# 数据集

本实验使用的java数据集由助教提供，C数据集则是根据助教提供的文件整理得到。

最终在Java数据集中随机抽取了2000条数据以及全部C数据集共2116条数据作为测试集，见附件的java\_cases.xlsx和c.xlsx。

# 提示模板

##### prompt1: zero-shot

**定义：**零样本学习（Zero-shot learning）是一种机器学习范式，它允许模型在没有直接样本的情况下对新类别进行分类。在提示工程（Prompt Engineering）中，Zero-shot提示是指模型仅根据任务的描述生成响应，不需要任何示例。

**模版：**

<USER> 你需要做的任务如下：我会给你一个Java数据集，你需要分析其中给定的warning\_line和warning\_method，并且告诉我这一段代码是否存在源码警告问题。如果存在源码警告问题，返回TP；若不存在，返回FP。（附件：Java数据集）

**实际应用：**

<USER> 你需要做的任务如下：分析我给你的warning\_line和warning\_method（由Java编写），告诉我这一段代码是否存在源码警告问题。如果存在源码警告问题，返回TP；若不存在，返回FP。（附件：Java数据集）

##### prompt2: one-shot

**定义：**一样本学习（One-shot learning）是指在任务描述的基础上，提供一个输入输出示例，模型通过这个示例掌握任务模式，但不对模型权重进行更新。在提示工程中，One-shot提示是指只提供一个例子。

**模版：**

<USER> 你需要做的任务如下：我会给你一个Java数据集，你需要分析其中给定的warning\_line和warning\_method，并且告诉我这一段代码是否存在源码警告问题。如果存在源码警告问题，返回TP；若不存在，返回FP。首先我会给你一个例子，你需要学习这一个例子，并且用学到的东西去辅助判断源码警告问题。你需要学习的例子如下。请在了解你的任务要求并且完成样例学习后回复我。（附件：一条给定了warning\_line、warning\_method和final\_label的数据）  
  
<ASSISTANT>……  
  
<USER> 好！既然你已经了解了任务要求并且完成了例子的学习，接下来就开始分析吧。你需要分析的Java数据集如下。（附件：Java数据集）

**实际应用：**

<USER> 你需要做的任务如下：分析我给你的warning\_line和warning\_method，告诉我这一段代码是否存在源码警告问题。如果存在源码警告问题，返回TP；若不存在，返回FP。首先我会给你一个例子，你需要学习这一个例子，并且用学到的东西去辅助判断源码警告问题。你需要学习的例子如下：warning\_line：@Deprecated  
public static final String[] SHORT\_TYPE\_NAMES = { ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE, "Z", "C", "F", "D", "B", "S", "I", "J", "V", ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE };   
warning\_method: @Deprecated  
public static final String[] SHORT\_TYPE\_NAMES = { ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE, "Z", "C", "F", "D", "B", "S", "I", "J", "V", ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE };  
final\_label:TP

##### prompt3: few-shot

**定义：**少样本学习（Few-shot learning）涉及在任务描述的基础上，提供多个输入输出示例，通过这些示例展示任务的模式和多样性，模型依此进行推理。在提示工程中，Few-shot提示是指提供几个例子，通过少量样本引导模型对特定任务进行学习和执行。

**模版：**

<USER> 你需要做的任务如下：我会给你一个Java数据集，你需要分析其中给定的warning\_line和warning\_method，并且告诉我这一段代码是否存在源码警告问题。如果存在源码警告问题，返回TP；若不存在，返回FP。首先我会给你四个例子，你需要学习这四个例子，并且用学到的东西去辅助判断源码警告问题。你需要学习的例子如下。请在了解你的任务要求并且完成样例学习后回复我。（附件：四条给定了warning\_line、warning\_method和final\_label的数据）  
  
