




# Week3



背景：Windows作为全世界份额第一的操作系统，必然会成为网络恶意攻击软件的重点。因此Windows在不断更新和自我迭代的路上，除了将操作系统的功能性提升意外，也还在费尽心思地确保其安全性能够跟上操作系统的升级。

到现在为止，Windows10已经内置了Defender、防火墙、SmartScan等多重网络安全和恶意软件防护手段。杀软的防护方法也从传统的扫描查杀、文件保护，进化到了主动防御和行为特征检测，再到现代的启发式、AI引擎防护。

## Load Big Dataset

- 转换数据类型，减小内存占用

## Feature engineering

- 根据特征值占比大小选择是否保留特征，比如对于二分类，某一个特征中出现次数最多的特征出现占比达到了90%以上，而label中正负类样本基本一致，可以初步判断对于推断没有帮助可以删除。
- 根据缺失值占比排序选择处理策略: 缺失值少则填充 or 缺失值占比大则删除

	Feature	Unique_values	Percentage of missing values	Percentage of values in the biggest category	type
28	PuaMode	2	99.974119	99.974119	category
41	Census_ProcessorClass	3	99.589407	99.589407	category
8	DefaultBrowsersIdentifier	1730	95.141637	95.141637	float16
68	Census_IsFlyingInternal	2	83.044030	83.044030	float16
52	Census_InternalBatteryType	78	71.046809	71.046809	category
...	...	...	...	...	...
1	ProductName	6	0.000000	98.935569	category
45	Census_HasOpticalDiskDrive	2	0.000000	92.281272	int8
54	Census_OSVersion	469	0.000000	15.845202	category
55	Census_OSArchitecture	3	0.000000	90.858045	category
82	HasDetections	2	0.000000	50.020731	int8

- 相关性分析,选择性剔除共线性特征
- Label Encoding:** LGBM基于直方图的方法可以高效处理分类特征的。
- One-Hot Encoder:** 利用稀疏矩阵存储非零元素的列和值，节省运算内存和利用
- '垂直堆叠'**在对大数据集特征稀疏化的过程中,按**批处理**将稀疏矩阵堆叠返回 单一稀疏矩阵

```
#Fit OneHotEncoder
ohe = OneHotEncoder(categories='auto', sparse=True, dtype='uint8').fit(train)

#Transform data using small groups to reduce memory usage
m = 100000
train = vstack([ohe.transform(train[i*m:(i+1)*m]) for i in range(train.shape[0] // m + 1)])
```

## LGBM 调优策略优化

- 多种交叉验证的策略:**stratified&Group K-Fold&Repeated K-Fold**

- 使用 **网格搜索** 或 **随机搜索** 来寻找最优的超参数

- 使用**SHAP** 来解释模型预测

SHAP 分析每个特征对模型输出的影响来解释预测

- **模型融合:**

尝试将多种集成学习的模型(adaboost&xgboost&random forest)

融合 (堆叠&加权平均)

```
# 训练 XGBoost 模型
xgb_model = xgb.XGBClassifier()
xgb_model.fit(X_train, y_train)

# 训练随机森林模型
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)

# 使用模型对测试集进行预测
pred_lgbm = modelLGB1.predict(X_test)
pred_xgb = xgb_model.predict_proba(X_test)[: , 1] # XGBoost 模型预测
pred_rf = rf_model.predict_proba(X_test)[: , 1] # 随机森林模型预测

# 合并预测结果 - 简单平均法
pred_ensemble = (pred_lgbm + pred_xgb + pred_rf) / 3

#加权平均
weights = [0.5, 0.3, 0.2] # 给 LightGBM 更高的权重
pred_ensemble_weighted = (weights[0] * pred_lgbm + weights[1] * pred_xgb + weights[2] * pred_
```