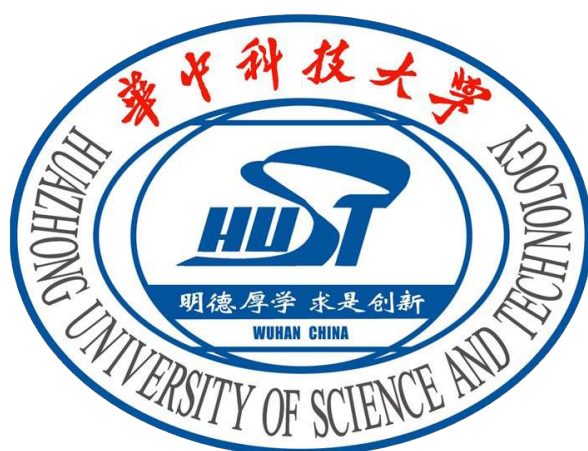


华中科技大学计算机科学与技术学院

《机器学习》 结课报告



专	业	<u>计算机科学与技术</u>
班	级	<u> </u>
学	号	<u> </u>
姓	名	<u> </u>
成	绩	<u> </u>
指导教师		<u>何琨</u>
时	间	<u>2024 年 5 月 17 日</u>

目录

1 实验要求	1
2 算法设计与实现	2
2.1 数据导入与预处理	2
2.2 逻辑回归模型训练	3
2.3 模型参数评估与可视化	4
2.4 填写预测提交文件	6
2.5 KNN 模型下的训练与比较	6
3 实验环境与平台	7
4 结果与分析	8
4.1 模型评价参数	8
4.2 学习曲线（损失函数下降）	8
4.3 预测概率分布	9
4.4 逻辑回归与 KNN 模型的优劣比较	9
4.5 ROC 曲线与 AUC 的值	10
4.6 Sample Submission Score	10
4.7 结果总结	10
5 个人体会	11
参考文献	12

1 实验要求

总体要求：

1. 控制报告页数，不要大段大段贴代码
2. 表格和插图请编号并进行交叉引用，表格使用三线表

任务要求：

1. 使用银行流失数据集进行二元分类，对于本系列的这一集，你的任务是预测客户是继续使用他们的帐户还是关闭它（例如，流失）。

2 算法设计与实现

2.1 数据导入与预处理

使用 pandas 的 `read_csv` 来读入数据，将无关的属性'id', 'CustomerId', 'Surname' 丢弃，并利用 `sklearn.model_selection` 中的 `train_test_split` 划分出占比 0.8 的训练集 `train` 和占比 0.2 的测试集 `test0`，划分好后，`train` 的数据结构如图 2.1 所示。

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
90879	822	Spain	Male	37.0	8	0.00	2	1.0	1.0	169038.02	0
24401	535	France	Male	37.0	5	0.00	2	1.0	1.0	140883.91	0
151368	817	France	Male	25.0	2	89994.71	1	1.0	0.0	159654.00	0
49205	577	Spain	Male	43.0	8	0.00	2	1.0	0.0	171060.01	0
117254	598	France	Female	35.0	3	0.00	2	1.0	1.0	59297.34	0
...
73349	626	France	Female	37.0	4	0.00	1	1.0	0.0	176712.59	0
109259	711	Germany	Female	28.0	7	150329.15	2	0.0	0.0	141533.19	0
50057	681	Spain	Male	48.0	7	142905.51	1	1.0	1.0	163581.67	0
5192	444	France	Female	36.0	7	0.00	1	0.0	1.0	138743.86	1
128037	802	France	Female	35.0	10	0.00	2	0.0	1.0	113597.64	0

