基于机器学习方法的健康活动状态检测

2016080106 社科 64 郑仁西

目录

1.研究背景与意义

1.1 研究背景

人类活动识别（Human Activity Recognition，HAR）是一个旨在识别个人有关活动的信息。它使人工和智能系统能够识别用户的身体活动，如步行、跑步和骑自行车等。随着科技的发展，人们使用智能手机以及智能手表来记录自己的活动状态已经成为了一种日常，对HAR的需求也在不断增加。例如，智能手机上的音乐播放软件可以使用HAR检测用户当前的活动，并推荐与活动相关的音乐。

而且，这一技术不仅仅应用在普通人身上，可能对与那些老年人与患有残疾的人来说，也是非常重要的。随着预期寿命的增加，越来越多的老年人在日常生活中表现出困难，可能会跌倒、走错路等。对于残疾人，世界上已有15%的人口生活在某种形式的残疾中，其中2%至 4%的人患有严重的残疾。因此，对于这些人来说，HAR 可以帮助这些人能够更快地监测（如跌倒监测）并及时与医疗机关进行联系。

目前，人类活动识别的方法主要分为基于计算机视觉（Computer Vision）与基于传感器的活动识别。基于视觉的HAR具有较高的识别准确率，但是另一方面它却有着功耗高，且涉及用户隐私的问题。而可穿戴式传感器通常不会面临隐私问题[1]。

使用传统机器学习方法的原因：

随着HAR的应用不断增长，在算法的实现方面，深度学习方法在HAR的应用表现出来高度的增长。而且在识别方面也表现出了非常高的精准度。

虽然如此，深度学习方法在大型的活动数据集上产生高精度的结果，但是在许多HAR的应用程序中，由于数据集较小，输入数据的维度较低等问题，传统的机器学习方法可能更加适用[2]。

2.数据集介绍

2.1 数据获取

MHEALTH (Mobile HEALTH) 数据集包括十名不同类型志愿者在进行多项体育活动时的身体运动和生命体征记录。 放置在受试者胸部、右手腕和左脚踝上的传感器用于测量不同身体部位所经历的运动，即加速度、转弯率和磁场方向。 位于胸部的传感器还提供 2-lead ECG 测量，可用于基本心脏监测、检查各种心律失常或查看运动对心电图的影响。即，该数据集包含 12 种运动，3 个传感器，10 个检测主题。

活动类别如下所示：

L1：站立不动（1 分钟）

L2：坐下和放松（1 分钟）

L3：躺下（1 分钟）

L4：步行（1 分钟）

L5：爬楼梯（1 分钟）

L6：腰部向前弯曲 (20 次)

L7：手臂正面抬高 (20 次)

L8：膝盖弯曲（蹲伏）(20 次）

L9：骑自行车（1 分钟）

L10：慢跑（1 分钟）

L11：跑步（1 分钟）

L12：前后跳跃 (20 次)

实验的过程与步骤具体如下：

收集的数据集包括 10 名不同轮廓的志愿者在进行 12 项体育活动时的身体运动和生命体征记录。传感器分别放置在受试者的胸部、右手腕和左脚踝上，并用松紧带固定。多个传感器的使用使我们能够测量不同身体部位所经历的运动，即加速度、转弯率和磁场方向，从而更好地捕捉身体动力学。位于胸部的传感器还提供 2 导联 ECG 测量值，这些测量值不用于开发识别模型，而是收集用于未来工作目的。例如，此信息可用于基本心脏监测、检查各种心律失常或查看运动对 ECG 的影响。所有传感模式都以 50 Hz 的采样率记录，这

被认为足以捕捉人类活动。每个会话都使用摄像机记录。考虑到每个活动涉及的身体部位的多样性（例如，手臂的正面抬高与膝盖弯曲）、动作的强度（例如，骑自行车与坐着和放松）及其执行速度或动态性（例如，跑步与静止）。这些活动是在实验室外环境中收集的，对必须执行这些活动的方式没有任何限制，但受试者在执行它们时应尽力而为。

