第四章 自适应推荐模型与实现

本章对系统中的自适应内容进行集中阐述。训练组学习者使用自适应规则进行词汇学习；测试组学习者使用自适应规则与自适应算法相结合的方式进行词汇学习。测试组自适应算法的确定结合特征工程以及模型训练与测试的准确度。在训练组学习者与测试组学习者进行词汇学习的过程中，本研究收集了学习数据，并在最后统计了所有教学实验词汇的词汇模型训练与测试的准确率。本章从自适应规则、特征工程、推荐类别、学习数据获取、算法模型方面分别进行阐述。

4.1 自适应规则

本文将从学习、练习、复习的方面对本研究所涉及自适应规则进行阐述。

第一，词汇学习阶段自适应。词汇学习阶段自适应包括制作难度逐渐提升的学习材料以及词汇学习入口方面的自适应。学习材料的制作见第五章学习材料部分，本节主要介绍词汇学习入口方面的自适应。

表 4.1 词汇学习阶段自适应规则

词汇初始掌握程度	学习入口	学习内容
我从没有见过这个词。	从字开始学习	字、词、搭配、句子、练习
我不知道这个词的意思。	从词开始学习	词、搭配、句子、练习
我知道这个词的意思，但是不知道它的搭配。	从搭配开始学习	搭配、句子、练习
我知道这个词的搭配，但是不知道如何在一个句子中使用这个词。	从句子开始学习	句子、练习
我可以在一个句子中使用这个词。	直接进行练习	练习

第二，词汇练习阶段自适应。如前文所述，每个目标词汇安排两个词汇搭配进行学习，词汇练习题包括词汇释义题、词汇搭配题、词汇填词题，相对应地，为每个目标词汇安排两个词汇搭配题。本研究将两个词汇搭配题看作一个整体，认为：（1）两个词汇搭配题只有全做对，才算词汇搭配题做对；（2）两个词汇搭配题中如果其中一个做错，除了做错的词汇搭配题给予反馈，即便另一个做对，也会给予知识点反馈。在词汇练习阶段与下文的复习阶段中的训练组均是如此处

理。下文中复习阶段中的测试组中正确的反馈也是如此处理，但是还要结合词汇自适应学习算法的推荐情况。

第三，词汇复习阶段自适应。在第二章提到本文认同词汇学习要 “重复 7 次及以上”才能掌握，因此，本研究假定学习者至少需要重复 7 次，才能掌握所学内容。基于此，本研究根据不同词汇掌握安排不同的复习次数。本研究复习阶段分为训练组与测试组。对于训练组学习者来说，按照下表进行词汇复习内容与次数的推荐。对于测试组学习者来说，正确的词汇复习内容与次数见下表，而实际的复习内容则根据由训练组学习数据进行训练得出的词汇模型推荐确定的。本研究还会计算对于测试集的模型推荐准确率。

表 4.2 训练组学习者词汇复习内容与次数

词汇初始掌握程度	复习内容和次数	备注
我从没有见过这个词。	搭配题和填词题各复习5次	如果在第一次学习后的练习，搭配题或者填词题做错了，那么搭配题或者填词题各复习6次。词汇释义题同样如此。 如果第一次练习或之后的复习时，题目做错，那么复习内容就添加词汇题目相应的学习资料。
我不知道这个词的意思。	搭配题和填词题各复习5次	同上
我知道这个词的意思，但是不知道它的搭配。	搭配题和填词题各复习4次	同上
我知道这个词的搭配，但是我不知道如何在一个句子中使用这个词。	搭配题和填词题各复习3次	同上
我可以在一个句子中使用这个词。	搭配题和填词题各复习2次	同上

4.2 特征工程

在第二章自适应学习理论部分提到，本研究的自适应学习推荐算法主要考虑学习者因素与学习对象因素。本研究学习者因素主要考虑词汇的整体认知水平，具体包括使用“学习者前测成绩”来反映学习者对于待学词汇的掌握准确度，使用“学习者前测所用时间”来反映学习者对于待学词汇的熟练程度。本研究学习对象因素包括学习者对于某个词汇的初始认知水平；学习者在练习和复习过程中

对于某个词汇在释义、搭配、填词方面的认知水平；本次实验中学习某个词汇的次数、本次实验中学习某个词汇所用的整体时间。

本研究为每个词汇[1]建立学习模型[2]。本研究基于词汇学习的流程，首先确认学习对象因素对于词汇建模是必要的，包括词汇学习模块中的某词汇初始掌握程度、词汇练习和复习模块中某词汇对应的三种词汇练习题的作答情况、本次实验中学习某个词汇的次数、本次实验中学习某个词汇所用的整体时间，基于此确定了词汇模型初始特征，如表 4.3。

表 4.3 词汇模型初始特征

特征分类	具体特征	备注
学习对象之“你知道这个词汇到什么程度”	A	代表“我从没有见过这个词”。如果是，则为1，否则为0。
	B	代表“我不知道这个词的意思”。如果是，则为1，否则为0。
	C	代表“我知道词这个词的意思，但是不知道它的搭配”。如果是，则为1，否则为0。
	D	代表“我知道这个词的搭配，但是我不知道如何在一个句子中使用这个词”。如果是，则为1，否则为0。
	E	代表“我可以在一个句子中使用这个词”
学习对象之学习记录	词汇释义题回答是否正确	如果第一次练习时，回答错误，标注为0，则进行六次复习。其他情况下，不进行实际复习，但是均标注为1，
	搭配题回答是否正确	回答正确，标注为1，否则标注为0。需要注意的是，只有目标词汇的两个词汇搭配题全部回答正确，才

[1] 教学实验涉及内容包含 65 个词汇。

[2] 使用三种算法建立不同的模型，并使用投票法进行模型融合，提升训练集的训练准确率。

		算回答正确。
学习对象之学习记录	填词题回答是否正确	回答正确，标注为1，否则标注为0。
	本次实验中学习某个词汇的次数	1-7次
	学习时间	统计学习某个词汇所用的整体时间，单位为分钟，保留小数点后一位。

