Mo Lab1 Report - Bipolar Disorder Detection

万晨阳 3210105327

实验内容介绍

实验背景

双相障碍检测，即通过医学检测数据预测病人是否双相障碍，或双相障碍治疗是否有效。医学数据包括医学影像数据与肠道数据。由于缺少医学样本且特征过多，因此选取合适的特征对双模态特征进行整合并训练合适的分类器进行模型预测具有较强的现实需求与医学意义。

实验要求

本实验需要我们完成少样本、多特征下的监督学习。主要要求如下：

1. 实现双模态特征选择与提取整合。
2. 选择并训练机器学习模型进行准确分类。
3. 分析不同超参数以及特征选择方法对模型的结果影响。

实验环境

使用的主要相关库如下：

- numpy
- sklearn

模型训练与改进

基本思路

- 对于高维特征的问题，`特征工程/特征选择`比模型的调参更加重要，降低特征向量维度并选取关键特征是模型泛化提升的重点。
- 对于小样本高维特征的问题，我们选择简单、可解释性高的模型。在本次任务中我们选择`Logistic回归`、`支持向量机`、`朴素贝叶斯`等模型进行尝试和比较。这些模型对于线性问题有比较好的表现，相比于较为复杂的黑箱更适合本任务。
- 进行严格的`交叉验证`。因为数据样本少，所以样本集合划分的随机性会对模型的训练过程造成很大的影响。我选择在训练和调参的过程中都加入交叉验证以降低数据集划分随机性带来的影响。

数据预处理与特征选择

归一化

在读入数据之后进行数据的标准化。利用 `MinMaxScaler` 进行数据的归一化操作，这样可以消除指标之间的量纲影响，便于不同单位或量级的指标能够进行比较和加权。

```
    scaler = MinMaxScaler()
    feature1 = pd.DataFrame(scaler.fit_transform(feature1_raw))
    feature2 = pd.DataFrame(scaler.fit_transform(feature2_raw))
```

特征选择

任务要求中明确提到了需要我们实现双模态特征选择与提取整合，我选择进行**特征**级融合，也即对于`feature1`和`feature2`分别提取重要特征后连接为新的特征向量并返回。对于选择特征的方法，我尝试了`PCA`、`Pearson correlation coefficient`、`Mutual information`三种方法， 利用`SelectKBest`选择得分最高的一定数量的特征。同时对于不同的特征数量，我也进行了遍历搜索，以保证在尽可能少的特征数目下有尽可能好的模型效果。最终我利用`Pearson correlation coefficient`在`feature1`中选择了3个特征，在`feature2`中选择了3个特征，筛选结果如下：

```
select feature1 name: [ 2060 2064 3912]
select feature2 name: [ 77 134 247]
new_features shape: (39, 6)
```

数据集划分

按照训练集（60%）、验证集（20%）、测试集（20%）的比例划分数据集。

```
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=47, s

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=42, st
```

数据增强

由于数据量过少，我们对于训练集加入高斯噪声进行复制，以达到数据增强、扩充训练集的效果。

```
def add_gaussian_noise(X, y, noise_stddev = 0.05):
    num_samples, num_features = X.shape
    noise = np.random.normal(loc=0.0, scale=noise_stddev, size=(num_samples,   num_features))
    X_noisy = X + noise
    X_augmented = np.vstack((X, X_noisy))
    y_augmented = np.vstack((y, y))
    return X_augmented, y_augmented

X_train, y_train = add_gaussian_noise(X_train, y_train)
```

模型选择与参数调整

模型选择

对于该问题，我尝试了使用`Logistic回归`、`支持向量机`、`朴素贝叶斯`三种模型分别独立进行训练，都达到了比较好的效果。为了提升模型的鲁棒性，我选择使用`VotingClassifier`将训练好的三个模型进行集成，采用加权软投票的方式输出结果。