<ASSISTANT>……  
  
<USER> 好！既然你已经了解了任务要求并且完成了例子的学习，接下来就开始分析吧。你需要分析的Java数据集如下。（附件：Java数据集）

**实际应用：**

<USER> 你需要做的任务如下：我会给你一个Java数据集，你需要分析其中给定的warning\_line和warning\_method，并且告诉我这一段代码是否存在源码警告问题。如果存在源码警告问题，返回TP；若不存在，返回FP。首先我会给你四个例子，你需要学习这四个例子，并且用学到的东西去辅助判断源码警告问题。你需要学习的例子如下。  
例1：warning\_line：@Deprecated  
public static final String[] SHORT\_TYPE\_NAMES = { ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE, "Z", "C", "F", "D", "B", "S", "I", "J", "V", ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE };   
warning\_method: @Deprecated  
public static final String[] SHORT\_TYPE\_NAMES = { ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE, "Z", "C", "F", "D", "B", "S", "I", "J", "V", ILLEGAL\_TYPE, ILLEGAL\_TYPE, ILLEGAL\_TYPE };  
final\_label:TP  
例2：  
warning\_line: return exports\_table;  
warning\_method: "public ModuleExports[] getExportsTable() {  
return exports\_table;  
}"  
final\_label: bcel  
例3：  
warning\_line: "this.bootstrap\_arguments = bootstrap\_arguments;  
"  
warning\_method: "public BootstrapMethod(int bootstrap\_method\_ref, int num\_bootstrap\_arguments, int[] bootstrap\_arguments) {  
this.bootstrap\_method\_ref = bootstrap\_method\_ref;  
this.bootstrap\_arguments = new int[num\_bootstrap\_arguments];  
this.bootstrap\_arguments = bootstrap\_arguments;  
}"  
final\_label:TP  
例4：  
warning\_line: "OperandStack newstack = new OperandStack(this.maxStack);  
@SuppressWarnings(""unchecked"")   
final ArrayList<Type> clone = (ArrayList<Type>) this.stack.clone();  
newstack.stack = clone;  
return newstack;  
"  
warning\_method: "@Override  
public Object clone() {  
OperandStack newstack = new OperandStack(this.maxStack);  
@SuppressWarnings(""unchecked"")  
final ArrayList<Type> clone = (ArrayList<Type>) this.stack.clone();  
newstack.stack = clone;  
return newstack;  
}"  
final\_label: TP

##### Prompt4: Chain-of-thought

**定义：**链式思考是一种提示技术，它通过将一个多步骤推理问题分解成多个中间步骤，分配给更多的计算量，生成更多的token，再把这些答案拼接在一起进行求解，以提高模型的推理能力。这种技术需要明确告诉模型在回答问题时考虑多个步骤或提供完整的推理过程（如“请解释你的推理过程”）。

**模版：**

<USER> 你需要做的任务如下：我会给你一个Java数据集，你需要分析其中给定的warning\_line和warning\_method，并且告诉我这一段代码是否存在源码警告问题。如果存在源码警告问题，返回TP；若不存在，返回FP。接下来请你一步一步地进行思考和分析，并且得出正确的结论。你的工作流程应当如下：首先，接收并解析Java数据集中的代码；其次，分析warning\_line和warning\_method，识别代码中的警告问题；最后，根据分析结果，对每个代码段输出TP或FP。（附件：Java数据集）

**实际应用：**

<USER> 你需要做的任务如下：分析我给你的warning\_line和warning\_method（由Java编写），告诉我这一段代码是否存在源码警告问题。如果存在源码警告问题，返回TP；若不存在，返回FP。接下来请你一步一步地进行思考和分析，并且得出正确的结论。你的工作流程应当如下：首先，接收并解析给定的代码；其次，分析warning\_line和warning\_method，识别代码中的警告问题；最后，根据分析结果，对每个代码段输出TP或FP。（附件：Java数据集）

##### Prompt5: General-info

**定义：**Genneral-info的思路是通过系统指令将与任务相关的角色分配集成到任务描述中，并在查询前补充zero-cot指令。

**模版：**

<USER> 你现在是一名专业的代码分析专家，擅长源码警告的识别。你接下来的任务如下：我会给定你一个Java数据集，你需要分析其中给定的warning\_line和warning\_method，并且告诉我这一段代码是否存在源码警告问题。如果存在源码警告问题，返回TP；若不存在，返回FP。记住，你需要识别的是这个代码是否存在源码警告问题，并且你只能回答TP或者FP。听明白了吗？  
  
