AI Project 2

张鸣昊
软件 83
2017011002

1. 介绍

本次实验主要分为两部分，第一部分用不同的分类算法完成了对给定数据集的学习，从而得到预测用户是否会订购产品；第一部分通过对蛙叫特征进行聚类分析，得到对青蛙科（Family）的划分。在两个任务中，算法都包括调用 sklearn 库的类和自己实现的类，通过从不同角度出发，充分对比了各类算法的优劣。同时对特征工程的提升、对超参数的对比选择、正负样本平衡等技巧，也使得结果不断优化。

2. 实验 1：银行精准销售解决方案

本任务目标是对样本进行二分类。所以我们应当采用有效的分类算法，在对给定样本的特征进行恰当处理之后，用最合适的参数来完成任务。

2.1 分类算法

实验中采用了以下几种 sklearn 中的算法：knn，linear regression，random forest，decision tree，gbdt（gradient boosting decision tree）以及 svm。这里既包含线性分类器（linear regression），也包含对于平面性质划分的分类器（svm），bagging 方法（random forest）和 boosting 方法（dt&gbdt）。在 classification.py 中，我实现了 ClassifyMethods 类，其中包括上面所提到的所有分类算法以及对他们统一进行评估的函数。此外，本人自己实现了 knn 算法，并针对本次实验给定数据集的特点进行了改进，实现了 weighted knn （WKNN 类），细节见下文。

2.2 自实现 KNN

本人实现了 KNN 类，并在内部计算中使用 numpy 的数据结构加速。具体来说，该类成员包括 neighbors 数量 k，类函数包括：neighbors()来获取所求点的邻居，_prediect()用来预测单个点的类别，predict()用来预测一个给定数据集的类别，除此之外还有一些用来展示的辅助函数。内部的度量包括曼哈顿距离 absolute_distance()和欧氏距离 euclidean_distance()。这些函数内部的运算全部由 numpy 数据结构和运算完成，充分利用 cpu 资源。具体见代码文件 KNN.py。

2.3 Weighted KNN

本次实验中给定数据集包括了 90%的负样本，正样本仅有 10%，而对于银行预测买家这件事儿来讲，我们显然更在意那些会买的人能否被我们准确拿捏，也就是说本次任务更重要的是**对正例的准确预测**。这对于普通的、无针对性改进的分类算法要求很高，单纯从算法和特征工程上来应对往往让人摸不到头脑。而做数据增广（data augmentation），欠抽样（under samplimg）以及过抽样（over sampling）等数据处理虽然无疑可以带来提升，但本人认为对本次大作业所期望我们学到的东西毫无帮助，故这里不考虑通过对数据进行改进的方法来提升。我们先来简单分析一下 k neighbours 类算法分类的原理。给定训练集的 train_x 和 train_y（数据和标签）后，当给定一个待预测样本 test_x 时，通过搜索最邻近的 k 个 train_x_i 后，以投票的方式完成对此样本类别的预测。但对于正负不平衡二分类问题来说，二维平面内负例特征占绝大部分，也就是说，我们输入一个待遇测的正例之后，周围搜索到的负例的期望仍然非常大。

如图，绿色点为正样本特征，但很多情况下周围掺杂橙色的负样本特征。

对于这种情况，假设类间距离更小的情况下，本人采取了根据距离加权投票：距离该待预测点越近，投票权重越大。具体的，我们用距离加一个小的正则项之后求倒数（加正则项是为了让距离很小的时候有意义，否则 weight 会变成无穷大）。这之后，我们还需要对其进行 normalization，让 weight 满足加权后求和期望不变：

$$\mathbb{E}\left[\sum_{i=1}^{k} w_i * \sigma_k\right] = (\sum_{i=1}^{k} w_i)\mathbb{E}\left[\sum_{i=1}^{k} \sigma_k\right] = \mathbb{E}\left[\sum_{i=1}^{k} \sigma_k\right] = 0.1k,\ \sigma_i \in \{0, 1\}$$