参数调整

我选择使用`F1 Score`作为评分，加入交叉验证进行网格搜索以自动调参。对于上述的`Logistic回归`、`支持向量机`、`朴素贝叶斯`三种模型以及模型集成后的投票分类器，选择调整的参数分别如下：

- Logistic Regression

```
parameters = {'C':[1, 10],
              'solver':('newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'),
              'max_iter':[100, 1000],
              'multi_class':('ovr', 'multinomial')}
```

- SVC

```
parameters = {'kernel':('linear', 'rbf'),
              'C':[1, 10],
```

```
                'gamma':[0.001, 0.0001],
                'max_iter':[100, 1000]}
```

- Naive Bayes

```
parameters = {'var_smoothing':[1e-09, 1e-08, 1e-07, 1e-06, 1e-05]}
```

- Voting Classifier

```
parameters = {'voting':['soft'],
              'weights':[[2,1,1], [1,2,1], [1,1,2]]}
```

以`Logistic Regression`为例，模型建立和训练的过程如下，我们利用网格搜索自动调参：

```
def model_and_train(model, parameters, X_train, y_train, scorer, kfold):
    grid_obj = GridSearchCV(estimator = model, param_grid = parameters, scoring = scorer, cv = kfold)
    plot_learning_curve(grid_obj, X_train, y_train, cv = kfold, n_jobs = 4)
    grid_obj.fit(X_train, y_train)
    best_clf = grid_obj.best_estimator_
    return best_clf

scorer = make_scorer(fbeta_score, beta=1)
kfold = KFold(n_splits = 10)

# logistic regression
lr = LogisticRegression(random_state = 42)
parameters = {'C':[1, 10],
              'solver':('newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'),
              'max_iter':[100, 1000],
              'multi_class':('ovr', 'multinomial')}
lr = model_and_train(lr, parameters, X_train, y_train, scorer, kfold)
```

而在三个模型都达到了比较好的效果后，我们进行模型的集成，并针对加权情况进行简单的调参：

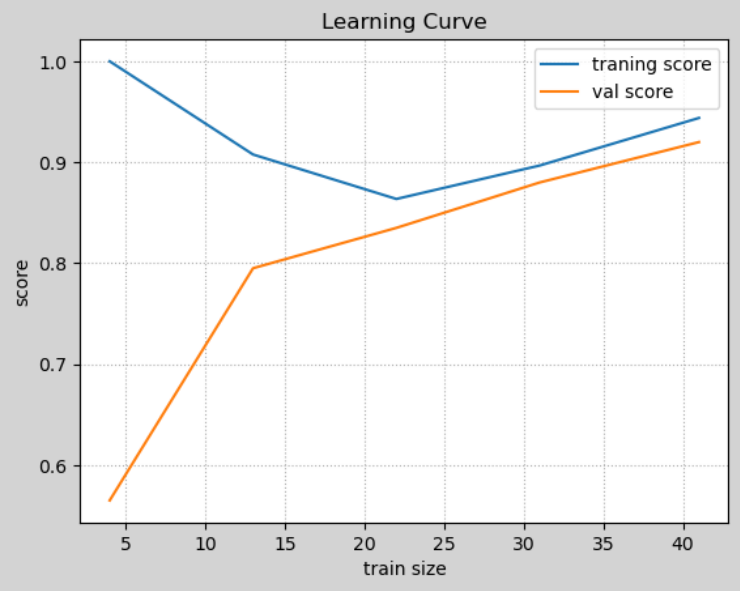
```
voting_clf = VotingClassifier(estimators=[('lr', lr),
                                          ('svc', svc),
                                          ('gs',gs)],
                                          voting = 'soft')
plot_learning_curve(voting_clf, X_train, y_train, cv=kfold, n_jobs=4)
grid_obj = GridSearchCV(estimator = voting_clf,
                        param_grid = {'weights':[[2,1,1], [1,2,1], [1,1,2]]},
                        scoring = scorer, cv = kfold)
grid_obj.fit(X_train, y_train)
best_clf = grid_obj.best_estimator_
```

