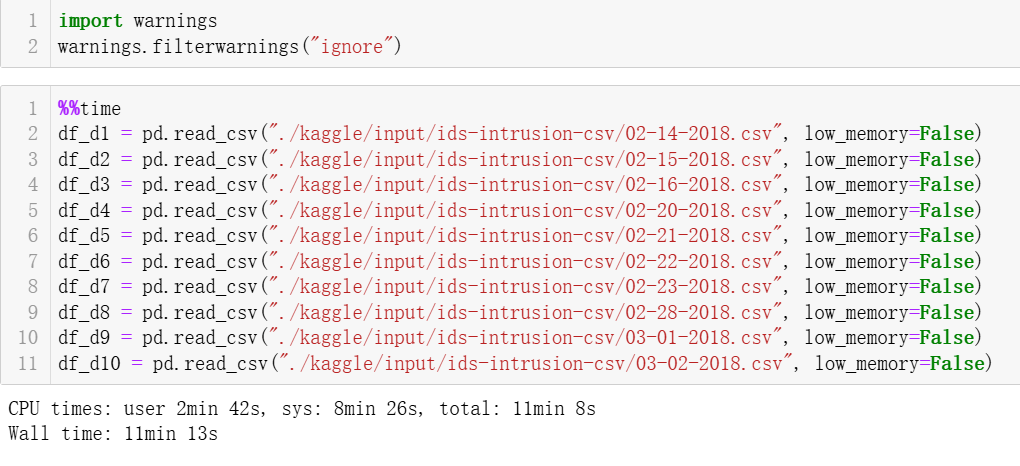
代码详细内容：

数据清理



这段代码用于忽略警告信息，使得在加载数据时，如果遇到一些非致命的警告（例如数据类型转换的警告），不会显示出来，保持输出的简洁

 这段代码分别读取了每个CSV文件，并将它们存储在多个Pandas数据框（df\_d1 到 df\_d10）中。

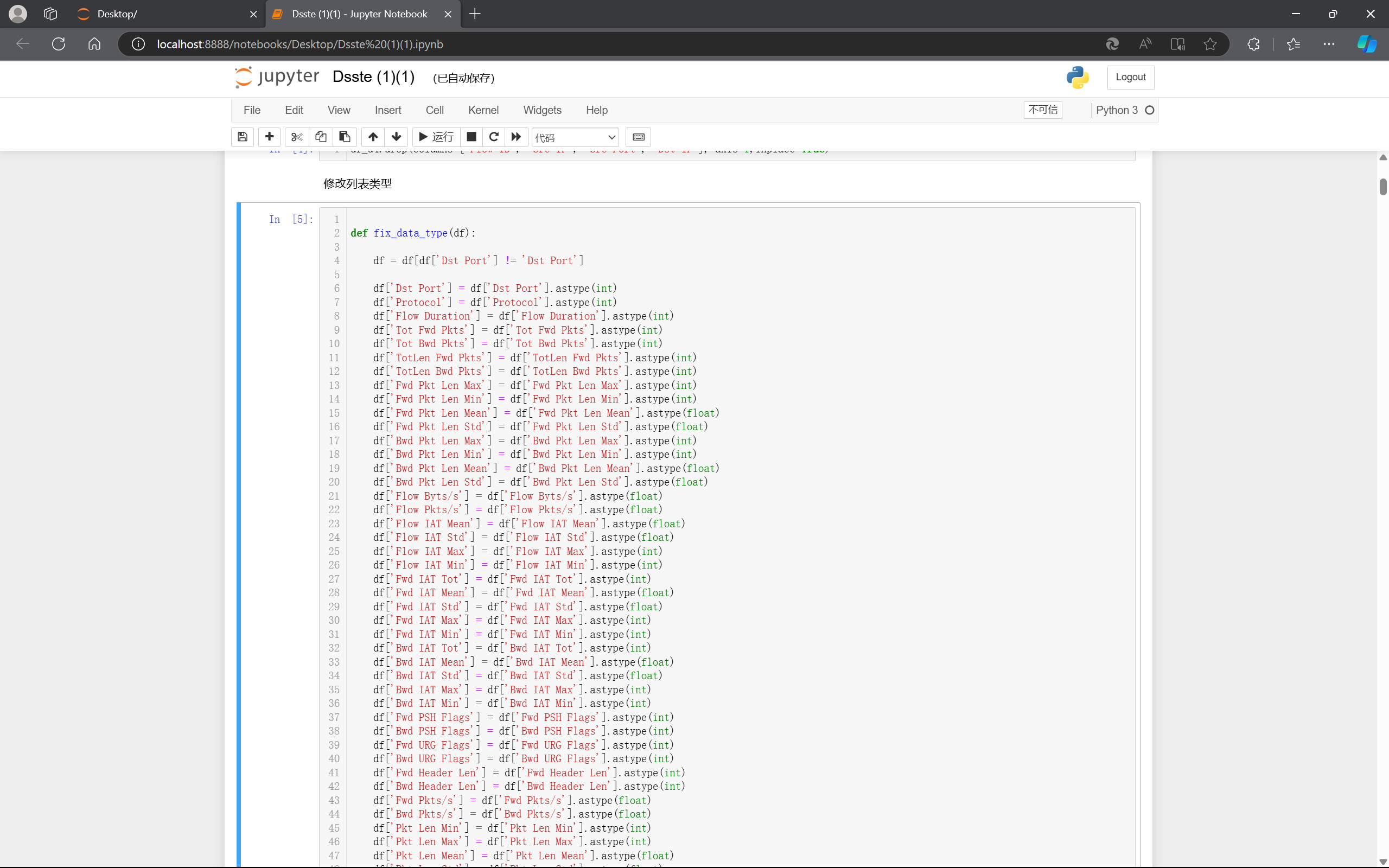
 pd.read\_csv() 是Pandas库中读取CSV文件的函数。

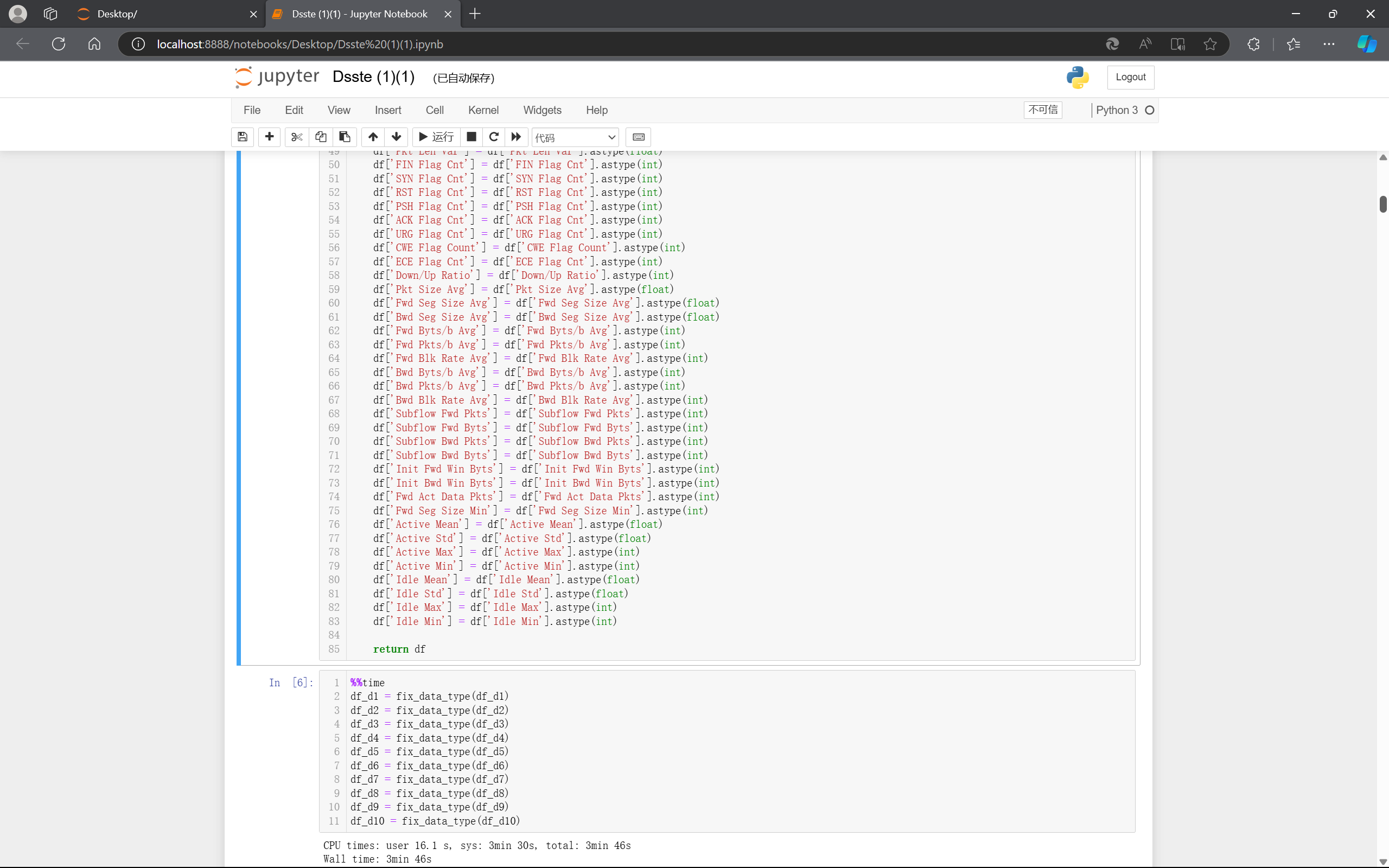
 参数low\_memory=False的作用是为了提高读取大文件时的效率，并防止分块读取数据时可能出现的混淆数据类型的问题。

 读取这些文件的时间输出显示总耗时超过11分钟，这表明这些文件体积较大，是典型的网络流量数据。

**这段代码在数据预处理中的作用**

* **数据加载和准备**：
  + 这段代码的主要目的是从硬盘中读取多个原始数据文件到内存中，以便在后续步骤中进行进一步的数据操作。
  + 在入侵检测系统（IDS）的数据处理中，通常需要处理大量的网络流量数据，这些数据会分布在多个文件中。因此，首先要将这些文件的数据加载到内存中，形成数据框（DataFrame）。
* **后续操作的准备**：
  + 将数据加载到内存中后，下一步通常是对这些数据框进行合并（数据集成）、清洗（去除无效数据或空值）、转换（将特征转换为适合模型的格式），以及规范化等操作。
  + 这段代码相当于是为后续的数据集成、清洗和特征选择做好了准备。