2.2 数据集描述

数据集中有 13 个属性，各个属性的含义如下。

alx：来自左脚踝传感器的加速度（X 轴）

aly：来自左脚踝传感器的加速度（Y 轴）

alz：来自左脚踝传感器的加速度（Z 轴）

glx：左脚踝传感器的陀螺仪（X 轴）

gly：来自左脚踝传感器的陀螺仪（Y 轴）

glz：左脚踝传感器的陀螺仪（Z 轴）

arx：来自右下臂传感器的加速度（X 轴）

ary：来自右下臂传感器的加速度（Y 轴）

arz：来自右下臂传感器的加速度（Z 轴）

grx：来自右下臂传感器（X 轴）的陀螺仪

gry：来自右下臂传感器的陀螺仪（Y 轴）

grz：来自右下臂传感器的陀螺仪（Z 轴）

subject：志愿者编号

Activity：对应的活动

3. 探索性分析

3.1 数据可视化

根据数据的分布图，容易看出对于大部分属性，数据集中存在严重的偏度问题，与正态分布相差很大。因此在对数据进行数据集划分前，先对数据集进行标准化处理。

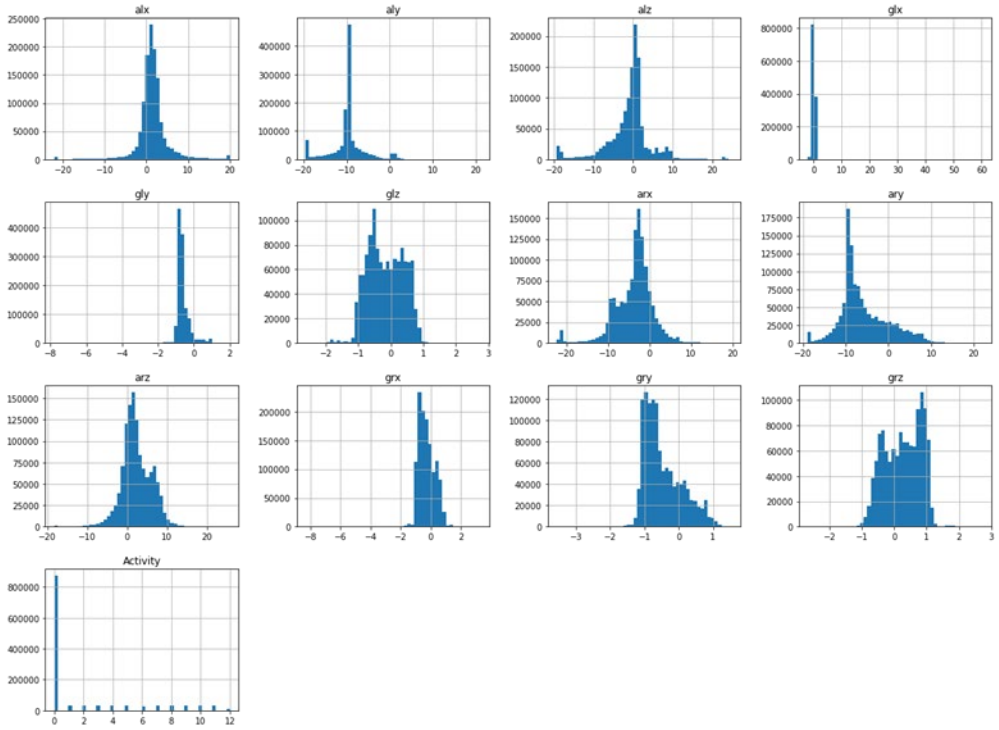

图-3.1 探索性分析

在数据标准化处理之后，对数据按照 3:2 的比例，将原数据集划分为训练集和测试集进行模型训练。

3.2 欠采样与过采样

在本实验中使用了采样的方法来处理数据不平衡的问题。

过采样顾名思义就是从样本少的类别中随机抽样，再将抽样得来的样本添加到数据集中，从而达到类别平衡。这种方法存在一个问题，因为重复采样往往会导致严重的过拟合。欠采样的思想同样比较简单，从多数类样本中随机选取一些剔除掉。这种方法的缺点是被剔除的样本可能包含着一些重要信息，致使学习出来的模型效果不好。在本实验中，基础数量设定为 30720，因为大多数运动的观测值个数均为这个数。对低于这个数量的运动进行过采样，