结果展示

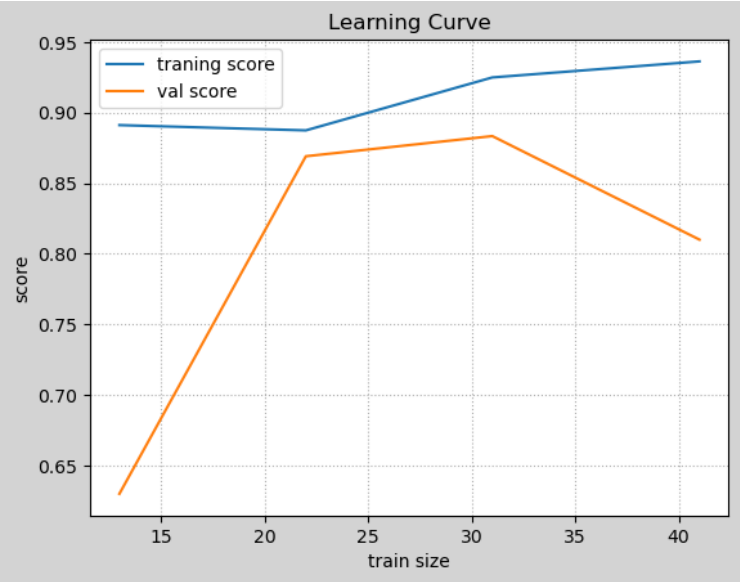
训练结果

经过训练和调参，以下为三个子模型的学习曲线：

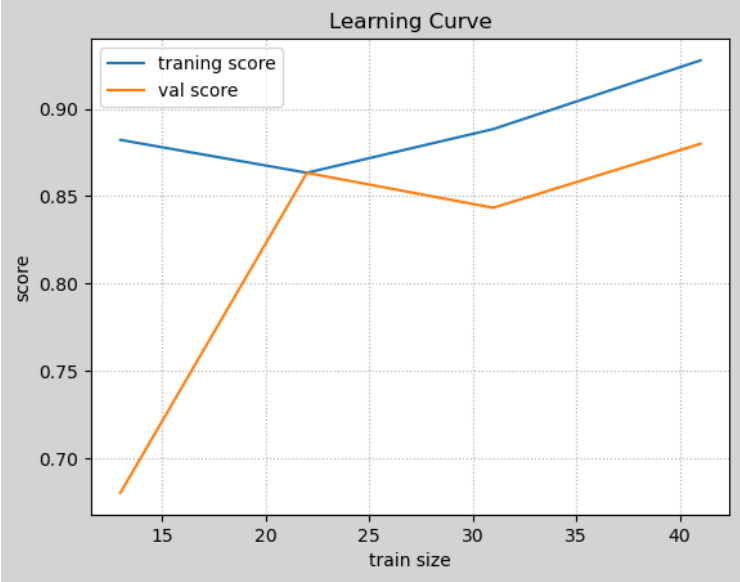
- Logistic Regression



- SVC

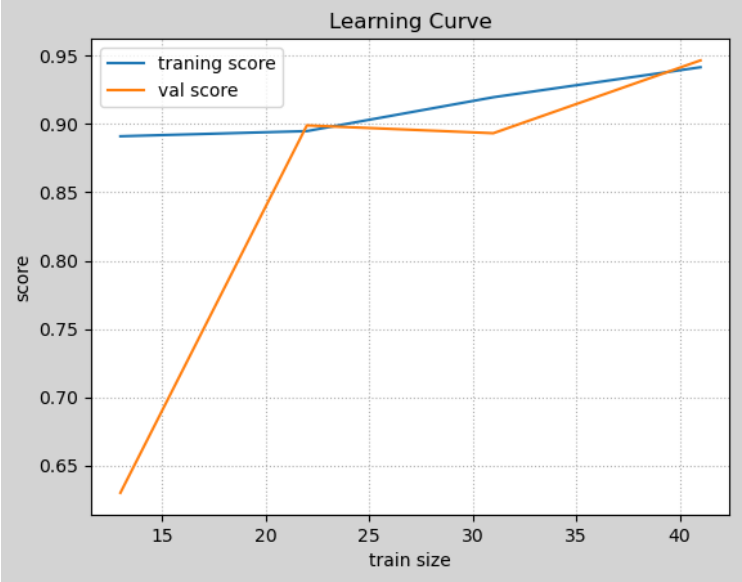


- Naive Bayes



可以看到受数据集样本过少的影响，存在一定的波动，但是在训练集和验证集上都取得了比较好的效果。

而对于集成后的模型，学习曲线的结果如下：



模型最终得分情况如下：

- validation data
  - Final accuracy score on the validation data: 0.8750
  - Recall score on validation data: 1.0000
  - Final F-score on the validation data: 0.8889
- test data
  - Accuracy on test data: 1.0000
  - Recall on test data: 1.0000
  - F-score on test data: 1.0000

我们在验证集和测试集上都取得了比较好的效果。

测试结果

测试点	状态	时长	结果
测试结果	✓	1s	测试成功，在10个测试样本中，准确率为 1.0, 召回率为 1.0, F-score为 1.0

结论分析与体会

超参数对模型的结果影响

由于我使用了模型集成的方法，所以对于子模型和集成模型的典型超参数进行分别分析。

1. Logistic Regression

- **正则化参数 (C):** 调整正则化参数以控制模型的复杂性。较小的C值会增加正则化强度，可能防止过拟合。本问题中由于使用集成模型，子模型的过拟合现象集成后被削弱，所以选择C适中的模型。
- **惩罚项 (penalty):** 选择合适的惩罚项（L1或L2）。L1惩罚通常导致稀疏权重，适用于特征选择。不过由于这一参数与求解器挂钩，所以两者同时作为组合进行调参。

2. SVM

- **核函数 (kernel):** 根据数据特征选择合适的核函数。常用的有线性核、多项式核和高斯核。本问题相对简单，使用复杂核函数的效果并不好，简单的线性分类就能够达到效果。

- **正则化参数 (C):** 调整C值来平衡分类错误和决策边界的平滑度。由于使用了集成，我们选择C适中的模型以保证有足够的复杂度同时不至于过拟合。
- **Gamma参数 (对于RBF核):** 控制单个训练样本的影响范围。较小的值会导致高斯函数的标准差增大，决策边界变得更加平滑。

3. Naive Bayes

- **平滑参数 (alpha):** 对于朴素贝叶斯，特别是多项式朴素贝叶斯，调整平滑参数以避免概率为零的情况。本模型训练结果为选择了较小的平滑参数。

4. 集成模型

- **投票策略 (voting):** 选择硬投票（majority voting）或者软投票（根据每个模型的概率加权投票）。由于我们的子模型比较少，所以在本问题中明显软投票带来的容错率更高。
- **权重分配 (weights):** 对于软投票，可以为每个模型分配不同的权重。这样可以减少对于每个模型的准确度的依赖，而偏向于某些效果更好的模型。

以下为最终的模型有关参数信息：

```
VotingClassifier(estimators=[('lr',
                              LogisticRegression(C=10, class_weight=None,
                                                 dual=False, fit_intercept=True,
                                                 intercept_scaling=1,
                                                 l1_ratio=None, max_iter=100,
                                                 multi_class='ovr', n_jobs=None,
                                                 penalty='l2', random_state=42,
                                                 solver='newton-cg', tol=0.0001,
                                                 verbose=0, warm_start=False)),
                             ('svc',
                              SVC(C=10, break_ties=False, cache_size=200,
                                  class_weight=None, coef0=0.0,
                                  decision_function_shape='ovr', degree=3,
                                  gamma=0.001, kernel='linear', max_iter=100,
                                  probability=True, random_state=42,
                                  shrinking=True, tol=0.001, verbose=False)),
                             ('gs',
                              GaussianNB(priors=None, var_smoothing=1e-09))],
                 flatten_transform=True, n_jobs=None, voting='soft',
                 weights=[2, 1, 1])
```

