机器学习概论-垃圾邮件分类实验报告

计 63 李志鹏 2016011283

2019 年 3 月 27 日

1 任务描述

使用朴素贝叶斯分类器对中文邮件进行垃圾邮件的分类。数据集为 64620 封中文邮件。要求手写核心代码，使用 5-交叉验证方法，并对实现的分类器的性能进行分析和讨论，尝试提升分类性能并解释。

2 实验设计

使用朴素贝叶斯算法，从训练集中提取 feature 并使用贝叶斯方法对测试集/验证集的邮件进行分类。按照要求，我使用了 5-交叉折叠验证的方式进行性能评测。此外，我使用了四种评价方式对结果进行评价，从不同角度展示分类器的性能。

2.1 朴素贝叶斯算法

朴素贝叶斯算法的原理概括如下。根据朴素贝叶斯的条件独立性假设，选取后验概率最大的类别作为分类结果

$$\widehat{y_k} = argmax_{w_i} P(x_k|w_i) \tag{1}$$

2.2 训练集/验证集的分割

按照实验要求使用 5-交叉验证方法，我没有分出传统意义上的测试集，而是每次随机将所有数据分为 5 份，每次随机选择一个作为测试集，其他四个作为数据集，得到五次分类结果后进行加和平均，结果作为当前分类器的最终性能。使用交叉验证方法的优点在于，避免了从测试集获得的反馈信息干扰了对于分类器的进一步优化和改进。可以较为客观的反应分类器的性能。

如无特殊说明，所有结果均是通过 5-交叉验证得到的。

2.3 性能评估方法

我采用了四种评价方式来对分类器的性能进行评价。

2.3.1 Accuracy

准确率 (Accuracy) 代表的是分类器整体的性能，在所有的测试集上分类正确的比例。但是当某一分类的数据太少，完全错分类依然可以达到很高的准确率。

2.3.2 Precision

精确率 (Precision) 是在所有分类为正的样本中有多少是正样本，代表分类为正的结果的可信度，用在期望得到的正分类全部正确的情况下。

2.3.3 Recall

召回率 (Recall) 是在所有正样本中被正确分类的比例，代表得到的正样本的比例，用在期望得到全部正样本的情况下。

2.3.4 F1-Measure

F-Measure 是对 Recall 和 Precision 的加权调和平均，因为 Recall 和 Precision 有时会呈现矛盾的情况，此时需要 F-Measure 进行综合的体现。

3 实验结果和讨论

下面对分类器的性能进行分析和讨论。讨论的顺序是按照我优化分类器时迭代的版本顺序，因此相关 Issue 的讨论顺序为 1-3-2-3。

3.1 Baseline

首先我实现了一个 naive 版本的分类器。我直接使用正则表达式在邮件中匹配出所有单个汉字，每个汉字作为独立的 feature 计算概率。同时，没有进行任何的优化，直接使用贝叶斯公式进行计算。使用了 100% 的训练集。

3.1.1 实验结果

得到的结果为

Accuracy	0.4550
Precision	0.9682
Recall	0.1842
F1-Measure	0.3095

表 1: Baseline 结果

3.1.2 性能分析和讨论

Baseline 版本只是实现了最基本的朴素贝叶斯分类器，没有加入任何的优化。可以看到得出的结果并不理想，这也意味着后面还有很大的提高空间。

3.2 数值计算优化

在计算后验概率时，显然每一个汉字出现的概率都是小于 1 的，那么当相当多个汉字的概率按照公式相乘后，得到的结果十分容易超出浮点数的存储限制，从而使得计算结果错误，进而影响分类结果。因此我在计算概率时对每个概率值取对数，将其结果累加，得到最终结果。

3.2.1 实验结果

Accuracy	0.9296
Precision	0.9540
Recall	0.9392
F1-Measure	0.9465

表 2: 数值计算优化结果

3.2.2 性能分析和讨论

很明显，将概率取对数对结果的改进是巨大的。说明在概率的累乘计算中很容易出现超出浮点数表示范围的情况，也符合我的猜想和预期。

如无特殊说明，以下所有结果均使用对数计算概率。

3.3 Issue1-训练集大小

直观来讲，用于学习的“经验”越多越充分，那么学习得到的“知识”也将越准确越鲁棒。“经验”体现在朴素贝叶斯算法上，即为计算频数的训练集大小。从本质上来讲，朴

素贝叶斯依赖于各个数据的“概率”的估计，通过频率估计概率，这要求频数达到一定的大小才可以保证，因此需要足够大的数据量才可以保证准确率。为了验证这一点，我分别从训练集中随机选择了 0.1%,1%,5%,50%,100% 的数据用于分类器的学习。

3.3.1 实验结果

学习不同大小训练数据的分类器的结果如下。

	Accuracy	Precision	Recall	F1-Measure
Min	0.7059	0.6579	0.8261	0.7692
Max	0.8571	0.9118	0.9259	0.8866
Average	0.8017	0.8170	0.8866	0.8468

表 3: 0.1% 训练集

	Accuracy	Precision	Recall	F1-Measure
Min	0.8581	0.9024	0.8863	0.9019
Max	0.9365	0.9634	0.9610	0.9518
Average	0.8993	0.9329	0.9187	0.9253