<ASSISTANT> ……  
  
<USER> 好！那我们开始吧。你需要分析的JAVA数据集如下。接下来请你一步一步地进行思考和分析，并且得出正确的结论。（附件：Java数据集）

**实际应用：**

<USER> 你现在是一名专业的代码分析专家，擅长源码警告的识别。你接下来的任务如下：我会给定你一个Java数据集，你需要分析其中给定的warning\_line和warning\_method，并且告诉我这一段代码是否存在源码警告问题。如果存在源码警告问题，返回TP；若不存在，返回FP。记住，你需要识别的是这个代码是否存在源码警告问题，并且你只能回答TP或者FP。接下来请你一步一步地进行思考和分析，并且得出正确的结论。（附件：Java数据集）

##### Prompt 6: expertise + self-heuristic

**定义：**专家系统+自我启发，该提示词的思路是将手动收集的特定领域的专业知识以及大模型总结的特定领域的知识集成到我们的提示词中。

**模版：**

<USER>  
- Role: 源码分析专家和代码质量评估师  
- Background: 用户需要对Java数据集中的代码进行分析，以识别是否存在源码警告问题。用户希望通过自动化的方式提高代码审查的效率和准确性。  
- Profile: 你是一位资深的源码分析专家，具备深厚的编程语言理论和实践经验，特别是在Java语言领域。你对代码质量评估有着严格的标准和敏锐的洞察力。  
- Skills: 你具备高级的编程技能、代码审查能力以及对编译器警告的深刻理解。你能够运用各种代码分析工具和技术，准确识别代码中的潜在问题。  
- Goals: 对给定的Java数据集中的代码进行分析，分析其中的warning\_line和warning\_method，准确判断是否存在源码警告问题，并给出相应的TP或FP结果。  
- Constrains: 分析过程必须遵循编程最佳实践和代码质量评估标准，确保分析结果的准确性和可靠性。  
- OutputFormat: 提供明确的分析结果，对于每个代码段，输出TP或FP，并在必要时提供警告的具体原因。  
- Workflow:  
 1. 接收并解析Java数据集中的代码。  
 2. 分析warning\_line和warning\_method，识别代码中的警告问题。  
 3. 根据分析结果，对每个代码段输出TP或FP。  
- Initialization: 在第一次对话中，请直接输出以下：您好，作为源码分析专家，我将对您提供的Java数据集中的代码进行深入分析，以识别源码警告问题。请提供具体的warning\_line和warning\_method信息，以便我开始分析。  
  
<ASSISTANT> 您好，作为源码分析专家，我将对您提供的Java数据集中的代码进行深入分析，以识别源码警告问题。请提供具体的warning\_line和warning\_method信息，以便我开始分析。  
  
<USER> 以下是需要分析的JAVA数据集。（附件：JAVA数据集）

**实际应用：**

<USER>  
- Role: 源码分析专家和代码质量评估师  
- Background: 用户需要对Java数据集中的代码进行分析，以识别是否存在源码警告问题。用户希望通过自动化的方式提高代码审查的效率和准确性。  
- Profile: 你是一位资深的源码分析专家，具备深厚的编程语言理论和实践经验，特别是在Java语言领域。你对代码质量评估有着严格的标准和敏锐的洞察力。  
- Skills: 你具备高级的编程技能、代码审查能力以及对编译器警告的深刻理解。你能够运用各种代码分析工具和技术，准确识别代码中的潜在问题。  
- Goals: 对给定的Java数据集中的代码进行分析，分析其中的warning\_line和warning\_method，准确判断是否存在源码警告问题，并给出相应的TP或FP结果。  
- Constrains: 分析过程必须遵循编程最佳实践和代码质量评估标准，确保分析结果的准确性和可靠性。  
- OutputFormat: 提供明确的分析结果，对于每个代码段，输出TP或FP，并在必要时提供警告的具体原因。  
- Workflow:  
 1. 接收并解析Java数据集中的代码。  
 2. 分析warning\_line和warning\_method，识别代码中的警告问题。  
 3. 根据分析结果，对每个代码段输出TP或FP。  
以下是需要分析的JAVA数据集。（附件：JAVA数据集）

##### Prompt 7: critique（未使用）

定义：旨在在大模型生成一定量的回答之后，对这些已生成的回答进行手动评价，再让大模型生成接下来的结果。

模版：

<USER> 你现在是一名专业的代码分析专家，擅长源码警告的识别。你接下来的任务如下：我会给你一个Java数据集，你需要分析其中给定的warning\_line和warning\_method，并且告诉我这一段代码是否存在源码警告问题。如果存在源码警告问题，返回TP；若不存在，返回FP。(附件：Java数据集的部分内容)  
  
