MWS Cup 2023
マルウェア分類解説

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
LINEヤフー 株式会社	愛甲健二	作問委員
	草間好輝	作問委員
早稲田大学	杉山孔亮	作問委員

今年の問題について

背景

今年の問題の方針

今年の問題はマルウェアの分類問題とした

	2022	2023
データの種類	Stringsのみ	FFRI Datasetの形式全項目
プラットフォーム	Kaggle（1チーム1人）	Kaggle（制限なし）
問題	マルウェア検知	マルウェア分類

Late Sub できるので 復習にどうぞ

今年のテーマ

以下の理由から半教師あり設定で、かつ「その他」のラベルを含むマルウェア分類にした

特徴	概要
検知ではなく分類	マルウェア検知以外のタスク
半教師あり	マルウェアのファミリのアノテーションは難しく、ラベルのないマルウェアの方が多い
「その他」ラベル	マルウェアは常に進化している

MWS Cup 2023
マルウェア分類

問題

データの形式

FFRI Datasetの形式とid, hashes, date, labelが異なる

	ID	HASHES	DATE	Label
train_labeled.jsonl	0~1504	md5, sha1, sha256 なし (null)	なし (null)	マルウェアの種類
train_unlabeled.jsonl	1505~2754			なし (null)
test.jsonl	2755~4004			

ここを予測

ファミリラベルに関する注意

今回ファミリラベルはMalware BazaarのSignatureをそのまま使用した

本来このような自動付与されるラベルはあまり評価には適していない

ラベルの正しさが問題

正しい場合、機械学習などせずにそれを使えばよい

正しくない場合、マルウェア分類ではなくSignature予測

ハンドラベリングを行って評価するべきだが、それ自体かなり困難

当初はベンダの解析レポートなどを活用しようとしたが、訓練データとのデータセットシフトが大きく問題として成立しなかった

これらの経緯からSignatureを使用しているため、本Cupの手法を元に論文等に応用する場合は細心の注意が必要である

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）

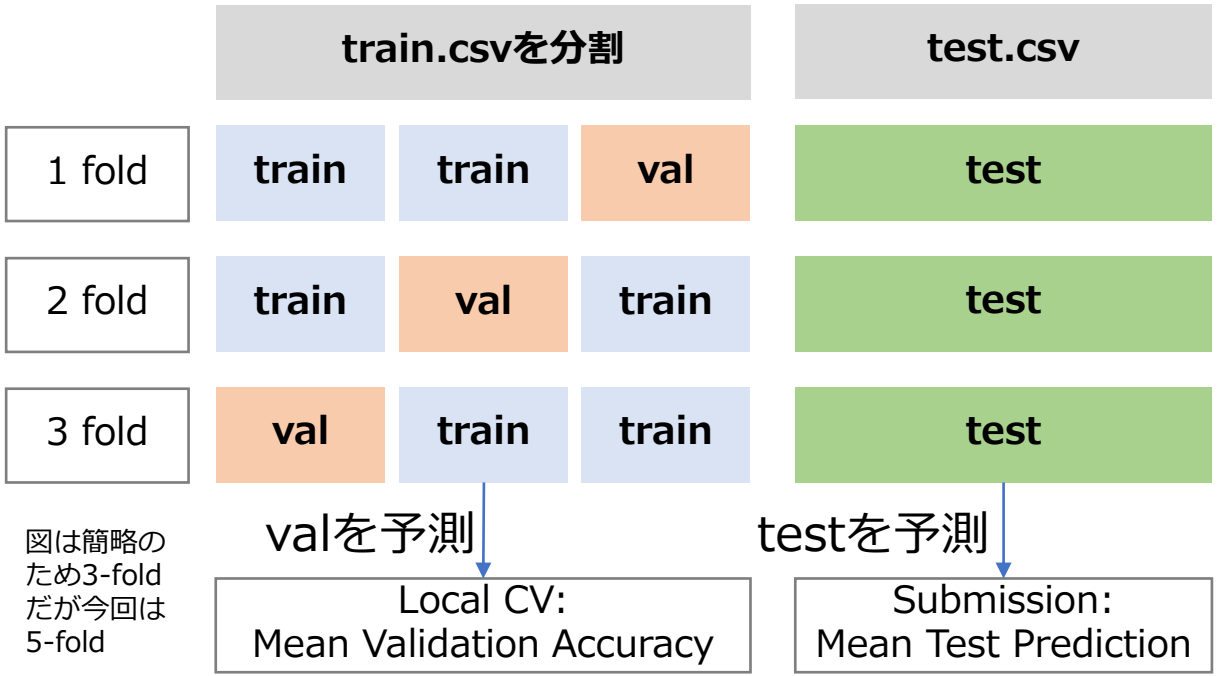
サンプルの回答の解説 (PB 0.568)

ラベル付きデータのみをFexrdを用いて分類

	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
使用している特徴量	LIEFやManalyzeなど使用したツールごとに数値をそのまま使用/Label Encoding/Feature Hashing 等の変換を行っている

CVとアンサンブル

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



CVとアンサンブル (PB 0.568 => 0.520!)