这里 sigma 代表 k 个近邻样本的类别投票，我们估计正样本约占 10%，所以直接用 0.1k 代表。期望可以拆开求和是因为对每个投票，概率分布是相同的。详细做法请看代码。做了正则化之后，我们便可以用 WKNN 进行更合理的分类预测。

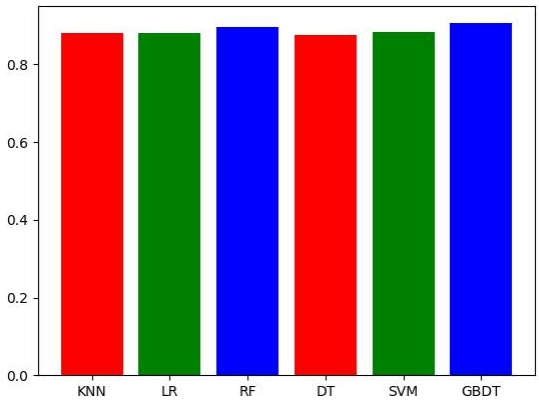
2.4 算法对比

我们首先对比 sklearn 中不同算法对该任务的预测能力。我们通过对正负样本的预测精度，召回率，F1 系数进行对比，充分考察各个算法的性能。实验中，我们对数据集以 4:1 的比例进行划分，对字符串数据进行映射为整数的处理，并做了归一化，这里我们先用除标签外所有数据进行尝试。以下数据反应了训练后的模型在测试集上的表现。数字为 sklearn 中 classification report 打印出来的 weighted avg 项得分：

Classifyer	Precision	Recall	F1 score
KNN	0.88	0.88	0.86
LR	0.78	0.88	0.83
Random Forest	0.88	0.90	0.88
Decision Tree	0.87	0.86	0.87
SVM	0.85	0.88	0.83
GBDT	0.89	0.90	0.89

除此之外，本人还对比了交叉验证的结果

KNN	LR	RF
0.88	0.88	0.90
DT	**SVM**	**GBDT**
0.87	0.88	0.91



可以大体看出，随机森林和 GBDT 这种 ensemble 展现出了较好的预测结果。但是，如果我们把全部样本当成训练集，并对全部样本进行预测，并只看对正例的绝对预测能力，我们就可以得到如下的结果：

Classifyer	Precision	Recall	F1 score
KNN	0.70	0.38	0.49
LR	0.00	0.00	0.00
Random Forest	0.99	0.92	0.96
Decision Tree	1.00	1.00	1.00
SVM	0.61	0.01	0.02
GBDT	0.92	0.92	0.92

可以看到，boosting 和 bagging 的拟合能力实际上是极强的，DT 方法居然能达到惊人的 1，而线性分类器对正样本完全没有拟合能力。这是因为线性模型用直线来拟合，必然会倾向与只在乎负样本，所以反倒是泛化能力要更好，因为至少随机要比完全过拟合到一种类别上令我们满意。

这里还要提一句 inference time。这些方法除了 SVM 之外，推理速度都很快，而 SVM 运行时间大概是其他方法的几十倍。不过从原理上我们不难理解，完全基于超平面划分的方法自然是要比例如概率模型方法要慢。

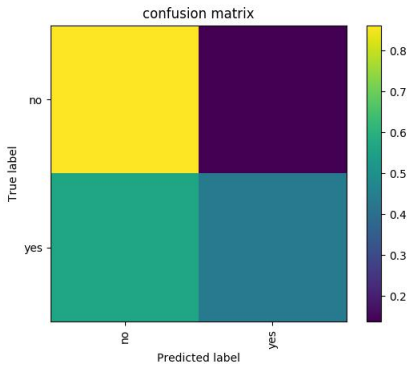
然后我们对比自实现的方法和 sklearn 内置 KNN 的效果：

Classifyer	Precision	Recall	F1 score
KNN(sklearn)	0.88	0.88	0.86
KNN(mine)	0.86	0.83	0.85
WKNN(mine)	0.87	0.80	0.83