本研究也在表 4.3 词汇模型初始特征的基础上，进行特征的进一步选取，以便通过算法训练与测试进行最优的特征选择。本节接下来主要探讨如何处理学习者因素特征与词汇掌握程度特征。学习者因素特征[3]包括学习者前测成绩、学习者前测所用时间。词汇掌握程度的计算方式为“假设一共要进行 7 次学习，（7-待学习的次数）/7”。

前文通过理论部分探讨了自适应学习推荐算法考虑学习者因素，本研究也通过实验，使用训练集完整数据以及测试集第一次复习[4]的数据，发现保留学习者因素特征，相对于不保留学习者因素特征，对于模型训练的准确度是起到正向作用的，因此，本研究保留学习者因素特征。

实验如下：

以教学实验词汇“发生”为例，按照不添加“学习者前测成绩”和“学习者前测所用时间”两个特征和添加“学习者前测成绩”和“学习者前测所用时间”两个特征，分别对训练集和测试集第一次复习数据进行不同算法的训练，程序分别见附录 Q 和附录 R。

附录 Q 和附录 R 主要步骤均如下：

（1）训练集数据预处理。包括训练集加载数据、训练集提取数据、训练集数据最小-最大规范化。

（2）测试集数据预处理。包括测试集加载数据、测试集提取数据、测试集数据最小-最大规范化。

（3）模型训练与预测。使用决策树[58]、基于决策树的 AdaBoost[58,59]、多项式朴素贝叶斯[60]、高斯朴素贝叶斯[61]、基于多项式朴素贝叶斯的 AdaBoost[59,60]、SVM[62]、基于 SVM 的 AdaBoost[59,62]、KNN[63]、基于 KNN 的 AdaBoost[59,63]、XGBoost[64]算法分别对训练集与测试集进行训练与预测。

[3] 由于多个学习者都会学习某词汇，因此词汇学习数据中会有多个学习者的学习数据，笔者在收集关于某词汇的学习数据时，使用了学习者编号对学习者进行区分。但是学习者编号对于词汇模型训练与测试并无意义，因此，学习者因素不考虑学习者编号。

[4] 测试集后续复习内容的确定需要根据算法来判断，在算法不能确定前，不能确定后续复习数据。通过实验证明，此处使用第一次复习的数据来进行判断，同样可以起到对比不同模型在测试集上准确度的效果。

通过表 4.4 可以看出，当添加学习者因素特征后，模型在训练集和测试集上的准确率大部分得到了提升。基于多项式朴素贝叶斯的 AdaBoost、高斯朴素贝叶斯、XGBoost 三种算法得出的模型准确度较好。

为了探索“掌握程度”的特征是否对准确率有作用，笔者在教学实验词汇“发生”的训练集和测试集第一次复习的数据上添加了“掌握程度”特征。如前文所述，假设词汇一共要进行 7 次学习，（7-待学习的次数）/7 即为词汇掌握程度。

由于前面通过特征选择的实验得出，学习者因素特征对模型的准确度起正向作用，因此，以添加了学习者因素特征的数据作为对照，在其基础上添加“掌握程度”特征，通过对比，来判断“掌握程度”特征的作用。在添加“掌握程度”特征后，分别对训练集和测试集第一次复习数据进行上述算法的训练与预测。程序整体思路与附录 R 大致相同，只是在训练集和测试集的数据选取时略有不同，因此便不再把程序添加到附录。

通过表 4.4 可以看到，添加“掌握程度”特征后，高斯朴素贝叶斯、XGBoost 两个算法的准确度略微有所提升，但是基于多项式朴素贝叶斯的 AdaBoost 算法在训练集上的准确率从 89.5%下降到了 73.7%，下降较为明显。而即便不添加“掌握程度”特征，高斯朴素贝叶斯、XGBoost 两个算法在训练集的准确率分别为 84.2%、94.7%，在测试集上的准确率分别为 100%、100%，准确度整体较好。因此，笔者在进行特征选择时，不添加“掌握程度”特征。

表 4.4 词汇模型特征选择准确率对比

	未添加学习者因素特征		添加学习者因素特征		添加掌握程度特征	
	训练集准确率	测试集准确率	训练集准确率	测试集准确率	训练集准确率	测试集准确率
决策树	100%	66.7%	100%	66.7%	100%	66.7%
多项式朴素贝叶斯	63.2%	100%	63.2%	100%	68.4%	100%
高斯朴素贝叶斯	73.7%	66.7%	84.2%	100%	89.5%	100%
SVM	68.4%	100%	63.2%	100%	63.2%	100%
KNN	68.4%	100%	63.2%	100%	63.2%	100%
基于决策树的 AdaBoost	100%	66.7%	100%	66.7%	100%	66.7%
基于多项	78.9%	100%	89.5%	100%	73.7%	100%

式朴素贝叶斯的AdaBoost						
基于SVM的AdaBoost	26.3%	33.3%	31.6%	33.3%	47.4%	100%
基于KNN的AdaBoost	47.4%	100%	47.4%	100%	47.4%	100%
XGBoost	89.5%	100%	94.7%	100%	100%	100%

表 4.5 词汇模型最终特征

特征分类	具体特征	备注
学习者因素	学习者前测成绩	0-100分
	学习者前测所用时间	0-70分钟
学习对象之“你知道这个词汇到什么程度”	A	代表“我从没有见过这个词”。如果是，则为1，否则为0。
	B	代表“我不知道这个词的意思”。如果是，则为1，否则为0。
	C	代表“我知道词这个词的意思，但是不知道它的搭配”。如果是，则为1，否则为0。
	D	代表“我知道这个词的搭配，但是我不知道如何在一个句子中使用这个词”。如果是，则为1，否则为0。
	E	代表“我可以在一个句子中使用这个词”
学习对象之学习记录	词汇释义题回答是否正确	如果第一次练习时，回答错误，标注为0，则进行六次复习。其他情况下，不进行实际复习，但是均标注为1，
	搭配题回答是否正确	回答正确，标注为1，否则标注为0。

	填词题回答是否正确	回答正确，标注为1，否则标注为0。
	本次实验中学习某个词汇的次数	1-7次
	学习时间	统计学习某个词汇所用的整体时间，单位为分钟，保留小数点后一位。

