2.2.2.1单个资产的网络攻防价值评估模型构建

**数据处理、指标体系怎么构建、初始的标签数据怎么来的、训练模型的选择（常用的多分类方法，一到两种候选的方法，初步尝试实验结果，最终结果）、调优的过程。**

**决策树怎么构建、根据现有数据集怎么调优**

**3.20前**

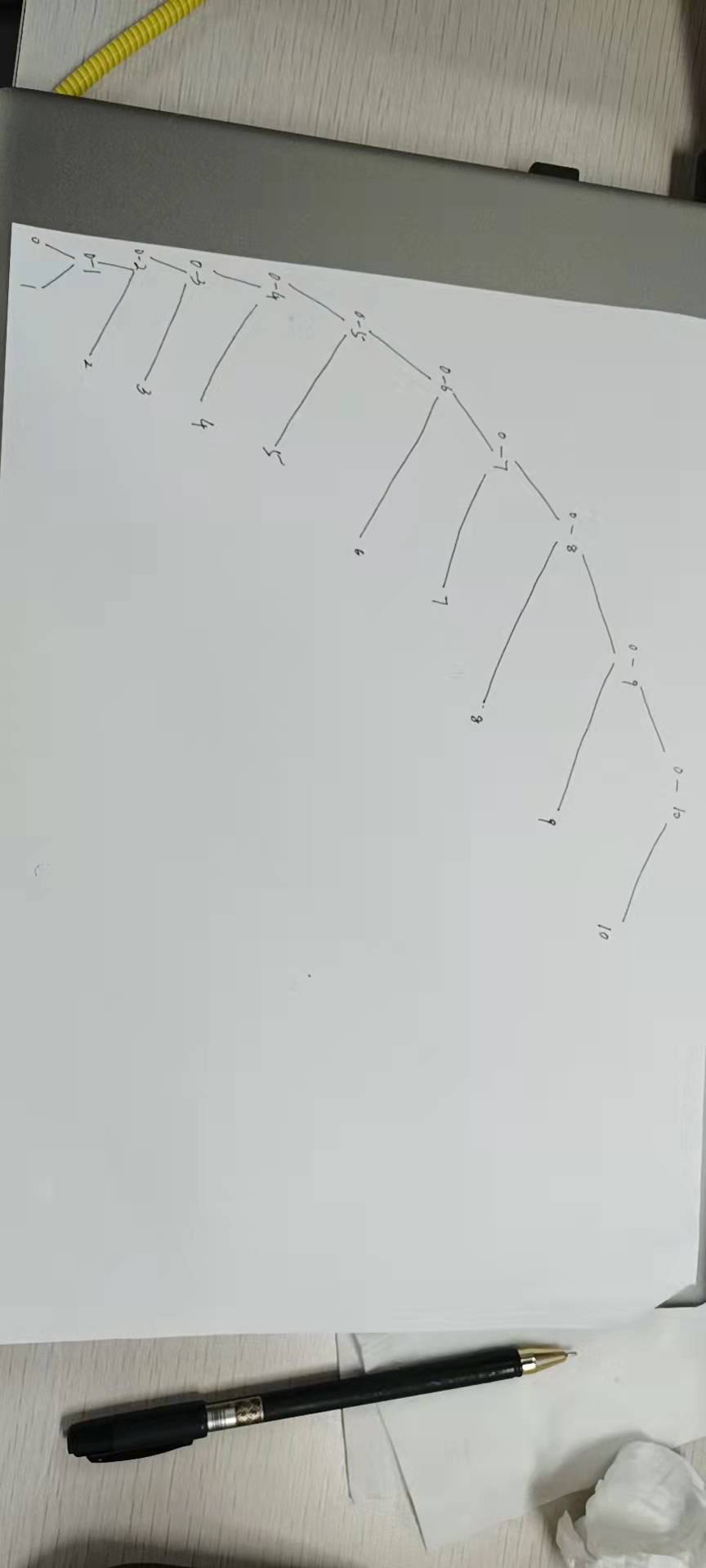
（1）网络资产评估模型构建

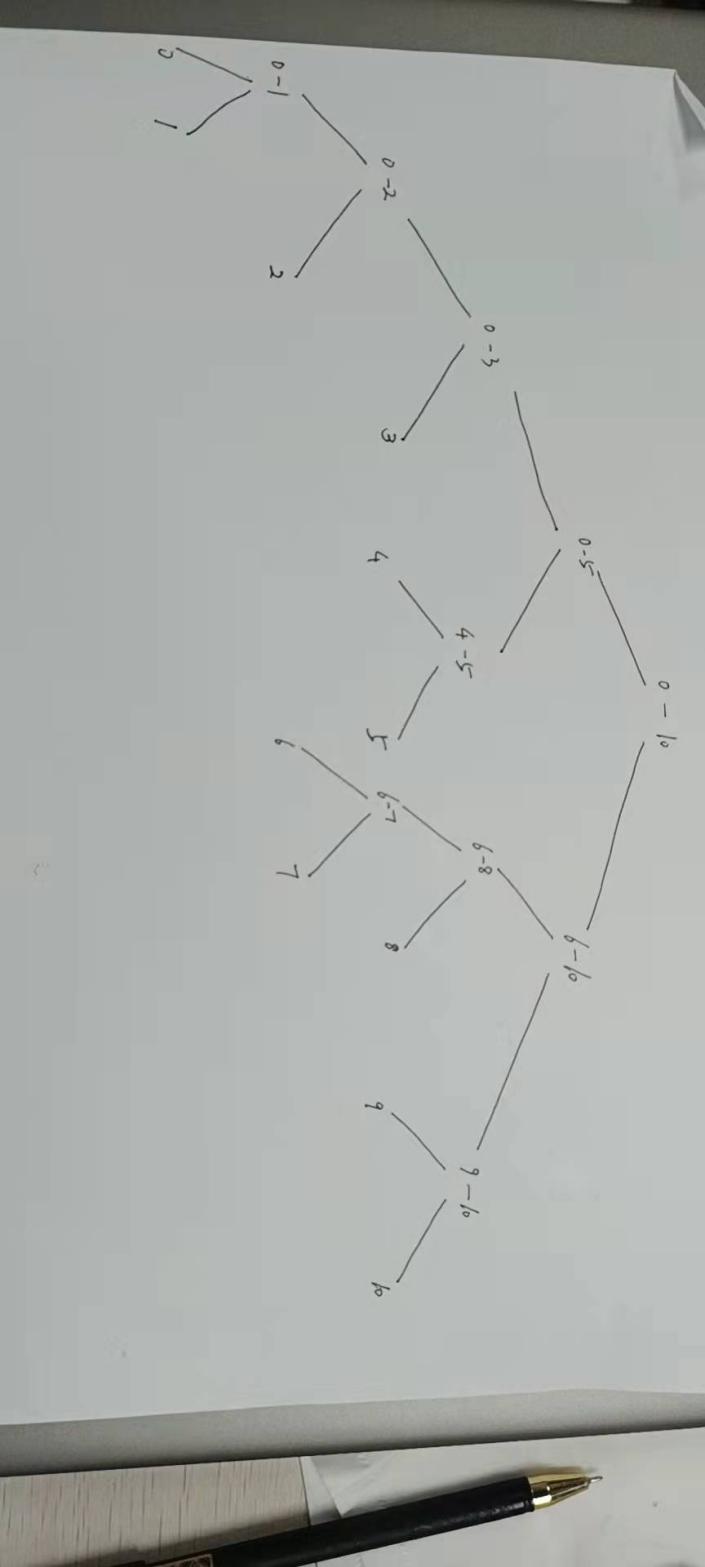
由于网络资产海量的特点，依赖人工方式对网络资产的脆弱性和重要性进行评估是不现实的，因此在技术上，通过引入机器学习的方法构建价值评估模型，对实际探测到的网络资产数据进行训练，并将对网络资产的价值评估转换为分类问题（价值等级0-10），运用多种算法分别构建不同的分类模型，并通过弱分类模型的集成优化和分类效果准确度的对比，选择最优的模型投入应用。表一是各分类模型实际分类效果的对比：

表一：模型效果对比

|  |  |  |  |
| --- | --- | --- | --- |
| 算法 | | 准确率 | Time |
| 神经网络 | ANN | 18%（过拟合，原始数据形式不适合神经网络学习，且数据量少） |  |
| 决策树 | LightGBM\*10-10 | 88.77% | 10 min 49 s |
| LightGBM\*10-10+SMOTE  (过采样0-10) | 88.94% | 9 min 23 s |
| LightGBM\*10-10+SMOTE  (只做过采样5，6) | 88.73% | 10 min 48 s |
