar(url)
        special_chars <- str_count(url, "[^A-Za-z0-9가-힣]")
        return(special_chars/length)
}

 phishing_data$special_chars_ratio <- sapply(phishing_data$ur
 normal_data$special_chars_ratio <- sapply(normal_data$url, s
```

```
boxplot(phishing_data$special_chars_ratio, normal_data$specia
```

<결과>




```
summary(phishing_data$special_chars_ratio)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
.08333 0.20833 0.22727 0.21313 0.23810 0.29412

summary(normal_data$special_chars_ratio)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
.06471 0.16667 0.18182 0.18514 0.20000 0.33333
```

▼ https 사용 여부

```
# 데이터 불러오기
setwd("C:/data_analysis/project")
data <- read.csv("phishing_data.csv")

# 프로토콜 여부 열 만들기
protocol <- as.data.frame(data$protocol)

# 프로토콜 여부 확인
protocol <- ifelse(grepl("^https", data$url), "0", "1")
```

```
# 결과 확인하기
print(protocol)
```

```
> count_of_0 <- sum(protocol == 0)
> print(count_of_0)
[1] 140
> count_of_1 <- sum(protocol == 1)
> print(count_of_1)
[1] 1846
```

→ “phishing_data” 를 사용했을 때의 결과

▼ 단축URL 서비스 사용 여부

```
library(urltools)
library(tidyverse)
library(openxlsx)

# 파일 불러오기
setwd("C:/data_analysis/project")
test_file <- "url2000.xlsx"
df_test <- read.xlsx(test_file)

# URL이 단축 서비스의 도메인을 포함하는지 여부를 확인하는 함수
is_shortened <- function(url) {
  domain <- urltools::domain(url)
  is_shortened <- domain %in% shortening_domains
  return(is_shortened)
}

# 단축 URL 서비스의 도메인 목록
shortening_domains <- c("bit.ly","kl.am","cli.gs","bc.vc","po
                        "adf.ly","x.co","1url.com","ad.vu","m
                        "shor7.com","yfrog.com","tinyurl.com'
                        "twitthis.com","buzurl.com","cur.lv",
                        "hurl.ws","om.ly","prettylinkpro.com'
```

```
                          "link.zip.net","doiop.com","url4.eu",
                          "go2l.ink","yourls.org","wp.me","goo
                          "u.bb","shorturl.at","han.gl","wo.gl"

# URL 컬럼에서 도메인을 추출하여 단축 URL 여부를 확인하고 새로운 열 추가
df_test <- df_test %>%
  mutate(shortened = sapply(url, is_shortened) %>% as.integer
  mutate(shortened = ifelse(shortened == 1, "1", "0"))  # 1을

# 결과 확인
print(df_test)
```

```
> zero_count <- sum(df_test$shortened == 0, na.rm = TRUE)
> sum(zero_count)
[1] 1828
> count <- sum(df_test$shortened == 1, na.rm = TRUE)
> sum(count)
[1] 158
```

→ “phishing_data”를 사용했을 때의 결과

▼ [도메인] 특수문자, 숫자, `.` , `-` 개수

- 도메인 추출 코드

```
extract_domain <- function(url) {
        # 프로토콜(https, http) 제거
        url <- gsub("^https?://", "", url)

        # www. 제거
        domain <- gsub("^(www\\.)?", "", url)

        # 다음 첫 슬래시(/) 나올 때까지 추출
        domain <- gsub("/.*", "", domain)
```

```
        return(domain)
}
```

- 개수 추출 코드

```
library(stringr)