表 4: 1% 训练集

	Accuracy	Precision	Recall	F1-Measure
Min	0.9133	0.9380	0.9230	0.9343
Max	0.9403	0.9655	0.9438	0.9545
Average	0.9255	0.9518	0.9355	0.9436

表 5: 5% 训练集

	Accuracy	Precision	Recall	F1-Measure
Min	0.9264	0.9504	0.9333	0.9439
Max	0.9324	0.9601	0.9443	0.9485
Average	0.9290	0.9538	0.9383	0.9460

表 6: 50% 训练集

	Accuracy	Precision	Recall	F1-Measure
Min	0.9274	0.9502	0.9357	0.9448
Max	0.9315	0.9552	0.9422	0.9481
Average	0.9299	0.9537	0.9399	0.9467

表 7: 100% 训练集

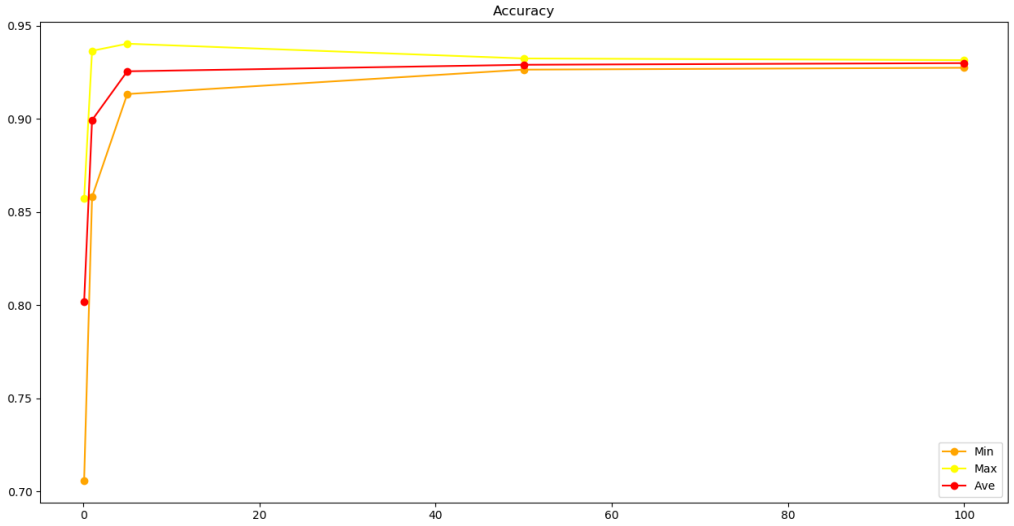

图 1: Accuracy

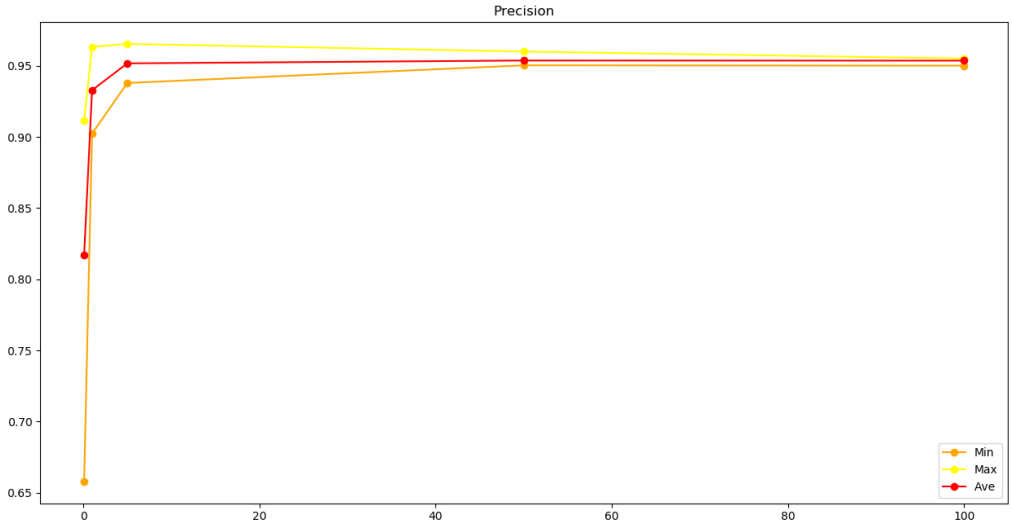

图 2: Precision

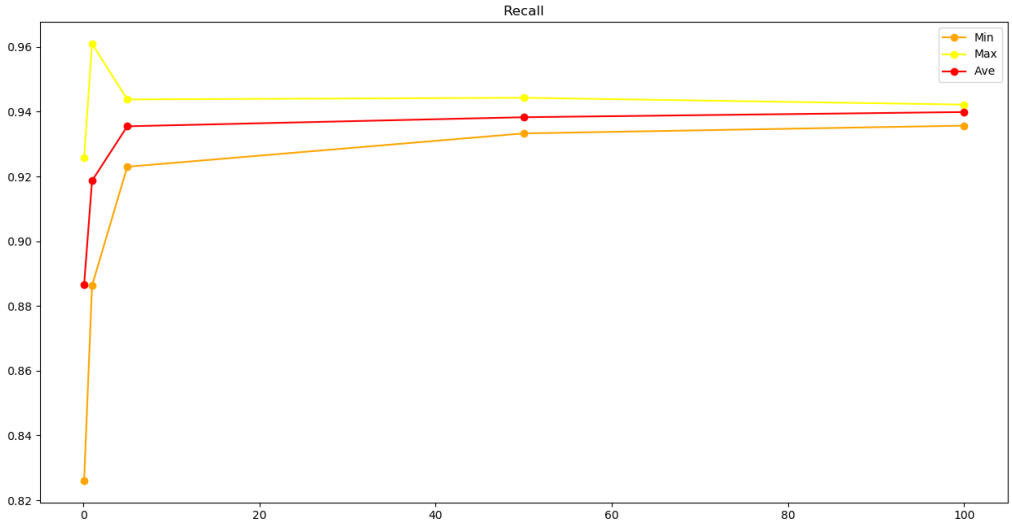

图 3: Recall

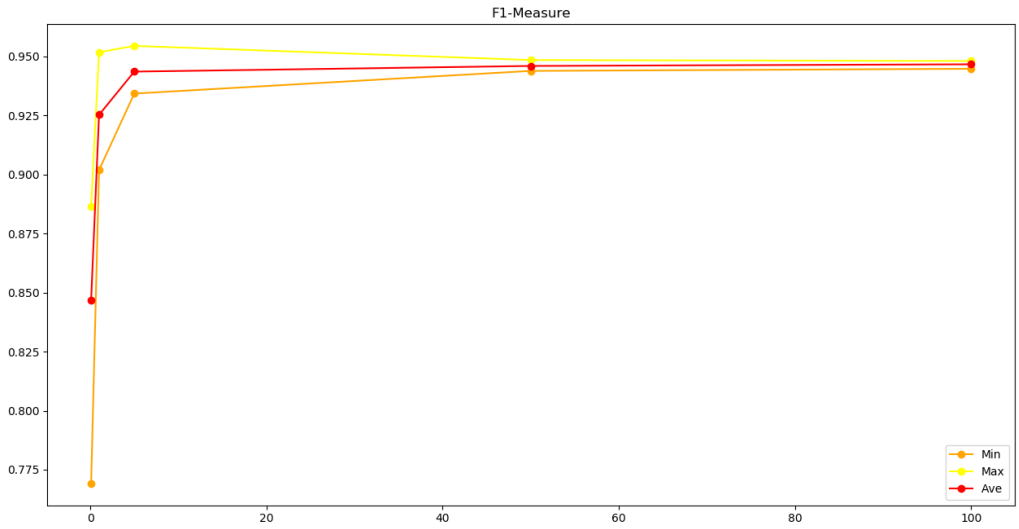

图 4: F1-Measure

3.3.2 性能分析和讨论

从表格和图像中很明显可以看出，随着训练集的增大，分类器的准确率变高。这说明训练集的大小对于分类器的准确性有着决定性的作用。

具体来看，使用 50% 的训练集和使用全集进行训练得到的结果几乎没有区别，使用 5% 的训练数据就已经能得到相当高的准确率了，这一方面说明当数据已经足够多时，更多的数据并不能提升朴素贝叶斯分类器的性能。