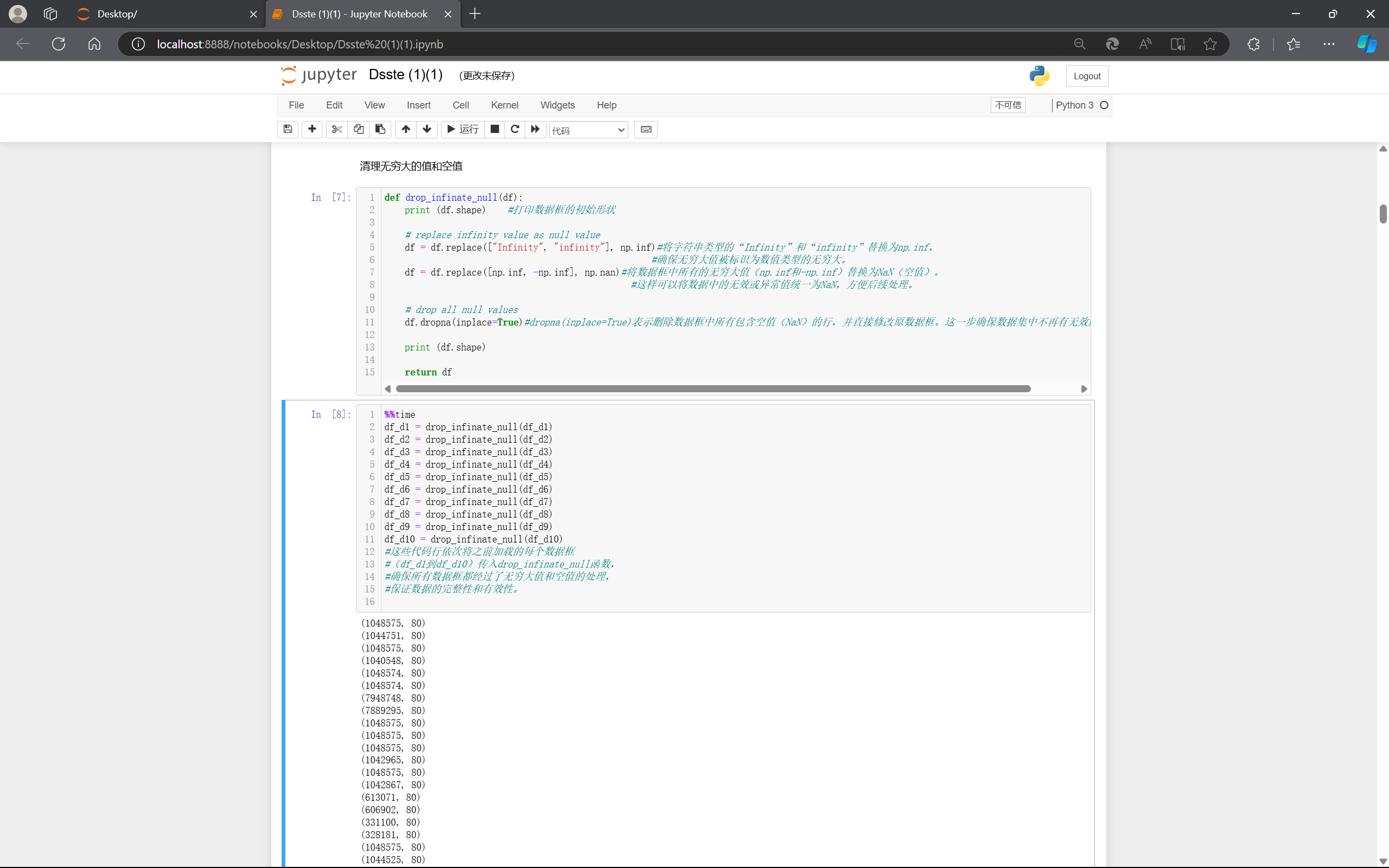


这行代码用于清除数据中的无效行。例如，如果某一行的Dst Port列出现了值为字符串"Dst Port"的情况（可能是因为某些文件的开头有列名重复），这行代码会将这些无效行过滤掉。