对高于这个数量的运动进行欠采样，从而来保证数据的平衡。

关于过采样方法， 本实验中采用了 SMOTE 方法（synthetic minority oversampling technique）。SMOTE 会随机选取少数类样本用以合成新样本，而不考虑周边样本的情况，从而保证生成的数据相对来讲比较平均。但这种方法也有不足，如果选取的少数类样本周围全是多数类样本，这类的样本可能是噪音，则新合成的样本会与周围的多数类样本产生大部分重叠，致使分类困难。

4. 分类算法

实验中的模型主要分为传统机器学习模型和深度学习模型。其中，机器学习模型包括逻辑回归，light-GBM，XGBoost，Catboost，投票分类器和 Stacking 算法。使用的深度学习模型是多层神经网络模型。

4.1.1 逻辑回归

逻辑回归是机器学习中一种常见的分类模型，其对于简单的分类问题具有良好的效果。其基本原理是采用 sigmoid 函数作为我们的预测函数，来预测条件概率 P（y = 1 | x）。在我们的问题中，sigmoid 函数的输出就是乘客存活下来的概率，范围在 [0，1] 之间。模型在训练的过程中，通过不断最小化极大似然代价函数，来提高模型预测的准确率。在训练的过程中，加入正则化项，可在一定程度上减轻模型过拟合。

4.1.2 LightGBM

LightGBM （Light Gradient Boosting Machine）是一个实现 GBDT 算法的框架，支持高效率的并行训练[3]。 GBDT (Gradient Boosting Decision Tree) 是机器学习中一个长盛不衰的模型，其主要思想是利用弱分类器（决策树）迭代训练以得到最优模型，该模型具有训练效果好、不易过拟合等优点。GBDT 在每一次迭代的时候，都需要遍历整个训练数据多次。如果把整个训练数据装进内存则会限制训练数据的大小；如果不装进内存，反复地读写训练数据又会消耗非常大的时间。LightGBM 提出的主要原因就是为了解决 GBDT 在海量数据遇到的问题，让 GBDT 可以更好更快的用于实践。在实验中，估计值个数设定为了 400。

4.1.3 XGBOOST

XGBoost 是陈天奇等人开发的一个开源机器学习项目，高效地实现了 GBDT 算法并进行了算法和工程上的许多改进[4]。XGBoost 本质上还是一个 GBDT，但是力争把速度和效率发挥到极致。XGBoost 的核心算法思想如下：

（1）不断地添加树，不断地进行特征分裂来生长一棵树，每次添加一个树，其实是学习一个新函数 f(x)，去拟合上次预测的残差；

（2）当我们训练完成得到 k 棵树，我们要预测一个样本的分数，其实就是根据这个样本的特征，在每棵树中会落到对应的一个叶子节点，每个叶子节点就对应一个分数；

（3）最后只需要将每棵树对应的分数加起来就是该样本的预测值。

在实验中，树的最大生长深度设置为了 5，学习率为 0.1，估计值个数设置为了 160.

4.1.4 CatBoost

CatBoost 是俄罗斯的搜索巨头 Yandex 在 2017 年开源的机器学习库，是 Boosting 族算法的一种[5, 6]。CatBoost 是一种基于对称决策树（oblivious trees）为基学习器实现的参数较少、支持类别型变量和高准确性的 GBDT 框架，主要解决的痛点是高效合理地处理类别型特征，CatBoost 是由 Categorical 和 Boosting 组成。此外，CatBoost 还解决了梯度偏差（Gradient Bias）及预测偏移（Prediction shift）的问题，从而减少过拟合的发生，进而提高算法的准确性和泛化能力。在实验中，估计值设置为了 100。

4.1.5 VotingClassifier

投票法是集成学习 里面针对分类问题的一种组合策略，其基本思想是选择算法中输出最多的那个类，硬投票是选择算法输出最多的标签，若标签数量相等，则按照升序的次序进行选择[7]。在本实验中采用的 hard voting 的方法，即根据少数服从多数来确定最终结果。