132027 rows x 11 columns

图 2.1: Dataframe of Train

将数据特征划分为分类特征和数值特征，其中分类特征为'Geography', 'Gender'，其余为数值特征。

对于数值特征，首先使用 `SimpleImputer` 填充缺失值，将其填充为数据的中位数，使用中位数填补缺失值可以减少由于缺失值带来的数据偏差，也使其免受极端值的影响。接下来使用 `StandardScaler` 将数据的均值调整为 0，标准差调整为 1，使得数据具有相同的尺度，其中 `StandardScaler` 按照如下公式对每个特征进行标准化：

$$\text{scaled value} = \frac{\text{value} - \mu}{\sigma} \quad (2.1)$$

其中， μ 是特征的均值， σ 是特征的标准差。

对于分类特征，首先使用 `SimpleImputer` 填充缺失值，将缺失值替换为字符串'missing'，随后使用 `OneHotEncoder` 独热编码，将分类特征转换为一组二进制（0 或 1）特征，使其可以应用于机器学习模型。

将其转化为 dataframe 形式，处理完的训练集 `train` 的数据结构由图 2.2 所示。

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geography_France	Geography_Germany	Geography_Spain	Gender_Female	Gender_Male	Exited
0	2.067045	-0.128462	1.062065	-0.883564	0.812788	0.570811	1.004365	1.123215	0.0	0.0	1.0	0.0	1.0	0
1	-1.513526	-0.128462	-0.006138	-0.883564	0.812788	0.570811	1.004365	0.564206	1.0	0.0	0.0	0.0	1.0	0
2	2.004665	-1.478287	-1.074341	0.548947	-1.013609	0.570811	-0.995654	0.936892	1.0	0.0	0.0	0.0	1.0	0
3	-0.989540	0.546451	1.062065	-0.883564	0.812788	0.570811	-0.995654	1.163362	0.0	0.0	1.0	0.0	1.0	0
4	-0.727547	-0.353433	-0.718273	-0.883564	0.812788	0.570811	1.004365	-1.055723	1.0	0.0	0.0	1.0	0.0	0
...
132022	-0.378223	-0.128462	-0.362206	-0.883564	-1.013609	0.570811	-0.995654	1.275596	1.0	0.0	0.0	1.0	0.0	0
132023	0.682225	-1.140831	0.705997	1.509334	0.812788	-1.751893	-0.995654	0.577097	0.0	1.0	0.0	1.0	0.0	0
132024	0.307949	1.108878	0.705997	1.391166	-1.013609	0.570811	1.004365	1.014877	0.0	0.0	1.0	0.0	1.0	0
132025	-2.648829	-0.240947	0.705997	-0.883564	-1.013609	-1.751893	1.004365	0.521714	1.0	0.0	0.0	1.0	0.0	1
132026	1.817527	-0.353433	1.774200	-0.883564	0.812788	-1.751893	1.004365	0.022428	1.0	0.0	0.0	1.0	0.0	0

132027 rows x 14 columns

图 2.2: Dataframe of Preprocessed Train

2.2 逻辑回归模型训练

给定数据集 $D = \{(x_1, y_1), (x_2, y_2) \dots (x_m, y_m)\}$, 其中 (x_i, y_i) 表示第 i 个样本, 每个数据有 n 个特征, 即 $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$, 类别 $y = \{0, 1\}$, 训练数据形成分类器, 将数据分为 0 或 1。

使 x_i 的 n 个特征为线性特征, 同时为了表示简洁, 将常数项 b 也整合进特征中, 将 $b = \theta_0, x_0 = 1$, 最终可以表示为如下形式:

$$z = \theta x + b = \sum_{i=0}^n [x_i \theta_i] \quad (2.2)$$

定义 Sigmoid 函数, Sigmoid 函数在逻辑回归中用于将线性组合的结果转换为概率值, 其公式如下:

$$y(z) = \frac{1}{1 + e^{-z}} \quad (2.3)$$

定义损失函数, 在逻辑回归中, 损失函数使用对数损失函数, 其公式如下:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))] \quad (2.4)$$

其中 m 是样本数量, $y^{(i)}$ 是第 i 个样本的真实标签, $h_{\theta}(x^{(i)})$ 是第 i 个样本的预测概率。

使用梯度下降法, 其是一种用于最小化损失函数的优化算法。在每次迭代中, 计算损失函数相对于参数的梯度, 并沿着梯度的反方向更新参数。其公式如下:

