Lab1-Classification and Regression[[1]](#footnote-1)\*

王佳慧1

1(计算机科学与技术，南方科技大学 深圳)

通讯作者: 王佳慧, E-mail: 11849057@sustc.mail.edu.cn

# 数据预处理

数据预处理是一项非常重要并且消耗时间的工作。在实验所给的数据中，我们将数据预处理分为3个步骤。

1. 读取csv格式的文件。在这一部分我们使用“ReadData”函数实现。由于在train.csv和val.csv中csv格式的分隔符是“；”，所以我们不能使用pd.read\_csv读取文件。那么在这里我们使用csv模块的“csv.reader”，通过分隔符“；”读取文件，并将读到的reader通过“numpy.array”结合“list”函数转化为array形式的数据。
2. **def** ReadData(file):
3. with open(file, 'r') as f:
4. newfile = csv.reader(f, delimiter = ';')
5. dataList = np.array(list(newfile))
6. **return** dataList
7. 对features进行归一化/正则化。在这一部分我们使用“getXY”函数实现。首先我们先将各个feature的最大值和最小值输出，如图1所示，可以看出有些数据的值很小，比如Disciplinary\_failure只分布在0~1之间，而Absenteeism\_time\_in\_hours 则分布在0~120之间，那么我们除了要（在分类时）对Absenteeism\_time\_in\_hours这一列的数据进行normalization还要避免使某些feature值过大导致在模型训练时产生误差。因此这一部分我们主要靠“sklearn.preprocessing”中的“MinMaxScaler”正则化所有的features。
8. min\_max\_scaler = MinMaxScaler()
9. X = min\_max\_scaler.fit\_transform(X)
10. OneHotEncoder转化数据。在这一部分我们使用“OneHotEncoding”函数实现。分析所有的features，其中Day\_of\_the\_week、Disciplinary\_failure、Drinker、Smoker、Seasons都是可以用OneHotEncoder转化的。但是当我们观察val.csv时，Seasons对应的这列数据只取了其中两个值。所以我们只对前面提到的四个feature进行归一化处理。
11. **def** OneHotEncoding(DataList, l):
12. **for** k **in** l:
13. X2 = [100 \* x **for** x **in** DataList[:,k]]
14. X2 = np.array(X2).reshape(-1,1)
15. enc = OneHotEncoder(sparse=False)
16. X2 = enc.fit\_transform(X2)
17. DataList = np.concatenate((DataList, X2), axis=1)
18. DataList = np.delete(DataList, l, axis=1)
19. **return** DataList

经过上面三个步骤的处理，我们的train数据由原来的540\*21变成540\*27。

1. l1 = [2, 10, 13, 14]
2. l2 = [3, 11, 14, 15]
3. X\_train\_class = OneHotEncoding(X\_train\_class, l1)
4. X\_val\_class = OneHotEncoding(X\_val\_class, l1)
5. X\_train\_reg = OneHotEncoding(X\_train\_reg, l2)
6. X\_val\_reg = OneHotEncoding(X\_val\_reg, l2)

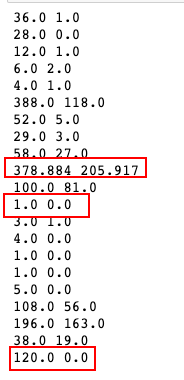


图1 feature的最大和最小值

# 分类和回归

在分类和回归我分别采用了三种模型进行比较，即SVM、MLP、Linear Model。这三种模型都是用sklearn实现的，下面进行具体分析。

## SVM

针对SVM，我分别使用了SVC和SVR进行分类和回归。其中:

* SVC。SVC的kernel有四种选择，我分别对这四种模型kernel进行了训练，得到了如表1所示不同的f1\_score。

1. **def** svmClass(X\_train\_class, Y\_train\_class, X\_val\_class, Y\_val\_class):
2. l = ['linear', 'rbf', 'sigmoid', 'poly']
3. micro\_f1\_max = 0
4. micro\_f1\_model = ""
5. **for** i **in** l:
6. clf = SVC(kernel=i)
7. clf.fit(X\_train\_class, Y\_train\_class)
8. clf\_predict = clf.predict(X\_val\_class)
9. **from** sklearn.metrics **import** f1\_score
10. micro\_f1 = f1\_score(Y\_val\_class, clf\_predict, average='micro')
11. **if** micro\_f1 > micro\_f1\_max:
12. micro\_f1\_max = micro\_f1
13. **return** micro\_f1\_model, micro\_f1\_max

* SVR。针对SVR的kernel进行训练也得到了mse。如表1所示。

表**1** SVM

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel | SVC |  | SVR |
| Linear | 0.38 |  | 181.714601392 |
| rbf | 0.21 |  | 192.305620336 |
| Sigmoid | 0.21 |  | 194.099671347 |
| Poly | 0.14 |  | 196.364791518 |

由表3我们可以看出，当kernel=“linear”时，对应的micro\_f1是最高的，为0.38，mse是最小的，近似为181。

## MLP

针对MLP，我分别使用了MLPClassifier和MLPRegression分析分类和回归。我设置了如表2所示的参数。

表**2** MLP

|  |  |  |  |
| --- | --- | --- | --- |
|  | MLPClassifier |  | MLPRegression |
| Solver | adam |  | sgd |
| Activation | tanh |  | relu |
| Alpha | 1e-5 |  | 1e-5 |
| Hidden\_layer\_sizes | (100, 30) |  | (100, 50) |
| Result | micro\_f1: 37% |  | Mse: 167 |

最终得到的分类预测精度micro\_f1为0.37，mse近似为167。

## Linear Model

针对Linear Model，我分别使用了SGDClassifier和LogisticRegression。参数设置如表3和表4所示。

表**3** SGDClassifier

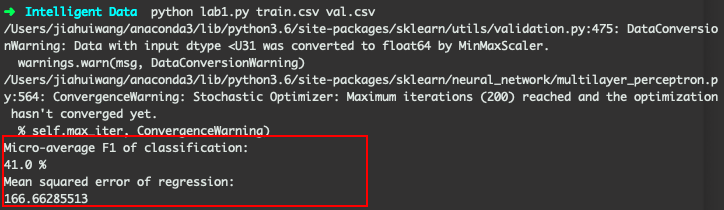
|  |  |  |
| --- | --- | --- |
|  | SGDClassifier |  |
| Max\_iter | 1000 |  |
| tol | 1e-3 |  |

表**4** LogisticRegression

|  |  |  |
| --- | --- | --- |
|  | LogisticRegression |  |
| solver | lbfgs |  |
| multi\_class | multinomial |  |
| Random\_state | 0 |  |

最终得到的分类预测精度micro\_f1为0.42，mse近似为178。

结合上面三种模型，我们最终输出最大的micro\_f1为41%（Linear model），最小的mse为167（MLP）。



1. [↑](#footnote-ref-1)