4.1.6 StackingClassifier

Stacking 方法是一种分层模型集成框架[8, 9]。以两层为例，首先将数据集分成训练集和测试集，利用训练集训练得到多个初级学习器，然后用初级学习器对测试集进行预测，并将输出值作为下一阶段训练的输入值，最终的标签作为输出值，用于训练次级学习器（通常最后一级使用 Logistic 回归）。由于两次所使用的训练数据不同，因此可以在一定程度上防止过拟合。在本实验中，仅有两层学习器，第二级学习模型师逻辑回归模型。

4.1.7 深度学习

近年来，深度学习在各个领域都表现出优秀的能力。与传统的模式识别方法不同，深度

学习可以极大的减轻设计特征的工作量，并且可以通过训练端到端的神经网络来学习更多高级和有意义的特征。此外，深层网络结构更适合无监督学习。

5. 实验结果

实验中使用了不同的分类模型，具体的模型结果如下所示。

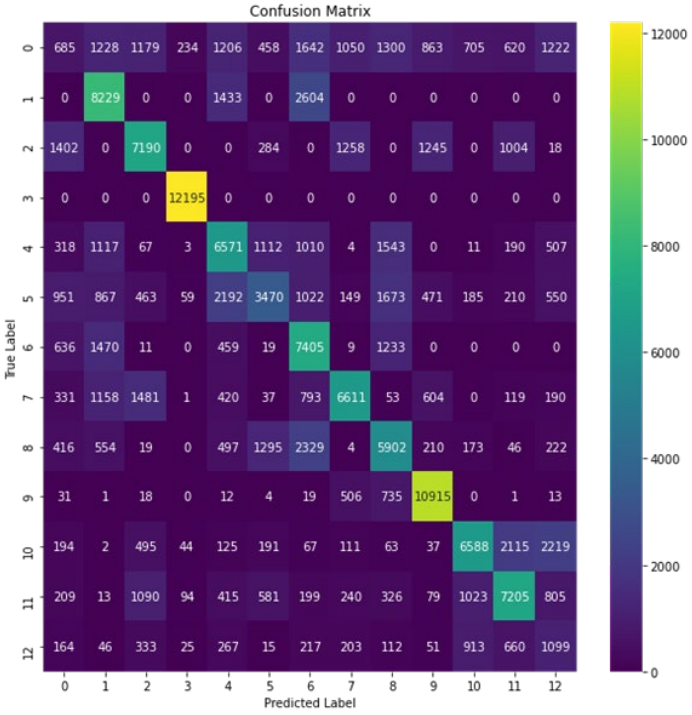

图 5.1：逻辑回归结果

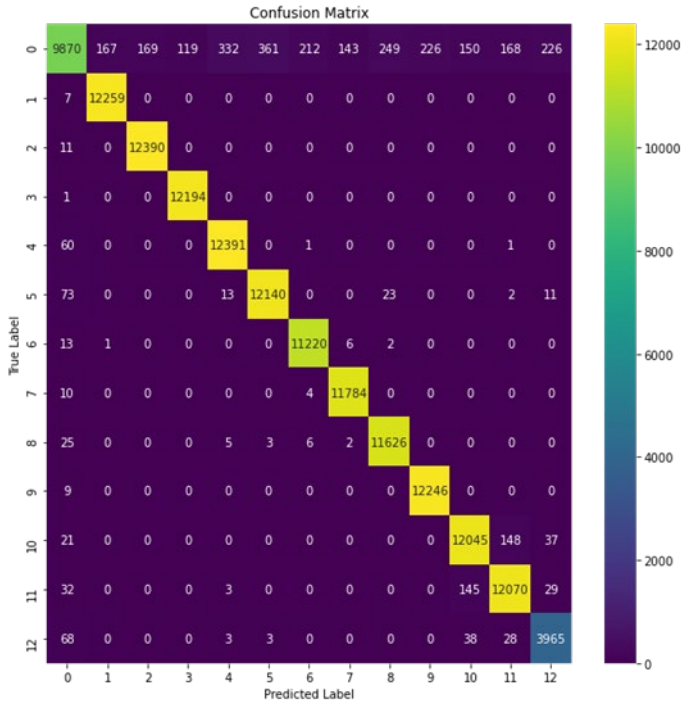

图 5.2：Light-GBM 结果

图 5.1 以混淆矩阵的方式展示了逻辑回归的分类结果，由图示可以看出结果并不好。进一步，根据运行结果显示，逻辑回归模型的测试集准确率仅有 56.21%，且用时 16s.