**转换各列的数据类型**

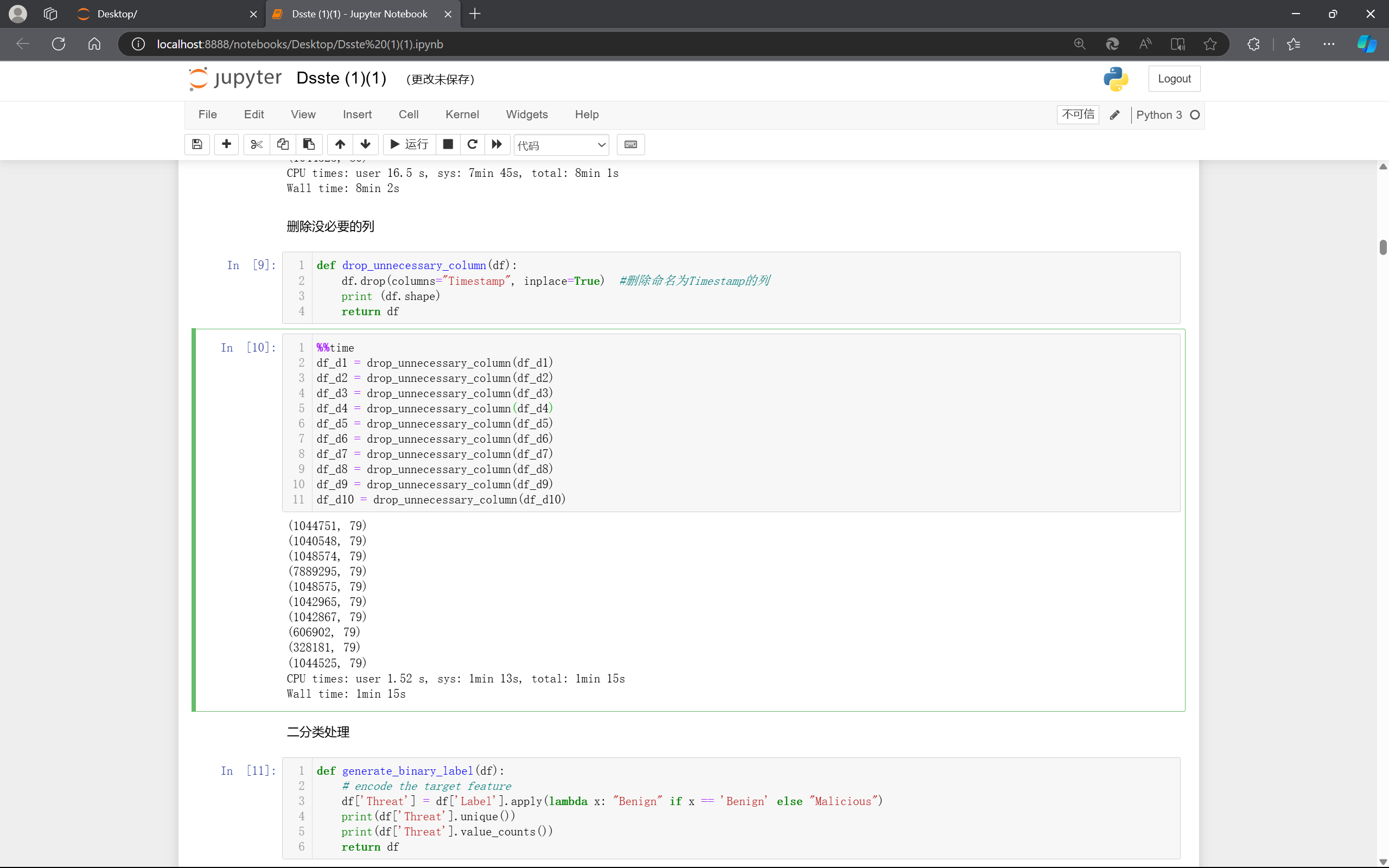
* 代码中针对每一个特征列（如Dst Port、Protocol、Flow Duration等），使用.astype()方法将其转换为适当的数据类型：
  + **整数类型（int）**：例如，端口号、协议、数据包数量等都应为整数。
  + **浮点数类型（float）**：例如，一些计算得到的平均值或标准差应为浮点数。
* 这种转换确保数据框中的数据类型符合特征的实际含义，并且能提高后续计算的效率和准确性。

这些代码行将之前加载的每个数据框（df\_d1到df\_d10）传入fix\_data\_type函数，确保每个数据框都完成了数据类型的修正和转换。



 输出显示了每个数据框在清洗前后的行数变化。例如，第一个数据框从1048575行减少到1044751行，这表明有一些行由于包含无效值而被删除了。

 通过比较每个数据框在清洗前后的行数，可以看到哪些数据框中的无效值较多，也可以判断数据集中不同文件的质量情况。



 **去除无关信息**：

* Timestamp（时间戳）列通常记录了每条流量样本的时间信息，在许多情况下，这些时间信息并不直接影响攻击检测或分类任务，可能被认为是**无关特征**。因此，删除这类列可以简化数据，避免模型训练时处理无用信息。

