DATA SCIENCE AZ IT BIZTONSÁGBAN

3. forduló

SOPHOS

A kategória támogatója: SOPHOS

Ismertető a feladathoz

A 3.forduló feladatait a hosszú hétvége miatt kivételesen szerda (11.02.) éjfélig tudod megoldani!

Érdemes ebben a fordulóban is játszanod, mert a következő forduló kezdetekor, 11.03-án 18 órától kiosztjuk az 1.-2.-3. fordulóban megszerzett badgeket!

A verseny közben az alábbi teljesítményeket díjazzuk:

- fordulógyőztes
- átlagnál jobb időeredmény
- átlag feletti pontszám
- hibátlan forduló

Szeretnénk rá felhívni figyelmedet, hogy az egyszer megkapott badge-eket nem vonjuk vissza, akkor sem, ha esetleg az adott fordulóban a visszajelzések alapján változások vannak.

Jó játékot!

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>

Az előző fordulóban két tokenizációs megközelítéssel tanítottunk egy-egy Logistic regression modellt, hogy el tudjuk dönteni URL-ekről, hogy gyanúsak-e, vagy sem.

Az töltsük le előző fordulóban megismert tanító adathoz tartozó teszt adatot az alábbi linkről:

- https://oitm-competition.s3.eu-west-2.amazonaws.com/round3/oitm_test_urls.csv

Az előző fordulóban tanított tokenizerek és modellek elérhetőek az alábbi URL-eken:

- https://oitm-competition.s3.eu-west-2.amazonaws.com/round3/ngram_vectorizer.pkl
- https://oitm-competition.s3.eu-west-2.amazonaws.com/round3/word_vectorizer.pkl

- https://oitm-competition.s3.eu-west-2.amazonaws.com/round3/lr_ngram.pkl
- https://oitm-competition.s3.eu-west-2.amazonaws.com/round3/lr_word.pkl

Ebben a fordulóban megnézzük, hogy melyik megközelités volt hatékonyabb.

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

1. feladat 0/1 pont

Mi a definíciója a True Positive Rate-nek (TPR)?

Válasz

- ○ TP/TN+FN
- ○ TP/TP+FP
- ● TP/TP+FN
  Ez a válasz helyes, de nem jelölted meg.
- ○ TN/TP+FN

Magyarázat

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

2. feladat 0/2 pont

A valóságban (jó esetben) sokkal kevesebb káros minta áll rendelkezésünkre, mint ártalmatlan. Az alábbi metrikák közül melyiket érdemes kerülni leginkább kiegyensúlyozatlan adathalmaz esetén?

Válasz

- [x] Accuracy
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Precision
- [ ] AUC
- [ ] F1 Score

Magyarázat

Példa: ha van 100 URL-ünk, amiből 99 benign, 1 káros, akkor az azonosan hamis függvénynek az accuracy-ja 99%

https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distributions/

3. feladat 0/1 pont

Az alábbiak közül melyek igazak a ROC görbe alatti területre (ROC AUC)?

Válaszok

- [x] Az AUC a görbe alatti területet méri és a True Positive Rate-et, valamint a False Positive Rate-et mutatja meg, különböző valószínűség küszöbértékekkel
  Ez a válasz helyes, de nem jelölted meg.
- [x] A tökéletes modell területe 1.0
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Az a modell, amelyik random eredményeket ad, 0-ás ROC görbe alatti területtel fog rendelkezni
- [ ] Az azonosan igaz modell, 0-ás ROC görbe alatti területtel fog rendelkezni

Magyarázat

https://en.wikipedia.org/wiki/Receiver_operating_characteristic

4. feladat 0/3 pont

Az előző fordulóban két tokenizációs módszerrel tanitottunk egy-egy Logistic regression modellt.

Mi a ROC AUC értéke a jobban teljesítő modellnek, három tizedesjegyig?