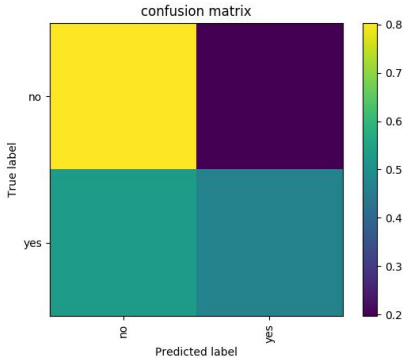
总体上，我们得分稍逊与 sklearn 方法，但差别不大，但对于我们更关心的正例预测，结果如下：

Classifyer	Precision	Recall	F1 score
KNN(sklearn)	0.46	0.24	0.31
KNN(mine)	0.43	0.42	0.34
WKNN(mine)	0.47	0.49	0.35

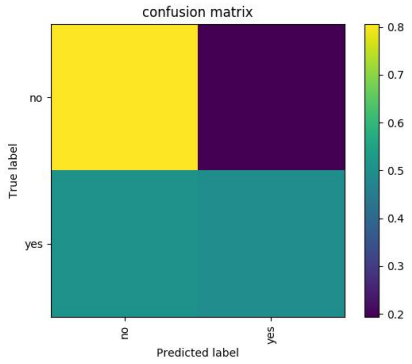
我们可以看到，召回率和 F1 得分这类真正能反应模型预测水平的数据上，自行实现的 KNN 更高，而 WKNN 也确实带来了提升。下面的混淆矩阵（confusion matrix）对这一特点展现的更具体：



KNN(sklearn)



KNN(mine)



WKNN(mine)

这里我们明显可以看出，对正例的预测能力有肉眼可见的提升。

最后是特征选择。除了将所有数据全用之外，我尝试理解数据，并分析得到：数据中除去标签的倒数五列数据，是客户与银行交流的相关数据，我认为这将会与是否购买产品具有最直接的相关性。于是，我使用这些数据进行实验，以下是**正样本分类**结果：

Classifyer	Precision	Recall	F1 score
KNN	0.53	0.32	0.40
LR	0.61	0.12	0.21
Random Forest	0.54	0.34	0.42
Decision Tree	0.49	0.34	0.41
SVM	0.60	0.17	0.26
GBDT	0.68	0.33	0.44

可以看到相对于之前的结果有了很明显的提高。此外，我尝试了一些单独列作为数据的实验，发现**效果并不好，本人猜测这是由于数据信息复杂性给出了一个分类性能的下界，在机器学习理论中也有相关的理论工作**，由于篇幅有限且不相关，在此不做全部展示。另外 L1 范数和 L2 范数对于结果影响不大，也不做详细讨论。代码中参数设置十分便捷，实验运行方便，助教可以自行尝试之。

3. 实验 2：青蛙叫声聚类分析

本任务是无监督学习中的聚类任务。这里由于所用特征已经做了归一化， 所以我们在初步的实验中不需要做过多的数据考虑，先以对比算法为主。

3.1 聚类算法

本次实验我采用了 sklearn 中的 kmeans，dbscan，optics 和 spectral clustering 四种方法，自行实现了 kmeans 算法。通过多种评估标准对比和分析，以及对结果进行可视化，充分展现了对各种聚类算法对这次实验的性能。