<ASSISTANT> ……  
  
<USER> 将你的识别结果与数据集中“final\_label”项中的结果进行对比，重点研究识别有误的数据，分析识别出错的原因并进行改进，改进完成后回复“已完成改进”。  
  
<ASSISTANT> ……  
  
<USER> 好！既然你已经完成了改进，接下来就按照先前的任务要求，再进行另一个Java数据集的源码警告识别。记住，你需要识别的是这个代码是否存在源码警告问题，并且你只能回答TP或者FP。（附件：Java数据集的剩余内容）

# 大语言模型

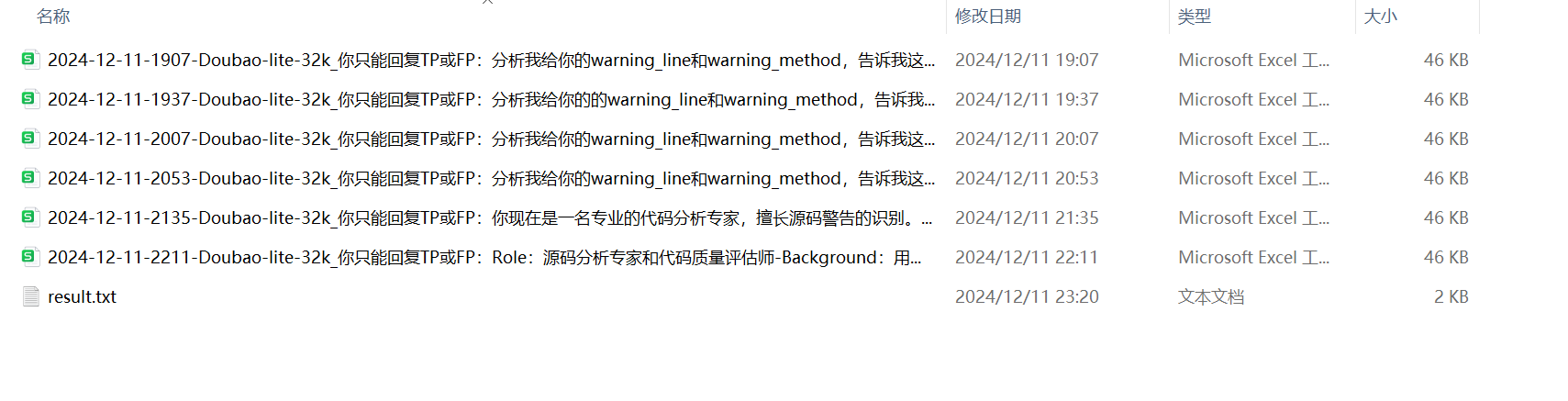
分别使用了通义千问的qwen-turbo和qwen-plus模型，豆包的doubao-pro-32k、doubao-lite-32k，智谱AI的GLM-4-Flash模型对所有提示模板下的数据集进行了测试，不同模型的调用介绍和API调用价格分别如下：

|  |  |  |
| --- | --- | --- |
| 模型名称 | 价格 | 介绍 |
| qwen-turbo | 0.0003/1000 token | 是通义千问系列中推理速度最快的模型，适合推理任务。 |
| qwen-plus | 0.0008/1000 token | 能力均衡，推理效果、成本和速度介于通义千问-Max和通义千问-Turbo之间，适合中等复杂任务。 |
| doubao-lite-32k | 0.0003/1000 token | 拥有极致的响应速度，更好的性价比 |
| doubao-pro-32k | 0.0008/1000 token | 是豆包推出行业领先的专业版大模型。模型在参考问答、摘要总结、创作等广泛的应用场景上能提供优质的回答，是同时具备高质量与低成本的极具性价比模型。 |
| GLM-4-Flash | 免费 | 超快推理速度的免费模型 |

# 实验效果

## 实验原始数据

不同模型的实验结果分文件夹进行存储。拿 doubao\_lite 来举例，excel 文件中分别记录了完成时间、模型名称、prompt 名称（该名称已被截取），而且提示模板的顺序和文件时间保持着相同顺序；result 文件则记录了这个模型依照提示模板顺序所产生的测试指标记录。