另一方面，当训练数据不足时，朴素贝叶斯分类器的性能不仅不够高，同时也不够稳定，不能作为成熟稳定的模型。我认为具体原因是因为，当训练数据不够多时，测试集中如果出现训练数据中没有的 feature，算法会忽略 (此时未加入平滑处理) 当前 feature，从而减少了所有对于最终结果有贡献的 feature 的数量，这就造成了整体表现的不稳定。

如无特殊说明，以下所有结果均使用了 100% 的训练集。

3.4 Issue3-Feature 提取

由于我在 baseline 版本中只使用了单个汉字作为 feature，所以接下来通过不断加入新的 feature 对分类器进行优化。

3.4.1 字词作为特征

首先我使用实验数据中已经有的 $data_cut$ 数据进行训练，略微修改正则表达式，使其匹配中文字词而不是单个汉字，将整个词作为 feature 进行频数统计。期望上通过词进行分类可以更加接近真实的语义，从而实现更好的分类。

得到的结果如下表

Accuracy	0.7836
Precision	0.8541
Recall	0.8124
F1-Measure	0.8327

表 8: 使用词作为特征

与使用单字的分类器相比，性能反而下降了，这与预期不符。我认为原因在于，作为单个汉字的组合，词的种类比字要更多，也就更容易出现测试集中某个词从未出现过，从而无法给出有效的分类信息。为了验证这一点，接下来我实现了 Laplace 平滑并做了验证。

3.5 Issue2-Laplace 平滑

Laplace 平滑，或者别的平滑处理，是为了解决出现零概率的情况。出现零概率的原因是由于训练数据有限，导致某个 feature 只在一个或几个 label 中出现，甚至根本没有出现过，于是导致在讨论某些 label 时，该 feature 的概率为 0。若不进行平滑处理，一个零概率 feature 会使贝叶斯公式计算出的概率直接为 0，使得该 label 直接被放弃，虽然可以通过忽略零概率 feature 来避免这一点，但这是以放弃训练信息为代价的，效果仍然不好。