3.2 实现

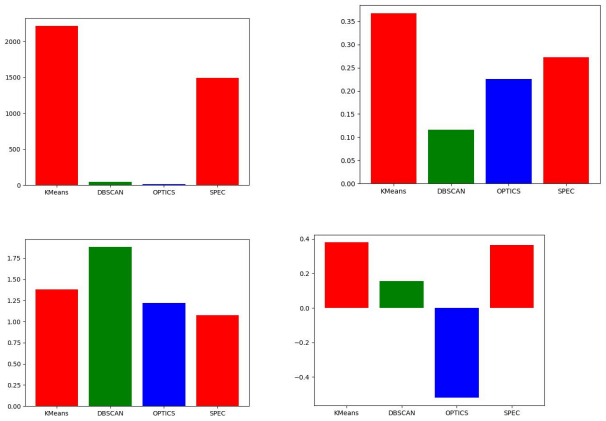
在 clustering.py 中，我实现了 ClusteringMethods 类，其中包括上面所提到的所有聚类算法以及对他们统一进行评估的函数。此外，在 KMeans.py 中，有我自己实现的 kmeans 算法，其中包括和 sklearn 接口一致的 fit()函数，用来对给定数据进行初步计算；以及 evaluation()函数，可以用多种指标对训练结果进行评估：davies bouldin score，calinski harabasz score，silhouette score，completeness score 以及 tsne 图。具体烦请助教查看代码。

3.3 实验

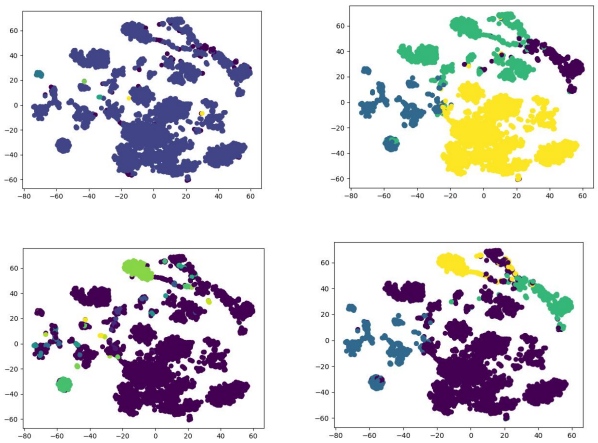
本次任务评估比分类任务简单，因为不存在考虑数据集的划分和验证等问题。我们首先用全部 MFCC 进行实验，由于数据已经做了归一化处理，我们直接将其输入使用即可。下表是相关结果：

Clustering	db	ch	sil	completeness
KMeans	1.39	2199.82	0.38	0.37
DBSCAN	1.61	49.25	0.12	0.10
OPTICS	1.15	16.23	-0.48	0.17
SPEC	1,08	1469.39	0.36	0.27

下面是 calinski harabasz score，davies bouldin score，completeness score，silhouette score 的柱形图：

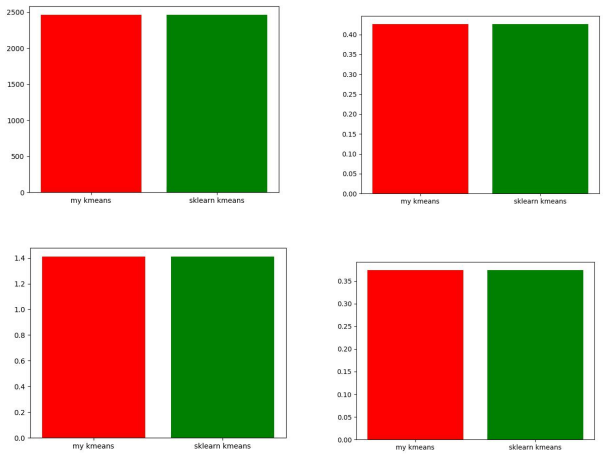


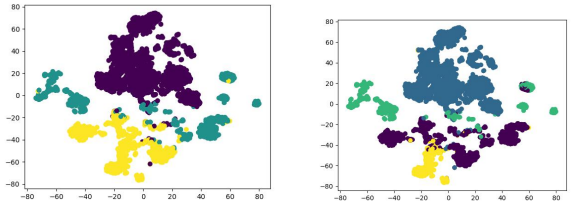
下面是 tsne 图（左到右、上到下依次为 DBSCAN，KMeans，OPTICS 和 SPEC）：



