## Выполнил: Лю Бовэнь

## Группа: ИУ5И-24М

## GradientBoostingClassifier

## LogisticRegression

```
[4]  #  导入必要的库
     import  numpy  as  np
     import  pandas  as  pd
     from  sklearn.model_selection  import  train_test_split
     from  sklearn.feature_extraction.text  import  CountVectorizer,  TfidfVectorizer
     from  sklearn.ensemble  import  GradientBoostingClassifier
     from  sklearn.linear_model  import  LogisticRegression
     from  sklearn.metrics  import  accuracy_score,  classification_report
```

```
[5]  #  加载数据集
     file_path  =  '/content/StudentsPerformance.csv'     #  在Google  Colab中运行时需要上传文件
     data  =  pd.read_csv(file_path)
```

```
[6]  #  选择文本特征和目标标签
     data['parental  level  of  education']  =  data['parental  level  of  education'].astype(str)
     X  =  data['parental  level  of  education']
     y  =  data['gender']     #  例如，以性别作为目标标签进行分类
```

```
[7]  #  将数据分为训练集和测试集
     X_train,  X_test,  y_train,  y_test  =  train_test_split(X,  y,  test_size=0.2,  random_state=42)
```

```
[8]  #  定义函数来训练和评估模型
     def  evaluate_model(vectorizer,  classifier,  X_train,  X_test,  y_train,  y_test):
             X_train_vec  =  vectorizer.fit_transform(X_train)
             X_test_vec  =  vectorizer.transform(X_test)

             classifier.fit(X_train_vec,  y_train)
             y_pred  =  classifier.predict(X_test_vec)

             accuracy  =  accuracy_score(y_test,  y_pred)
             report  =  classification_report(y_test,  y_pred)

             return  accuracy,  report
```

[9]
```
# 使用CountVectorizer进行向量化
count_vectorizer = CountVectorizer()
```

[10]
```
# 使用TfidfVectorizer进行向量化
tfidf_vectorizer = TfidfVectorizer()
```

[11]
```
# 定义分类器 GradientBoostingClassifier
gb_classifier = GradientBoostingClassifier(random_state=42)
```

[12]
```
# 定义分类器 LogisticRegression
lr_classifier = LogisticRegression(max_iter=1000, random_state=42)
```

```
# 评估模型
print("CountVectorizer + GradientBoostingClassifier")
count_gb_accuracy, count_gb_report = evaluate_model(count_vectorizer, gb_classifier, X_train, X_test, y_train, y_test)
print(f"Accuracy: {count_gb_accuracy}\n")
print(count_gb_report)

print("CountVectorizer + LogisticRegression")
count_lr_accuracy, count_lr_report = evaluate_model(count_vectorizer, lr_classifier, X_train, X_test, y_train, y_test)
print(f"Accuracy: {count_lr_accuracy}\n")
print(count_lr_report)

print("TfidfVectorizer + GradientBoostingClassifier")
tfidf_gb_accuracy, tfidf_gb_report = evaluate_model(tfidf_vectorizer, gb_classifier, X_train, X_test, y_train, y_test)
print(f"Accuracy: {tfidf_gb_accuracy}\n")
print(tfidf_gb_report)

print("TfidfVectorizer + LogisticRegression")
tfidf_lr_accuracy, tfidf_lr_report = evaluate_model(tfidf_vectorizer, lr_classifier, X_train, X_test, y_train, y_test)
print(f"Accuracy: {tfidf_lr_accuracy}\n")
print(tfidf_lr_report)
```

CountVectorizer + GradientBoostingClassifier
Accuracy: 0.49

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| female | 0.48 | 0.81 | 0.61 | 97 |
| male | 0.51 | 0.18 | 0.27 | 103 |
| accuracy | | | 0.49 | 200 |
| macro avg | 0.50 | 0.50 | 0.44 | 200 |
| weighted avg | 0.50 | 0.49 | 0.43 | 200 |

