DATA SCIENCE AZ IT BIZTONSÁGBAN

2. forduló

SOPHOS

A kategória támogatója: SOPHOS

Ismertető a feladathoz

Útmutató:

- A **radio button-os kérdésekre** egy helyes válasz van.
- **Ha lejár a feladatlap ideje, a rendszer AUTOMATIKUSAN** beküldi azt az addig megjelölt válaszokkal.
- Az **adatbekérős feladatokra NEM jár részpontszám**, csak a feleletválasztósakra.
- **Badge-ket** a 4.forduló után kapsz majd először.
- Az **adatbekérős kérdéseknél** igyekeztünk minden variációt megadni (kisbetű, nagybetű, szóköz), de ha mégis eltérést tapasztalsz a megoldásokban, kérjük, jelezd felénk!

+1: Azért szólunk, hogy senkit ne a végén érjen meglepetés: a játék nem tipp-mix és csapatkategória sincs! Természetesen akinek nem inge...

Jó versenyzést kívánunk!

FIGYELEM!

A következőkben URL-ekkel fogunk dolgozni. Az adathalmaz különböző források egyesítésével készült:

- Malicious:

1. https://urlhaus.abuse.ch/
2. https://phishtank.org/

- Benign

1. https://data.mendeley.com/datasets/gdx3pkwp47/2

A malicious URL-ek nem szintetikusak, ténylegesen vírusokat és/vagy phishing site-ot hosztolnak.

A véletlen klikkelés és megnyitást elkerülendő, minden URL-ben a pontokat "." "[.]"-ra cseréltük.

<FIGYELEM vége>

Egy kollégánk felkeres minket; előfizetett egy új apple szolgáltatásra paypalon keresztül, egy barátja ajánlotta emailen keresztül, és minket se szeretne kihagyni.

Az alábbi URL-eket használta:

- http://1stopmoney[.]com/paypal-login-secure/websc[.]php
- http://31[.]14[.]136[.]202\secure[.]apple[.]id[.]login/Apple/login[.]php

Az első reakciónk után,

felhivjuk a kollégánk figyelmét, hogy nagy valószínűséggel phishing áldozatává vált.

Ha egy ember ránézésre meg tudja mondani egy URL-ről, hogy gyanús, hiszen hiába tartalmazza a paypal és apple szavakat, nem cégek regisztrált domainjen vannak, akkor egy ML algoritmus is lehetséges, hogy meg tud birkózni vele.

Az elkövetkezőkben URL klasszifikáló modelleket fogunk építeni, hátha fel tudjuk hívni kollégánk figyelmét legközelebb a lehetéges veszélyre - mielőtt rájuk klikkel.

A feladatokban több, gyanús URL-ek detektálására szolgáló machine learning (ML) modellt fogunk építeni.

A fordulóhoz szükséges adatokat az alábbi linken érhetjük el:

https://oitm-competition.s3.eu-west-2.amazonaws.com/round2/oitm_train_urls.csv

A probléma mélyebb leírása:

https://ai.sophos.com/2021/05/11/eli10-hunting-for-malicious-urls-with-character-level-embeddings-and-convolutions/

Egy logistic regression alapkonfigurációval kezdünk.

A kód futtatására alkalmas környezet legyen készenlétben:

Ajánljott környezet: Python3.6

- scikit-learn==0.24.2
- pandas==1.1.5

- numpy==1.19.5

A következő csomag importok előfeltételei a feladatsornak:

- import pandas as pd
- from sklearn.feature_extraction.text import TfidfVectorizer
- from sklearn.linear_model import LogisticRegression
- import numpy as np

Felhasznált idő: 00:00/40:00 **Elért pontszám:** 0/16

1. feladat 0/1 pont

Töltsük le a training adathalmazt és ismerkedjünk meg vele:

https://oitm-competition.s3.eu-west-2.amazonaws.com/round2/oitm_train_urls.csv

A csatolt training set hány URL-je hosztolja 2022 egyik legelterjedtebb malware-jét, az Emotet (aliasoktól most tekintsünk el)?

Válasz

A helyes válasz:

794

Magyarázat

```
import pandas as pd
df_train = pd.read_csv('oitm_train_urls.csv')
len(df_train[df_train['tags'].str.contains('emotet', case = False)])
```

2. feladat 0/1 pont

A szövegelemzés az ML algoritmusok egyik elterjedt alkalmazási területe. Azonban nyers szöveges adathalmazokat nem lehet közvetlenül feldolgozni ML modellekkel, mivel legtöbbjük meghatározott méretű numerikus adatot tartalmazó vektorokat vár, nem pedig változó hosszúságú szövegeket.