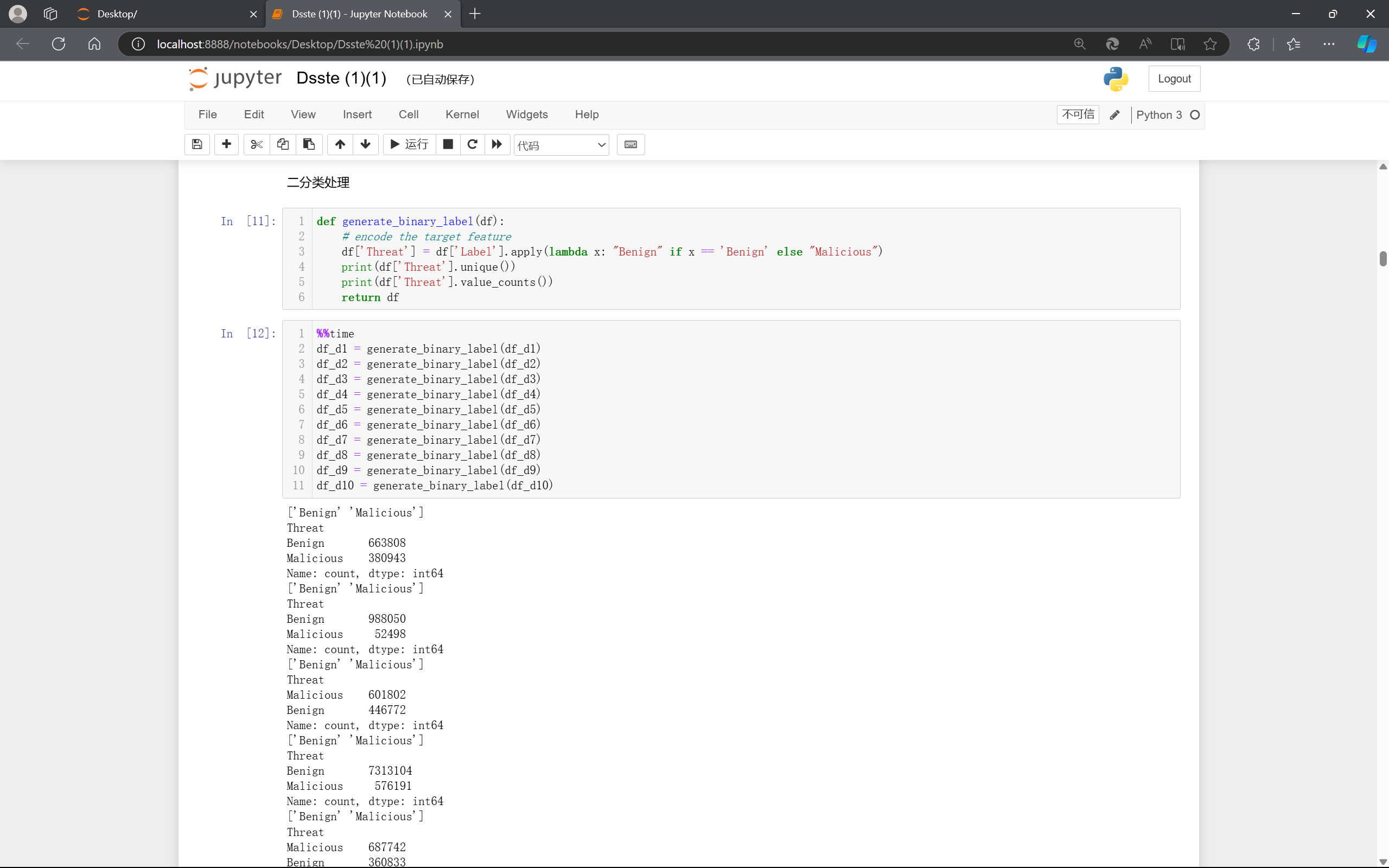
 **减少数据维度**：

* 删除不必要的列有助于**减少数据维度**，这在处理高维数据集时尤其重要。减少数据集中的列数可以降低模型的复杂度，提高训练速度，并减少过拟合的风险。

 **优化计算资源**：

* 通过去除冗余或无关的特征，可以减少数据框的大小，从而节省内存资源和计算时间。这对于处理大规模数据集尤其有用，因为它可以显著提高数据处理和模型训练的效率。

二分类



将标签只分类两种，一种是正常流量Benign另一种是恶意流量Malicious

 **将多类别标签转换为二元标签**：

* 在CSE-CIC-IDS2018数据集中，Label列可能包含多个不同的攻击类型（如DDoS、Web Attack、Infiltration等），以及Benign（正常流量）。通过将所有攻击类型归为"Malicious"，将正常流量归为"Benign"，这段代码将原本的多类别问题简化为一个二元分类问题。