Laplace 平滑的处理方式为给所有零概率 feature 加一个固定的 lambda 偏移，同时也在分母加上 label 的数量乘以 lambda，保证所有概率加起来为 1。lambda 的大小也会影响 Laplace 平滑的效果，为了找到 lambda 对平滑效果的影响，我测试了从 10^{-30} 到 10^{30} 的 lambda，以自然对数为底，给出了性能曲线。

3.5.1 实验效果

Accuracy	0.9921
Precision	0.9944
Recall	0.9938
F1-Measure	0.9941

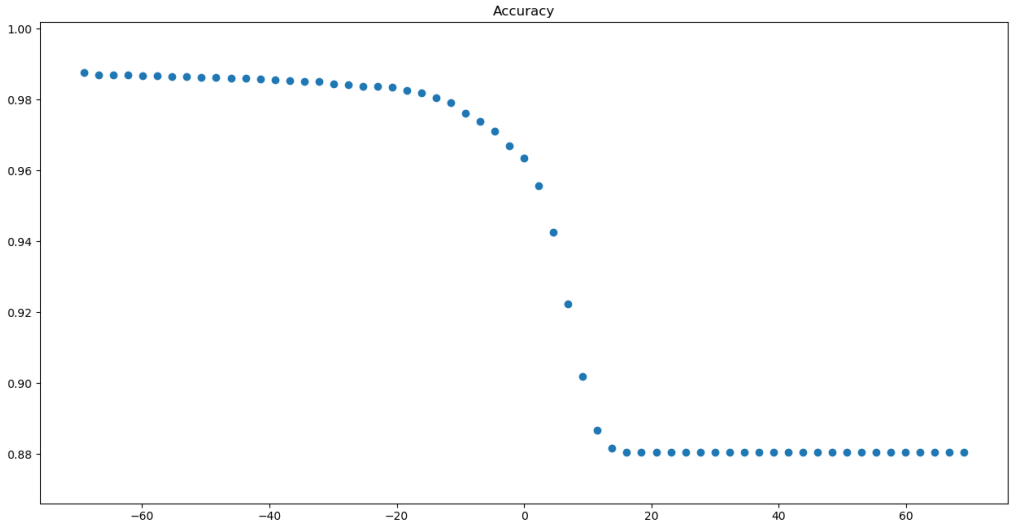

图 5: Laplace 平滑的 Accuracy

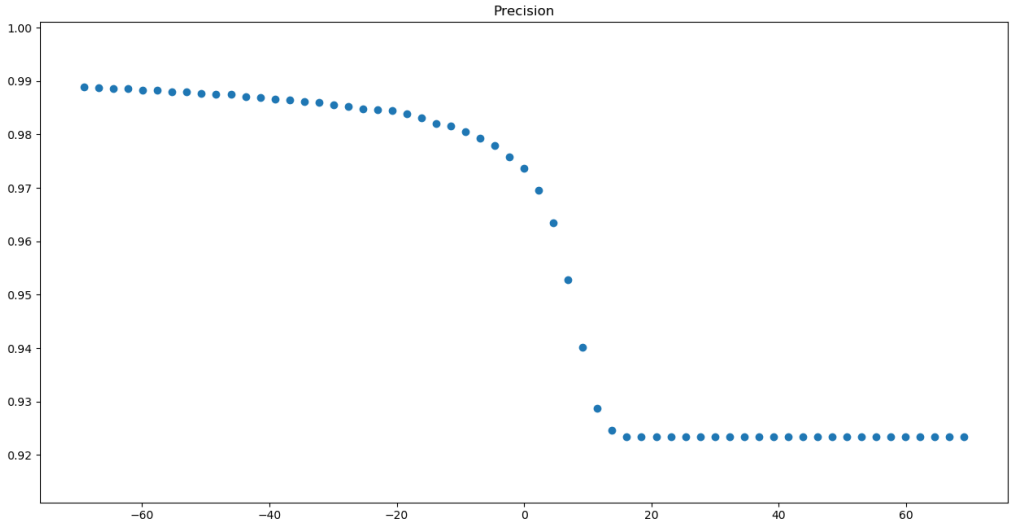

图 6: Laplace 平滑的 Precision

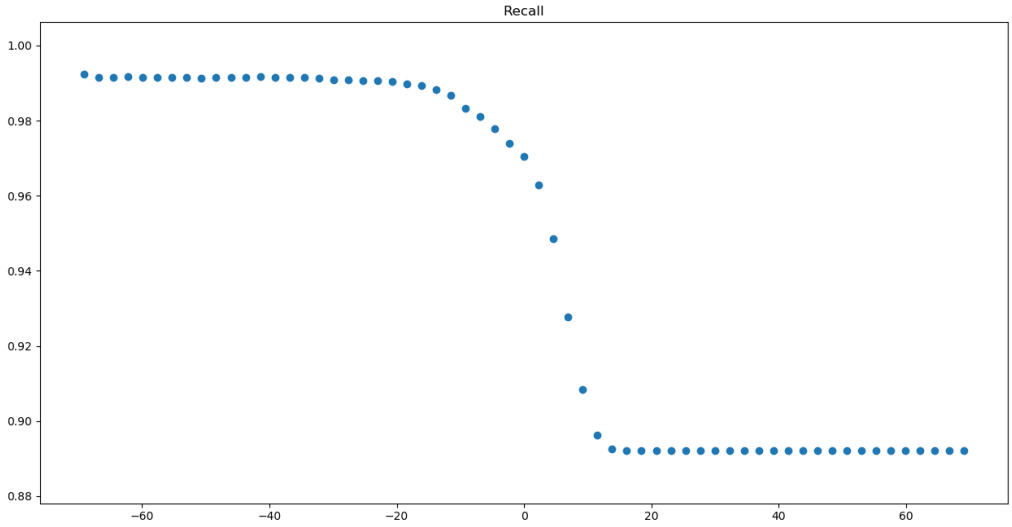

图 7: Laplace 平滑的 Recall

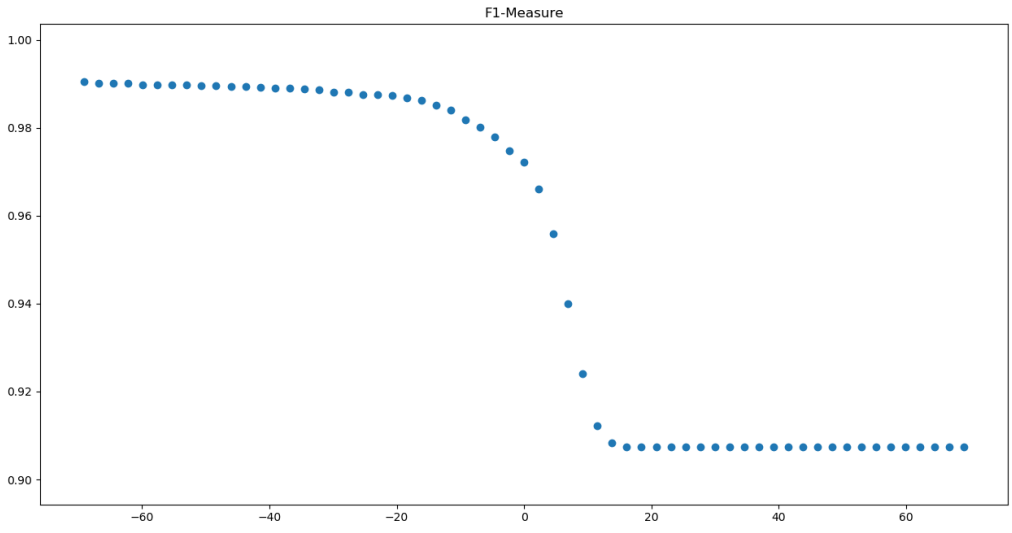

图 8: Laplace 平滑的 F1-Measure

3.5.2 性能分析和讨论

很明显，lambda 取较小的值时 (10^{-30})，分类器的性能较好，且随着 lambda 的增大，性能不断下降。我认为原因是，当 lambda 取较大值时，使得所有 feature 的概率值十分接近，实际上削弱了非零概率的 feature 的信息，因此整体性能变差。

加入 Laplace 平滑后，分类器性能又有了大幅的提升，这验证了之前的猜想。

3.6 Issue3-Feature 提取-续

下面继续讨论更多 feature 的选取。我选取了发件邮箱 (From)，是否包含链接 (url) 和 X-Mailer 这三个特征。同时我也放弃了时间等 feature。我放弃时间的原因在于，我认为垃圾邮件和正常邮件的发送时间并不会有实质的区别，虽然正常邮件几乎不会在半夜发送，但是垃圾邮件也没有必须要在半夜发送的理由，甚至我认为垃圾邮件在半夜发送的概率并不会比正常邮件在半夜发送的概率要大。如果加入了发送时间作为特征，反而可能会屏蔽掉一些在非正常时间发送的及其重要的正常邮件，因此我放弃了时间这个特征。