梯度计算公式:

$$\nabla J(\theta) = \frac{1}{m} X^T (h_{\theta}(X) - y) \quad (2.5)$$

参数更新公式:

$$\theta = \theta - \alpha \nabla J(\theta) \quad (2.6)$$

其中, α 是学习率, X 是特征矩阵, y 是真实标签向量, $h_{\theta}(X)$ 是预测概率向量。

模型训练过程中，首先需要初始化，随后设置学习率为 0.1 和迭代次数为 3000 次作为逻辑回归的参数，使用梯度下降进行迭代后得到一组训练出的参数向量 w ，其中梯度下降部分的代码如下：

```

1 def gradient_descent(X, y, w, learning_rate, num_iterations):
2     m = len(y)
3     cost_history = []
4     for i in range(num_iterations):
5         h = sigmoid(X.dot(w))
6         gradient = X.T.dot(h - y) / m
7         w -= learning_rate * gradient
8
9         if i % 50 == 0:
10             cost = compute_cost(X, y, w)
11             cost_history.append(cost)
12             #print(f'Cost after iteration {i}: {cost}')
13     return w, cost_history

```

2.3 模型参数评估与可视化

模型训练完成后，分别计算模型的准确率 (Accuracy)，精确率 (Precision)、召回率 (Recall)、F1 等，首先计算混淆矩阵，然后再根据混淆矩阵的结果，计算评估参数，其参数分为四部分：

TP：模型预测为 1，真实值符合模型预测，也为 1

TN：模型预测为 0，真实值符合模型预测，也为 0

FP：模型预测为 1，真实值不符合模型预测，为 0

FN：模型预测为 0，真实值不符合模型预测，为 1

则个评估参数可表示为如下公式：

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (2.7)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2.8)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2.9)$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.10)$$