4.3 推荐类别

有可能的推荐类别如下表所示。

表 4.6 推荐类别

类别编号	类别具体内容[5]	备注
类别1	12	搭配题、填词题
类别2	12a	搭配题、填词题、搭配题学习资料
类别3	12b	搭配题、填词题、填词题学习资料
类别4	12ab	搭配题、填词题、搭配题学习资料、填词题学习资料
类别5	1	搭配题
类别6	1a	搭配题、搭配题学习资料
类别7	2	填词题
类别8	2b	填词题、填词题学习资料
类别9	0	无，也即停止这个词汇的学习。

[5] 1 代表搭配题，2 代表填词题，3 代表词汇释义题；a 代表搭配题对应的学习资料，b 代表填词题对应的学习资料，c 代表词汇释义题对应的学习资料。在实验进行中，主要推荐的类别为类别 1-类别 9。类别 10-类别 18 为在类别 1-类别 9 的基础上加了释义题。类别 19-类别 27 为在类别 10-18 的基础上加了释义题学习资料。是否推荐释义题或释义题学习资料，主要是看在第一次学习后的练习中是否有做对释义题的题目，如果做错了，则给予推荐，否则则不予推荐。而由于是即时练习，一般情况下都能做对。没有做对的情况，比如学习者在判断对于一个词的掌握程度时，认为自己掌握了词的意思，从搭配开始学习，也就是跳过了词义的学习资料，那么在这种情况下就可能做错，但是这种出错的情况极少。除了根据第一次练习对词汇释义题进行是否掌握的判断外，后面不再予以测试，而是将重点放到搭配题和填词题上。因为，搭配题和填词题的难度是大于词汇释义题的。如果搭配题和填词题都掌握的情况下，词汇释义题是一定掌握了的，这点会在实验后测中进行进一步验证。

类别10	123	搭配题、填词题、释义题
类别11	12a3	搭配题、填词题、搭配题学习资料、释义题
类别12	12b3	搭配题、填词题、填词题学习资料、释义题
类别13	12ab3	搭配题、填词题、搭配题学习资料、填词题学习资料、释义题
类别14	13	搭配题、释义题
类别15	1a3	搭配题、搭配题学习资料、释义题
类别16	23	填词题、释义题
类别17	2b3	填词题、填词题学习资料、释义题
类别18	3	释义题
类别19	123c	搭配题、填词题、释义题、释义题学习资料
类别20	12a3c	搭配题、填词题、搭配题学习资料、释义题、释义题学习资料
类别21	12b3c	搭配题、填词题、填词题学习资料、释义题、释义题学习资料
类别22	12ab3c	搭配题、填词题、搭配题学习资料、填词题学习资料、释义题、释义题学习资料
类别23	13c	搭配题、释义题、释义题学习资料
类别24	1a3c	搭配题、搭配题学习资料、释义题、释义题学习资料
类别25	23c	填词题、释义题、释义题学习资料
类别26	2b3c	填词题、填词题学习资料、释义题、释义题学习资料

类别27	3c	释义题、释义题学习资料

4.4 学习数据获取

在确定好特征以及推荐类别后，笔者首先按照 4.1 节中的自适应规则进行复习内容的推荐。笔者为每个词汇的学习都建立了一个工作表，由于有 65 个词汇，那么就有65个工作表。训练集学习者会对每个词汇进行1-6次不同次数的复习，直到六次复习全部完成，训练集 65 个工作表的数据收集完成。由于对于每个词汇来说，复习的次数是不同的，那么 65 个工作表的行数也是不同的。经过统计，训练集 65 个工作表一共含有 1085 行数据。每行数据有 14 列，那么一共有 15190 个数据。为保证数据准确获得[6]，笔者独立完成每次复习中的如下工作：出题、批改试卷、复习推荐、数据收集。

对于测试集来说，与测试集不同的是，会根据算法模型来推荐下一步学习内容。笔者对训练集每个词汇都建立单独的模型[7]，实验根据每个词汇的模型对测试集中每个词汇的学习进行复习推荐。在经过 1-6 次复习后，数据收集完成。为了保证数据准确性，训练集数据同样由笔者单独收集完成。相比于训练集，测试集数据的获取增加了训练集算法模型训练与测试的工作。经过统计，测试集 65 个工作表一共含有 1150 行数据。每行数据有 15 列，那么一共有 17250 个数据。

4.5 算法模型

在 4.2 节中，介绍了基于多项式朴素贝叶斯的 AdaBoost、高斯朴素贝叶斯、XGBoost 三种算法得出的模型的准确率较好，上述三种算法的一种会在应用于某个词汇的数据时表现不好，因此，笔者决定对三种算法进行模型融合，使用投票法来判定最后的训练与测试结果。笔者对含有三种算法的三个程序进行了稍微修改，以做到只需修改要查询的词汇以及词汇数据的行数，即可在用于不同词汇时进行程序的复用。使用了上述三种算法的程序代码见附录 S。三种算法模型融合后的推荐正确率见表 4.7。每个词汇的行数、推荐正确行数、推荐正确率等信息见附录 T。

表 4.7 模型融合后的推荐正确率

标题	行数/比率
训练集词汇全部行数	1085

[6] 由于给实验学习者解释数据收集方式，会花费大量时间，而且会占用学习者的时间，因此笔者决定独立进行收集。

[7] 具体见 4.5 算法模型一节。

训练集词汇推荐正确行数	999
训练集词汇推荐正确率	92.07%
测试集词汇被算法推荐的全部行数	890
测试集词汇被算法推荐正确的行数	629
测试集词汇被算法推荐正确的比率	70.67%
测试集词汇被实际推荐正确的行数	750
测试集词汇被实际推荐正确的比率	84.27%