3.6.1 From 的特征

由于垃圾邮件的发送者大多使用免费注册的邮箱，与正常邮件的多种邮箱域名有所区别，因此我认为可以将邮箱域名作为一个 feature 加入到分类器中。

实现的分类器的性能如下。

Accuracy	0.9949
Precision	0.9969
Recall	0.9954
F1-Measure	0.9962

表 9: 加入发件邮箱特征

准确率提升了 0.28%，说明发件邮箱确实为一个区分垃圾邮件的有效 feature。

3.6.2 url 的特征

由于垃圾邮件往往包括一些广告或推销链接，因此是否包含 url 也是一个垃圾邮件的特征，所以将 url 加入到 feature 中。

结果如下。

Accuracy	0.9955
Precision	0.9969
Recall	0.9963
F1-Measure	0.9966

表 10: 加入 url 特征

准确率再次提高了 0.1% 左右，说明这的确是一个有效的较为好用的 feature。此外，url 还提升了 Recall 指标，这表明 url 在正分类中是一个十分鲜明的特征。

3.6.3 X-Mailer 的特征

X-Mailer 是邮件发送平台的特征，正常邮件的 X-Mailer 大多为 Microsoft, FoxMail 等平台，而垃圾邮件来自类似 VolleyMail 等平台，因此这也可以作为一个区分垃圾邮件的特征。

结果如下。

Accuracy	0.9969
Precision	0.9974
Recall	0.9979
F1-Measure	0.9976

表 11: 加入 X-Mailer 特征

准确率再次提升，达到了 99.69%。

3.6.4 特征权重

上述三个额外加入的特征，他们和邮件中出现的词显然提供的信息重要性是不等同的，因此要对其进行加权。但若权重过大，显然也会影响分类器性能，因此我在 1-20 的范围内的权重进行测是，找出最佳权重。