 **简化分类任务**：

* 将标签二值化后，可以简化模型的训练和测试过程，尤其是在模型的初步构建阶段。这样可以让模型更快地找到区分Benign和Malicious流量的特征模式。

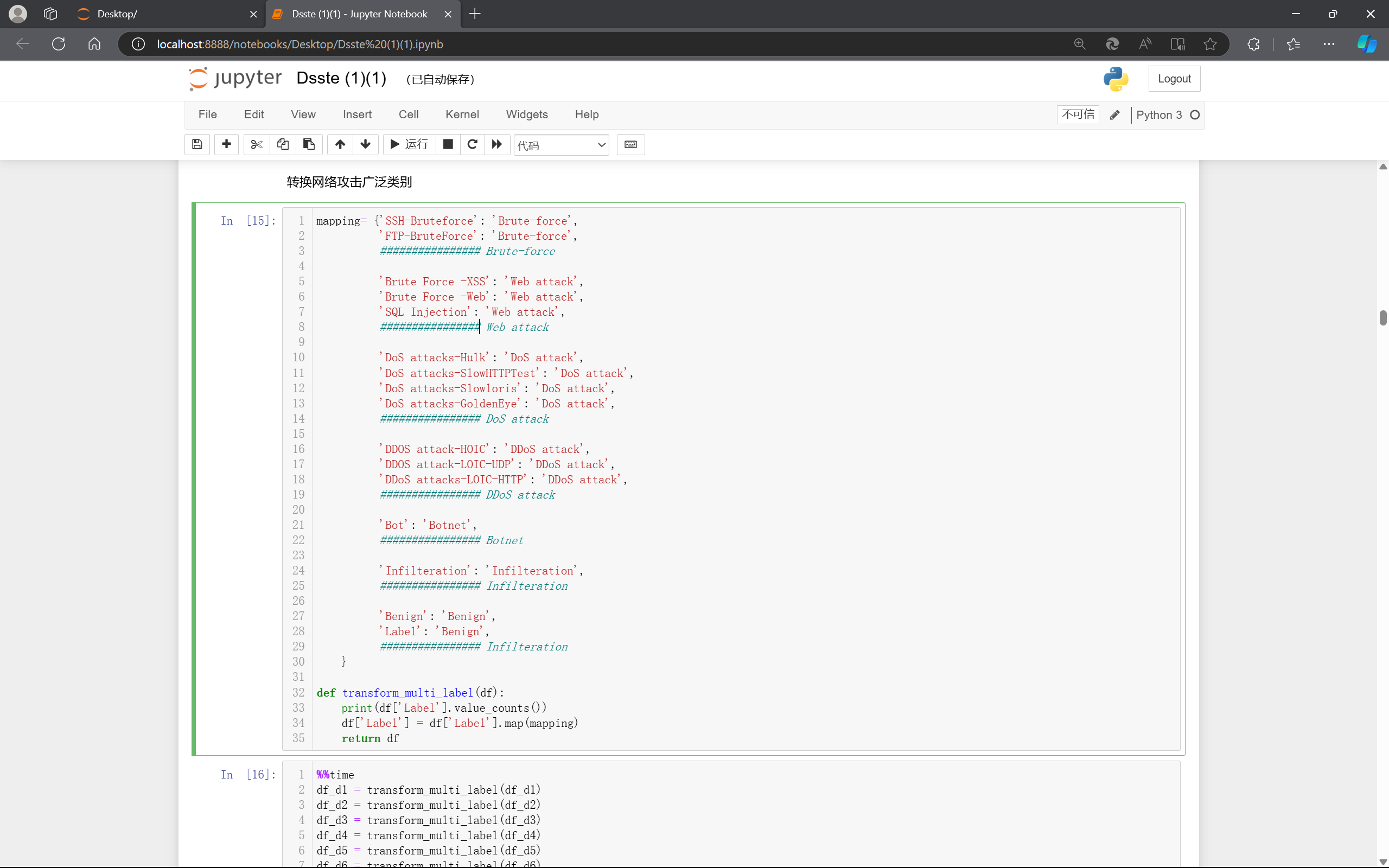
 **数据不平衡评估**：

* 通过value\_counts()打印出每个类别的数量，研究人员可以了解数据集中Benign和Malicious样本的分布情况。这对于处理数据不平衡问题非常重要。如果Malicious类别占比过低，则可能需要进一步的处理（例如过采样或欠采样）来平衡数据集。

输出的内容解释

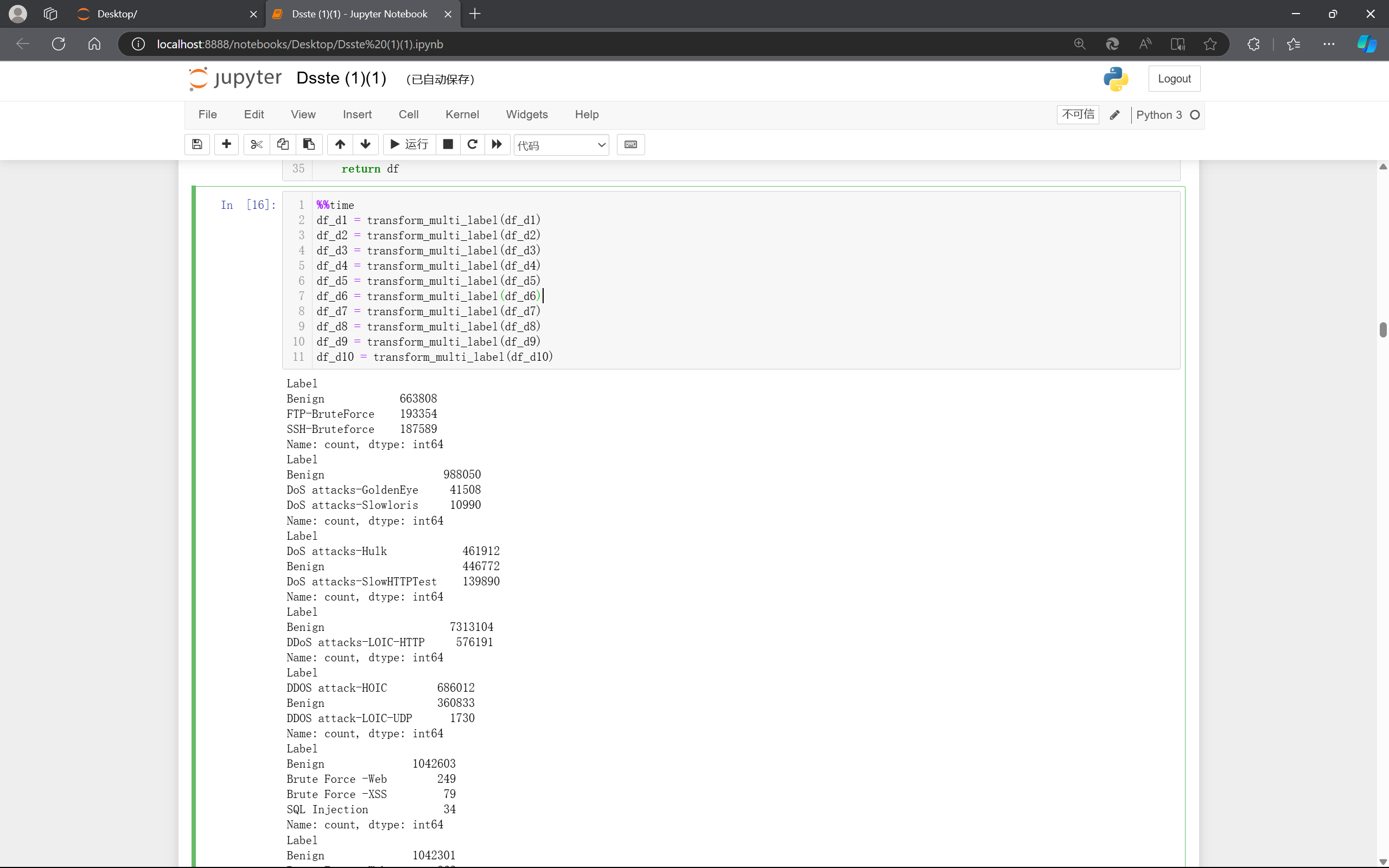
 输出中显示了每个数据框在Threat列中的唯一值（即['Benign', 'Malicious']），确认了二值化操作成功。