附录 Q 不添加学习者因素特征进行词汇模型训练与预测

```
    # 训练集数据预处理
    # 训练集加载数据
    import pandas as pd

    df = pd.read_excel(
        '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(训练集-用于加工).xlsx',
        header=None)
    df

    # 训练集提取数据
    fasheng_train = df.iloc[2:21, 4:15]
    fasheng_train

    fasheng_train = fasheng_train.reset_index(drop=True)

    fasheng_train.iloc[:, 10] = fasheng_train.iloc[:, 10].replace({
        1: 5,
        12: 1,
        '12a': 2,
        '2b': 8,
    })

    fasheng_train

    # 训练集数据规范化
    fasheng_train_x = fasheng_train.iloc[:, 0:10]
    fasheng_train_x

    fasheng_train_y = fasheng_train.iloc[:, 10]
    fasheng_train_y
    from sklearn.preprocessing import MinMaxScaler
```

```
    ss = MinMaxScaler()
    fasheng_train_x = ss.fit_transform(fasheng_train_x)
    fasheng_train_x

    # 测试集数据预处理
    # 测试集加载数据
    fasheng_test = pd.read_excel(
        '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(测试集-用于加工).xlsx',
        header=None)
    fasheng_test

    # 测试集提取数据
    fasheng_test_x = fasheng_test.iloc[6:9, 4:14]
    fasheng_test_x

    fasheng_test_y = fasheng_test.iloc[6:9, 15]
    fasheng_test_y

    fasheng_test_y = fasheng_test_y.replace({12: 1})
    fasheng_test_y

    # 测试集数据规范化
    from sklearn.preprocessing import MinMaxScaler

    fasheng_test_x = ss.transform(fasheng_test_x)
    fasheng_test_x

    # 模型训练与预测
    # 决策树
    # 决策树训练
    from sklearn.tree import DecisionTreeClassifier
    # 构造 ID3 决策树
```

```
    clf = DecisionTreeClassifier(criterion='entropy')
    # 决策树训练
    clf.fit(fasheng_train_x, fasheng_train_y)
    # 决策树预测
    fasheng_train_x_predict_label = clf.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
    # 得到训练集决策树准确率
    acc_decision_tree = round(clf.score(fasheng_train_x,
fasheng_train_y), 6)
    print(u'score 准确率为 %.4lf' % acc_decision_tree)

    # 决策树测试集预测
    fasheng_test_x_predict_label = clf.predict(fasheng_test_x)
    print(fasheng_test_x_predict_label)

    # 得到测试集决策树准确率
    acc_decision_tree = round(clf.score(fasheng_test_x,
fasheng_test_y), 6)
    print(u'score 准确率为 %.4lf' % acc_decision_tree)

    # 基于决策树的 AdaBoost
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import AdaBoostClassifier

    clf = DecisionTreeClassifier(criterion='entropy')
    # AdaBoost 分类器
    ada = AdaBoostClassifier(base_estimator=clf, n_estimators=200)
    ada.fit(fasheng_train_x, fasheng_train_y)
    fasheng_train_x_predict_label = ada.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)

    from sklearn import metrics

    print(metrics.accuracy_score(fasheng_train_y,
```

```
fasheng_train_x_predict_label))

    # 测试集预测

    prediction = ada.predict(fasheng_test_x)
    print(prediction)

    # 测试集预测准确率
    print('准确率:', metrics.accuracy_score(fasheng_test_y,
prediction))

    # 多项式朴素贝叶斯
    # 多项式贝叶斯分类器
    from sklearn.naive_bayes import MultinomialNB

    clf = MultinomialNB(alpha=0.001).fit(fasheng_train_x,
fasheng_train_y)
    # 多项式朴素贝叶斯预测
    fasheng_train_x_predict_label = clf.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
    # 得到训练集朴素贝叶斯准确率
    acc_bayes_train = round(clf.score(fasheng_train_x,
fasheng_train_y), 6)
    print(u'score准确率为 %.4lf' % acc_bayes_train)

    # 多项式朴素贝叶斯预测
    fasheng_test_x_predict_label = clf.predict(fasheng_test_x)
    print(fasheng_test_x_predict_label)

    # 得到测试集决策树准确率
    acc_bayes_train = round(clf.score(fasheng_test_x, fasheng_test_y),
6)
    print(u'score准确率为 %.4lf' % acc_bayes_train)
```

```
    # 高斯朴素贝叶斯
    # 高斯贝叶斯分类器
    from sklearn.naive_bayes import GaussianNB

    clf = GaussianNB().fit(fasheng_train_x, fasheng_train_y)
    # 朴素贝叶斯预测
    fasheng_train_x_predict_label = clf.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
    # 得到训练集朴素贝叶斯准确率
    acc_bayes_train = round(clf.score(fasheng_train_x,
fasheng_train_y), 6)
    print(u'score 准确率为 %.4lf' % acc_bayes_train)

    # 高斯朴素贝叶斯预测
    fasheng_test_x_predict_label = clf.predict(fasheng_test_x)
    print(fasheng_test_x_predict_label)

    # 得到测试集决策树准确率
    acc_bayes_train = round(clf.score(fasheng_test_x, fasheng_test_y),
6)
    print(u'score 准确率为 %.4lf' % acc_bayes_train)

    # 基于多项式朴素贝叶斯的 AdaBoost
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.ensemble import AdaBoostClassifier

    clf = MultinomialNB(alpha=0.001)
    # AdaBoost 分类器
    ada = AdaBoostClassifier(base_estimator=clf, n_estimators=200)
    ada.fit(fasheng_train_x, fasheng_train_y)
    fasheng_train_x_predict_label = ada.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)

    from sklearn import metrics
```

```
    print(metrics.accuracy_score(fasheng_train_y,
fasheng_train_x_predict_label))

    # 测试集预测

    prediction = ada.predict(fasheng_test_x)