图 5.2 以混淆矩阵的方式展示了 Light-GBM 的结果，由图示可以看出结果非常好。进一步，根据运行结果显示，Light-GBM 模型的测试集准确率为 97.75%，且用时 18s.

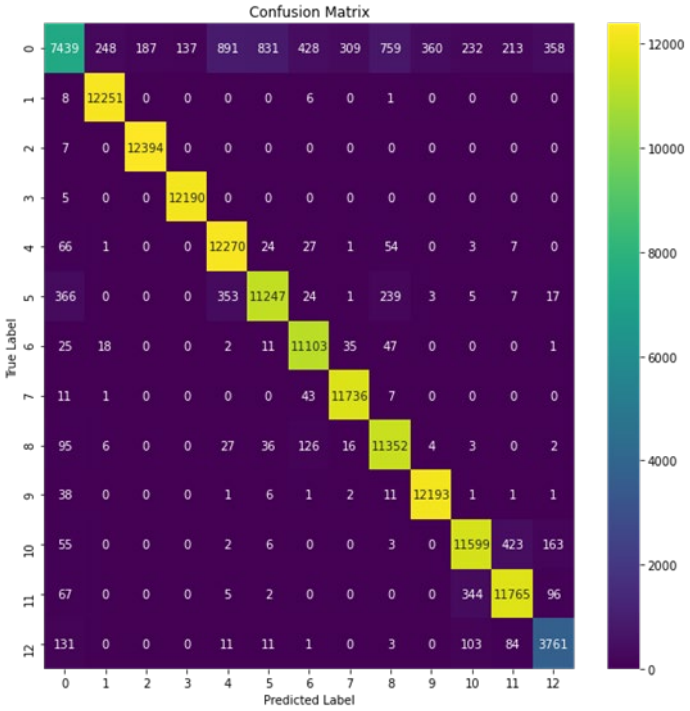

图 5.3：XGBoost 结果

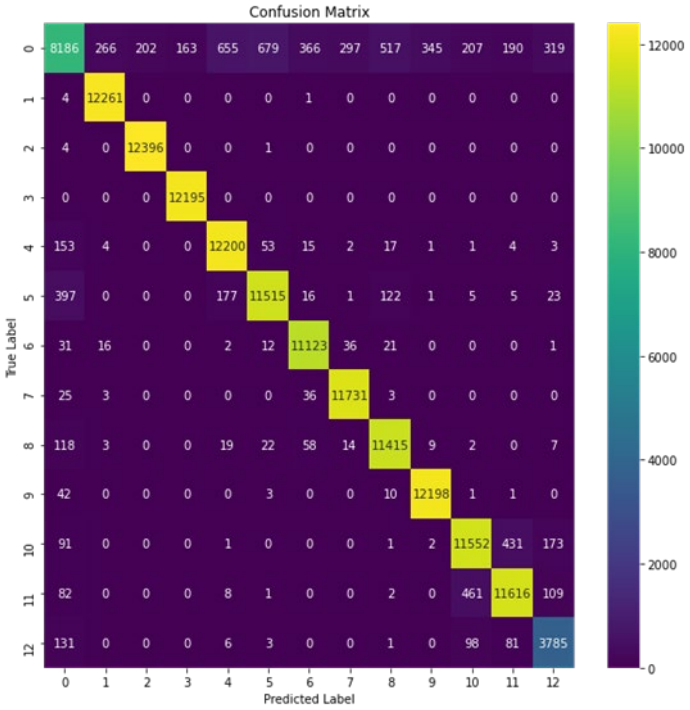

图 5.4：CatBoost 结果

图 5.3 以混淆矩阵的方式展示了 XGBoost 的结果，由图示可以看出结果非常好。进一

步，根据运行结果显示，XGBoost 模型的测试集准确率为 94.47%，但用时 104s，说明 XGBoost 方法效率相较于 LightBoost 比较低。

图 5.4 以混淆矩阵的方式展示了 CatBoost 的结果，由图示可以看出结果非常好。进一步，根据运行结果显示，CatBoost 模型的测试集准确率为 95.06%，用时最短仅 15s，说明 CatBoost 方法在所有的基于树分类的模型中，效率是最高的。

