**实验 恶意DGA域名检测系统的简单实现**

1. **实验目的**
   1. 熟悉Python语言的使用
   2. 掌握使用传统机器学习算法进行恶意域名检测的方法
   3. 搭建简单的域名检测可视化界面
2. **背景知识**
   1. **僵尸网络与域名生成算法（DGA， Domain Generation Algorithm）**

僵尸网络最早出现在1999年，PrettyPark是首个具有僵尸网络特性的恶意程序[4]，后来伴随着科技的发展，僵尸网络也在不断进化，被控主机数逐年增加，被控主机分布范围也逐步扩大。2016年Mirai僵尸网络攻击导致美国大范围网络瘫痪，2020年8月29日Emotet僵尸网络卷土重来，通过发送大量恶意垃圾邮件传播恶意软件。

早期的僵尸网络中，被控主机通过轮询硬编码IP地址或域名与C&C服务器建立通信，但这种方式很容易被防御者发现，一旦IP地址被列入黑名单，僵尸网络就被彻底关闭。为了更好的躲避防御者的检测、更加隐蔽地进行通信，攻击者在僵尸网络中引入了Domain-Flux技术，通过域名系统（Domain Name System，DNS）频繁更改IP地址和域名之间的映射，从而隐藏C&C服务器的真实位置，对此，防御者提出了“Sinkhole”方法，通过合法途径将所有恶意域名指向C&C服务器以外的一个特定主机，从而阻止僵尸网络。为了进一步提高僵尸网络中C&C服务器与被控主机之间通信的可靠性以及鲁棒性，域名生成算法（Domain Generation Algorithm，DGA）应运而生，并很快成为僵尸网络的核心组成部分，图1是僵尸网络利用域名生成算法连接被控主机和C&C服务器的示意图。