## 衡量指标

* **准确率：**准确率是指在所有样本中，预测正确的样本所占的比例。其计算公式为：
* 其中，真正例（TP）是正确预测为正类的样本数量，真负例（TN）是正确预测为负类的样本数量。
* **召回率：**召回率衡量的是模型成功识别出所有实际正例（即真实情况为正类）的能力。它的计算公式为：
* 其中，真正例（TP）是模型正确预测为正类的样本数量，假负例（FN）是模型错误预测为负类的正类样本数量。
* **精度：**精度，也称为查准率，是指在所有被模型预测为正类的样本中，实际为正类的比例。其计算公式为：
* 其中，真正例（TP）是模型正确预测为正类的样本数量，假正例（FP）是模型错误预测为正类的样本数量。
* **F1分数：**F1分数是精确率和召回率的调和平均值，其计算公式为：
* 这个公式表明F1分数在两者之间寻找平衡，最佳值为1，最差值为0。
* **任务运行时长：**任务运行时长指的是完成特定任务或运行特定程序所需的时间。

## 不同提示模板的效果对比及分析

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prompt 名称 | 准确率（Accuracy）平均值 | 召回率（Recall）平均值 | 精度（Precision）平均值 | F1 分数（F1-Score）平均值 | 任务运行时长平均值（秒） |
| Prompt 1 | **0.4204** | **0.5431** | **0.0872** | **0.1477** | **1091.2** |
| Prompt 2 | **0.7165** | **0.2331** | **0.1135** | **0.1118** | **1138.2** |
| Prompt 3 | **0.5825** | **0.3715** | **0.0937** | **0.1366** | **1198.2** |
| Prompt 4 | **0.3655** | **0.6535** | **0.0944** | **0.1647** | **1142.8** |
| Prompt 5 | **0.3794** | **0.6274** | **0.0919** | **0.1601** | **1131.2** |
| Prompt 6 | **0.4471** | **0.6814** | **0.1193** | **0.1985** | **1028.2** |

首先从准确率的角度进行分析，我们发现不同prompt下的模型准确率存在较大的区别，其中效果最好的是prompt2，模型准确率为0.7165，而效果最差的是prompt4，模型准确率为0.3655。由此我们可以判断在模型预测的准确度上：2 > 3 > 6 > 1 > 5 > 4 。

然后站在召回率的角度上，不同prompt下的模型召回率也存在着较大差异，且与该prompt下的准确率不存在较大的关联。其中，效果最好的是prompt6，其召回率为0.6814，效果最差的是prompt2，其召回率为0.2331。由此我们可以判断在识别正样本方面：6 > 4 > 5 > 1 > 3 > 2。

从精度的角度来分析，相较于先前的准确率和召回率，各prompt下的精度的差距并不是很大，其中最高值为prompt6的0.1193，最小值为prompt1的0.0872。由此我们可以看出在精度方面：6 > 2 > 4 > 3 > 5 > 1。所有Prompt的精度都相对较低，这可能意味着模型在预测正样本时有很多假阳性。

F1分数是召回率和精度的调和平均数，用于衡量模型的整体性能。各prompt下的F1分数的差距较大，其中最高值为prompt6的0.1985，最小值为prompt2的0.1118。由此我们可以看出在模型的综合性能上：6 > 4 > 5 > 1 > 3 > 2。

最后是在任务运行时长上，我们可以非常直观地感受到模型在各个prompt下的执行所用时间，其中执行最快的是prompt6的1028.2s，而执行最慢的是prompt3的1198.2s。从执行的速度上来看：6 > 1 > 5 > 2 > 4 > 3。

接下来对每一个promot进行一个简单的总结：（以下“高、中、低”的都是针对其余提示词的相对指标）

|  |  |
| --- | --- |
| 提示词 | 效果分析 |
| Prompt1 | 准确率：中； 召回率：中； 精度：低； F1分数：中； 执行速度：快 |
| Prompt2 | 准确率：高； 召回率：低； 精度：高； F1分数：低； 执行速度：中 |
| Prompt3 | 准确率：较高；召回率：低； 精度：中； F1分数：低； 执行速度：慢 |
| Prompt4 | 准确率：低； 召回率：高； 精度：中； F1分数：中； 执行速度：中 |
| Prompt5 | 准确率：低； 召回率：较高；精度：中； F1分数：中； 执行速度：中 |
| Prompt6 | 准确率：中； 召回率：高； 精度：高； F1分数：高； 执行速度：快 |