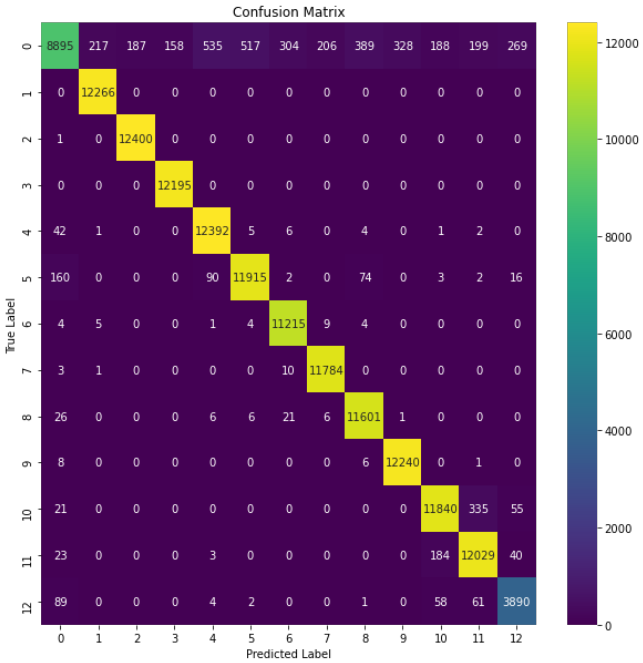

图 5.5：Hard-Voting 结果

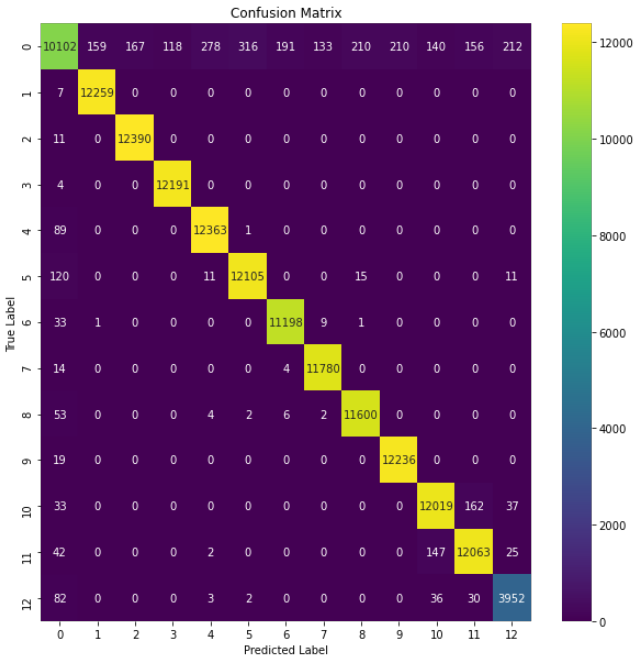

图 5.6： Stacking 结果

图 5.5 以混淆矩阵的方式展示了 Hard-Voting 的结果，由图示可以看出结果非常好。

进一步，根据运行结果显示，Hard-Voting 模型的测试集准确率为 96.72%，但用时 159s，说明 Voting 方法相较于树模型效率较低。

图 5.6 以混淆矩阵的方式展示了 Stacking 的结果，由图示可以看出结果非常好。进一步，根据运行结果显示，Stacking 模型的测试集准确率为 97.79%，用时最长为 841s，说明 stacking 方法效率是最低的。