# 특수문자 개수 추출
phishing_domain$special_chars <- str_count(phishing_domain

# 숫자 개수 추출
phishing_domain$numbers <- str_count(phishing_domain$domai

# 점(dot) 개수 추출
phishing_domain$dot_chars <- str_count(phishing_domain$dom

# -(dash) 개수 추출
phishing_domain$dash_chars <- str_count(phishing_domain$do

# 정상 데이터도 동일하게 진행
```

- 결과
  - 특수문자 개수

```
summary(phishing_domain$special_chars)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1.000   2.000   2.000   1.884   2.000   3.000
summary(normal_domain$special_chars)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1.000   1.000   1.000   1.359   2.000   5.000
```

피싱도메인 : 거의 대부분 2에 몰려있음
정상도메인 : 거의 대부분이 1~2에 있고, 4-5개 갖는 이상치 있음

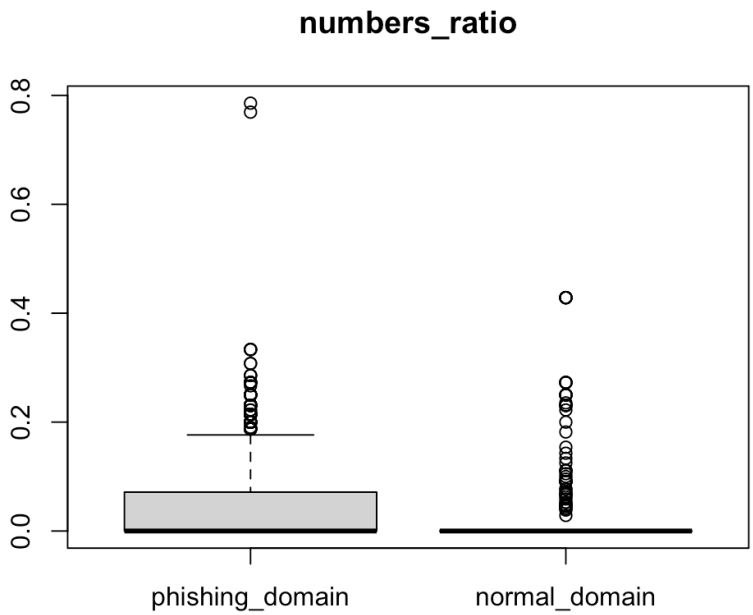
  - 숫자 개수

```
summary(phishing_domain$numbers)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.0000  0.0000  0.0000  0.6898  1.0000 11.0000
summary(normal_domain$numbers)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.0000  0.0000  0.0000  0.0625  0.0000  4.0000
```

피싱도메인 : 거의 대부분 0~1개. 이상치의 범위가 4~5개까진 정상도메인과 비슷하나, 10-11개까지 나오는 이상치 존재

정상도메인 : 거의 대부분 0개. 1-4개까지 있긴 함

- 혹시 몰라서 비율로도 통계 내 봤으나, 개수와 비슷한 양상을 보임



  - 점 개수

```
summary(phishing_domain$dot_chars)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1.000   2.000   2.000   1.877   2.000   3.000
summary(normal_domain$dot_chars)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1.000   1.000   1.000   1.286   2.000   4.000
```

피싱도메인 : 거의 2개에 몰려있음
정상도메인 : 1~2개에 몰려있는데, 1개인 곳이 더 많음.

- -(dash) 개수 → 피싱도메인에는 하나도 존재하지 않고, 정상도메인에는 111개 데이터가 최대 3개 가짐

▼ 확장자 여부

- url2000

```
install.packages("urltools")
library(readxl)
library(urltools)

df <- read_excel("url2000.xlsx")

has_extension <- function(url) {
   extensions <- c('.txt', '.exe', '.js')
   for (ext in extensions) {
     if (grepl(ext, url)) {
       return(1) # 확장자 포함
     }
   }
   return(0) # 미포함
 }


df$has_extension <- sapply(df$url, has_extension)

print(df)
```

```
> table(df$has_extension)

   0    1
1987   13
```

```
install.packages("urltools")
library(readxl)
library(urltools)

df <- read.csv("phishing_data.csv", fileEncoding = "euc-kr")
```

```
> table(df$has_extension)

   0    1
1980    6
```

▼ Path 깊이 → 일단 넣어 놓지만 추후 성능이 좋지 않을 경우 빼는 걸로

- url2000

```
install.packages("urltools")
library(readxl)
library(urltools)

df <- read_excel("url2000.xlsx")

