《数据挖掘应用实验》课程报告

实验 3

yunwuyue lzu_email Student ID

Lanzhou University

Date: 2023 年 6 月 5 日

1 实验数据集

1.1 IRIS

IRIS 数据集 [1] 最早由英国统计学家和生物学家 Ronald Fisher 于 1936 年介绍，该数据集成为统计学和机器学习中广泛应用的基准数据集之一。

IRIS 数据集常用于机器学习和模式识别任务中，尤其是用于分类算法的测试和性能评估。通过对四个特征进行分析和模型训练，可以尝试根据特征值预测鸢尾花的品种。

数据规模上，IRIS 数据集包含 150 个样本，共计 5 列数据，部分数据如表1所示。

1.1.1 特征描述

IRIS 数据集种每个样本有 4 个特征，如下所示：

1. sepal length（花萼长度）
2. sepal width（花萼宽度）
3. petal length（花瓣长度）
4. petal width（花瓣宽度）

1.1.2 标签描述

IRIS 数据集中存在 1 种标签 class(鸢尾花品种)，该标签将每个样本标记为以下三个类别之一：

- Setosa
- Versicolor
- Virginica

1.2 Breast Cancer Wisconsin Data Set

Breast Cancer Wisconsin (Diagnostic) 数据集 [2] 由威斯康星大学医学院的 Dr. William H. Wolberg 等人收集并整理，用于乳腺癌诊断任务。

表 1: IRIS 数据集（部分）

sepal length	sepal width	petal length	petal width	class
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5.0	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa

数据规模上，该数据集包含 599 个样本，每个样本有 30 个特征和 1 个标签 (诊断结果)，还有一列数据表示 ID Number，共计 32 列数据。

其部分数据如表2所示。

1.2.1 特征描述

Breast Cancer Wisconsin 数据集中的特征是从乳腺肿块的数字化图像中计算得出的，共有 30 种，包含以下 10 种真实特征：

1. radius（半径）
2. texture（质地）
3. perimeter（周长）
4. area（面积）
5. smoothness（平滑度）
6. compactness（紧密度）
7. concavity（凹度）
8. concave points（凹点）
9. symmetry（对称性）
10. fractal dimension（分形维数）

每种真实特征又分为平均值、误差、最差值 3 项，如‘radius’这种真实特征又分为 Mean Radius, Radius SE, Worst Radius3 项，因此一共有 30 种特征。

1.2.2 标签描述

Breast Cancer Wisconsin 数据集中存在 1 种标签，为诊断结果（Diagnosis），该标签将每个样本标记为两个类别之一：良性（B）或恶性（M），表示乳腺肿块的诊断结果。

需要注意的是，该数据集中只有 Diagnosis 列的数据为 string 类型，因此在后续处理中需要将 Diagnosis 列转换为数值型数据，实验中，这一步是通过调用 sklearn.preprocessing.LabelEncoder 库中的 LabelEncode 类实现的。

表 2: Breast Cancer 数据集（部分）

	ID number	Diagnosis	Mean Radius	Mean Texture	Mean Perimeter
0	842517	M	20.57	17.77	132.90
1	84300903	M	19.69	21.25	130.00
2	84348301	M	11.42	20.38	77.58
3	84358402	M	20.29	14.34	135.10
4	843786	M	12.45	15.70	82.57
...	...	...	...	...	...
563	926424	M	21.56	22.39	142.00
564	926682	M	20.13	28.25	131.20
565	926954	M	16.60	28.08	108.30
566	927241	M	20.60	29.33	140.10
567	92751	B	7.76	24.54	47.92

2 相关算法简介

2.1 随机森林模型

随机森林是一种集成学习方法，通过组合多个决策树来构建一个更强大的分类器或回归器。它采用自助采样法对原始训练数据集进行有放回抽样，生成多个不同的训练子集，然后针对每个子集独立地构建决策树。最后，通过对各个决策树的预测结果进行投票或平均，来获得最终的预测结果。