| LightGBM\*10-10+SMOTE  (只做过采样5，5：6) | **89.14%** | 10 min 47 s |
| LightGBM\*10-10+SMOTE  (只做过采样6，6：5) | **89.14%** | 10 min 47 s |
| LightGBM\*10-6 | 87.89% | 11 min 26s |
| LightGBM\*10-6+SMOTE  (只做过采样6，6：5) | 88.5% | 11 min 32 s |
| LightGBM | 80.0% | 6 min 11 s |
| LightGBM\*8-10 | 85.9% | 7 min 17 s |
| LightGBM\*8-10+SMOTE | 85.6% | 7 min 19 s |
| LightGBM\*8-6 | 85.9% | 7 min 55 s |
| LightGBM\*8-6+SMOTE | 86.4% | 6 min 23 s |
| XGBoost | 67.2% | 11min 20 s |
| 朴素贝叶斯 |  | 46.2%（数据间的相关性影响分类结果） |  |
| 随机森林 |  | 74.3%（投票选择，个别准确率高的决策树结果被多数准确率较低的决策树影响） |  |
| SVM | SVM直接多分类 | 54.5%（实现多分类精度不高，且速度慢，少用于多分类问题） | 13 min 7 s |
| SVM\*10\_10 | 61.9% | 1 min 42 s |
| SVM\*10\_6 | 60.4% | 1 min 36 s |
| Else | OLS | 48.9%（数据间无线性关系） |  |