 另外，每个数据框中Threat列的类别数量也被打印出来。例如，第一个数据框中Benign有663808个样本，Malicious有380943个样本。这些信息可以帮助判断数据集的类别不平衡程度。



 这个mapping字典将原始数据集中各种攻击类型和正常流量映射为标准化标签。例如，所有的SSH-Bruteforce和FTP-BruteForce都被映射为'Brute-force'，而各种Web攻击（如Brute Force -XSS、SQL Injection等）都被归类为'Web attack'。

 这种映射方式可以将原本的多种标签标准化为少数几类，有助于简化分类任务。

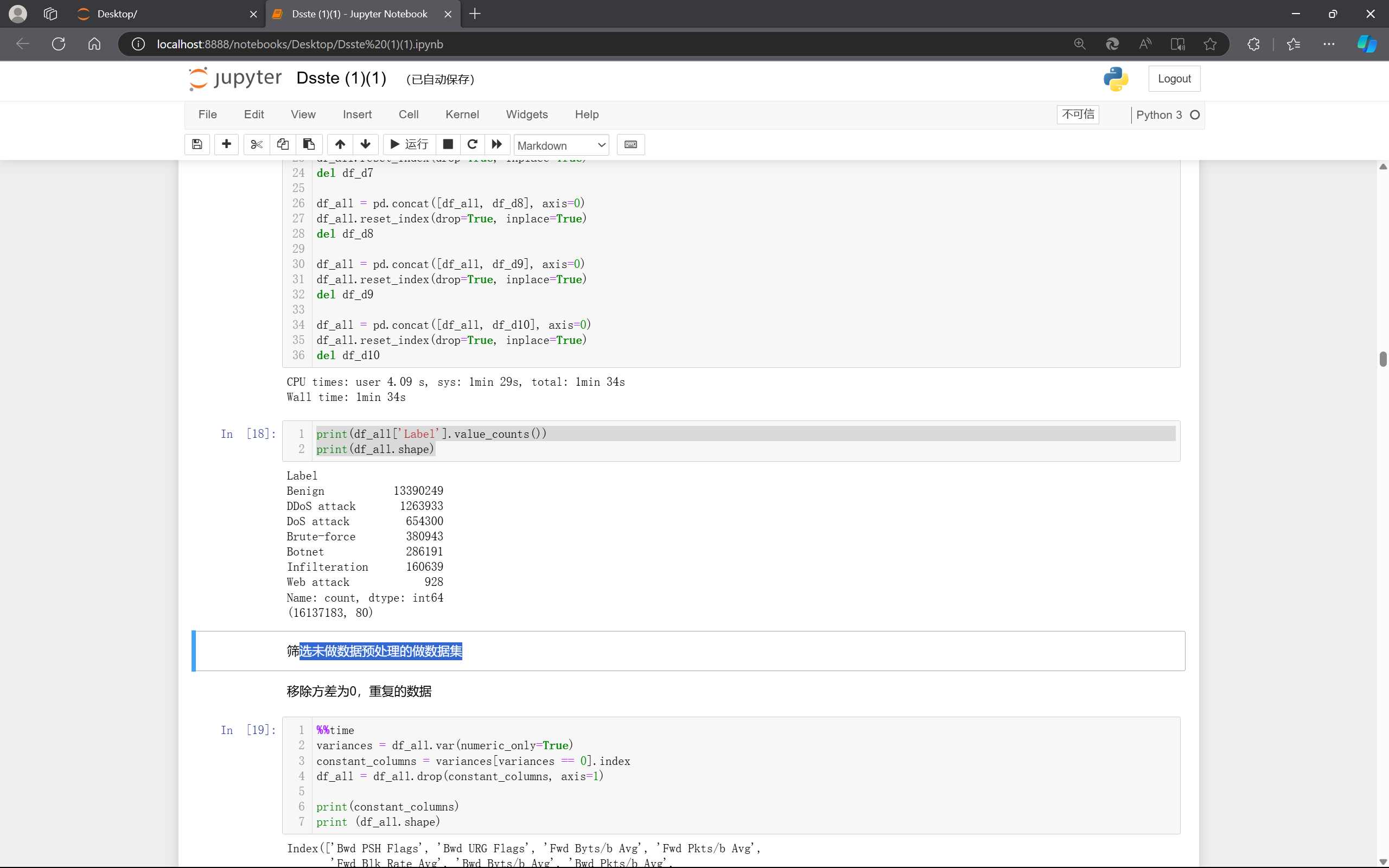


* + 这段代码将transform\_multi\_label函数依次应用到每个数据框df\_d1到df\_d10上，确保每个数据框中的Label列都经过了标准化的映射处理。
  + 函数的作用是使用之前定义的mapping字典，将原始数据集中多种形式的标签合并为几类标准标签（例如将各种类型的Web攻击都映射为"Web attack"）。
  + 这样做可以确保所有的数据框在标签上保持一致性，使得后续的数据处理和模型训练能够使用相同的标签结构。

1. **%%time魔术命令**
   * %%time是Jupyter Notebook中的魔术命令，用于测量整个代码块的执行时间。它显示了从df\_d1到df\_d10这十个数据框标签转换的总耗时。
   * 这对于评估标签转换步骤的性能和效率很有帮助，特别是在处理大型数据集时，可以确保该步骤在合理的时间内完成。