假设随机森林由 T 个决策树组成，每个决策树用函数 $h_t(x)$ 表示。随机森林的预测结果可以表示为：

$$H_T(x) = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

其中，$H_T(x)$ 是随机森林的预测函数。每个决策树的预测结果通过投票或平均来得到最终的预测结果。

伪代码见下列算法1。

Algorithm 1: 随机森林算法

Input: 训练数据集 $S = (x_i, y_i)_{i=1}^m$，基学习算法 L，决策树数量 T，每棵树的特征子集大小 k

for $t \leftarrow 1$ *to* T **do**

从 S 中有放回地抽样生成训练子集 S_t; 从 S_t 中随机选择 k 个特征作为该树的输入特征; 使用训练子集 S_t 构建决策树 h_t，其中的特征只考虑选定的 k 个特征;

end

Output: 随机森林模型 $H_T(x)$

2.2 Stacking 模型

Stacking 是一种集成学习方法，通过将多个基学习器的预测结果作为输入，构建一个元学习器来获得最终的预测结果。Stacking 的核心思想是将基学习器的输出作为新的特征输入到元学习器中，使其能够学习到基学习器之间的关系和权重。基学习器可以是不同的分类器或回归器，而元学习器可以是任何适合的机器学习算法。

假设训练数据集 $S = (x_i, y_i)_{i=1}^m$，Stacking 模型由 K 个基学习器和一个元学习器组成。每个基学习器 h_k 的输出可以表示为:

$$Z_k = h_k(X)$$

其中，X 是输入特征矩阵。将基学习器的输出作为元学习器的输入特征，可以表示为:

$$Z = [Z_1, Z_2, ..., Z_K]$$

元学习器的预测结果可以表示为:

$$Y' = g(Z)$$

其中，$g(\cdot)$ 是元学习器的函数。

伪代码见下列算法2。

Algorithm 2: Stacking 算法

Input: 训练数据集 $S = (x_i, y_i)_{i=1}^m$，基学习算法 L，元学习算法 M，基学习器数量 K

for $k \leftarrow 1$ *to* K **do**

使用 L 构建基学习器 h_k，使用训练数据集 S 进行训练; 计算基学习器 h_k 在训练数据集上的预测结果 $Z_k = h_k(X)$;

end

将所有基学习器的预测结果 $Z = [Z_1, Z_2, ..., Z_K]$ 作为新的特征矩阵; 使用 M 构建元学习器 g，使用特征矩阵 Z 和对应的真实标签 Y 进行训练;

Output: Stacking 模型 g

2.3 AdaBoost 分类器

Boosting 是一族可将弱学习器提升为强学习器的算法，弱学习器是指泛化性能略优于随机猜测的学习器，例如在二分类问题上精度略高于 50% 的分类器。

该族算法的工作机制是先在初始训练集上训练出一个基学习器 (base learner)，再根据基学习器的表现对训练样本的权重分布进行调整，使得先前基学习器预测错的样本在后续受到更多关注，然后基于调整后的权重分布训练下一个基学习器；如此重复进行，直至基学习器数目达到事先指定的值 T，最终将这 T 个基学习器进行加权线性组合。AdaBoost [3] 就是该族算法最著名的代表。

设训练数据集 $S = \{(x_i, y_i)\}_{i\in[m]}$, 基学习器 $h_t : X \to Y$, 其加权线性组合为

$$H_T(x) = \sum_{t\in[T]} \alpha_t h_t(x)$$

在算法的第 t 轮，当前的分类器组合为 $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$，经验指数损失

$$\begin{aligned}\sum_{i\in[m]} exp(-y_i H_t(x_i)) &= \sum_{i\in[m]} exp(-y_i H_{t-1}(x_i) - y_i\alpha_t h_t(x_i)) \\ &= \sum_{i\in[m]} exp(-y_i H_{t-1}(x_i)) exp(-y_i\alpha_t h_t(x_i)) \\ &\propto \sum_{i\in[m]} D_{t-1}(i) exp(-y_i\alpha_t H_t(x_i))\end{aligned}$$