我们可以看到，各类指标排序续本一样，KMeans 在所有算法中效果最好，DBSCAN 和 DBSCAN 的改进 OPTICS 完全不奏效。这是因为它们是基于密度的聚类算法，而我们的特征分布算是比较均匀，所以还是 KMeans 算法更为合适。

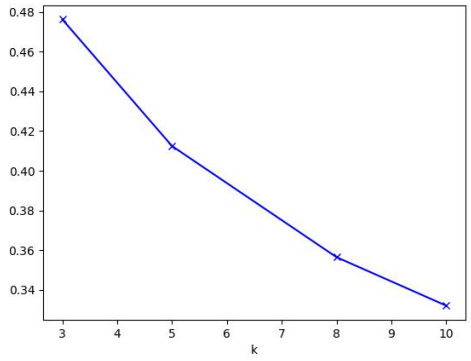
下面是自己实现的 KMeans 算法与 sklearn 中实现的对比:





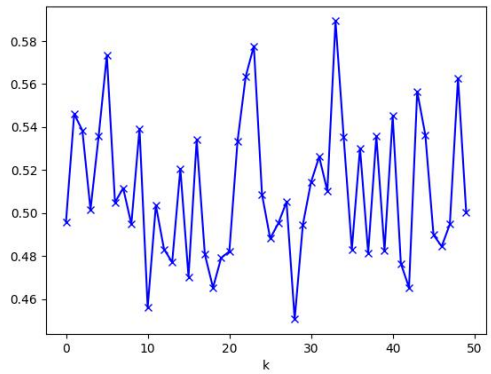
我们可以看到，各项指标基本一致，tsne 图除了颜色不同，分类基本一致。

下面我们对于超参数 k，来用平均畸变系数来评估：

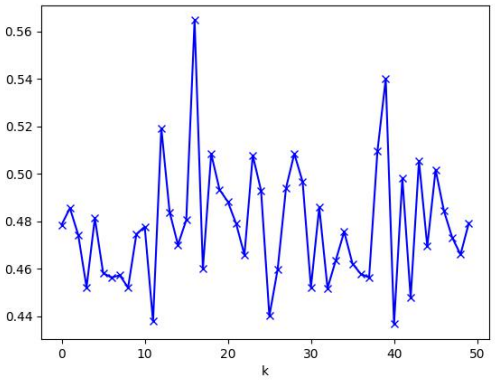


这里可以看到，当 k 变大时，畸变系数逐渐变小。故可以得出：对于聚类问题，k 越大，对整体的处理更加平滑顺畅。

关于特征选取，本人实验做法如下：由于 22 列数据不能直接看出有什么区别，故我选取了在 0-21 范围内长度为 10 的随机数作为下标，随机 抽取 10 列作为训练特征，共测试 50 组随机数据，畸变系数结果如下：



可以看到其实对于不同的特征，差别不是很大（大概 0.45-0.6）范围内；对于随机抽取 20 列的实验，结果如下：



实际上也没有看出很大的差别。对于 5,15 的实验，结果相仿。故本人得出的结论是，这个任务所提供的数据每个列差别不大，而对于特征数量差别也不大，而本身 MFCC 的特征就是经过了妥当处理的，所以特征工程/选择也是一件成本高回报低的事情。

而对于 MFCC 的特征适用的度量方式，本人在网上查到的资料大体说明欧氏距离就是最合适的方法，且在不用欧式距离的情况下，kmeans 的收敛性也不确定有理论保证，故实验中采用的度量均为 L2 距离。

4. 结论

本次实验两个任务代表了机器学习两大方法：有监督和无监督方法。对于有标数据的分类预测问题，即应该注重算法本身的设计，又需要结合具体的数据集来做合适的调整；对于无监督聚类任务，对特征的分析尤为重要。总体来说，机器学习任务实际上就是数据处理任务，我们不能只盯着各种算法，还要结合数据本身，才能很好地驾驭它们。同时，我们也应用各类评价指标来评估一批算法，这样才能多角度、全方位的认识他们。最后感谢老师助教提供的精心设计的作业，也感谢一些给予我建议的同学。