Válaszok

A helyes válasz:

0.988

.988

98.8%

Magyarázat

```
lr_ngram = pickle.load(open('lr_ngram.pkl', 'rb'))
lr_word = pickle.load(open('lr_word.pkl', 'rb'))


ngram_vectorizer = pickle.load(open('ngram_vectorizer.pkl', 'rb'))
word_vectorizer = pickle.load(open('word_vectorizer.pkl', 'rb'))


df_test = pd.read_csv('oitm_test_urls.csv')


test_ngram_features = ngram_vectorizer.transform(df_test['url'])
test_word_features = word_vectorizer.transform(df_test['url'])


preds_ngram = lr_ngram.predict_proba(test_ngram_features)[:, 1]
preds_word = lr_word.predict_proba(test_word_features)[:, 1]


df_test['preds_ngram'] = preds_ngram
df_test['preds_word'] = preds_word

for y_pred, y_true, legend_label in [(preds_ngram, df_test['is_malware'], 'ngram'), (preds_word, df_test['is_

    fpr, tpr, _ = metrics.roc_curve(y_true, y_pred)
    auc = metrics.roc_auc_score(y_true, y_pred)


    print(f'{legend_label}: {auc}')
```

5. feladat 0/3 pont

Mi az n-gram Model True Positive Rate-je 10^-3-nál, három tizedesjegyig?

Válaszok

A helyes válasz:

0.825

.825

82.5%

1.000

Magyarázat

```
fpr, tpr, thresholds = metrics.roc_curve(df_test['is_malware'], preds_ngram)


def tprs_at_fprs(fpr_desired, fpr, tpr):
    tpr_desired = np.zeros(len(fpr_desired))
    hi = len(fpr)

    for pos, val in enumerate(fpr_desired):
        idx = bisect_right(fpr, val, 0, hi)
        idx = (idx if idx != hi else -1) - 1
        if idx >= 0:
            tpr_desired[pos] = tpr[idx]

    return tpr_desired


tprs_at_fprs([0.001], fpr, tpr)
```

A feladatot lehetett úgy is értelmezni, hogy a threshold legyen 1e-3.

Így threshold=1e-3 esetén 3 tizedesjegyre kerekítve 1.000 jön ki a TPR-re:

```
y_predicted = y_score >= 1e-3
 cm = sklearn.metrics.confusion_
matrix(y_true, y_predicted)
 tn, fp, fn, tp = cm.ravel()
 tpr = tp/(tp+fn)
 print(tpr)
```

6. feladat 0/3 pont

Mi a 10^-3 False Positive Rate-hez threshold határérték az n-gram modellnél, három tizedesjegyig?

Válaszok

A helyes válasz:

0.593

.593

59.3%

Magyarázat

```
fpr, tpr, thresholds = metrics.roc_curve(df_test['is_malware'], preds_ngram)




def thresholds_at_fprs(fpr_desired, fpr, thresholds):
```

```
    thresholds_desired = np.zeros(len(fpr_desired))
    hi = len(fpr)
    for pos, val in enumerate(fpr_desired):
        idx = bisect_right(fpr, val, 0, hi)
        idx = (idx if idx != hi else -1) - 1
        if idx >= 0:
            thresholds_desired[pos] = thresholds[idx]
    return thresholds_desired


thresholds_at_fprs([0.001], fpr, thresholds)
```

7. feladat 0/1 pont

Amennyiben úgy döntöttünk, hogy 10^-4 False Positive Rate-et szeretnénk megengedni a modelljeinknek a 'rottentomatoes[.]com/uwbrvz8quj/a[.]exe' URL-t mindkét modell pozitívként jelzi, az 'exe' és a tanító halmazban gyakran malicious mintákban előforduló 'uwbrvz8quj' string miatt. Hogyha módosítjuk kicsit az URL-t, a következőre: 'rottentomatoes[.]com/uwbrvz8quk/a[.]exe', melyik modell jelezne be az alábbi URL-re, hogy gyanús, továbbra is rögzített 10^-4 FPR mellett?