OVR





MVM

现有的数据脆弱性共2.4w条数据，12个属性，重要性共1.7w条数据，9个属性。构建模型的目标为根据资产的多个属性分别实现资产的重要性和脆弱性分值评估，这是一个多分类问题。

常用的[分类器](https://so.csdn.net/so/search?q=%E5%88%86%E7%B1%BB%E5%99%A8&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/BigDataEngineer/article/details/_blank)包括支持向量机（SVM）、贝叶斯、线性回归、随机森林、神经网络、决策树，针对现有的数据，所有的属性已被处理为离散型的整数型数据，脆弱性和重要性分值为0-10的11分类，。

支持向量机（SVM）：SVM是实现分类的模型，但SVM实现多分类精度不高，且速度慢，很少用于多分类问题。对本实验数据，若仅简单的用SVM实现11分类的问题时，准确度不高；若将多分类问题转化成多个二分类问题时，虽然算法的复杂有所上升，但相比简单的多分类，准确度有所提高，但由于在将多分类问题转换成二分类问题的过程中，总有一个多分类问题的存在，这就大大降低了整个算法的准确度。

朴素贝叶斯模型分类算法需要基于条件独立性假设的前提,这是一个理想状态,在实现多分类的过程中，数据之间的独立性越强,其分类结果越准确。但在实际应用中数据属性间会存在联系,从而降低了分类准确性,为了获取各个属性之间的相关系数，我们计算了各个属性之间的相关系数，以此来表示属性之间的依赖关系，表1为与脆弱性相关的各个属性的相关系数，表2为与重要性相关的各个属性的相关系数，相关系数越大，表示属性间的依赖关系越强。

表二：脆弱性属性间的依赖关系

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 相关系数 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 |
| S1 | \ | 0.213 | 0.069 | 0.077 | 0.259 | 0.443 | 0.049 | 0.229 | 0 | 0 | 0 | 0 |
| S2 | \ | \ | 0.280 | 0.101 | 0.269 | 0.230 | 0.034 | 0.361 | 0 | 0 | 0 | 0 |
| S3 | \ | \ | \ | 0.127 | 0.156 | 0.112 | 0.005 | 0.456 | 0 | 0 | 0 | 0 |
| S4 | \ | \ | \ | \ | 0.123 | 0.092 | 0.002 | 0.278 | 0.456 | 0 | 0 | 0 |
| S5 | \ | \ | \ | \ | \ | 0.608 | 0.150 | 0.431 | 0 | 0 | 0 | 0 |
| S6 | \ | \ | \ | \ | \ | \ | 0.107 | 0.312 | 0 | 0 | 0 | 0 |
| S7 | \ | \ | \ | \ | \ | \ | \ | 0.030 | 0 | 0 | 0 | 0 |
| S8 | \ | \ | \ | \ | \ | \ | \ | \ | 0 | 0 | 0 | 0 |
| S9 | \ | \ | \ | \ | \ | \ | \ | \ | \ | 0 | 0 | 0 |
| S10 | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | 0 | 0 |
| S11 | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | 0 |
| S12 | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ |

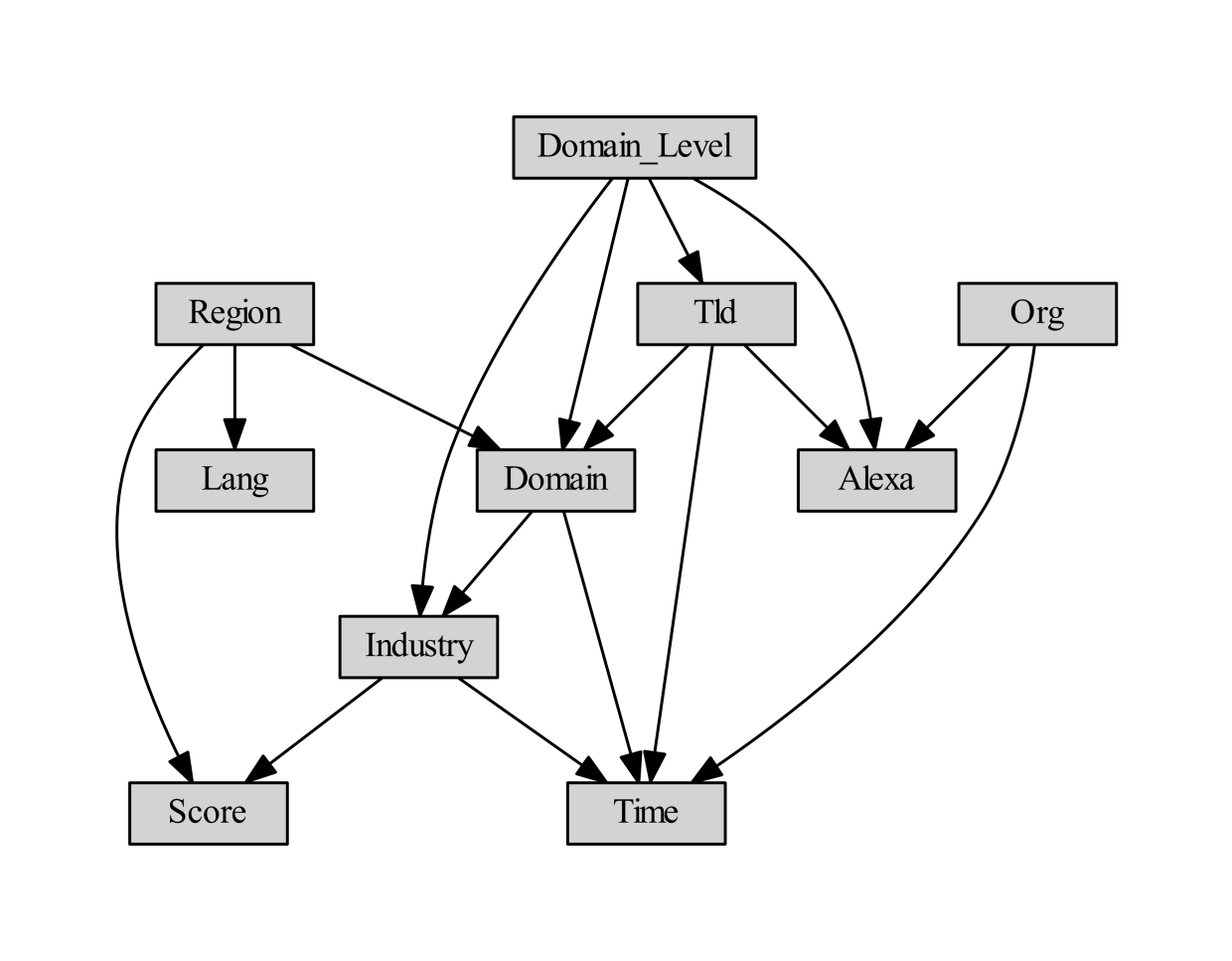
注：S1:操作系统,S2:Web容器,S3:Web应用,S4:识别的组件数量,S5:漏洞编号,S6:漏洞危害等级,S7:漏洞发现时间,S8:开发语言,S9:弱口令,S10:防火墙,S11:云主机,S12:CDN

表三：重要性属性间的依赖关系

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 相关系数 | L1 | L2 | L3 | L4 | L5 | L6 | L7 | L8 | L9 |
| L1 | \ | 0.019 | 0.022 | 0.083 | 0.116 | 0.151 | 0.276 | 0.336 | 0.059 |
| L2 | \ | \ | 0 | 0.001 | 0 | 0 | 0.001 | 0.002 | 0.002 |
| L3 | \ | \ | \ | 0 | 0 | 0 | 0 | 0.002 | 0.002 |
| L4 | \ | \ | \ | \ | 0.013 | 0.181 | 0.023 | 0.037 | 0.012 |
| L5 | \ | \ | \ | \ | \ | 0.025 | 0.241 | 0.247 | 0.005 |
| L6 | \ | \ | \ | \ | \ | \ | 0.047 | 0.086 | 0.029 |
| L7 | \ | \ | \ | \ | \ | \ | \ | 0.751 | 0.006 |
| L8 | \ | \ | \ | \ | \ | \ | \ | \ | 0.012 |
| L9 | \ | \ | \ | \ | \ | \ | \ | \ | \ |