    print(prediction)

    # 测试集预测准确率
    print('准 确 率 : ', metrics.accuracy_score(fasheng_test_y,
prediction))

    # SVM
    from sklearn import metrics
    from sklearn import svm
    # 创建SVM分类器
    model = svm.SVC()
    # 用训练集做训练
    model.fit(fasheng_train_x, fasheng_train_y)
    # 训练集预测
    fasheng_train_x_predict_label = model.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
    # 训练集预测准确率
    print('准确率: ',
          metrics.accuracy_score(fasheng_train_y,
fasheng_train_x_predict_label))

    # 用测试集做预测
    prediction = model.predict(fasheng_test_x)
    print(prediction)

    # 测试集预测准确率
    print('准 确 率 : ', metrics.accuracy_score(fasheng_test_y,
```

```
prediction))

    # 基于SVM的AdaBoost
    # 基于SVM的AdaBoost分类器
    from sklearn import svm
    from sklearn.ensemble import AdaBoostClassifier

    clf = svm.SVC()
    # AdaBoost分类器
    ada = AdaBoostClassifier(base_estimator=clf,
                             n_estimators=200,
                             algorithm='SAMME')
    ada.fit(fasheng_train_x, fasheng_train_y)
    fasheng_train_x_predict_label = ada.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)

    from sklearn import metrics

    print(metrics.accuracy_score(fasheng_train_y,
fasheng_train_x_predict_label))

    # 测试集预测

    prediction = ada.predict(fasheng_test_x)
    print(prediction)

    # 测试集预测准确率
    print('准 确 率 : ', metrics.accuracy_score(fasheng_test_y,
prediction))

    # KNN
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn import metrics
```

```
    # 创建 KNN 分类器
    knn = KNeighborsClassifier()
    # 训练
    knn.fit(fasheng_train_x, fasheng_train_y)
    # 训练集预测
    fasheng_train_x_predict_label = knn.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
    # 训练集预测准确率
    print('准确率: ',
          metrics.accuracy_score(fasheng_train_y,
fasheng_train_x_predict_label))

    # 测试集预测
    predict_y = knn.predict(fasheng_test_x)
    print(predict_y)
    print("KNN 准确率: %.4lf" % metrics.accuracy_score(fasheng_test_y,
predict_y))

    # 基于 KNN 的 AdaBoost
    # 基于 KNN 的 AdaBoost 分类器
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.ensemble import AdaBoostClassifier

    clf = svm.SVC()
    # AdaBoost 分类器
    ada = AdaBoostClassifier(base_estimator=clf,
                             n_estimators=50,
                             algorithm='SAMME')
    ada.fit(fasheng_train_x, fasheng_train_y)
    fasheng_train_x_predict_label = ada.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)

    from sklearn import metrics
```

```
    print(metrics.accuracy_score(fasheng_train_y,
fasheng_train_x_predict_label))

    # 测试集预测

    prediction = ada.predict(fasheng_test_x)
    print(prediction)

    # 测试集预测准确率
    print('准 确 率 : ', metrics.accuracy_score(fasheng_test_y,
prediction))

    # XGBoost
    # Excel 文件中工作表的名字
    search_word = "发生"
    # 训练集工作表行数
    train_rows = 21

    import pandas as pd
    from sklearn.preprocessing import MinMaxScaler

    # 读取训练集工作表内容
    word_train = pd.read_excel(
        '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the
learning record of words(训练集-用于加工).xlsx',
        header=None,
        sheet_name=search_word)
    word_train

    # 训练集 X
    word_train_x = word_train.iloc[2:train_rows, 4:14]

    # 训练集 X 数据规范化
    ss = MinMaxScaler()
```

```
    word_train_x = ss.fit_transform(word_train_x)
    word_train_x

    # 训练集 y
    word_train_y = word_train.iloc[2:train_rows, 14].replace({
        0: 0,
        12: 1,
        '12a': 2,
        1: 3,
        '2b': 4
    })
    word_train_y

    # 读取测试集工作表内容
    word_test = pd.read_excel(
        '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(测试集-用于加工).xlsx',
        header=None,
        sheet_name=search_word)

    word_test

    # 测试集 X
    word_test_x = word_test.iloc[6:9, 4:14]
    word_test_x

    # 测试集 X 数据规范化
    word_test_x = ss.transform(word_test_x)
    word_test_x

    # 测试集 y
    word_test_y = word_test.iloc[6:9, 15].replace({
        0: 0,
        12: 1,
```

```
        '12a': 2,
        1: 3,
        '2b': 4,
        '12b': 5
    })
    # word_test_y

    import numpy as np
    from sklearn.metrics import zero_one_loss
    from sklearn import metrics

    # XGBoost 算法
    from XGBoost import XGBClassifier

    clf = XGBClassifier()

    clf.fit(word_train_x, word_train_y)
    word_train_x_predict_label = clf.predict(word_train_x)
    print("训练集预测：", word_train_x_predict_label)

    print("训练集准确率：",
          metrics.accuracy_score(word_train_y,
word_train_x_predict_label))

    # 测试集预测

    prediction = clf.predict(word_test_x)
    print("测试集预测：", prediction)

    # 测试集预测准确率
    print('测试集准确率：', metrics.accuracy_score(word_test_y,
prediction))
```