# URL에서 경로 세그먼트 수를 세는 함수 정의
count_path_segments <- function(url) {
  parsed_url <- url_parse(url)
  path_segments <- strsplit(parsed_url$path, "/")[[1]]
  # 첫 번째 요소가 공백이면 무시하고 나머지 요소의 개수 반환
  if (length(path_segments) > 1 && path_segments[1] == "") {
    return(length(path_segments) - 1)
  } else {
    return(length(path_segments))
  }
```

```
}

# 각 URL의 경로 세그먼트 수 계산
segment_counts <- sapply(df$url, count_path_segments)

# 결과 데이터프레임에 추가
df$segment_count <- segment_counts

print(df)
```

```
> table(df$segment_count)

  1   2   3   4   5   6   7   8   9
745 588 359 187  84  25   6   4   2
```

```
> summary(df$segment_count)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   2.000   2.204   3.000   9.000
```

```
     1      2      3      4      5      6      7      8      9
0.3725 0.2940 0.1795 0.0935 0.0420 0.0125 0.0030 0.0020 0.0010
```

```
install.packages("urltools")
library(readxl)
library(urltools)

df <- read.csv("phishing_data.csv", fileEncoding = "euc-kr")

# URL에서 경로 세그먼트 수를 세는 함수 정의
count_path_segments <- function(url) {
  parsed_url <- url_parse(url)
  path_segments <- strsplit(parsed_url$path, "/")[[1]]
  # 첫 번째 요소가 공백이면 무시하고 나머지 요소의 개수 반환
  if (length(path_segments) > 1 && path_segments[1] == "") {
    return(length(path_segments) - 1)
  } else {
```

```
    return(length(path_segments))
  }
}

# 각 URL의 경로 세그먼트 수 계산
segment_counts <- sapply(df$url, count_path_segments)

# 결과 데이터프레임에 추가
df$segment_count <- segment_counts

print(df)

# segment_count 열의 빈도 계산
segment_count_frequency <- table(df$segment_count)

# 클래스별 비율
segment_count_ratio <- prop.table(segment_count_frequency)

print(segment_count_ratio)
```

```
> summary(df$segment_count)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   1.000   1.001   1.000   3.000
```

```
> table(df$segment_count)

   1    3
1985    1
```

```
> head(df)
                      url isphishing segment_count
1 http://masf.krhes.boston          Y             1
2   con08.nu8w.love 외11건          Y             1
3           coz.ul3u.show           Y             1
4   https://donate.do/h73t          Y             1
5          oabe.fyy8.plus           Y             1
6           mp.fnb4.media           Y             1
```

▼ 피싱 단어 빈도

```
library(stringr)

#데이터 불러오기
setwd("C:/data_analysis/project")
data <- read.csv("phishing_data.csv")

# URL 열 추출
phishing_urls <- data$url


# URL에서 단어 추출하는 함수
extract_words_from_url <- function(url) {

  # 제외할 문자열 패턴 설정(프로토콜, 단축 url 서비스)
  exclude_patterns <- c("https?", "http?","www",
                        "bit.ly","kl.am","cli.gs","bc.vc","po
                        "adf.ly","x.co","1url.com","ad.vu","m
                        "shor7.com","yfrog.com","tinyurl.com"
                        "twitthis.com","buzurl.com","cur.lv",
                        "hurl.ws","om.ly","prettylinkpro.com"
                        "link.zip.net","doiop.com","url4.eu",
                        "go2l.ink","yourls.org","wp.me","goo.
                        "u.bb","shorturl.at","han.gl","wo.gl"
                        "com","ly","gl","kr","org")

  # 추출한 URL 부분에서 단어 추출 (제외할 패턴을 포함하지 않는 단어만 추
  words <- unlist(str_extract_all(phishing_urls, paste0("\\b(

  return(words)
}

# 각 URL에서 단어 추출
all_words <- unlist(lapply(phishing_urls, extract_words_from_

# 단어 빈도 계산
```

```
word_freq <- table(all_words)

# 상위 10개 단어 추출
top_words <- head(sort(word_freq, decreasing = TRUE), 10)

print(top_words)
```

<피싱 사이트 결과>

```
> print(top_words)
all_words
  hair  miami   casa beauty   cash  quest    gay agency  media   skin
738792 285984 228390 141006 135048 121146 115188 111216 109230 107244
```

<정상 사이트 결과>