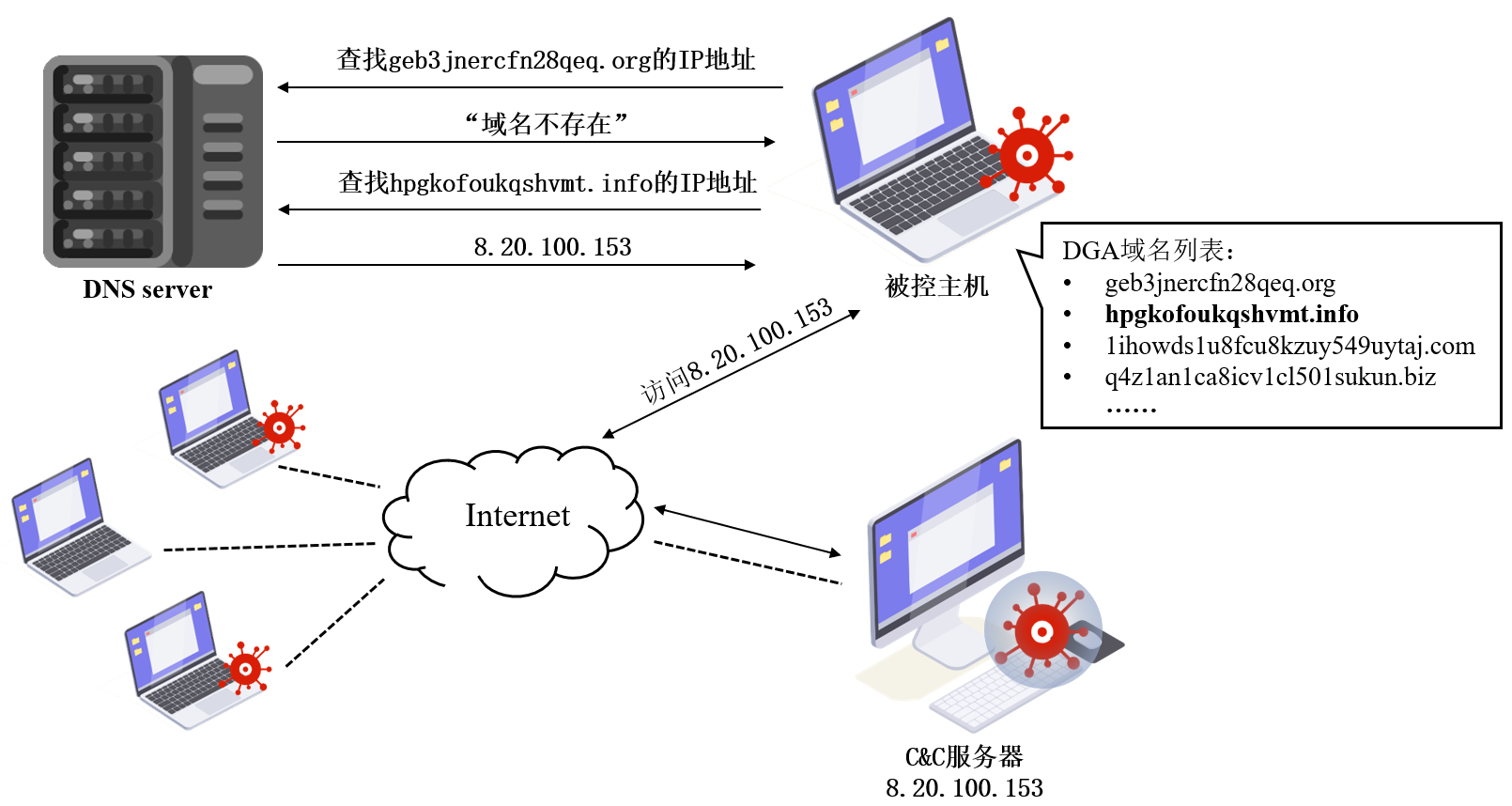


图1 僵尸网络利用域名生成算法建立通信网络示意图

图1中被控主机和C&C服务器利用域名生成算法生成大量恶意域名，僵尸网络管理员将这些域名作为C&C服务器的候选集，从中选取极少数域名进行注册，作为C&C服务器与被控主机之间的“交汇”点。被控主机依次在域名系统中查询这些域名的IP地址，如果该域名没有被注册，域名系统将返回“域名不存在”，如果查询的域名是被注册的恶意域名，域名系统返回该域名对应IP地址，即C&C服务器的有效IP地址，接下来被控主机利用该IP地址与C&C服务器建立连接。域名生成算法每次生成的域名数量庞大，因此防御者无法完全覆盖并拦截所有DGA域名，也导致Sinkhole防御方法失效，并且域名生成算法产生的域名的生存周期较短，这使得传统的黑白名单检测方式也无法有效应对。因此目前绝大多数僵尸网络都使用域名生成算法以及DGA域名构建弹性的C&C基础结构。

DGA域名是被控主机与C&C服务器建立连接的核心要素，这意味着有效检测DGA域名可以帮助网络安全组织发掘僵尸网络，防御恶意软件，对于维护互联网网络安全具有重要意义。

* 1. **DGA域名的定义与特点**

域名（Domain name）是一串由数字、字母以及特殊符号（“.”、“-”和“\_”）构成的字符串序列。域名具有树形结构，也被称为域名空间。域名中的‘.’是分割符，用于划分内部层次结构，分隔符之间的部分叫做标签。域名中最右侧的标签被称为顶级域名（Top Level Domain，TLD），如“com”。顶级域名左侧紧靠的标签被称为二级域名（Second Level Domain，SLD），如“example.com ”中的“example”，越靠近域名左侧，标签所属层级越低。

一般情况下，域名中包含两个或两个以上标签，对英文字符大小写不敏感，即“BAIDU.com”等同于“baidu.com”。域名的最大长度为255，但是在现实场景中，为了方便人们的记忆，绝大多数域名的长度都远小于255。