附录 R 添加学习者因素特征进行词汇模型训练与预测

```
# 训练集
# 重新加载数据
import pandas as pd

df = pd.read_excel(
    '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(训练集-用于加工).xlsx',
    header=None)

df

df_fasheng = df.iloc[2:21, 2:15]
df_fasheng = df_fasheng.reset_index(drop=True)
df_fasheng

# 提取数据
df_fasheng.iloc[:, 12] = df_fasheng.iloc[:, 12].replace({
    1: 5,
    12: 1,
    '12a': 2,
    '2b': 8,
})
df_fasheng

# 数据规范化
fasheng_train_x = df_fasheng.iloc[0:19, 0:12]
fasheng_train_x

fasheng_train_y = df_fasheng.iloc[:, 12]
fasheng_train_y

# 用于贝叶斯算法的数据预处理
```

```
from sklearn.preprocessing import MinMaxScaler

ss = MinMaxScaler()
fasheng_train_x = ss.fit_transform(fasheng_train_x)
fasheng_train_x

fasheng_train_y = df_fasheng.iloc[:, 12]
fasheng_train_y

# 测试集
# 加载数据
fasheng_test = pd.read_excel(
    '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(测试集-用于加工).xlsx',
    header=None)
fasheng_test

# 提取数据
fasheng_test_x = fasheng_test.iloc[6:9, 2:14]
fasheng_test_x

fasheng_test_y = fasheng_test.iloc[6:9, 15]
fasheng_test_y

fasheng_test_y = fasheng_test_y.replace({12: 1})
fasheng_test_y

# 数据规范化
from sklearn.preprocessing import MinMaxScaler

fasheng_test_x = ss.transform(fasheng_test_x)
fasheng_test_x

# 模型训练
```

```
    # 决策树
    from sklearn.tree import DecisionTreeClassifier
    # 构造 ID3 决策树
    clf = DecisionTreeClassifier(criterion='entropy')
    # 决策树训练
    clf.fit(fasheng_train_x, fasheng_train_y)
    # 决策树预测
    fasheng_train_x_predict_label = clf.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
    # 得到决策树准确率
    acc_decision_tree = round(clf.score(fasheng_train_x,
fasheng_train_y), 6)
    print(u'score 准确率为 %.4lf' % acc_decision_tree)

    # 决策树测试集预测
    fasheng_test_x_predict_label = clf.predict(fasheng_test_x)
    print(fasheng_test_x_predict_label)

    # 得到测试集决策树准确率
    acc_decision_tree = round(clf.score(fasheng_test_x,
fasheng_test_y), 6)
    print(u'score 准确率为 %.4lf' % acc_decision_tree)

    # 基于决策树的 AdaBoost
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import AdaBoostClassifier

    clf = DecisionTreeClassifier(criterion='entropy')
    # AdaBoost 分类器
    ada = AdaBoostClassifier(base_estimator=clf, n_estimators=200)
    ada.fit(fasheng_train_x, fasheng_train_y)
    fasheng_train_x_predict_label = ada.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
```

```
    from sklearn import metrics

    print(metrics.accuracy_score(fasheng_train_y,
fasheng_train_x_predict_label))

    # 测试集预测

    prediction = ada.predict(fasheng_test_x)
    print(prediction)

    # 测试集预测准确率
    print('准确率：', metrics.accuracy_score(fasheng_test_y,
prediction))

    # 多项式朴素贝叶斯
    # 多项式贝叶斯分类器
    from sklearn.naive_bayes import MultinomialNB

    clf = MultinomialNB(alpha=0.001).fit(fasheng_train_x,
fasheng_train_y)
    # 多项式朴素贝叶斯预测
    fasheng_train_x_predict_label = clf.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
    # 得到训练集朴素贝叶斯准确率
    acc_bayes_train = round(clf.score(fasheng_train_x,
fasheng_train_y), 6)
    print(u'score准确率为 %.4lf' % acc_bayes_train)
    # 多项式朴素贝叶斯预测
    fasheng_test_x_predict_label = clf.predict(fasheng_test_x)
    print(fasheng_test_x_predict_label)

    # 得到测试集决策树准确率
    acc_bayes_train = round(clf.score(fasheng_test_x, fasheng_test_y),
6)
```

```
    print(u'score 准确率为 %.4lf' % acc_bayes_train)

    # 高斯朴素贝叶斯
    # 高斯贝叶斯分类器
    from sklearn.naive_bayes import GaussianNB

    clf = GaussianNB().fit(fasheng_train_x, fasheng_train_y)
    # 朴素贝叶斯预测
    fasheng_train_x_predict_label = clf.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
    # 得到训练集朴素贝叶斯准确率
    acc_bayes_train = round(clf.score(fasheng_train_x,
fasheng_train_y), 6)
    print(u'score 准确率为 %.4lf' % acc_bayes_train)

    # 高斯朴素贝叶斯预测
    fasheng_test_x_predict_label = clf.predict(fasheng_test_x)
    print(fasheng_test_x_predict_label)

    # 得到测试集决策树准确率
    acc_bayes_train = round(clf.score(fasheng_test_x, fasheng_test_y),
6)
    print(u'score 准确率为 %.4lf' % acc_bayes_train)

    # 基于多项式朴素贝叶斯的 AdaBoost
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.ensemble import AdaBoostClassifier

    clf = MultinomialNB(alpha=0.001)
    # AdaBoost 分类器
    ada = AdaBoostClassifier(base_estimator=clf, n_estimators=200)
    ada.fit(fasheng_train_x, fasheng_train_y)
    fasheng_train_x_predict_label = ada.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
```

```
    from sklearn import metrics

    print(metrics.accuracy_score(fasheng_train_y,
fasheng_train_x_predict_label))

    # 测试集预测

    prediction = ada.predict(fasheng_test_x)
    print(prediction)

    # 测试集预测准确率
    print('准 确 率 : ', metrics.accuracy_score(fasheng_test_y,