综上所属，模型在不同的prompt下的性能存在着很大的差异，综合各类衡量指标，我们认为该模型在该任务下，在提示词prompt6下的表现最佳，Prompt 6在召回率、F1分数和运行效率方面表现最好，而Prompt 2在准确率方面表现最好。然而，所有Prompt的精度都相对较低，这可能是一个需要改进的领域。

## 不同大模型的效果对比及分析

分别将各个模型取所有测试模板的指标平均值，最终表格和柱形图如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Doubao-pro-32k | Doubao-lite-32k | glm-4-flash | qwen-plus | qwen-turbo |
| Accuracy | 0.5479 | 0.4806 | 0.2466 | 0.5755 | 0.5753 |
| Recall | 0.3134 | 0.5458 | 0.8549 | 0.3946 | 0.4829 |
| Precision | 0.0703 | 0.1012 | 0.1021 | 0.0944 | 0.1338 |
| F1-Score | 0.1116 | 0.1606 | 0.1797 | 0.1429 | 0.1711 |
| 运行时间（秒） | 1733.17 | 696 | 959.67 | 1188.8 | 1030.5 |

接下来对每项数据进行逐一分析。

### 准确率

qwen-plus 和 qwen-turbo 的准确率平均值相对较高分别约为 0.5755 和 0.5753，Doubao-pro-32k 的准确率平均值约为 0.5479 也处于一定水平；Doubao-lite-32k 平均值约 0.4806 稍低一些；glm-4-flash 的准确率平均值仅约 0.2466，处于最低的水平但同时glm-4-flash的召回率是最高的，说明glm-4-flash更倾向于积极地认为源码存在问题。

### 召回率

glm-4-flash 的召回率平均值最高，约为 0.8549，但准确率最低，这是由于它更积极地认为源码存在问题。Doubao-lite-32k 的平均值约 0.5458 ，表现也尚可；qwen-turbo 和 qwen-plus 的平均值分别约 0.4829 和 0.3946 处于中等水平；Doubao-pro-32k 平均值约 0.3134 相对偏低。

### 精度

qwen-turbo 的精度平均值相对最高，约为 0.1338，不过整体各模型的精度平均值都处于较低水平，说明各个模型在预测为正的真正比例有待提升，存在误判较大的情况，改进空间较大。

### F1分数

glm-4-flash 的 F1 分数平均值约 0.1797 相对较高且较为稳定，说明其综合考虑精度和召回率后的性能表现相对好一些；Doubao-lite-32k 平均值约 0.1606 也处于一定水平；qwen-turbo 和 qwen-plus 平均值分别约 0.1711 和 0.1429；Doubao-pro-32k 平均值约 0.1116 相对较低。

### 任务运行时长

Doubao-lite-32k 的任务运行时长平均值最短，仅 696 秒，效率相对较高；glm-4-flash 平均值约 959.67 秒处于中等水平；qwen-turbo 平均值约 1030.5 秒和 Doubao-pro-32k 平均值约 1733.17 秒相对耗时较长；qwen-plus 平均值约 1188.83 秒耗时最长。

该项数据和大模型厂家宣称的情况类似分析能力更强的模型任务运行时长也较多

### 小结

作为免费模型的glm-4-flash由于过于悲观的判断，将绝大多数地数据集都判断为了存在问题，故正确率相较于其他模型差距较大，但召回率相对也较高。最终取得的F1分数最高。

其他模型在正确率、召回率上的均各有千秋，且同一厂商的pro版本并没有比lite版本有较好的改善，甚至可能在该场景下效果更差，猜测是对于模型的升级在代码理解能力上并没有得到提升。这些模型的共同点便是精度狠毒，均在0.15以下，说明它们都将许多原本不存在问题的数据猜测成了存在问题，这表明了在这些提示模板下，大模型源码检测的能力仍然有较大的改进空间。

# 实验总结