其中 $D_{t1}(i) \propto exp(y_i H_{t-1}(x_i))$ 是上一轮的分类器组合 $H_{t-1}(x)$ 在样本 (x_i, y_i) 上的归一化指数损失 (使得 D_{t-1} 在训练数据集上构成一个分布), 亦是本轮样本 (x_i, y_i) 的权重。注意到 $y_i, h_t(x_i) \in 1$, 进一步化简有

$$\begin{aligned}\sum_{i\in[m]} exp(-y_i H_t(x_i)) &\propto \sum_{i\in[m]} D_{t-1}(i) exp(-y_i\alpha_t H_t(x_i)) \\ &= exp(-\alpha_t) \sum_{i\in[m]} D_{t-1}(i)\mathbb{I}(y_i = h_t(x_i)) + exp(\alpha_t) \sum_{i\in[m]} D_{t-1}(i)\mathbb{I}(y_i \neq h_t(x_i)) \\ &= exp(-\alpha_t) \sum_{i\in[m]} D_{t-1}(i) + (exp(\alpha_t) - exp(-\alpha_t)) \sum_{i\in[m]} D_{t-1}(i)\mathbb{I}(y_i \neq h_t(x_i))\end{aligned}$$

注意到第一项与 h_t 无关，故

$$h_t = \arg_h \min \sum_{i\in[m]} D_{t-1}(i)\mathbb{I}(y_i \neq h(x_i))$$

即第 t 轮选取的基分类器 h_t 应最小化加权错误率，此外令关于 α_t 的梯度

$$-exp(-\alpha_t) \sum_{i\in[m]} D_{t-1}(i) + (exp(\alpha_t) + exp(-\alpha_t)) \sum_{i\in[m]} D_{t-1}(i)\mathbb{I}(y_i \neq h_t(x_i)) = 0$$

可得

$$exp(2\alpha_t) = \frac{\sum\limits_{i\in[m]} D_{t-1}(i)}{\sum\limits_{i\in[m]} D_{t-1}(i)\mathbb{I}(y_i \neq h(x_i))}$$

$$\Rightarrow \alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$$

其中

$$\epsilon_t = \frac{\sum\limits_{i\in[m]} D_{t-1}(i)\mathbb{I}(y_i \neq h_t(x_i))}{\sum\limits_{i\in[m]} D_{t-1}(i)} = \text{加权错误率}$$

得到 α_t 和 h_t 后，下一轮的样本权重

$$D_t(i) \propto exp(-y_i H_t(x_i)) \propto D_{t-1}(i)exp(-y_i\alpha_t h_t(x_i))$$

伪代码见下列算法3。

Algorithm 3: AdaBoost 算法

Input: 训练数据集 $S = \{(x_i, y_i)\}_{i=1}^m$，基学习算法 L，迭代轮数 T

初始化权重分布 $D_1(i) \leftarrow \frac{1}{m}$;

for $t \leftarrow 1$ *to* T **do**

  $h_t \leftarrow L(S, D_t)$ 在 S 上以权重 D_t 训练基分类器 ;

  $\epsilon_t \leftarrow P(x \sim D_t, y)\mathbb{I}(h_t(x) \neq y)$ 计算加权错误率 ;

  if $\epsilon_t > 0.5$ **then**

    中止算法;

  end

  $\alpha_t \leftarrow \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ 计算 h_t 的权重系数 ;

  for $i \leftarrow 1$ *to* m **do**

    if $h_t(x_i) = y_i$ **then**

      $D_{t+1}(i) \leftarrow D_t(i)\exp(-\alpha_t)$ 更新权重 ;

    end

    else

      $D_{t+1}(i) \leftarrow D_t(i)\exp(\alpha_t)$ 更新权重 ;

    end

  end

  归一化权重 $s \leftarrow \sum_{i=1}^m D_t(i)$;

  for $i \leftarrow 1$ *to* m **do**

    $D_t(i) \leftarrow \frac{D_t(i)}{s}$;

  end

end

Output: sign $\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$

3 实验过程

3.1 数据预处理

根据 IRIS 数据集的说明iris.names与 wdbc 数据集的说明wdbc.names两个文件，对于两个数据集：

1. 缺少索引（index）
2. 不存在缺失值（Missing attribute values）

从各自的说明文件中，我们获取到两个数据集的列名，为方便后续的操作，在两个数据集中插入索引，并将修改后的数据集分别保存为“iris_modified.csv”，“cancer_modified.csv”。