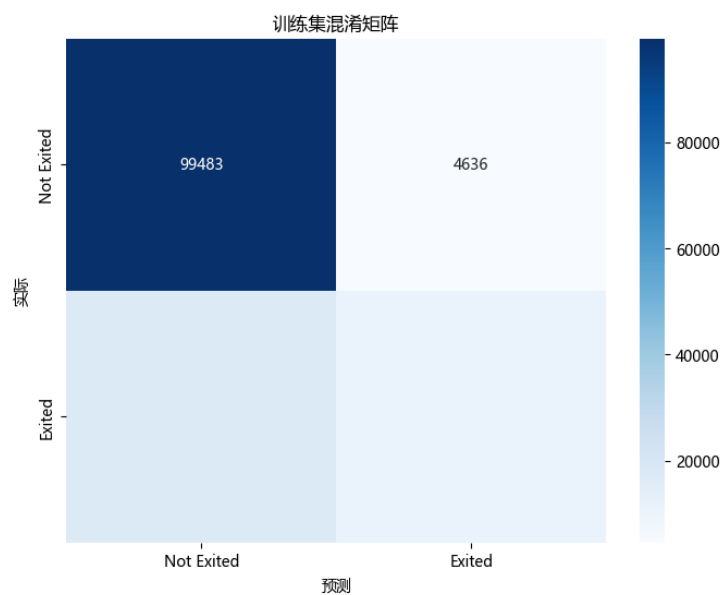


图 2.3: 混淆矩阵

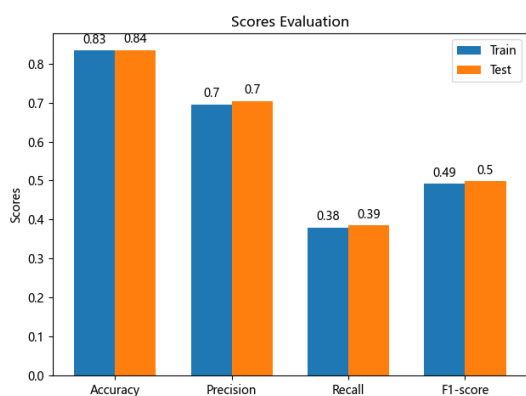


图 2.4: 模型评价参数

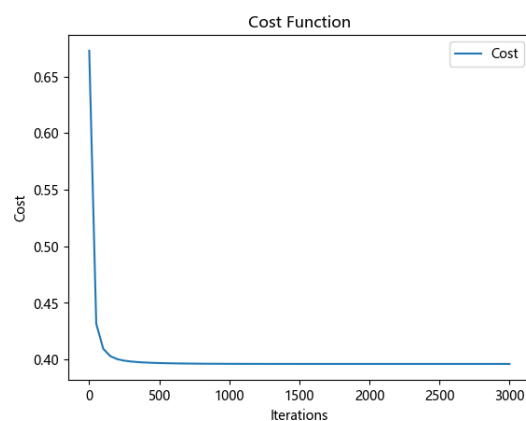


图 2.5: 损失函数变化曲线

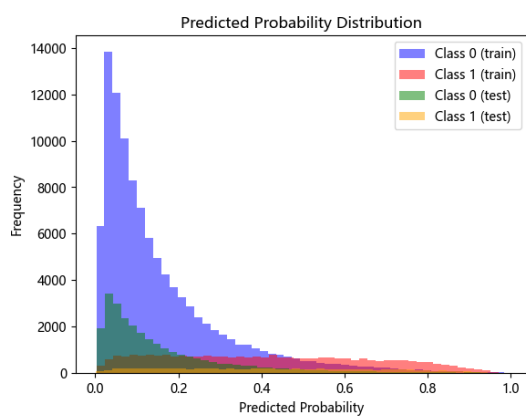


图 2.6: 预测概率分布

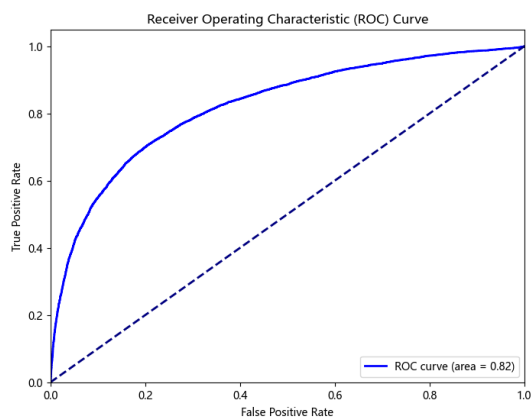


图 2.7: ROC 曲线和 auc

由如下公式计算真阳性率 TPR 和假阳性率 FPR，并以此为纵轴和横轴绘制 ROC

曲线。

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2.11)$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (2.12)$$

AUC 的值通过计算 ROC 曲线下的面积来获得，其公式为：

$$\text{AUC} = \int \text{TPR} d(\text{FPR}) \quad (2.13)$$

2.4 填写预测提交文件

将 test 数据集导入训练好的模型中，并将得到的结果写入 sample_submission.csv 文件中。

2.5 KNN 模型下的训练与比较

训练 KNN 模型，将其参数设置为 $n_neighbors = 5$ ，表示对于每个测试样本，选择最近的 5 个训练样本，投票决定测试样本的类别。其中 KNN 类设置的关键部分代码如下：

```
1 def __predict(self, x):
2     distances = [np.linalg.norm(x - x_train) for
3                  x_train in self.X_train]
4     k_indices = np.argsort(distances)[:self.n_neighbors]
5     k_nearest_labels = [self.y_train[i] for i in k_indices]
6     most_common = np.bincount(k_nearest_labels).argmax()
7     return most_common
```

由于 KNN 算法的运行时间过长，其运行时间通常为逻辑回归算法的数百倍，内存需求和运算复杂度已经成为模型训练的瓶颈，但是在选取部分数据进行训练和测试时，训练准确率到达了 0.877，其相比于逻辑回归来说得到了很大的提升。