A probléma orvoslására már léteznek kész programcsomagok. A Scikit-learn egy hasznos ML modul, ami a legelterjedtebb megoldásokat implementálja. A következőkben ezzel a könyvtárral fogunk dolgozni:

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Vektorizálásnak nevezzük azt az általános folyamatot, amelynek során szöveges dokumentumok gyűjteményét numerikus feature vektorokká alakítjuk.

ML tanítása során legalább két adathalmazzal dolgozunk: train és test.

Egy realisztikus szimuláció esetén, algoritmustól függetlenül, melyik a megfelelő módja a vektorizáló "fittelésének"?

Válasz

- [x] fit(training_data)
  Ez a válasz helyes, de nem jelölted meg.
- [ ] fit(test_data)
- [ ] fit(training_data && test_data)

Magyarázat

Tanítási fázisban nem használhatunk ki semmilyen ismeretet a teszthalmazról.

3. feladat 0/2 pont

A következő sklearn.feature_extraction.text TfidfVectorizer beállítással fitteljük az adatunkat:

```
word_vectorizer = TfidfVectorizer(
  strip_accents='unicode',
  analyzer='word',
  token_pattern=r'\w{1,}',
  stop_words='english',
  max_features = 5000
)
```

A tokenizer ezzel a beállitással, első naív megközelítésként, az angol szavakat fogja elkódolni (memória takarékosság céljából csak 5000-et).

Nézzük a következő minta URL-t:

'scikit-learn[.]org/stable/modules/generated/sklearn[.]feature_extraction[.]text[.]TfidfVectorizer[.]html'

Tokenizáljuk az URL-t az újonnan fittelt word analyzer tokenizálónkkal.

Ha újra konstruáljuk az URL-t a megtartott tokenekből, hány százalékát tudjuk lefedni az eredeti URL-nek (rekonstruált URL hossza / eredeti URL hossza, két tizedes jegyig)?

Válaszok

A helyes válasz:

23

.23

0.23

Magyarázat

```
df_train = pd.read_csv('../oitm_train_urls.csv')

url = 'scikit-learn[.]org/stable/modules/generated/sklearn[.]feature_extraction[.]text[.]TfidfVectorizer[.]ht


word_vectorizer.fit(df_train['url'])
train_word_features = word_vectorizer.transform(df_train['url'])
```

```
def get_tokens(vectorizer, sample):
  vectorized = vectorizer.transform([sample]).toarray()[0]
  feature_names = vectorizer.get_feature_names()

  return [feature_names[i] for i in np.nonzero(vectorized)[0]]

def remove_tokens(tokens, sample):
    for token in tokens:
        sample = sample.replace(token, '')

    return sample

tokens = get_tokens(word_vectorizer, url)
leftover = remove_tokens(tokens, url)

(len(url) - len(leftover)) / len(url)
0.23076923076923078
```

4. feladat 0/2 pont

Ismételten fitteljük az adatunkat, ezúttal egy másik beállítással:

```
ngram_vectorizer = TfidfVectorizer(
  strip_accents='unicode',
  analyzer='char',
  ngram_range=(3, 3),
  max_features = 5000
)
```

A tokenizer ezzel a beállitással az összes 3-gram-ot fogja elkódolni (memória takarékosság céljából csak 5000-et).

Nézzük ismét a minta URL-t:

'scikit-learn[.]org/stable/modules/generated/sklearn[.]feature_extraction[.]text[.]TfidfVectorizer[.]html'

Tokenizáljuk az URL-t az újonnan fittelt word analyzer tokenizálónkkal.

Ha újra konstruáljuk az URL-t a megtartott tokenekből, mekkora részét tudjuk lefedni az eredeti URL-nek (rekonstruált URL hossza / eredeti URL hossza, két tizedesjegyig)?

Válaszok

A helyes válasz:

75

.75

0.75

Magyarázat

```
df_train = pd.read_csv('../oitm_train_urls.csv')

ngram_vectorizer.fit(df_train['url'])


def get_tokens(vectorizer, sample):
  vectorized = vectorizer.transform([sample]).toarray()[0]
  feature_names = vectorizer.get_feature_names()

  return [feature_names[i] for i in np.nonzero(vectorized)[0]]

def remove_tokens(tokens, sample):
    for token in tokens:
        sample = sample.replace(token, '')

    return sample

url = 'scikit-learn[.]org/stable/modules/generated/sklearn[.]feature_extraction[.]text[.]TfidfVectorizer[.]ht

tokens = get_tokens(ngram_vectorizer, url)
leftover = remove_tokens(tokens, url)

(len(url) - len(leftover)) / len(url)
```

5. feladat 0/0 pont

Logistic regression esetén mondhatjuk-e, hogy ha egy tanult együttható (coef) súlya nagyobb, mint egy másiké, akkor az a változó a fontosabb szerepet játszik az eredményben?

