MWS Cup 2022 課題3 解説

株式会社ＦＦＲＩセキュリティ
https://www.ffri.jp

作問メンバーについて

紹介

今年の作問体制

今年は以下のメンバーにご協力いただきました（敬称略）

所属・氏名		役割
株式会社ＦＦＲＩセキュリティ	茂木裕貴	作問チームリーダー
	中川恒	MWS Cup副委員長
LINE株式会社	愛甲健二	作問委員
	小野颯真	作問委員
	草間好輝	作問委員
	華徳凱	作問委員

今年の問題について

背景

今年の問題の方針

今年の問題はKaggle上でStringsを用いたマルウェアの検知のみに絞った

	2018 - 2021	2022
データの種類	FFRI Datasetの形式全項目	Stringsのみ
プラットフォーム	独自	Kaggle
問題	表層解析問題と分類問題	分類問題のみ

Late Sub できるので 復習にどうぞ

今年のテーマ

以下の理由からStringsに絞った検知にした

特徴	概要
試行錯誤しやすい	特徴量を絞ることで方向性が定まる NLPの各種手法を試せる
データ量を増やす	データサイズの削減により検体の件数 を昨年の約2倍に
論文に基づく設定	Stringsを用いた検知は複数の論文が 存在する

MWS Cup 2022 課題3

問題

データの形式

ID, Strings, Labelの3つ

	ID	Strings	Label
train.csv	0~1999	(例) !This program cannot be run in DOS mode.¥nRich-¥n.text …	0/1
test.csv	0~4999	(例) !This program cannot be run in DOS mode.¥nRich-¥n.text …	

ここを予測

未来のデータでの評価

いわゆるConcept Driftへの対処を狙った

背景

- 機械学習によるマルウェア検知は過去の検体を学習し、未来の検体を検知するという時系列性がある
- 特にマルウェアは検知回避等のため「進化」していくことから、時間とともに検知モデルの検知精度が下がっていくことが想定される

制約

- エンドポイントのアンチウイルスソフトをWebサービスのように1日10回アップデートするのは極めて極めて困難である
- 過去のモデルで数年後の未来のマルウェアを検知できれば理想的である

上記を踏まえた問題設定

- 2020年のデータで訓練し、2022年のデータを予測
- FFRI Dataset 2022などを用いれば有利なため、外部データの使用は原則禁止とした

課題3 解説

解説と議論

ヒント・模範解答についての注意

他の問題と違ってDSコンペに「解答」はない

模範解答も普通はない（KaggleにせよProbSpaceにせよ普通ホストは出さない）

今回のヒントや模範解答はあくまで1つのやり方を提示しているに過ぎない

サンプルやヒントは一切手が付けられないことのないように出しているだけで、この方法に誘導したいわけでは一切ないし、模範解答のやり方が「正しい」訳ではない

参加者にはより良い作問のためWrite-upにご協力いただきたいです（本コンペのDiscussionやNotebookの公開・提出）

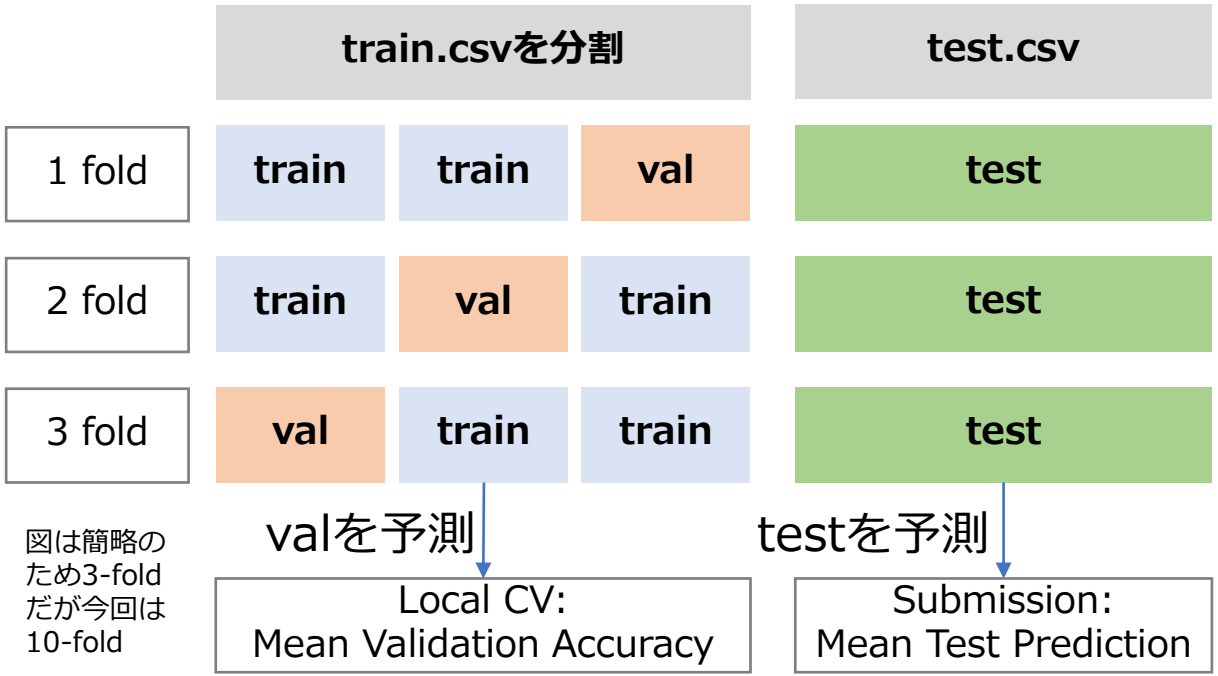
サンプルの回答の解説 (PB 0.79)