3 实验环境与平台

本次 bank churn 实验的实验环境与平台配置如表 3.1 所示。

表 3.1: 实验环境与平台配置

名称	配置信息
操作系统	Windows 11, version 23H2
开发语言	Python 3.11.7 packaged by Anaconda, Inc.
开发平台	Jupyter Notebook 7.0.8
CPU	AMD Ryzen 7 5800H with Radeon Graphics
GPU0	NVIDIA GeForce RTX 3050 Laptop GPU
GPU1	AMD Radeon(TM) Graphics
内存	16G

4 结果与分析

本次实验通过逻辑回归和 KNN 两种机器学习模型实现了银行客户流失的预测，最后模型的结果由图 2.4 至图 2.7 可视化地给出，根据图像内容，进行结果分析。

4.1 模型评价参数

准确率 (Accuracy): 训练集: 0.83 测试集: 0.84, 发现准确率在训练集和测试集上非常接近, 这表明模型在这两组数据上的表现一致, 同时得分较高, 表明没有明显的过拟合或欠拟合现象。

精确率 (Precision): 训练集: 0.7 测试集: 0.7, 精确率在训练集和测试集上也是一致的, 并且得分较高, 表明模型能够准确预测正类样本。

召回率 (Recall): 训练集: 0.38 测试集: 0.39, 召回率较低, 表明模型在检测正类样本时存在较高的漏检率。

F1-score: 训练集: 0.49 测试集: 0.5, F1 得分是精确率和召回率的调和平均数, 综合了模型的精确率和召回率。这个得分处于中游水平, 表明模型在整体性能上还有提升空间。

4.2 学习曲线 (损失函数下降)

学习曲线显示损失函数值随迭代次数的变化。在迭代初期, 损失函数值快速下降, 表明模型在快速学习, 在大约 1000 次迭代后, 损失函数值趋于平稳, 表明模型已经收敛。

该曲线与学习率也存在关联, 该图是在学习率为 0.1 时绘制出的, 学习率过高或过低时都会对训练产生负面影响。

学习率过低时, 每次参数更新的步长非常小, 训练过程变得非常缓慢, 需要更多的迭代次数才能达到收敛。同时其由于更新步长过小, 模型可能会过早陷入局部最优点, 而无法跳出局部最优去探索全局最优解。而且由于需要更多的迭代次数才能收敛, 学习率过低会浪费大量的计算资源和时间。

学习率过高时, 可能会导致模型参数在更新时跳跃过大, 无法在损失函数的方向上稳定下降, 导致训练过程中的损失值震荡甚至增加, 更极端的情况下参数更新可能会跳过最优点, 使得模型无法收敛到全局最优或局部最优, 使得损失函数的值越来越大, 参数更新发散, 无法找到有效的模型参数。

通常来讲, 在适宜的学习率范围下, 学习率越高, 学习曲线收敛地越快。

4.3 预测概率分布

图 2.6 显示了预测概率的分布情况。可以看到训练集的正类和负类样本的预测概率分布有明显的区分，这表明模型在训练集上的表现较好，测试集的正类和负类样本的预测概率分布也有较好的区分，但略有重叠，表明模型在测试集上的预测能力与训练集相似。

4.4 逻辑回归与 KNN 模型的优劣比较

逻辑回归的准确率为 0.835，而 KNN 算法的准确率为 0.877，但是 KNN 算法的运行时间时逻辑回归算法的数百倍，应当综合实际情况，在处理不同问题上对两种算法合理选择。

逻辑回归的优点：

1. 简单易解释，是线性模型，每个特征的回归系数可以直观地解释为对客户流失概率的影响。
2. 训练速度快，适用于大规模数据集。

缺点：

1. 逻辑回归假设特征和输出之间存在线性关系，因此对于非线性数据表现较差。
2. 逻辑回归对异常值较为敏感，其可能对模型产生较大影响。

KNN 的优点：

1. 是非参数模型，不做任何关于数据分布的假设，适用于非线性数据。
2. 可以处理任意形状的决策边界，在处理复杂分类问题时有较好的表现。

缺点：

1. KNN 在预测阶段需要计算测试样本与所有训练样本的距离，计算复杂度高，同时需要存储所有训练数据，内存占用大。
2. 选择合适的 `k_neighbors` 值对模型性能有很大影响，需要通过交叉验证等方法进行调优。

从上面的分析可以看出，逻辑回归和 KNN 在银行客户流失问题上的优劣取决于数据的特征和具体应用场景。如果数据量大，特征之间关系较为线性，且需要对模型结果有清晰解释，那么逻辑回归表现更优异；如果数据量较小，特征之间关系复杂且非线性，模型需要捕捉这种复杂关系，KNN 表现将会更好。