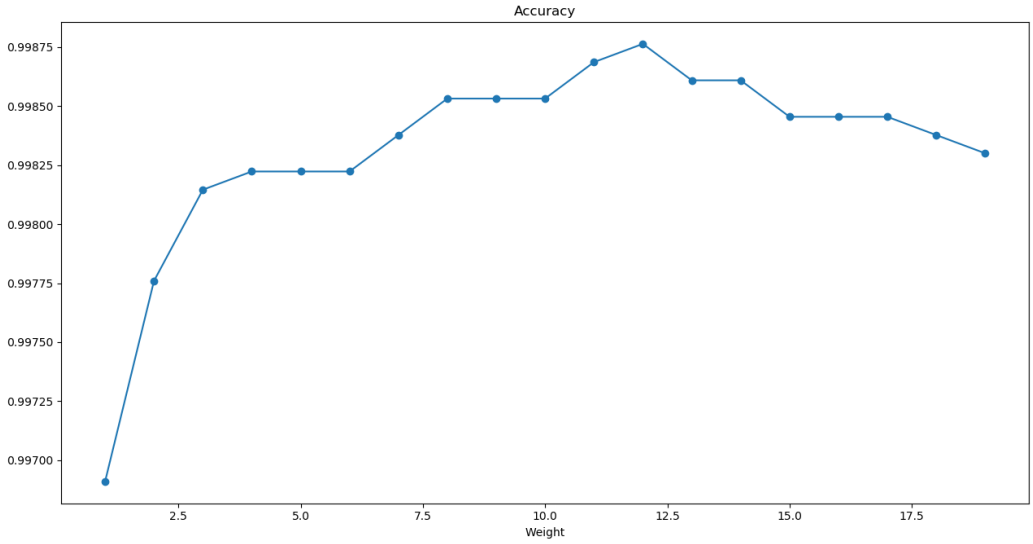

图 9: 不同 Weight 的 Accurancy

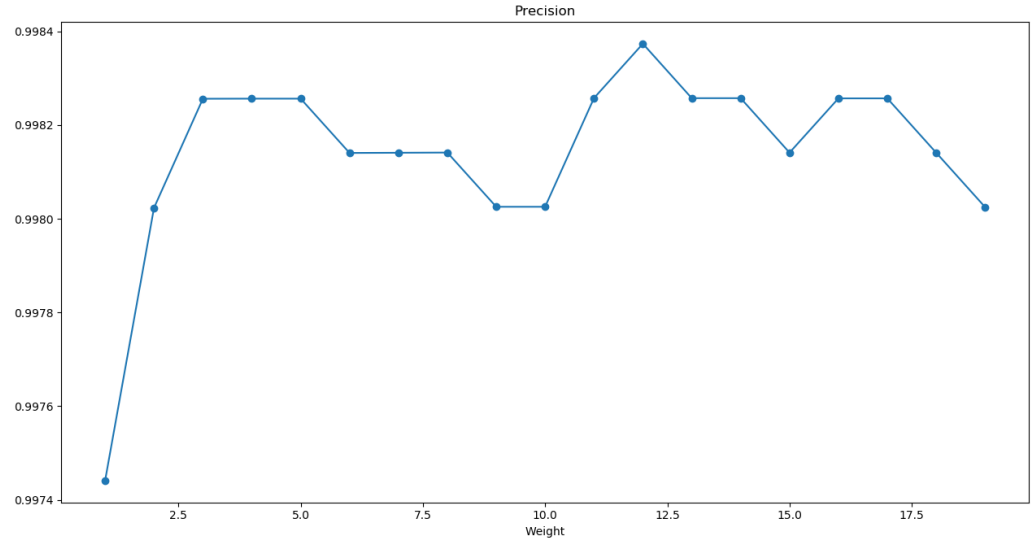

图 10: 不同 Weight 的 Precision

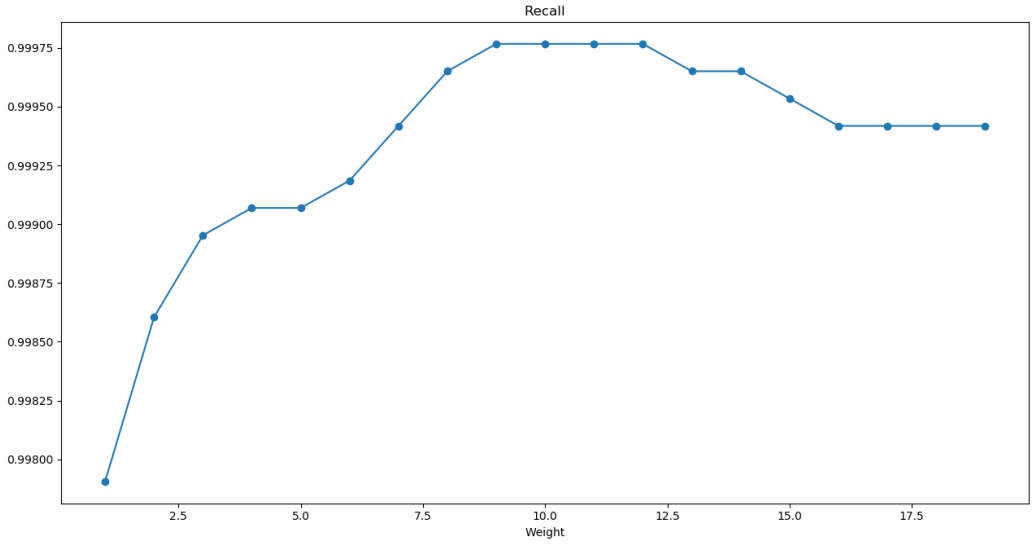

图 11: 不同 Weight 的 Recall

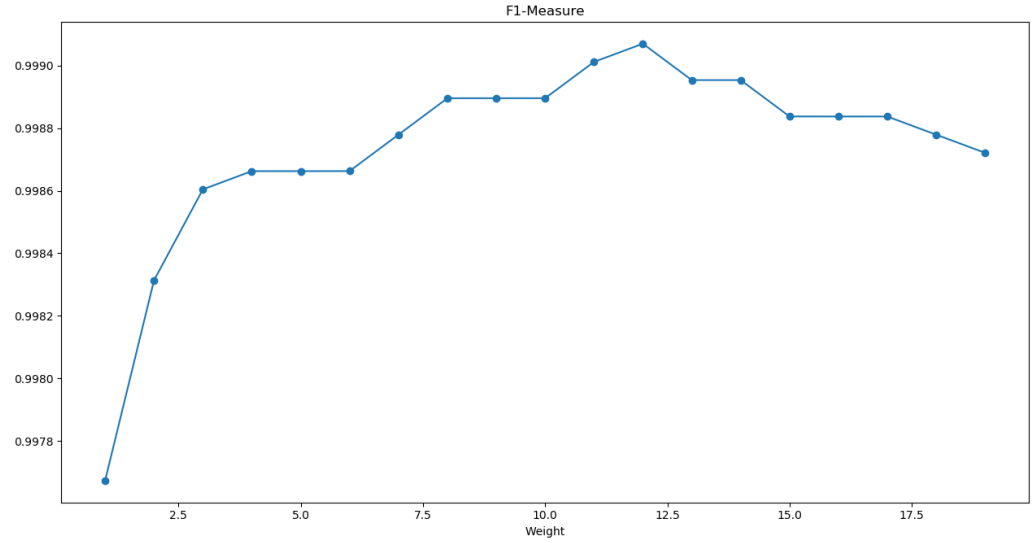

图 12: 不同 Weight 的 F1-Measure

可以发现，当 weight 取 12 时，分类器的性能最佳。