```
> print(top_words)
all_words
   html      en    wiki     htm     php wikipedia      ca    2011
 638000  250000  210000  208000  158000  152000  146000  144000
    net     the
 122000  120000
```

```
# 필요한 패키지 불러오기
library(readr)

#데이터 불러오기
setwd("C:/data_analysis/project")
data <- read.csv("phishing_data.csv")

# URL 열 추출
phishing_urls <- data$url

# 확인할 단어들 (피싱 사이트에서 빈출되는 단어)
phishing_words <- c("hair", "miami", "casa", "beauty", "cash'
# 확인할 단어들 (정상 사이트에서 빈출되는 단어)
# normal_words <- c("html", "en", "wiki", "htm", "php", "wiki
```

```
# 결과를 저장할 벡터 초기화
results <- logical(length(phishing_urls))

# 각 단어에 대해 반복문 실행
for (i in seq_along(phishing_words)) {
  results <- results | grepl(phishing_words[i], phishing_urls
}

# 결과를 "1"(피싱)과 "0"(정상)으로 변환
results <- as.integer(results)

# 결과 출력
print(results)
```

포함O → “1”, 포함X → “0”

피싱파일 ← 피싱 중복 단어

```
> sum(results == 0)
[1] 930
> sum(results == 1)
[1] 1056
```

피싱_파일 ← 정상 중복_단어

```
> sum(results == 1)
[1] 332
> sum(results == 0)
[1] 1654
```

정상 파일 ← 피싱 중복 단어

```
> sum(results == 1)
[1] 813
> sum(results == 0)
[1] 1187
```

정상 파일 ← 정상 중복 단어

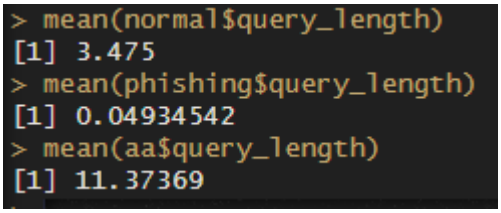
```
> sum(results == 0)
[1] 824
> sum(results == 1)
[1] 1176
```

▼ 쿼리 길이 구하기 (가지고 있는 피싱 url 쿼리가 너무 짧게 나와서 정상url 추출한 파일에서 피싱 url 2000개 추출하여 같이 진행)

<쿼리 길이 구하기>

```
library(stringr)

# 데이터 불러오기
normal <- read.csv("C://data_prac/normal_data.csv", header =

# URL에서 쿼리 부분 추출
query <- str_extract(normal$url, "\\?.*")

# 쿼리 부분의 길이 계산
query_length <- ifelse(is.na(query), 0, nchar(query))

# 결과 출력
print(paste("쿼리 길이:", query_length))

# normal 데이터프레임에 query_length 열 추가
normal$query_length <- query_length

# 수정된 데이터를 CSV 파일로 저장
write.csv(normal, file = "C://data_prac/normal_query.csv", ro



#------------------------------------------------------------
# 데이터 불러오기
phishing <- read.csv("C://data_prac/phishing_data.csv", fileE
head(phishing)
# URL에서 쿼리 부분 추출
query <- str_extract(phishing$url, "\\?.*")

# 쿼리 부분의 길이 계산
query_length <- ifelse(is.na(query), 0, nchar(query))

# 결과 출력
print(paste("쿼리 길이:", query_length))
```

```
# normal 데이터프레임에 query_length 열 추가
phishing$query_length<- query_length

# 수정된 데이터를 CSV 파일로 저장
write.csv(phishing, file = "C://data_prac/phishing_query.csv'



#-----------------------------------------------
aa <- read.csv("C://data_prac/aa.csv", header = TRUE)