DGA域名是由域名生成算法产生的伪随机字符序列。攻击者在构造DGA域名时主要对二级域名（SLD）进行操作，并随机选择一个顶级域名与之拼接形成DGA域名，因此在进行DGA域名检测任务时，主要对二级域名和顶级域名进行分析，忽略其他子段。

目前出现频率较高的DGA域名大约有45种，其中大多数是基于随机种子的，这类DGA域名与正常域名在字符长度分布、可读性、字符串构成等方面存在较大的差异，因此容易被检测。还有一部分域名是基于词典生成的，如Suppobox。相较于前一类DGA域名，基于词典生成的DGA域名在字符长度分布方面与正常域名相似，因此相对来说检测难度更大。表1中列出了部分DGA家族以及对应的实例。

表1 常见的DGA方法以及实例表

|  |  |  |
| --- | --- | --- |
| **DGA** | **DGA生成方式** | **实例** |
| Zeus | 年、月、日、[0-999]的MD5编码 | krhafeboleyhiytrwuduzlbucutwt  vsmfcabubenvibwolvgihirvmz |
| Conficker | 以世界时间为随机种子的生成器 | fabdpdri  sfqzqigzs |
| Kraken | 长度为[6-10]的随机字符串 | rhxqccwdhwg  huwoyvagozu |
| Srizbi | 基于异或操作的时间转换器 | wqpyygsq  tqdaourf |
| Torpig | 以当前时间和“8”为随机种子的生成器 | 16ah4a9ax0apra  12ah4a6abx5apra |
| kwyjibo | 基于马尔可夫的英文字符 | Overloadable  Refillingman |

1. **实验原理**
2. **特征工程**
   1. **结构特征**

该类特征代表了域名在字符串结构组成上的特点。表2对这些特征进行了总结，并以d1和d2为例说明这些特征的具体取值，d1=baidu.com是一个众所周知的正常域名，d2 = 159vthg1nqmpuyh6. beauty是一个DGA域名。

表2 域名结构特征表

| # | 特征 | 取值类型 | 意义 | d1 | d2 |
| --- | --- | --- | --- | --- | --- |
| 1 | domain\_len | 整数 | 域名总长度 | 9 | 23 |
| 2 | sld\_len | 整数 | 二级域名长度 | 5 | 16 |
| 3 | tld\_len | 整数 | 顶级域名长度 | 3 | 6 |
| 4 | tld\_dga | 布尔 | 是否包含某些恶意顶级域名 | 0 | 1 |

* 1. **语言特征**

该类特征主要关注正常域名和DGA域名在语言模式上的差异，对于机器学习分类具有重要作用。

| # | 特征 | 取值类型 | 意义 | d1 | d2 |
| --- | --- | --- | --- | --- | --- |
| 5 | uni\_domain | 整数 | 域名字符总数（去重） | 8 | 17 |
| 6 | uni\_sld | 整数 | 二级域名字符总数（去重） | 5 | 14 |
| 7 | uni\_tld | 整数 | 顶级域名字符总数（去重） | 3 | 6 |
| 8 | dig\_sld | 整数 | 二级域名数字总数 | 0 | 5 |
| 9 | dig\_ratio\_sld | 实数 | 二级域名数字占比 | 0.0 | 0.31 |
| 10 | sym\_ratio\_sld | 实数 | 二级域名特殊字符占比 | 0.0 | 0.0 |
| 11 | hex\_ratio\_sld | 实数 | 二级域名十六进制字符占比 | 0.6 | 0.31 |
| 12 | vow\_ratio\_sld | 实数 | 二级域名元音字母占比 | 0.6 | 0.06 |
| 13 | con\_ratio\_sld | 实数 | 二级域名辅音字符占比 | 0.4 | 0.63 |
| 14 | rep\_char\_ratio\_sld | 实数 | 二级域名重复字符/字符总数（去重） | 0.0 | 0.14 |
| 15 | cons\_con\_ratio\_sld | 实数 | 二级域名连续辅音占比 | 0.0 | 0.63 |
| 16 | cons\_dig\_ratio\_sld | 实数 | 二级域名连续数字占比 | 0.0 | 0.19 |
| 17 | gib\_value\_sld | 布尔 | 二级域名Gib成文检测 | 1 | 0 |