3.2 随机森林与 AdaBoost

3.2.1 分类

用于进行分类任务的代码文件是，在导入必要的库（$pandas$，$sklearn$）后，定义了一系列主要函数，包括：

- load_data()：加载数据集，分别加载了经过预处理的 iris 数据集和 wdbc 数据集。
- split_data(data, target_column, test_size)：划分数据集为训练集和测试集，其中 data 是输入的数据集，target_column 是目标变量的列名，test_size 是测试集所占比例。
- train_and_predict(X_train, X_test, y_train, y_test)：使用随机森林分类器和 AdaBoost 分类器对训练集进行训练，并在测试集上进行预测。返回随机森林和 AdaBoost 的预测结果。
- evaluate_accuracy(y_test, predictions)：计算预测结果的准确率。
- main()：主函数，用于执行整个流程。在主函数中，首先调用 load_data() 加载数据集，然后定义了一组测试集大小（0.2、0.4 和 0.6），接着进行循环。在每次循环中，打印测试集大小，然后调用。

3.2.2 回归

用于进行回归任务的代码文件是 “regression.py” 文件，在导入必要的库（$pandas$，$sklearn$）后，主函数由以下步骤组成：

- 导入数据集。即’iris_modified.csv’ 与’cancer_modified.csv’ 文件。
- 选择目标属性。分别为’Iris’ 数据集中的’petal_width’ 和’Breast Cancer’ 数据集中的’Mean Radius’。
- 划分数据集。对于’Iris’ 数据集，通过 drop 函数去除了’target’ 列，得到了特征矩阵 iris_X 和目标向量 iris_y。而对于’Breast Cancer’ 数据集，通过 drop 函数去除了’Diagnosis’ 列。划分比例为测试集占总数据集的 20%，并且设置了一个随机种子值（random_state=42）以保证可复现性。
- 使用随机森林方法。创建了随机森林回归器对象 rf_regressor，对训练集进行训练后使用模型进行预测，得到了 rf_iris_predictions、rf_cancer_predictions。

- 使用 AdaBoost 方法。创建了 AdaBoost 回归器对象 adaboost_regressor，其中 base_estimator 参数设置为 DecisionTreeRegressor，对训练集进行训练后使用模型进行预测，得到了 adaboost_iris_predictions、adaboost_cancer_predictions。
- 性能评估。输出了评估结果，分别是随机森林在'Iris' 数据集和'Breast Cancer' 数据集上的均方误差，以及 AdaBoost 在两个数据集上的均方误差。

3.3 Stacking 方法

用于实现 Stacking 模型的代码文件是 "stacking_model.py" 文件，此文件中使用到的库较多，除了基础的 $numpy$，$pandas$ 之外，还有以下库：

- sklearn.preprocessing.LabelEncoder：该库中的 LabelEncoder 类用于将类别型的标签进行编码，在本文件中，使用 LabelEncoder 类将 cancer 数据中的'Diagnosis' 列转换为数值型数据。
- sklearn.model_selection.train_test_split：该库中的 train_test_split 函数用于将数据集划分为训练集和测试集，也是较常用的库。
- sklearn.ensemble.GradientBoostingClassifier：该库中的 GradientBoostingClassifier 类用来实现梯度提升分类器模型。
- sklearn.ensemble.AdaBoostClassifier：该库中的 AdaBoostClassifier 类用来实现 AdaBoost 方法。
- sklearn.linear_model.LogisticRegression：该库中的 LogisticRegression 类用于实现逻辑回归模型。
- sklearn.metrics.accuracy_score：该库中的 accuracy_score 函数用于计算分类模型的准确率。

实现的 Stacking 模型包含以下参数：base_classifiers（基分类器列表）、meta_classifier（元分类器）、X_train、y_train、X_val、y_val 和 X_test。

模型训练过程如下：首先根据 X_train 和 y_train 对基分类器进行训练，并使用这些基分类器对 X_val 进行预测，将预测结果存储在 base_classifier_predictions 中。然后，使用 meta_classifier 对 base_classifier_predictions 和 y_val 进行训练。接下来，函数使用训练好的基分类器对 X_test 进行预测，将预测结果存储在 base_classifier_predictions_test 中。最后，函数使用训练好的 meta_classifier 对 base_classifier_predictions_test 进行预测，将预测结果返回。