# URL에서 쿼리 부분 추출
query <- str_extract(aa$url, "\\?.*")

# 쿼리 부분의 길이 계산
query_length <- ifelse(is.na(query), 0, nchar(query))

# 결과 출력
print(paste("쿼리 길이:", query_length))

# normal 데이터프레임에 query_length 열 추가
aa$query_length <- query_length

# 수정된 데이터를 CSV 파일로 저장
write.csv(aa, file = "C://data_prac/aa_query.csv", row.names
```

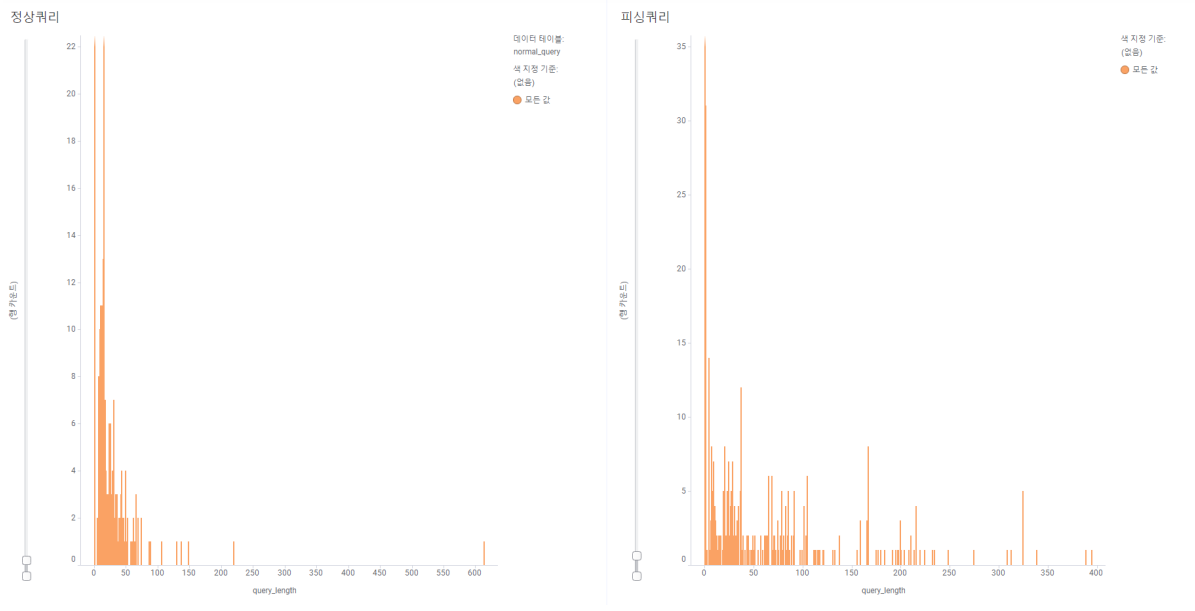


<0을 제외한 쿼리 길이 평균 구하기>

```
normal <- read.csv("C://data_prac/normal_query.csv", heade
head(normal)
# 0을 제외한 대문자 비율 추출
```

```
non_zero_query_length<- normal$query_length[normal$query_l

# 평균 계산
normal_non0mean_query_length<- mean(non_zero_query_length)

# 결과 출력
print(normal_non0mean_query_length)

#---------------------------------
phishing <- read.csv("C://data_prac/phishing_query.csv", h
head(phishing)

# 0을 제외한 대문자 비율 추출
non_zero_query_length<- phishing$query_length[phishing$que

# 평균 계산
phishing_non0mean_query_length<- mean(non_zero_query_lengt

# 결과 출력
print(phishing_non0mean_query_length)
#---------------------------------------
cc <- read.csv("C://data_prac/cc_query.csv", header = TRUE
head(cc)

# 0을 제외한 대문자 비율 추출
non_zero_query_length<- cc$query_length[cc$query_length !=

# 평균 계산
cc_non0mean_query_length<- mean(non_zero_query_length)

# 결과 출력
print(cc_non0mean_query_length)
```

```
> print(normal_non0mean_query_length)
[1] 27.2549
> # 결과 출력
> print(phishing_non0mean_query_length)
[1] 6.125
> # 결과 출력
> print(aa_non0mean_query_length)
[1] 69.31707
```



쿼리 길이의 평균은 정상/피싱/피싱 2000개 추출 순으로 3.5/0.05/11.37가 나왔다.(정상에 0이 많아서 평균 낮게 나왔을수도 있음) → 쿼리 길이가 0을 제외하고 다시 평균 구했더니 27/6/69 로 쿼리 길이는 유의미.

(상위가지: 0이다,0이아니다/하위가지 0이 아닐 때: 60이하면 정상, 60초과면 피싱의심)

▼ whois 등록 여부