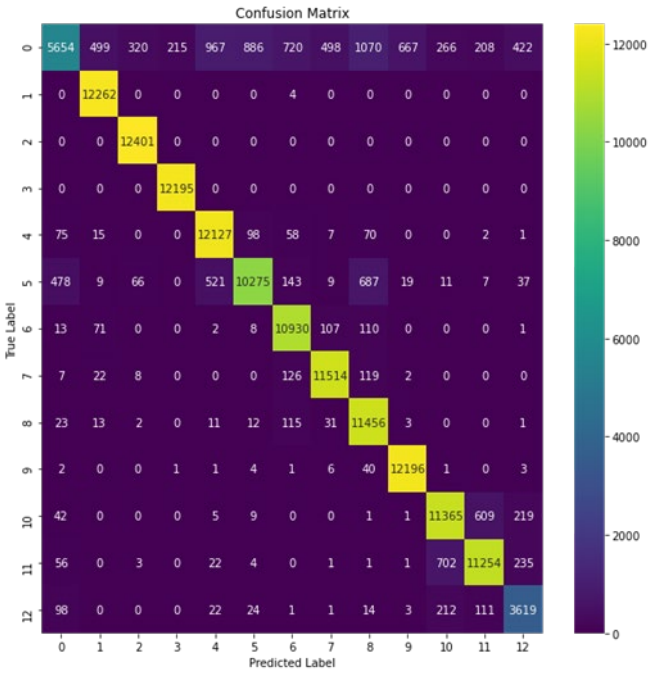

图 5.7：深度学习模型（多层神经网络）结果

图 5.7 以混淆矩阵的方式展示了多层神经网络的结果，由图示可以看出结果非常好。进一步，根据运行结果显示，深度学习模型的测试集准确率为 91.76%，但用时 221s。各方面相比较其他模型，并没有特别突出的地方。

将上述所有模型结果汇总至下表 5.1：

模型	准确率	用时（s）
逻辑回归	56.21%	16
Light-GBM	97.75%	18
XGBoost	94.47%	104
CatBoost	95.06%	15
Hard-Voting	96.72%	159
Stacking	97.79%	841
深度学习	91.76%	221

表 5.1：模型表现汇总

6.总结

根据个模型结果可以看出，逻辑回归模型的准确率最低，表现最差，仅有 56.21%。集成算法中的 Stacking 算法准确率最高，表现最佳，为 97.79%，但是其用时最长为 841s。基于树模型的各类算法，准确率和用时均比较优秀。如 Light-GBM 算法，其准确率达到了 97.75%，但其用时仅 18s，更适合于实践应用。除此之外，深度学习算法并没有表现出优秀的表现，其准确率低于基于树模型的算法和集成算法。分析其原因，一方面可能因为数据集不足，并没有训练得到最优的深度学习模型，另一方面因为仅仅使用了多层神经网络的简单深度学习算法。为了深一步的探索，后期可以尝试使用复杂的神经网络算法，如 CNN[10]等。

7. 参考文献

[1] Chen, Liming, et al. "Sensor-based activity recognition." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42.6 (2012): 790-808.

[2] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436-444.

[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

[4] Tianqi Chen, Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. KDD ’16, August 13-17, 2016, San Francisco, CA, USA

[5] Anna Veronika Dorogush, Andrey Gulin, Gleb Gusev, Nikita Kazeev, Liudmila Ostroumova Prokhorenkova, Aleksandr Vorobev "Fighting biases with dynamic boosting". arXiv:1706.09516, 2017.

[6] Anna Veronika Dorogush, Vasily Ershov, Andrey Gulin "CatBoost: gradient boosting with categorical features support". Workshop on ML Systems at NIPS 2017.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

[8] Tang, J., S. Alelyani, and H. Liu. "Data Classification: Algorithms and Applications." Data Mining and Knowledge Discovery Series, CRC Press (2015): pp. 498-500.

[9] Wolpert, David H. "Stacked generalization." Neural networks 5.2 (1992): 241-259.

[10] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton.” ImageNet Classification with Deep Convolutional Neural Networks”. Communications of the ACMVolume 60Issue 6June 2017 pp 84–90https://doi.org/10.1145/3065386