在最佳权重下的性能如下。

Accuracy	0.9987
Precision	0.9983
Recall	0.9997
F1-Measure	0.9990

表 12: 加入 X-Mailer 特征

3.7 最终性能比较

下面的图展示了我实现的朴素贝叶斯分类器的不同版本的性能。

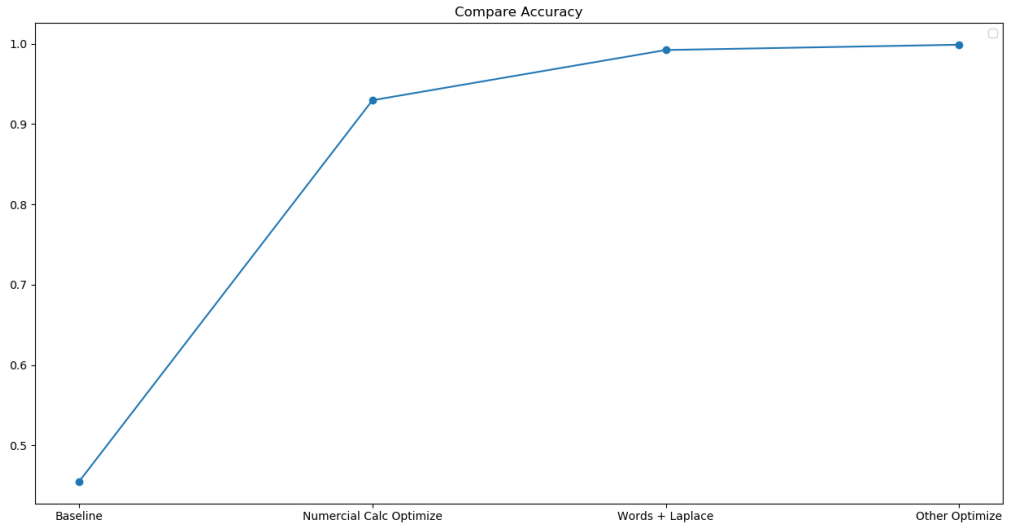

图 13: 不同版本的 Accurancy

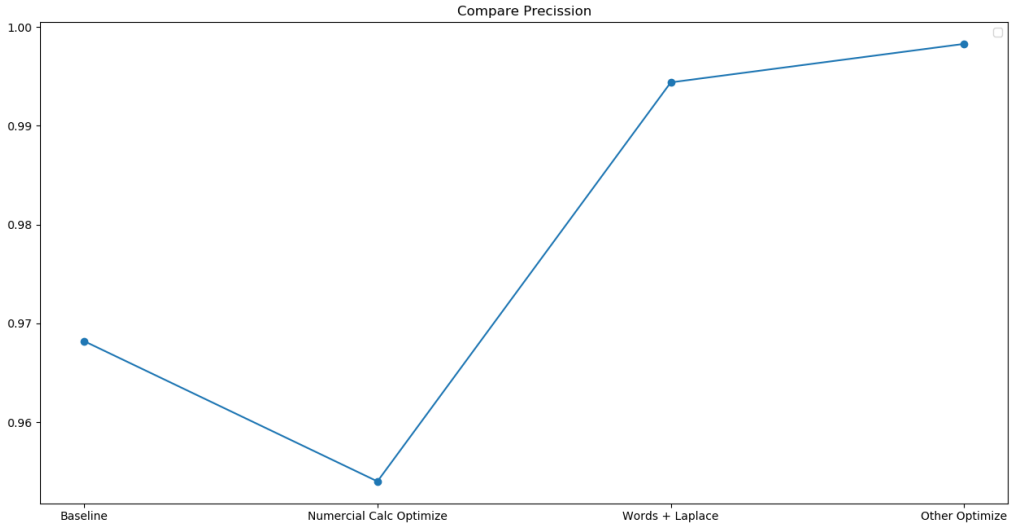

图 14: 不同版本的 Precision