1. **实验环境**

操作系统：不限

语言：Python3.6

编译器：PyCharm或其他

1. **实验内容**
   1. **理解代码结构，配置环境（10分）**
2. 代码结构

|\_\_\_\_\_data

|\_\_\_\_\_sample——样本文件夹

|\_\_\_\_\_sample\_black.csv——恶意域名样本

|\_\_\_\_\_sample\_white.csv——正常域名样本

|\_\_\_\_\_modle——模型文件夹（存放训练好的模型及归一化尺度）

**|\_\_\_\_\_**feature——特征文件夹（存放已提取特征的训练集和测试集）

|\_\_\_\_\_static  
 |\_\_\_\_\_\_gib\_model.pki——Gib成文检测依赖文件

|\_\_\_\_\_feature\_extraction.py——特征提取代码

|\_\_\_\_\_detection.py——模型训练、测试及域名检测代码

|\_\_\_\_\_requirements——依赖包及版本

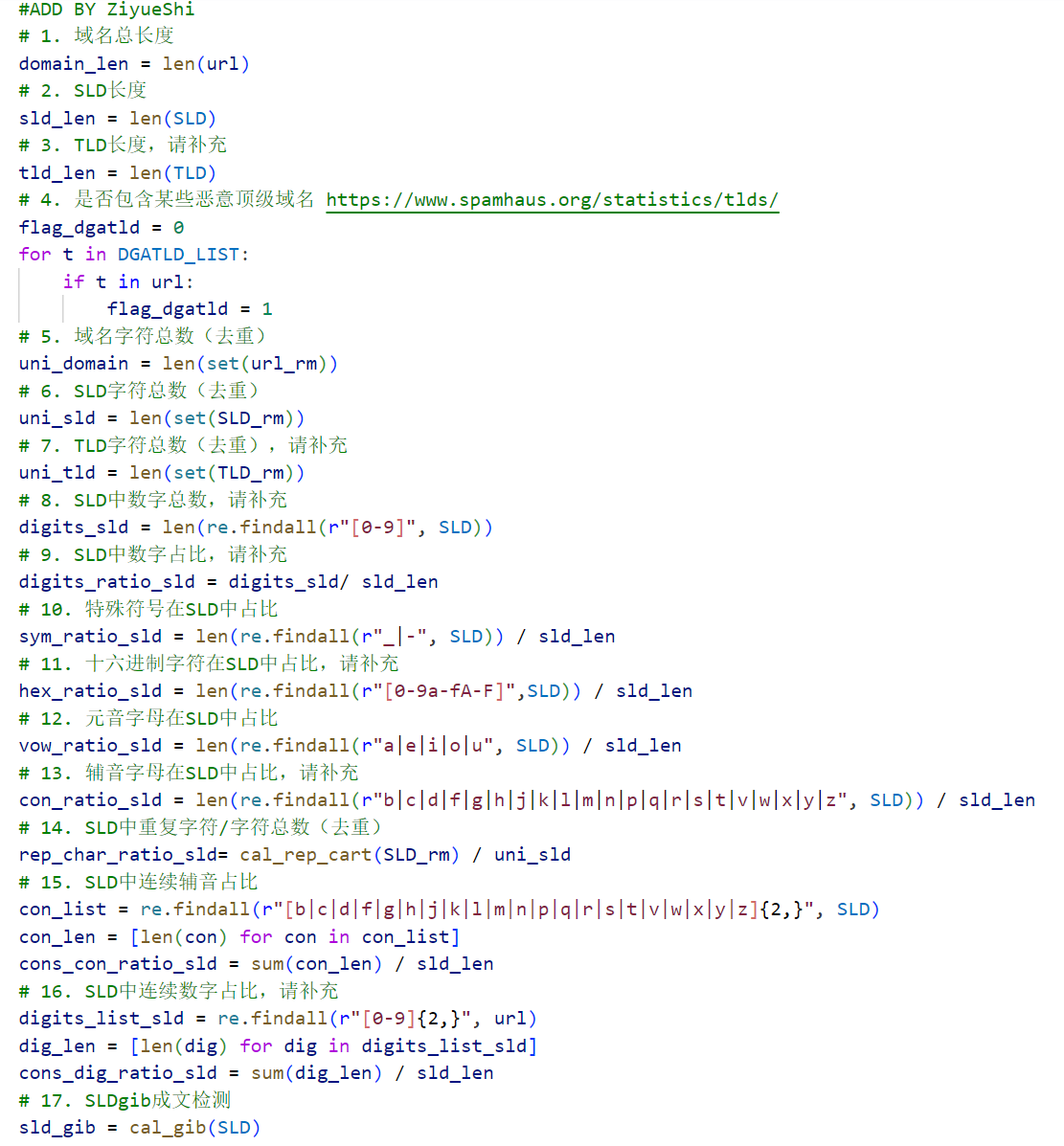
1. 环境配置

* 安装Python3.6（Ubuntu18.04系统已内嵌Python3.6，Windows系统需自行下载安装<https://www.python.org/downloads/>）
* 使用pip安装requirements中的依赖包



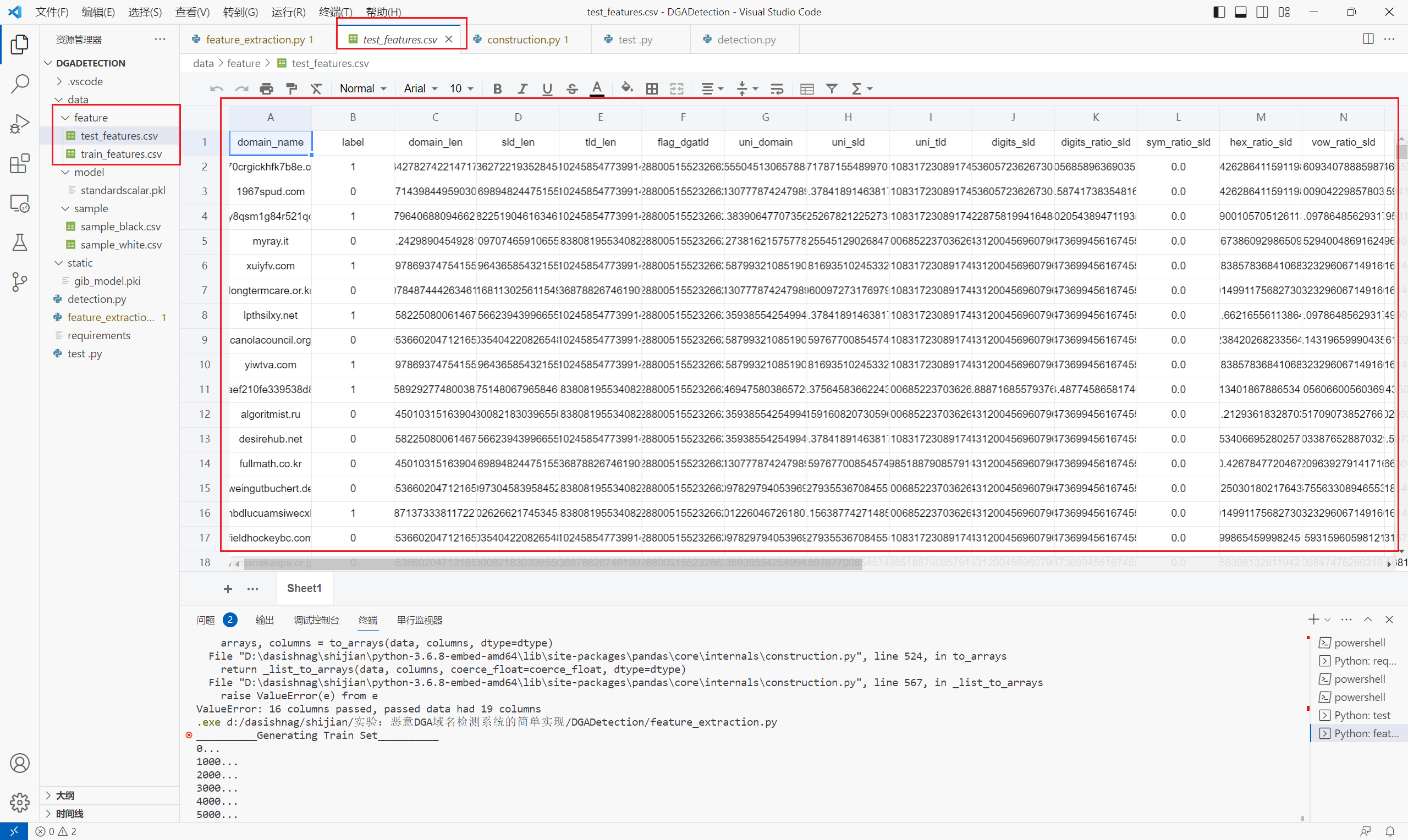
* 1. **补充feature\_extraction.py文件（30分）**