prediction))

    # SVM
    from sklearn import metrics
    from sklearn import svm
    # 创建SVM分类器
    model = svm.SVC()
    # 用训练集做训练
    model.fit(fasheng_train_x, fasheng_train_y)
    # 训练集预测
    fasheng_train_x_predict_label = model.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
    # 训练集预测准确率
    print('准确率: ',
          metrics.accuracy_score(fasheng_train_y,
fasheng_train_x_predict_label))

    # 用测试集做预测
    prediction = model.predict(fasheng_test_x)
    print(prediction)
```

```
    # 测试集预测准确率
    print('准 确 率 : ', metrics.accuracy_score(fasheng_test_y,
prediction))

    # 基于 SVM 的 AdaBoost
    # 基于 SVM 的 AdaBoost 分类器
    from sklearn import svm
    from sklearn.ensemble import AdaBoostClassifier

    clf = svm.SVC()
    # AdaBoost 分类器
    ada = AdaBoostClassifier(base_estimator=clf,
                             n_estimators=200,
                             algorithm='SAMME')
    ada.fit(fasheng_train_x, fasheng_train_y)
    fasheng_train_x_predict_label = ada.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)

    from sklearn import metrics

    print(metrics.accuracy_score(fasheng_train_y,
fasheng_train_x_predict_label))

    # 测试集预测

    prediction = ada.predict(fasheng_test_x)
    print(prediction)

    # 测试集预测准确率
    print('准 确 率 : ', metrics.accuracy_score(fasheng_test_y,
prediction))

    # KNN
    from sklearn.neighbors import KNeighborsClassifier
```

```
    from sklearn import metrics

    # 创建KNN分类器
    knn = KNeighborsClassifier()
    # 训练
    knn.fit(fasheng_train_x, fasheng_train_y)
    # 训练集预测
    fasheng_train_x_predict_label = knn.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
    # 训练集预测准确率
    print('准确率: ',
          metrics.accuracy_score(fasheng_train_y,
fasheng_train_x_predict_label))

    # 测试集预测
    predict_y = knn.predict(fasheng_test_x)
    print(predict_y)
    print("KNN准确率: %.4lf" % metrics.accuracy_score(fasheng_test_y,
predict_y))

    # 基于KNN的AdaBoost
    # 基于KNN的AdaBoost分类器
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.ensemble import AdaBoostClassifier

    clf = svm.SVC()
    # AdaBoost分类器
    ada = AdaBoostClassifier(base_estimator=clf,
                             n_estimators=50,
                             algorithm='SAMME')
    ada.fit(fasheng_train_x, fasheng_train_y)
    fasheng_train_x_predict_label = ada.predict(fasheng_train_x)
    print(fasheng_train_x_predict_label)
```

```
    from sklearn import metrics

    print(metrics.accuracy_score(fasheng_train_y,
fasheng_train_x_predict_label))

    # 测试集预测

    prediction = ada.predict(fasheng_test_x)
    print(prediction)

    # 测试集预测准确率
    print('准 确 率 : ', metrics.accuracy_score(fasheng_test_y,
prediction))

    # XGBoost
    # Excel 文件中工作表的名字
    search_word = "发生"
    # 训练集工作表行数
    train_rows = 21

    import pandas as pd
    from sklearn.preprocessing import MinMaxScaler

    # 读取训练集工作表内容
    word_train = pd.read_excel(
        '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the
learning record of words(训练集-用于加工).xlsx',
        header=None,
        sheet_name=search_word)
    word_train

    # 训练集 X
    word_train_x = word_train.iloc[2:train_rows, 2:14]
```

```
    # 训练集 X 数据规范化
    ss = MinMaxScaler()
    word_train_x = ss.fit_transform(word_train_x)
    word_train_x

    # 训练集 y
    word_train_y = word_train.iloc[2:train_rows, 14].replace({
        0: 0,
        12: 1,
        '12a': 2,
        1: 3,
        '2b': 4
    })
    word_train_y

    # 读取测试集工作表内容
    word_test = pd.read_excel(
        '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(测试集-用于加工).xlsx',
        header=None,
        sheet_name=search_word)

    word_test

    # 测试集 X
    word_test_x = word_test.iloc[6:9, 2:14]
    word_test_x

    # 测试集 X 数据规范化
    word_test_x = ss.transform(word_test_x)
    word_test_x

    # 测试集 y
    word_test_y = word_test.iloc[6:9, 15].replace({
```

```
        0: 0,
        12: 1,
        '12a': 2,
        1: 3,
        '2b': 4,
        '12b': 5
    })
    # word_test_y

    import numpy as np
    from sklearn.metrics import zero_one_loss
    from sklearn import metrics

    # XGBoost 算法
    from XGBoost import XGBClassifier

    clf = XGBClassifier()

    clf.fit(word_train_x, word_train_y)
    word_train_x_predict_label = clf.predict(word_train_x)
    print("训练集预测：", word_train_x_predict_label)

    print("训练集准确率：",
          metrics.accuracy_score(word_train_y,
word_train_x_predict_label))

    # 测试集预测

    prediction = clf.predict(word_test_x)
    print("测试集预测：", prediction)

    # 测试集预测准确率
    print('测试集准确率：', metrics.accuracy_score(word_test_y,
prediction))
```