**这段代码在数据预处理中的作用**

1. **确保所有数据框标签的一致性**：
   * 这段代码保证所有数据框都进行了相同的标签映射和标准化处理。这样，所有的数据框在标签上保持一致，有助于避免在后续分析和模型训练中因标签不统一而产生错误。
2. **简化分类任务**：
   * 通过将原始数据集中的多种标签合并为标准化的几类（例如，将各种Web攻击映射为"Web attack"），这段代码将多分类问题简化为少数几类的分类问题。这不仅有助于提高模型的准确性，还能让模型更好地学习和识别模式。
3. **检查和验证标签分布**：
   * 通过在transform\_multi\_label函数中打印Label列的值分布情况，这段代码可以帮助研究人员了解每个数据框中的标签分布，并识别可能存在的类别不平衡问题。这对于后续的模型优化和数据平衡处理有很大帮助。



 **Benign**（正常流量）：13,390,249 个样本。这是数据集中占比最高的类别。

 **DDoS attack**：1,263,933 个样本。

 **DoS attack**：654,300 个样本。

 **Brute-force**（暴力破解攻击）：380,943 个样本。

 **Botnet**（僵尸网络）：286,191 个样本。

 **Infilteration**（渗透攻击）：160,639 个样本。

 **Web attack**（网络攻击）：928 个样本。  
也就是论文中的图 ，这是lable列的，不是之前二分类Threat列的内容

大概流程图

**数据预处理流程图**

**1. 数据加载**

* **步骤**：将所有CSV文件依次加载为单独的数据框（df\_d1到df\_d10）。
* **目的**：将原始数据导入Python环境，便于后续的清理和处理。
* **操作**：pd.read\_csv() 加载数据。

**2. 数据清洗**

* **步骤 1**：数据类型修正（fix\_data\_type 函数）
  + **目的**：确保各列的数据类型正确，减少类型不匹配错误。
  + **操作**：将数值列转换为整数或浮点型，去除非数值数据（如字符串 "Dst Port"）。
* **步骤 2**：处理无穷大值和缺失值（drop\_infinate\_null 函数）
  + **目的**：清除或填充无穷大值和缺失值，确保数据完整性。
  + **操作**：将无穷大值替换为缺失值（NaN），然后删除包含缺失值的行。
* **步骤 3**：删除无关列（drop\_unnecessary\_column 函数）
  + **目的**：去除与分析无关的列，如时间戳Timestamp列，减少数据维度。
  + **操作**：删除指定的无关列。

**3. 标签转换**

* **步骤**：多类别标签标准化（transform\_multi\_label 函数）
  + **目的**：将不同的攻击标签映射为少数几类（如Brute-force、DoS attack等），确保标签一致性。
  + **操作**：通过mapping字典将Label列的多类别标签归类为标准类别。

**4. 二分类标签生成**

* **步骤**：生成二分类标签（generate\_binary\_label 函数）
  + **目的**：将所有恶意流量标记为"Malicious"，正常流量标记为"Benign"，创建一个新的Threat列以供二分类分析。
  + **操作**：根据Label列的值生成Threat列。

**5. 内存优化**

* **步骤**：减少内存占用（reduce\_mem\_usage 函数）
  + **目的**：降低数据框的内存使用，特别是大规模数据集时提高效率。
  + **操作**：将数值列转换为更小的类型（如float32、int8等）以减少内存占用。

**6. 数据集合并**

* **步骤**：逐步合并数据框
  + **目的**：将所有单独的数据框合并成一个完整的数据集（df\_all），方便统一处理。
  + **操作**：使用pd.concat()将数据框按行合并，并删除已合并的数据框以释放内存。

**7. 特征清理**

* **步骤 1**：删除常数列
  + **目的**：移除方差为零的列（即所有值都相同的列），减少数据冗余。
  + **操作**：计算每列方差，删除方差为零的列。
* **步骤 2**：查找和删除重复列
  + **目的**：去除与其他列完全相同的列，进一步减少冗余信息。
  + **操作**：使用双重循环检查各列相同性，删除被标记为重复的列。

**8. 保存清理后的数据**

* **步骤**：保存清理和优化后的数据框
  + **目的**：将最终清理好的数据框保存为CSV文件，便于后续分析和模型训练。
  + **操作**：to\_csv()将数据框保存为df\_all\_nosample.csv文件。

逻辑回归

Logistic Regression（逻辑回归）是一种基础的线性分类模型，通过最大化数据的似然函数来预测目标分类的概率。它适用于二分类或多分类任务，常用于基准测试或对比分析。我们在本实验中将逻辑回归模型作为参考基准模型，以评估其在检测恶意流量方面的效果。结果显示，逻辑回归模型的准确率为0.881，F1分数为0.782，尽管表现尚可，但相比其他复杂模型（如深度强化学习模型G-RXAD）的效果略逊一筹。这表明逻辑回归在处理复杂网络流量分类任务时可能有局限性，而更复杂的模型可以提供更高的检测准确率和更好的分类平衡。

Knn

K近邻算法（K-Nearest Neighbors, KNN）是一种基于距离的非参数分类模型，通过计算待分类样本与训练集中最近邻样本的距离来确定其类别。KNN在小数据集和低维空间中表现良好，但在高维度和大数据集中计算开销较高。我们在本实验中将KNN作为对比基准模型之一，以评估其在检测恶意流量方面的表现。实验结果表明，KNN模型的准确率为0.927，F1分数为0.907，显示出较好的分类效果，但与其他更复杂的模型（如G-RXAD）相比略有不足。这也表明，尽管KNN在分类任务中有一定的优势，但在复杂的网络流量场景下，其性能可能不及深度学习或强化学习模型。