CountVectorizer + LogisticRegression
Accuracy: 0.49

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| female | 0.48 | 0.81 | 0.61 | 97 |
| male | 0.51 | 0.18 | 0.27 | 103 |
| accuracy | | | 0.49 | 200 |
| macro avg | 0.50 | 0.50 | 0.44 | 200 |
| weighted avg | 0.50 | 0.49 | 0.43 | 200 |

TfidfVectorizer + GradientBoostingClassifier
Accuracy: 0.49

TfidfVectorizer + GradientBoostingClassifier
Accuracy: 0.49

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| female | 0.48 | 0.81 | 0.61 | 97 |
| male | 0.51 | 0.18 | 0.27 | 103 |
| accuracy | | | 0.49 | 200 |
| macro avg | 0.50 | 0.50 | 0.44 | 200 |
| weighted avg | 0.50 | 0.49 | 0.43 | 200 |

TfidfVectorizer + LogisticRegression
Accuracy: 0.49

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| female | 0.48 | 0.81 | 0.61 | 97 |
| male | 0.51 | 0.18 | 0.27 | 103 |
| accuracy | | | 0.49 | 200 |
| macro avg | 0.50 | 0.50 | 0.44 | 200 |
| weighted avg | 0.50 | 0.49 | 0.43 | 200 |

```
# 结论
results = {
        'CountVectorizer + GradientBoostingClassifier': count_gb_accuracy,
        'CountVectorizer + LogisticRegression': count_lr_accuracy,
        'TfidfVectorizer + GradientBoostingClassifier': tfidf_gb_accuracy,
        'TfidfVectorizer + LogisticRegression': tfidf_lr_accuracy
}

best_model = max(results, key=results.get)
print(f"The best model is {best_model} with accuracy {results[best_model]}")
```

```
The best model is CountVectorizer + GradientBoostingClassifier with accuracy 0.49
```

## Выводы:

В этом эксперименте мы использовали показатели успеваемости студентов.Для набора данных csv в качестве текстовой характеристики был выбран "уровень образования родителей", а "пол" использовался в качестве целевой метки для задач классификации.Мы использовали два метода векторизации текста (CountVectorizer и TfidfVectorizer) и два классификатора

(GradientBoostingClassifier и LogisticRegression) для обучения и оценки модели классификации.

Как видно из результатов эксперимента, коэффициент точности классификации всех моделей составляет 0,49.Это показывает, что использование различных методов векторизации текста (CountVectorizer и TfidfVectorizer) и классификаторов (GradientBoostingClassifier и LogisticRegression) не оказывает существенного влияния на результаты классификации.
Вот несколько возможных причин и предложений:

1.Недостаточные характеристики данных：

В этом эксперименте в качестве единственного текстового признака использовался только "уровень образования родителей", и, возможно, он не в полной мере отражает ключевую информацию о гендерной классификации.
Вы можете попробовать ввести другие текстовые или числовые функции, чтобы повысить производительность модели.

2.Векторизация объектов и выбор модели：

Из-за относительно простой текстовой информации в функции "уровень образования родителей" эффекты CountVectorizer и TfidfVectorizer не сильно отличаются.
Вы можете попробовать другие методы извлечения признаков, такие как встраивание слов или модели глубокого обучения, чтобы повысить эффективность классификации.

3.Дисбаланс данных：

Коэффициент запоминания показывает, что коэффициент запоминания для женщин достигает 0,81, в то время как для мужчин он составляет всего 0,18, что может свидетельствовать о дисбалансе в соотношении полов в наборе данных.
Вы можете попробовать использовать методы балансировки данных, такие как избыточная или недостаточная выборка, чтобы повысить производительность модели.

Таким образом, наилучшей моделью является комбинация CountVectorizer и GradientBoostingClassifier, но общий показатель точности невелик. В будущем производительность классификации может быть улучшена за счет улучшения разработки функций и оптимизации модели.