```
import whois
import pandas as pd
# 1 : 정상 / -1 : 피싱
def Domain_registration_length(domain) :
    try :
        total_date = get_total_date(domain)
        if total_date is None or total_date <= 365 :
            return -1
        else :
```

```
            return 1
    except (whois.parser.PywhoisError, AttributeError) :
        return None  # 오류 발생 시 None 반환

# domain 유효 기간 계산
def get_total_date(domain) :
    domain_info = whois.whois(domain)
    if domain_info is None:
        return None

    expiration_date = domain_info.expiration_date
    updated_date = domain_info.updated_date

    if expiration_date is None or updated_date is None:
        return None

    if isinstance(expiration_date, list):
        expiration_date = expiration_date[0]
    if isinstance(updated_date, list):
        updated_date = updated_date[0]

    total_date = (expiration_date - updated_date).days
    return total_date
#-------------------------------------------------------
# CSV 파일 경로 설정
csv_file_path = "C://data_prac/bb.csv"

# CSV 파일 불러오기
data = pd.read_csv(csv_file_path, header=0)

# 각 도메인의 유효 기간 판별하여 새로운 열에 추가
data['Domain_registration_length'] = data['domain'].apply(Dom

# 새로운 CSV 파일로 저장
new_csv_file_path = "C://data_prac/bb_with_registration_statu
data.to_csv(new_csv_file_path, index=False)
```

```
print("새로운 CSV 파일이 생성되었습니다.")
```

참고자료

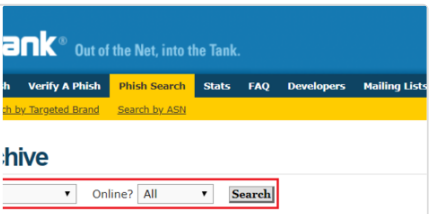
피싱 URL 속성 (by 혜린)

블로그 & 깃헙

피싱사이트, 정상사이트 데이터셋(Alexa, PhishTank, OpenPhish)

기존에 만들어진 데이터셋은 오래되었고, 최신의 데이터를 추가하기 어렵기에 우리 프로젝트만의 새로운 데...

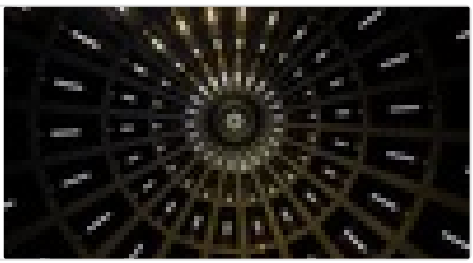
https://m.blog.naver.com/is_king/221540148072



Phishing Detection with Machine Learning

Explore and run machine learning code with Kaggle Notebooks | Using data from No attached data sources

https://www.kaggle.com/code/akritiupadhyayks/phishing-detection-with-machine-learning



[논문읽기] URL 주요특징을 고려한 악성URL 머신러닝 탐지모델 개발

URL 주요특징을 고려한 악성URL 머신러닝 탐지모델 개발 1. 서론 코로나 19를 상황을 이용한 악성, 피싱 URL 생성이 급증되었고, 이에 따라 악성 URL을 신속하게 탐지하고 대응 할 필요가 증가하였다. 위의 문제를 해결하기 위해 인

https://velog.io/@sojeong630/논문읽기-URL-주요특징을-고려한-악성URL-머신러닝-탐지모델-개발

논문

문자열 기반의 피싱 URL 탐지 방법 비교 (2018년 한국소프트웨어종합학술대회 논문집)

머신러닝을 활용한 피싱 사이트 탐지 방안 발표자료

IDF와 문자열 특징을 이용한 머신러닝 기반 악성 URL 탐지 (석사학위논문)