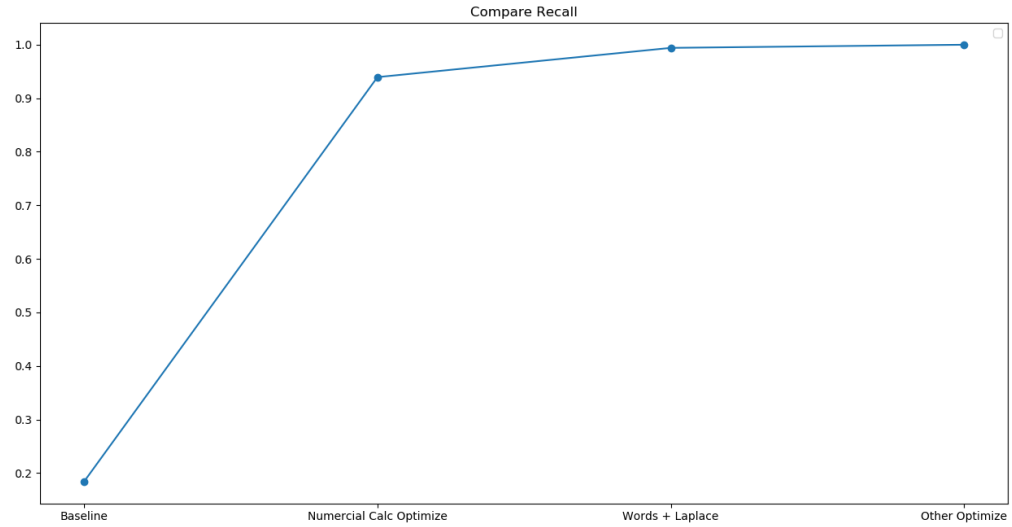

图 15: 不同版本的 Recall

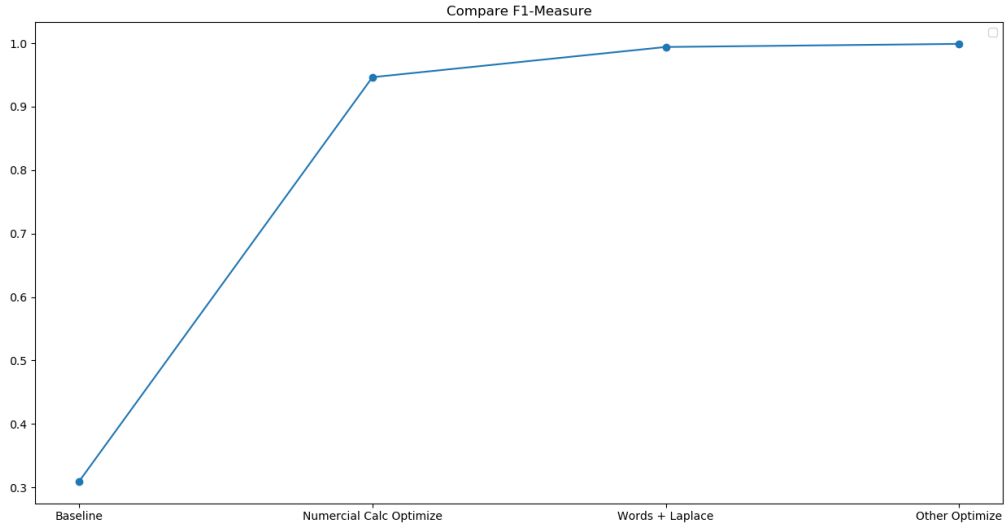

图 16: 不同版本的 F1-Measure

4 实验总结

我最终完成的朴素贝叶斯分类器对于垃圾邮件的分类准确率达到了 99.87%，并且在编写完善代码的过程中尝试了很多新东西，并且去了解和分析结果和现象，从而进一步了解了算法的理论基础和实际应用中的一些问题和隐患。我觉得最后的准确率可能并不是最重要的，重要的是不断分析优化的过程，不断去分析现象思考原因的过程，这样一个过程才是学习知识最重要的部分。