1. 结合注释学习并理解feature\_extraction.py文件中的各功能函数
2. 在已给代码的基础上补充函数get\_feature()，提取域名的至少17维特征



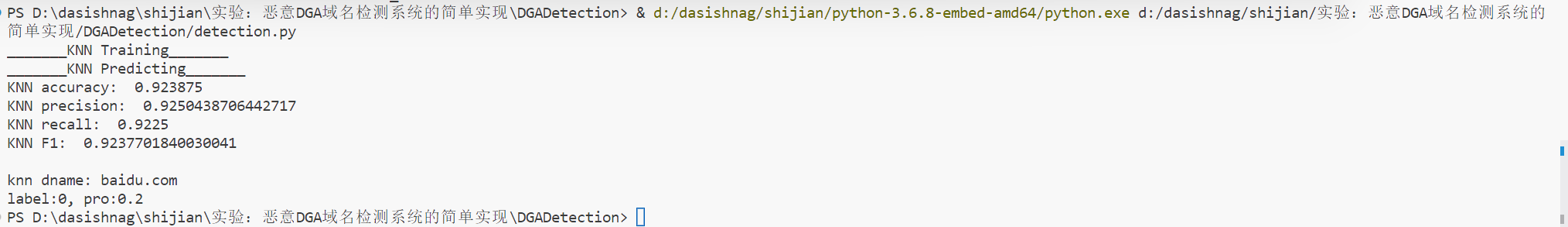
1. 在主函数中运行数据集生成函数dataset\_generation()，成功生成训练数据及测试数据