Válasz

- (●) Igen
  Ez a válasz helyes, de nem jelölted meg.
- ( ) Nem

Magyarázat

Kedves Versenyzők!

A kérdést 0 pontosra állítottuk, mivel értelmezés függvényében az igen és a nem mellett is lehet érvelni:

az együtthatók akkor összehasonlíthatóak hatás szempontjából, ha a magyarázó változók ugyanazon a skálán vannak. Például, ha az adatgeneráló folyamatot/valóságot a logit(esemény) = konstans + 1 * magasság méterben + 2 * szélesség méterben formula írja le - ahol az együtthatók alapján a szélesség az erősebb magyarázó változó -, és a becslés során a szélességet centiméterben adjuk meg, akkor annak az együtthatója átskálázódik, és 2 helyett 0.02 lesz. Ekkor, habár továbbra is a szélesség a domináns változó, pusztán az együttható alapján nem ez adódik.

Köszönjük a megértést!

https://towardsdatascience.com/a-simple-interpretation-of-logistic-regression-coefficients-e3a40a62e8cf

6. feladat 0/3 pont

Fitteljünk egy Logistic regression modellt a word analyzer által tokenizált szavainkra a következő módon:

```
from sklearn.linear_model import LogisticRegression
lr_word = LogisticRegression(penalty='l2')
lr_word.fit(train_word_features, df_train['is_malware'])
```

A fenti módszerrel trainelt modell számára melyik lesz a legártalmatlanabb szó? Más szavakkal a legfontosabb benign feature?

Válaszok

A helyes válasz:

edu

.edu

[.]edu

Magyarázat

```
def get_feature_importances(lr, vectorizer, ascending =False):
  importance_df = pd.DataFrame()
  feature_names = vectorizer.get_feature_names()
  coefs = lr.coef_[0]

  importance_df['feature_names'] = feature_names
  importance_df['coefs'] = coefs
  importance_df = importance_df.sort_values(by='coefs', ascending = ascending).head(25)

  return importance_df

get_feature_importances(lr_word, word_vectorizer, ascending = True).head(10)
```

7. feladat 0/3 pont

Fitteljünk egy Logistic regression modellt a word analyzer által tokenizált szavainkra a következő módon:

```
from sklearn.linear_model import LogisticRegression
lr_ngram = LogisticRegression(penalty='l2')
lr_ngram.fit(train_ngram_features, df_train['is_malware'])
```

Mi a három legfőbb indikátora a rosszindulatúságnak (maliciousness) az n-gram modellben?

Válaszok

A helyes válasz:

]ex,]cn, co/

co/,]cn,]ex

]cn,]ex, co/

]ex]cn co/

co/]cn]ex

co/,]cn,]ex

]ex,]cn,co/

Magyarázat

```
def get_feature_importances(lr, vectorizer, ascending =False):
  importance_df = pd.DataFrame()
  feature_names = vectorizer.get_feature_names()
  coefs = lr.coef_[0]

  importance_df['feature_names'] = feature_names
  importance_df['coefs'] = coefs
  importance_df = importance_df.sort_values(by='coefs', ascending = ascending).head(3)

  return importance_df


get_feature_importances(lr_ngram, ngram_vectorizer)
```

8. feladat 0/4 pont

Melyek a lehetséges következtetések, amiket a "]cn" token feature importance értékéből vonhatunk le?

Válaszok

- [x] A benign és malware minták különböző forrásokból származnak
  Ez a válasz helyes, de nem jelölted meg.
- [x] Egy regionálisan elfogult detektort építettünk
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Nem traineltük a modellt elég sok iteráción keresztül, hogy minden TLD megfelelő fontossággal tudjon hozzájárulni az eredményhez
- [ ] Concept drift problémába futottunk bele

Magyarázat

A "]cn" token a .cn top level domaint jelenti. Másképpen, a modellnek minden Kínában hostolt URL gyanús lesz. A benign része az adathalmaznak nem fedett le kínai site-okat.

Másképpen: Sampling bias

KÉSZÍTETTE cone

Megjelenés

Világos