Fexrdで使用されている特徴量を用いた

	Fexrdについての説明
概要	FFRI Datasetと同形式のデータから特徴量を作成するライブラリ
URL	https://github.com/FFRI/FEXRD
文字列から使用している特徴量	ヒストグラム エントロピー （改行で区切ったときの）平均長 Registryの個数 URLの個数 パス（Cドライブ）の個数

CVとアンサンブル (PB 0.79 => 0.80)

各Foldで作成したモデルでアンサンブルする（平均を取るだけ）
Cross Validationする以上Freeで計算できる
Kaggle等ではよくやられている（以降の数値はこれを前提にする）



図は簡略のため3-foldだが今回は10-fold

ヒントの解説：Packer (PB 0.80 => 0.836)

「Packerの数」を数えるのが比較的効いた

検証時は以下の3つしか見なかった（それでも効いたため）が、下の記事にあるようなPackerを全部試してみる&判定方法を改善すると良い

```
def has_packer(st):
    return (
        sum("UPX" in s for s in st.split("\n"))
        + sum("vmprotect" in s.lower() for s in st.split("\n"))
        + sum("aspack" in s.lower() for s in st.split("\n"))
    )
```

cf. https://qiita.com/y_oyama/items/d0899e3e4dc88a83aaef

ヒントの解説：ヒント1~3 (PB 0.836=>0.843)

ヒント1~3の統計情報もまとめて「多少」効いた
コードは模範解答をご覧ください！

ヒント	解説
アルファベットの割合	文字.isalpha()で文字がアルファベットか判定できるため、これを用いて計算できる
Punctuationの割合	string.punctuationでpunctuationのリスト（正確には文字列）が得られるため、これを用いて計算できる
「単語」の平均長さ	文字列.split()で「単語」のリストが得られる（Fexrdの方は改行でsplitしているため、「文」の平均長）ため、これを用いて計算できる

Bonus：stringsifter

ヒント1~3は全てstringsifterで使われているもの

今回は追加ファイルなく気軽に使え（て効く）るものをヒントに出した

	stringsifterについての説明
概要	検体から抽出した文字列をマルウェア解析に関係するかランク付けするツール
URL	https://github.com/mandiant/stringsifter
使用している特徴量	前頁の3つ keyloggerっぽい文字列があるか Hiveっぽい文字列があるか URLっぽい文字列があるか etc…

TFIDF (PB 0.843 => 0.90)

（検証では）最も効いた。そもそも単体でTest Acc 0.88程度行く
Kaggle Notebookで（メモリが）ギリギリ動く

```
all_f = []
for d in notebook.tqdm(df_train["Strings"].values):
    all_f.append(" ".join(d.split("\n")))
for d in notebook.tqdm(df_test["Strings"].values):
    all_f.append(" ".join(d.split("\n")))
vectorizer = TfidfVectorizer()
vec_tfidf = vectorizer.fit_transform(all_f)
```

TFIDFの話

今回の模範解答ではtrain+testでTFIDFしている

一般にある統計情報を計算するのにtrain+testで計算するかどうかというのは議論のあるトピックである（いわゆるleakageとの関連、コンペなのか実務なのか論文なのか…）

本資料では深入りしませんので例えば以下を参考にしてください

門脇大輔、阪田隆司、保坂桂佑、平松雄司、『Kaggleで勝つデータ分析の技術』、技術評論社、2019年 初版第2刷p124~p126のColumn

Bonus : Character-based CNN

Character-based CNNでもTest Acc 0.85~0.86程度出た
（そのときは
https://github.com/uvipen/Character-level-cnn-pytorch
をベースに実装した）
アンサンブルに使うと効くのではないか

Kaggle Notebookで動くか・精度を出しつつ動くように調整できるかは未検証

ちなみにtransformer/LSTM-CNNはあまり良い数値は出ませんでしたが、これもチューニング次第かもしれないので興味あれば試してみてください

Towards：次回以降（サステナビリティ）

作問にご協力いただける方を募集しています！！！

	2018 - 2021	2022
分類問題のサステナビリティ	• Fexrdの出力を食わせるだけで高精度 • 次回は模範解答からスタートされうる	• Stringsのみを用いた • trainデータとtestデータの時間をずらした

来年以降は未定

問題を「解く」ではなく「作る側」

- DSコンペに参加者ではなく作問側として参加する機会は希少
- より良い問題、より良い模範解答ができればMWSコミュニティそしてサイバーセキュリティ業界全体にプラス