主函数由以下步骤组成：

- load_data() 函数：加载数据集，返回 DataFrame。
- split_data() 函数：将数据集划分为训练集、验证集和测试集。它接受三个参数：data（输入的数据集）、target_column（目标变量的列名）和 test_size（测试集所占比例）。
- stacking_model() 函数：这个函数用于创建 Stacking 模型。
- main() 函数：主函数。它首先调用 load_data() 函数加载数据集，并调用 split_data() 函数将数据集划分为训练集、验证集和测试集。接下来，定义了三个基分类器和元分类器。然后，使用 LabelEncoder 对 y_train、y_val 和 y_test 进行编码。最后，调用 stacking_model() 函数获取模型的预测结果，并输出预测结果在测试集上的准确率。

3.4 结果分析

3.4.1 分类

在分类问题中，混淆矩阵可以用来计算多个评估指标，判断模型在各个类别上的分类表现。在混淆矩阵中，行代表真实标签，列代表模型的预测结果。使用“plot_confusion_matrix.py”文件绘制出混淆矩阵1。

根据“classification.py”文件的运行结果与生成的混淆矩阵，进行以下分析：

首先，我们看到在使用 Test Size 为 0.2 时，Stacking 模型在 Breast Cancer 数据集上的准确率为 0.9649。相比之下，随机森林模型的准确率为 0.9561，AdaBoost 模型的准确率为 0.9123。可以看出，Stacking 模型在这个数据集上表现最好，其准确率明显优于单个基础分类器。

接下来，我们观察 Test Size 为 0.4 时的结果。在 Iris 数据集上，随机森林和 AdaBoost 模型的准确率均为 0.9833，而在 Breast Cancer 数据集上，随机森林和 AdaBoost 模型的准确率分别为 0.9649 和 0.9386。这表明随机森林和 AdaBoost 模型在这个数据集上表现相似且较好。

最后，当 Test Size 为 0.6 时，随机森林和 AdaBoost 模型在 Iris 数据集上的准确率均为 0.9667。在 Breast Cancer 数据集上，随机森林模型的准确率为 0.9531，而 AdaBoost 模型的准确率为 0.9413。这显示出随机森林模型在这个数据集上略优于 AdaBoost 模型。