注：L1:Tld,L2:Time,L3:Alexa,L4:Domain,L5:Org,L6:Domain\_Level,L7:Lang,L8:Region,

L9:Industry



图一

图一是以网络资产重要性评估属性为例，在贝叶斯网络模型构建中由爬山算法进行结构学习，生成的贝叶斯网络最优模型，可以直观的观察到各属性间的依赖关系，在实验中，由于数据的数据量大、分类类别多的特性，训练数据十分占用内存，并且属性间存在很强的依赖关系，因此，贝叶斯网络在该项目中不适合作为评估模型。

线性回归：采用最小二乘法实现线性回归，显然数据属性与类别之间没有明显的线性关系，因此线性回归的方法并不适合。

随机森林：随机森林是一种由决策树构成的继承算法，在获取结果时采用投票的方式决定最终分类结果，在评估的过程中，可能有多数准确度不高的决策树，掩盖了真实的结果。

神经网络：神经网络比较适合处理复杂的自然信号，比如图像，声音，自然语言。在机器学习中，数据大概可以分成四大类：图像，序列，图和表格数据。其中，前3类数据有比较明显的模式，比如图像和图的空间局部性，序列的上下文关系和时序依赖等。而表格数据常见于各种工业界的任务，如广告点击率预测，推荐系统等。在表格数据中，每个特征表示一个属性，如性别，价格等等，特征之间一般没有明显且通用的模式。神经网络适合的是前三类数据，也就是有明显模式的数据。本项目的数据属于表格类型的数据，不适合神经网络的学习。

决策树：决策树也是一种非常常用的分类方法，决策树的主要思想是从一个无规则的样本集中推导出一个分类规则，其可以以树的形式表示，内部节点表示特征或分类指标，一个叶子节点表示一类。决策树算法逻辑清晰、层次分明、直观,其分类规则便于人们的理解和实现, 是一个相对友好的分类算法。决策树算法分类精度高, 采用决策树分类算法在数据的挖掘过程中,每个节点对应一个分类规则,可以准确将每个数据分类到叶节点。并且决策树算法运行高效, 用时较少。此外，该算法非常适合于表格类型的数据，且对于连续性和离散型的数据都可以进行分类，并且在二分类和多分类的准确度都优于SVM，所以最终选择了以决策树为基础来进行多分类模型的实现。GBDT、xgboost、lightgbm都能够非常高效的应用到分类、回归的问题上。三者在准确度和运行速度上都是逐步提升的，lightgbm是目前速度最快、准确度最高的算法，因此，我们最终选取了lightgbm模型。

在使用lightgbm模型进行多分类的过程中，我们发现，二分类的准确度远远高于多分类，因此，我们将模型做了一些改进，共有九个二分类模型和一个多分类模型，共有两种方案，一种是自右向左逐个分类，一种是所有二分类都从正中间的位置进行划分，在做了初步实验后，我们发现，自右向左逐步划分类别的准确度更高一点，因此选择该模型进行分类。由于在分类的过程中，数据不平衡的问题也会影响到最终的准确度，因此，需要对不平衡数据做处理。进行不平衡处理的方式有两种，过采样和欠采样处理，考虑到数据量不大的问题，最终选择过采样进行处理。过采样共有两种处理方式：SMOTE（合成少数过采样技术）和ROS（随机过采样技术），由于随机过采样的随机性会影响最终分类结果，因此选取更为合适的SMOTE方法。由于考虑到过采样处理后各个类别的数据要处于相对均衡的状态，因此有两种方案，一种是在读取数据后，直接将所有类别一起做过采样处理，使得每个类别的个数均衡，一种是在类别中间（即5或6）处进行过采样处理，出于严谨，分别单独在5处，6处以及同时在5、6处进行了三种过采样处理，分类结果显示，单独在5或6处进行过采样处理的准确度相同且达到最高，由此，确定了最终的分类模型。

lightgbm算法：

①基于Histogram的决策树算法：基于feature直方图寻找最优分割点，GBDT是遍历全部数据寻找最优分割点，其时间复杂度远远高于基于直方图查找的时间复杂度。

②GOSS：如果一个实例的梯度很小，说明这个实例得到很好的训练，其训练误差很小。不同梯度的样本选取不同的采样率进行训练，大大提高了训练速度，也在一定程度上提高了算法的准确度。

③EFB：在训练的过程中，将几个相互独立的特征组合成一个新的特征，构建新的特征组合的feature直方图，在保证训练准确度的情况下提高了训练速度。