附录 S 基于多项式朴素贝叶斯的 AdaBoost、高斯朴素贝叶斯、XGBoost

```
    # 基于多项式朴素贝叶斯的 AdaBoost
    # Excel 文件中工作表的名字
    search_word = "发生"
    # 训练集工作表行数
    train_rows = 21

    import pandas as pd
    from sklearn.preprocessing import MinMaxScaler

    # 读取训练集工作表内容
    word_train = pd.read_excel(
        '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(训练集-用于加工).xlsx',
        header=None,
        sheet_name=search_word)
    word_train

    # 训练集 X
    word_train_x = word_train.iloc[2:train_rows, 2:14]

    # 训练集 X 数据规范化
    ss = MinMaxScaler()
    word_train_x = ss.fit_transform(word_train_x)
    word_train_x

    # 训练集 y
    word_train_y = word_train.iloc[2:train_rows, 14].replace({
        12: 1,
        '12a': 2,
        '12b': 3,
```

```
        '12ab': 4,
        1: 5,
        '1a': 6,
        2: 7,
        '2b': 8,
        0: 0
    })
    word_train_y

    # 读取测试集工作表内容
    word_test = pd.read_excel(
        '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(测试集-用于加工).xlsx',
        header=None,
        sheet_name=search_word)

    word_test

    # 测试集 X
    word_test_x = word_test.iloc[6:10, 2:14]
    word_test_x

    # 测试集 X 数据规范化
    word_test_x = ss.transform(word_test_x)
    word_test_x

    # 测试集 y
    word_test_y = word_test.iloc[6:10, 15].replace({
        12: 1,
        '12a': 2,
        '12b': 3,
        '12ab': 4,
        1: 5,
        '1a': 6,
```

```
        2: 7,
        '2b': 8,
        0: 0
    })
    # word_test_y

    word_train_x
    word_train_y

    # 模型训练
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.ensemble import AdaBoostClassifier
    import matplotlib.pyplot as plt
    import numpy as np
    from sklearn.metrics import zero_one_loss
    from sklearn import metrics

    # 设置 AdaBoost 迭代次数
    n_estimators = 200

    clf = MultinomialNB(alpha=0.001)
    # AdaBoost 分类器
    ada = AdaBoostClassifier(base_estimator=clf,
n_estimators=n_estimators)
    ada.fit(word_train_x, word_train_y)

    # 调节图像大小、清晰度
    plt.figure(figsize=(10, 8), dpi=150)

    # 设置 plt 正确显示中文
    plt.rcParams['font.sans-serif'] = ['SimHei']

    ada_err = np.zeros((n_estimators, ))
    # 遍历每次迭代的结果 i 为迭代次数, pred_y 为预测结果
```

```
    for i, pred_y in enumerate(ada.staged_predict(word_train_x)):
        # 统计错误率
        ada_err[i] = zero_one_loss(pred_y, word_train_y)
    # 绘制每次迭代的 AdaBoost 错误率
    plt.plot(np.arange(n_estimators) + 1,
             ada_err,
             label='AdaBoost 错误率',
             color='orange')
    plt.xlabel('迭代次数')
    plt.ylabel('错误率')
    leg = plt.legend(loc='upper right', fancybox=True)
    print('AdaBoost 分类器迭代次数与错误率：')
    plt.show()

    # 朴素贝叶斯与 AdaBoost 分类器
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.ensemble import AdaBoostClassifier
    from sklearn import metrics

    clf = MultinomialNB(alpha=0.001)
    # AdaBoost 分类器
    ada = AdaBoostClassifier(base_estimator=clf, n_estimators=75)
    ada.fit(word_train_x, word_train_y)
    word_train_x_predict_label = ada.predict(word_train_x)
    print("训练集预测：", word_train_x_predict_label)

    print("训练集准确率：",
          metrics.accuracy_score(word_train_y,
word_train_x_predict_label))

    # 测试集预测

    prediction = ada.predict(word_test_x)
    print("测试集预测：", prediction)
```

```
    # 测试集预测准确率
    print('测试集准确率：', metrics.accuracy_score(word_test_y, prediction))


    # 高斯朴素贝叶斯
    # Excel 文件中工作表的名字
    search_word = "发生"
    # 训练集工作表行数
    train_rows = 21

    import pandas as pd
    from sklearn.preprocessing import MinMaxScaler, StandardScaler

    # 读取训练集工作表内容
    word_train = pd.read_excel(
        '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(训练集-用于加工).xlsx',
        header=None,
        sheet_name=search_word)
    word_train

    # 训练集 X
    word_train_x = word_train.iloc[2:train_rows, 2:14]

    # 训练集 X 数据规范化
    # ss = MinMaxScaler()
    ss = StandardScaler()
    word_train_x = ss.fit_transform(word_train_x)
    word_train_x

    # 训练集 y
    word_train_y = word_train.iloc[2:train_rows, 14].replace({
```

```
        12: 1,
        '12a': 2,
        '12b': 3,
        '12ab': 4,
        1: 5,
        '1a': 6,
        2: 7,
        '2b': 8,
        0: 0
    })
    word_train_y

    # 读取测试集工作表内容
    word_test = pd.read_excel(
        '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(测试集-用于加工).xlsx',
        header=None,
        sheet_name=search_word)

    word_test

    # 测试集 X
    word_test_x = word_test.iloc[6:10, 2:14]
    word_test_x

    # 测试集 X 数据规范化
    word_test_x = ss.transform(word_test_x)
    word_test_x

    # 测试集 y
    word_test_y = word_test.iloc[6:10, 15].replace({
        12: 1,
        '12a': 2,
        '12b': 3,
```

```
        '12ab': 4,
        1: 5,
        '1a': 6,
        2: 7,
        '2b': 8,
        0: 0
    })
    # word_test_y

    # 高斯朴素贝叶斯
    from sklearn.naive_bayes import GaussianNB
    from sklearn import metrics

    clf = GaussianNB()
    clf.fit(word_train_x, word_train_y)
    word_train_x_predict_label = clf.predict(word_train_x)
    print("训练集预测：", word_train_x_predict_label)

    print("训练集准确率：",
          metrics.accuracy_score(word_train_y,
word_train_x_predict_label))

    # 测试集预测

    prediction = clf.predict(word_test_x)
    print("测试集预测：", prediction)

    # 测试集预测准确率
    print('测试集准确率：', metrics.accuracy_score(word_test_y,
prediction))



    # XGBoost
    # Excel文件中工作表的名字
```

```
search_word = "发生"
# 训练集工作表行数
train_rows = 21

import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# 读取训练集工作表内容
word_train = pd.read_excel(
    '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(训练集-用于加工).xlsx',
    header=None,
    sheet_name=search_word)
word_train

# 训练集X
word_train_x = word_train.iloc[2:train_rows, 2:14]

# 训练集X数据规范化
ss = MinMaxScaler()
word_train_x = ss.fit_transform(word_train_x)
word_train_x

# 训练集y
word_train_y = word_train.iloc[2:train_rows, 14].replace({
    0: 0,
    12: 1,
    '12a': 2,
    1: 3,
    '2b': 4
})
word_train_y

# 读取测试集工作表内容
```

```
    word_test = pd.read_excel(
        '/Users/kangzhengwei/Desktop/学习资料/02 实验组学习/the learning record of words(测试集-用于加工).xlsx',
        header=None,
        sheet_name=search_word)

    word_test

    # 测试集X
    word_test_x = word_test.iloc[6:10, 2:14]
    word_test_x

    # 测试集X数据规范化
    word_test_x = ss.transform(word_test_x)
    word_test_x

    # 测试集y
    word_test_y = word_test.iloc[6:10, 15].replace({
        0: 0,
        12: 1,
        '12a': 2,
        1: 3,
        '2b': 4,
        '12b': 5
    })
    # word_test_y

    import numpy as np
    from sklearn.metrics import zero_one_loss
    from sklearn import metrics

    # XGBoost算法
    from XGBoost import XGBClassifier
```

```
    clf = XGBClassifier()

    clf.fit(word_train_x, word_train_y)
    word_train_x_predict_label = clf.predict(word_train_x)
    print("训练集预测：", word_train_x_predict_label)

    print("训练集准确率：",
          metrics.accuracy_score(word_train_y,
word_train_x_predict_label))

    # 测试集预测

    prediction = clf.predict(word_test_x)
    print("测试集预测：", prediction)

    # 测试集预测准确率
    print('测试集准确率：', metrics.accuracy_score(word_test_y,
prediction))
```