特征选择方法对于模型的结果影响

我尝试了如下的方法，并从理论以及实际效果来分析了他们的特点以及对结果的影响。

1. 相关性系数排序:
   与其他方法相比选取的特征不够显著，漏掉了部分重要的非线性特征导致结果不佳。

- **优势：** 可以用来查看每个特征与目标变量之间的线性相关性。高相关性的特征可能对模型的预测有积极影响。
- **缺点：** 只捕捉线性关系，可能无法捕捉非线性关系。

2. 卡方检验:
   相对较好，但是统计方法对于更复杂的关系缺少解释能力，不够直观。

- **优势：** 适用于分类问题，可以用来评估特征和目标变量之间的关联性。
- **缺点：** 只考虑了特征和目标变量之间的统计独立性，可能忽略了更复杂的关系。

3. 互信息法:
   相对较好，但是计算比较慢。一开始我是基于互信息法进行研究的，效果和皮尔逊相关系数相比差距不大。二者的提取结果基本一致。

- **优势：** 能够捕捉更广泛的关系，包括非线性关系。
- **缺点：** 计算相对较复杂，可能在高维空间下变得昂贵。

4. t-SNE:

- **优势：** 适用于可视化高维数据，可以帮助发现数据中的聚类结构。
- **缺点：** 主要用于可视化，不太适合直接用于特征选择。

5. PCA (主成分分析):
   漏掉了部分重要的非线性特征导致结果不佳。

- **优势：** 通过线性变换将原始特征转换为一组线性无关的主成分，有助于减小数据的维度。
- **缺点：** 可能丢失一些非线性关系，不太适用于捕捉非线性特征。

6. 皮尔逊相关系数:

- **优势：** 衡量了两个变量之间的线性相关性，可以用于选择与目标变量强相关的特征。
- **适用性：** 在特征之间存在线性关系时效果较好，不适用于捕捉非线性关系。

我最终选择了基于皮尔逊相关系数进行双模态特征选择和融合。优势如下：

- **直观性：** 它提供了直观的线性相关性度量，易于理解。
- **计算效率：** 计算速度相对较快，尤其适用于小样本高维数据。
- **方向性：** 除了强度，还提供了特征与目标变量之间的方向性。
- **过滤冗余特征：** 可以帮助排除高度相关的特征，减小模型的复杂性。
- **可解释性：** 通过皮尔逊相关系数选择的特征更容易被解释，有助于理解模型的特征重要性。

个人体会

- 对于机器学习而言，海量的数据往往是监督学习能训练出好的模型的关键。而对于小样本的监督学习任务，我们首先要做的还是尽可能从有限的样本中获取尽可能多的知识，比如在模型的训练过程中加入先验知识等。对于本任务，我们虽然没有加入先验知识进行模型训练的能力，但是进行数据增强也是可选的方案。
- 对于高维特征的数据，特征选择工作是较为重要的。好的特征选择能够降低训练的难度并且提升模型的泛化能力。对于小样本的场景来说，特征的选择和数据的处理是比模型的训练调参更加重要的。
- 对于本任务，由于数据量少，所以数据集划分的情况对模型的效果有较大的影响（尽管已经加入了交叉验证），比如我的实验是在`random_state = 42`的情况下进行的，我尝试使用了别的随机状态进行划分，结果有的变得更好而有的变得更坏。事实上由于我们的测试集和验证集数量太少，以至于无法完全合理的评价模型的优劣，我们所得到的也只是相对较优的一个模型。