実はアンサンブルをやるとスコアが下がる

今回は「その他」のラベルの付いたデータが5つしかなく、5-foldでは各Foldの分類器があまり強くないためと考えられる

Local CVの数値だけ評価のために記録しておき、全データで訓練したモデルをSubmitする方法もある

この資料ではアンサンブルモデルを使用する。理由は後述

ヒントの解説：Balanced Weight (PB 0.568 => 0.611)

ラベル付きデータをラベル毎に見ると、「その他」のデータが極端に少ないため、何らかの調整が必要そう

最も簡単な策はWeightの調整

まだラベル付き訓練データを全て使った単体のモデルの方がCV時の各Foldの分類器の平均よりPBは高いが、差は小さく、LBは逆転している

Weightの調整により各Foldの分類器の性能が十分高くなった

以降この資料ではアンサンブルモデルの方を前提とする

```
# Weightを調整
train_weight = compute_sample_weight(class_weight='balanced',
y=y_train).astype('float32')
lgb_train = lgb.Dataset(X_train, y_train, weight=train_weight)
```

ヒントの解説：MaxProbによるOOD検知 (PB 0.568 => 0.614)

5クラス分類器を作成したうえで、予測クラスのスコアがしきい値を超えない場合に「その他」扱いをする

こちらの方が若干（CV時の各Foldの分類器の平均で比べると）Weightを調整した場合よりPBが高い

```
# Weightはそのまま
lgb_train = lgb.Dataset(X_train, y_train)
# 5クラス分類
lgbm_param = {"objective": "multiclass", "verbose": -1, "num_class": 5, "metric": "multi_logloss"}
model = lgb.train(lgbm_param, lgb_train, valid_sets=lgb_val, verbose_eval=False)
predict = predict_with_best_iteration(model)
y_pred_val = predict(X_val)
y_pred_val_class = np.argmax(y_pred_val, axis=1)
y_pred_test_max = np.max(y_pred_val, axis=1)
# 予測スコアがしきい値より小さい場合は「その他」と判定
for i in range(len(y_pred_val)):
    if y_pred_test_max[i] < 0.85:
        y_pred_val_class[i] = 5
```

ヒントの解説：Pseudo Labeling (PB 0.614 => 0.645)

ラベルの予測値を使用してモデルを訓練する

```
# ラベル無しデータおよびテストデータの予測値をラベルとして訓練
X_train_with_un_test = np.vstack([X_train, X_un, X_lb[lb_idx], X_pb[pb_idx]])
y_train_with_un_test = np.vstack([
  y_train.reshape(y_train.shape[0], 1),
  final_pred_un_class.reshape(final_pred_un_class.shape[0], 1),
  final_pred_lb_class[lb_idx].reshape(final_pred_lb_class[lb_idx].shape[0],
1),
  final_pred_pb_class[pb_idx].reshape(final_pred_pb_class[pb_idx].shape[0], 1)
])
```

ヒントの解説：Yaraと文字列 (PB 0.645 => 0.670)

ラベルに紐づいたYaraの情報が使えると強い

今回はStringsが使いやすい

またこれをヒントに文字列をジッと見ると、NanoCoreはそのまま”NanoCore”の文字列がある場合に気付く

```python
class StringsFeatureExtractor(FeatureExtractor):
    feature_name = "strings"
    # 他の文字列としての単純な統計情報はあまり効かなかった
    @staticmethod
    def average_word_length(strings: List[str]) -> int:
        return average_word_length("\n".join(strings))

    @staticmethod
    def has_nanocore_sig(strings: List[str]) -> int:
        return 1 if "NanoCore" in "\n".join(strings) else 0

    @staticmethod
    def has_getantivirus_sig(strings: List[str]) -> int:
        return 1 if "GetAntiVirus" in "\n".join(strings) else 0
```

Bonus：上手くいかなかった手法

Stringsの統計情報

TFIDF

　　上２つは昨年の問題で効いたが、今回はあまり効かなかった

Focal Loss

　　サンプルのスコアより下がった

FexrdのManalyzeの特徴量の修正

　　FexrdのWarningを見るとUnknown扱いされるケースがある

　　しかしこれを修正するとむしろスコアが下がった

Conclusion

データをよく確かめること

- ラベルの偏り
- NanoCoreの文字には気付けたかもしれない

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！！！

	2022	2023
分類問題のサステナビリティ	• Strings以外で単体で検知に使うフィールドがない • 次回は模範解答からスタートされうる	• マルウェア検知ではなく分類問題とした • 半教師あり設定とし工夫の余地を増やした

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