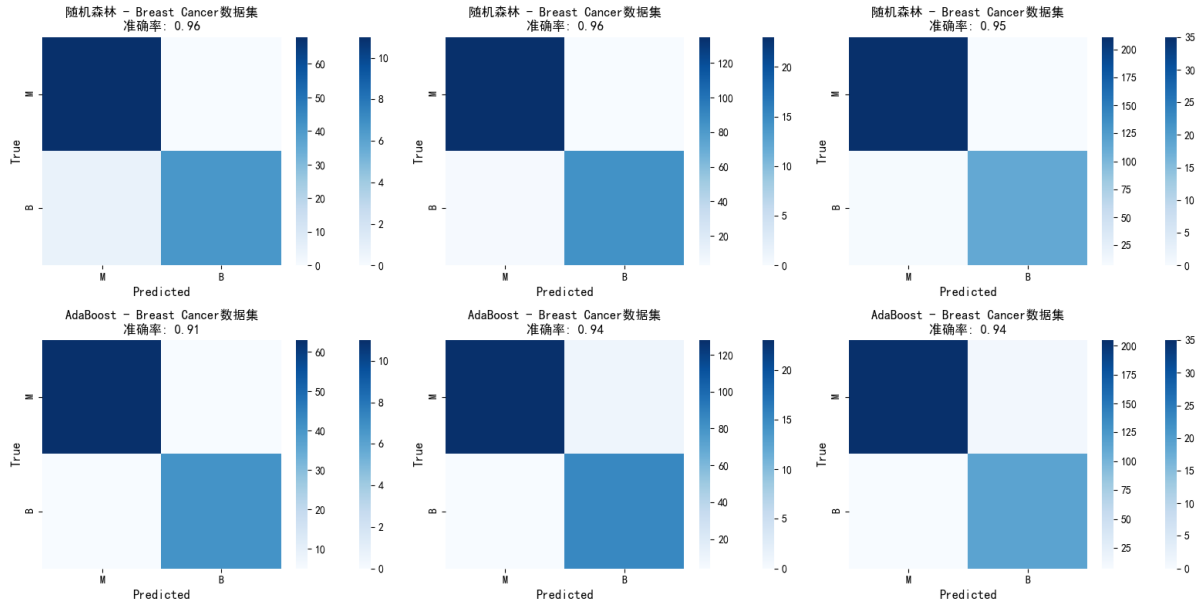

图 1: 混淆矩阵

综上所述，不同的测试集划分大小会对模型的性能产生影响。在这些情况下，Stacking 模型在 Breast Cancer 数据集上的性能相对较好，而随机森林模型在不同的数据集上都表现良好。这些结果表明了不同模型在不同数据集上的表现差异，同时也显示了使用集成学习方法（如 Stacking）能够提高模型性能的潜力。

3.4.2 回归

首先，对于 Iris 数据集，随机森林模型的均方误差为 0.000168，而 AdaBoost 模型的均方误差为 0.0。这表示随机森林模型在预测 Iris 数据集上的性能略优于 AdaBoost 模型。较小的均方误差值表明模型的预测结果与实际值之间的差异较小。

其次，对于 Breast Cancer 数据集，随机森林模型的均方误差为 0.0237，而 AdaBoost 模型的均方误差为 0.0064。在这种情况下，AdaBoost 模型在预测 Breast Cancer 数据集上表现更好，其均方误差较小，预测结果与实际值之间的差异较小。

综上所述，对于回归任务，不同的模型在不同数据集上可能会有不同的性能表现。在这种情况下，随机森林模型在 Iris 数据集上略优于 AdaBoost 模型，而 AdaBoost 模型在 Breast Cancer 数据集上表现更好。均方误差的较小值表明模型对于预测任务的准确性较高。然而，需要注意的是，这些结果仅限于使用给定数据集和设置下的性能比较，具体的结果可能会受到其他因素的影响，如数据质量、模型调参等。

4 结论与思考

在完成实验作业的过程中，我发现有以下几点需要注意的地方：

1. 数据预处理：在进行回归操作之前，需要对数据进行预处理，主要是需要将 wdbc 数据集中的 diagnosis 列转换为数值型数据，这也是本次实验首先需要完成的工作。在实验开始时我疏忽了这点，造成了代码错误。
2. 使用数据集：从UCI下载数据时，除了 data 格式的文件，还有 names 格式的说明文件，善用说明文件可以帮助我们理解数据集的特征、获得缺失值情况等，从而快速了解数据集；
3. 提高数学能力：本次实验中，实现 AdaBoost 模型是调用了 sklearn 库实现的，因此使用简洁的代码即可。但在查阅资料过程中，我发现此模型的实现是较为复杂的，涉及到较多的公式，我领悟到要深入研究本专业的知识离不开精深的数学知识。
4. 模型评估：在进行回归与分类实验之后，我发现对于 iris 数据集，在测试集比例为 0.2 时，预测准确率可以达到 100%, 有些“过拟合”，数据集较小是其中一个因素。因此，如果要在现实中进行数据分析，需要尽量扩大数据规模，从而获得更准确的分析结果。

5 致谢

本文的 LaTeX 模板来源于 ElegantPaper。

本文中部分知识引用自课程 数据挖掘应用实验中的课件。

参考文献

[1] R. A. Fisher. *Iris*. UCI Machine Learning Repository. DOI: 10.24432/C56C76. 1988.

[2] William Wolberg. *Breast Cancer Wisconsin (Diagnostic)*. UCI Machine Learning Repository. DOI: 10.24432/C5DW2B. 1995.

[3] Robert E. Schapire et al. “Boosting the margin: A new explanation for the effectiveness of voting methods”. In: *Proceedings of the 14th International Conference on Machine Learning*. Nashville, TN, 1997, pp. 322–330.