Válasz

- (●) N-gram
  Ez a válasz helyes, de nem jelölted meg.
- ( ) Word
- ( ) Egyik sem

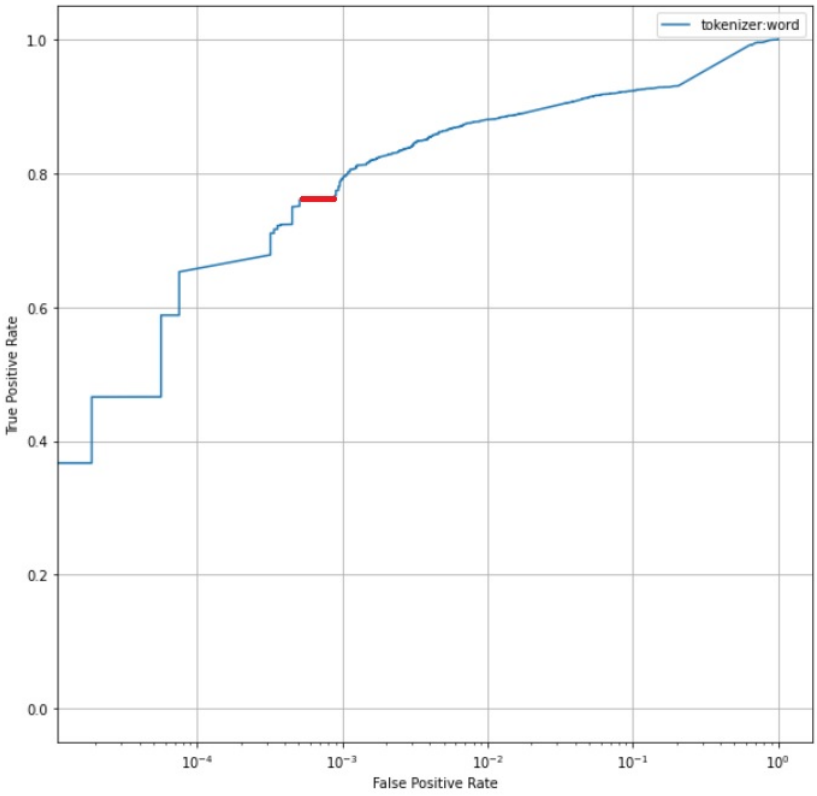
Magyarázat

```
fpr, tpr, thresholds = metrics.roc_curve(df_test['is_malware'], preds_ngram)

print(f"Ngram Treshhold: {thresholds_at_fprs([0.0001], fpr, tpr)[0]}")
print(f"Ngram pred {lr_ngram.predict_proba(ngram_vectorizer.transform(['rottentomatoes[.]com/uwbrvz8quk/a[.]e


fpr, tpr, thresholds = metrics.roc_curve(df_test['is_malware'], preds_word)

print(f"Word Treshhold: {thresholds_at_fprs([0.0001], fpr, tpr)[0]}")
print(f"Word pred {lr_word.predict_proba(word_vectorizer.transform(['rottentomatoes[.]com/uwbrvz8quk/a[.]exe'
```

Az Ngram tokenizációs módszer ellenálóbbnak bizonyul a betűcsere ellen, amit akár tekinthetünk egy egyszerű "obfuszkációs" kisértletnek.

8. feladat 0/2 pont

Melyik token felelős az alábbi ROC görbén pirossal jelölt szakaszon (fpr 0.0005 és 0.0009) található fals pozitív minták edményéért leginkább a word tokenizer által adott tokeneken tanított logistic regressionnel?



Válaszok

A helyes válasz:

cn

.cn

[.]cn

Magyarázat

```
df_test[(df_test['preds_word'] < thresholds_at_fprs([0.0005], fpr, thresholds)[0]) & (df_test['preds_word'] >
```