4.5 ROC 曲线与 AUC 的值

ROC 曲线 (Receiver Operating Characteristic curve) 是用来评估二分类模型性能的重要工具, 它展示了分类模型在不同阈值下的性能表现。

其横轴是 FPR (假阳性率), 纵轴是 TPR (真阳性率), 曲线从 (0, 0) 开始, 到 (1, 1) 结束, 曲线越接近左上角 (FPR 低, TPR 高), 模型的性能越好。

AUC 值表示 ROC 曲线下的面积, 可以用来量化表示模型的性能, AUC 值越接近 1, 表示模型的性能越好。

该模型的 $AUC = 0.82$, 表示模型在区分正类和负类样本时都有较高的能力, 模型能有效地区分不同类别的样本, 并且在实际应用中也具有较好的效果。

4.6 Sample Submission Score




Submission and Description		Private Score 	Public Score 	Selected
 sample_submission.csv	Pending (after deadline) · now · bank churn LR	0.81922	0.81442	<input type="checkbox"/>

图 4.1: 提交得分

4.7 结果总结

经过多次实验尝试, 修改学习率和迭代次数已经无法有效地提升分类准确率, 合理推断出数据之间应该有更复杂的非线性关系, 从 Kaggle 上其他人提交的代码中了解到, 尝试更复杂的模型 (如随机森林、梯度提升树、XGBoost、LightGBM 等) 应当还能做到评估参数的进一步提高。

5 个人体会

在 bank churn 的实验任务过程中，我手动实现了逻辑回归模型的数据预处理、训练、预测以及模型评估的一系列步骤，手动实现的方法帮助我深入理解了机器学习模型的内部工作机制，也让我对于逻辑回归算法的理解更加深刻。

手动实现逻辑回归算法，包括进行参数初始化、梯度下降优化、sigmoid 以及损失函数的计算，比直接使用库函数更有助于掌握算法的细节和数学原理。

最后通过诸如 matplotlib 和 seaborn 可视化工具，我直观地观察和理解了模型的行为和性能，其不仅有助于结果的展示，帮助我发现潜在的问题和改进方向，也使我真切的看到了自己训练出的模型所得到的结果，我也因此内心充盈了成就感。

当然，在实验过程中我仍存在许多困惑和不足，在未来的学习和改进的方向方面，我想尝试在更复杂的数据集和模型上应用这些实现方法，并探讨如何优化算法的性能。同时也深入探索更多的模型评估指标，如 PR 曲线、F-beta 得分等，以全面评估模型的性能。关于程序的运行时间方面，我也想尝试进行算法优化，提升计算效率和模型训练速度。

总而言之，在手动实现逻辑回归模型进行银行客户是否流失的二元分类的整个过程中，从数据预处理、模型训练最后到评估指标的计算和可视化，我都对机器学习的基本原理和实际应用有了更深入的理解，也产生了更加浓厚的兴趣，我相信这些积累都将为我以后得学习和研究打下坚实的基础。

参考文献

- [1] 周志华. 机器学习: 第 3 章. 清华大学出版社, 2016.
- [2] Walter Reade, Ashley Chow. (2024). Binary Classification with a Bank Churn Dataset . Kaggle. <https://kaggle.com/competitions/playground-series-s4e1>
- [3] 李航, 统计学习方法, 清华大学出版社, 2012
- [4] Scikit-learn, scikit-learn: machine learning in Python —scikit-learn 1.1.3 documentation