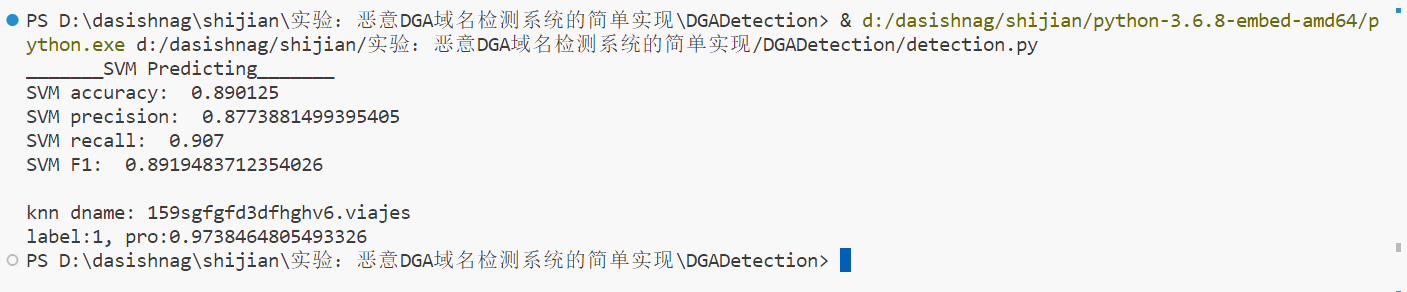


* 1. **补充detection.py文件（30分）**

1. 结合注释及KNeighborsClassifier官方文档（https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier）学习并理解自定义类KNNClassifier的功能，完成KNN分类器的训练，得到测试集在该模型上的表现。



1. 仿照自定义类KNNClassifier，参考SVC官方文档（https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html）编写类SVMClassifier，支持向量机分类器的训练、测试和其他功能。在此基础上，完成SVM分类器的训练，得到测试集在该模型上的表现。



1. 思考：模型如何调优？

超参数调优，特征工程优化，

* 1. **简单可视化（20分）**

**这里使用PyQt5进行可视化，在源代码目录下提供了DGA.exe可供使用。**

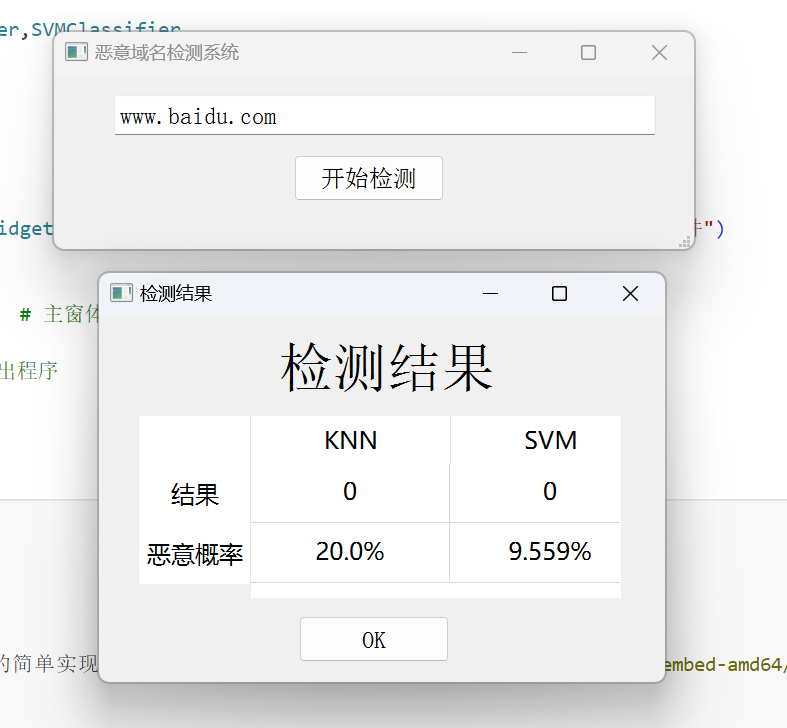
* 可输入域名字符串（对字符串有初步的合法性检查，如输入有误应给出提示）

合法性检查：



* 返回检测结果及恶意概率

这里将小数变成了百分数，感觉更易于理解



* 1. **扩展实验（10分）**

1. 思考还可以提取哪些有区分度的域名特征，在代码中实现
2. 尝试使用不同算法（传统机器学习算法、神经网络等）训练更多的分类器，比较不同分类器的效果及优缺点
3. 思考在一个检测任务中，多个模型的检测结果如何集成，除了简单的投票外是否还有其他策略，在代码中实现
4. **实验报告**

**实验时间：**2023.9.4-2023.9.8

**报告内容：**

1. 要体现实验过程（实验的控制台输出截图、可视化界面使用截图等）
2. 有对关键代码的分析或比较详细的代码注释
3. 实验心得总结

**提交方式：**将代码和实验报告一并打包提交到邮箱dcy@nankai.edu.cn