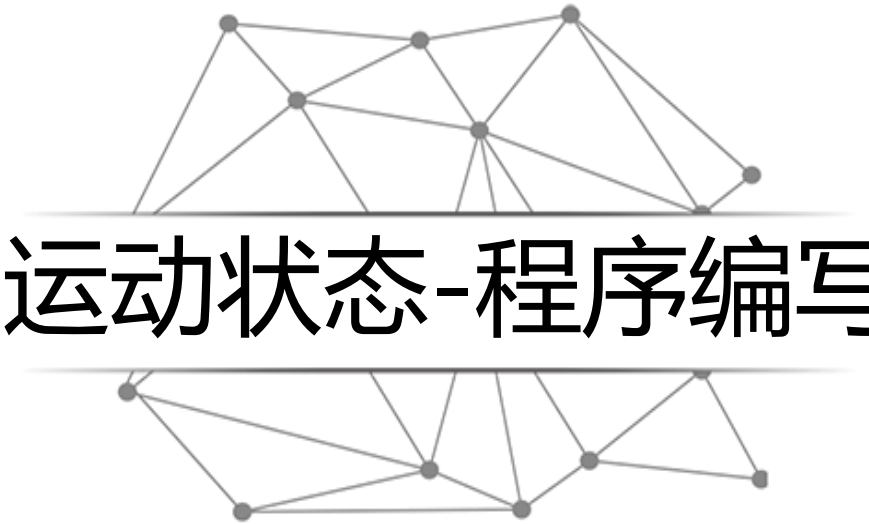
监督学习

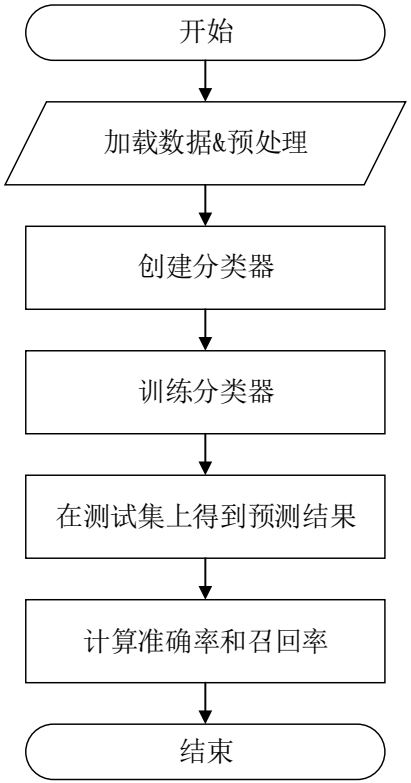
ML14

礼欣
www.python123.org



运动状态-程序编写

算法流程



- 需要从特征文件和标签文件中将所有数据加载到内存中，由于存在缺失值，此步骤还需要进行简单的数据预处理。
- 创建对应的分类器，并使用训练数据进行训练。
- 利用测试集预测，通过使用真实值和预测值的比对，计算模型整体的准确率和召回率，来评测模型。

模块导入

```
1  #-*- coding:utf-8 -*-
2
3  import numpy as np
4  import pandas as pd
5
6  from sklearn.preprocessing import Imputer
7  from sklearn.model_selection import train_test_split
8  from sklearn.metrics import classification_report
```

- 导入numpy库和pandas库。
- 从sklearn库中导入预处理模块Imputer
- 导入自动生成训练集和测试集的模块train_test_split
- 导入预测结果评估模块classification_report

模块导入

```
10
11  from sklearn.neighbors import KNeighborsClassifier
12  from sklearn.tree import DecisionTreeClassifier
13  from sklearn.naive_bayes import GaussianNB
```

接下来，从sklearn库中依次导入三个分类器模块：K近邻分类器KNeighborsClassifier、决策树分类器DecisionTreeClassifier和高斯朴素贝叶斯函数GaussianNB。

数据导入函数

```
16  def load_dataset(feature_paths, label_paths):
17      """读取特征文件列表和标签文件列表中的内容，归并后返回
18      """
19      feature = np.ndarray(shape=(0,41))
20      label = np.ndarray(shape=(0,1))
```

- 编写数据导入函数，设置传入两个参数，分别是特征文件的列表feature_paths和标签文件的列表label_paths。
- 定义feature数组变量，列数量和特征维度一致为41；定义空的标签变量，列数量与标签维度一致为1。

数据导入函数

```
21        for file in feature_paths:
22            # 使用逗号分隔符读取特征数据，将问号替换标记为缺失值，文件中不包含表头
23            df = pd.read_table(file, delimiter=',', na_values='?', header=None)
24            # 使用平均值补全缺失值，然后将数据进行补全
25            imp = Imputer(missing_values='NaN', strategy='mean', axis=0)
26            imp.fit(df)
27            df = imp.transform(df)
28            # 将新读入的数据合并到特征集合中
29            feature = np.concatenate((feature, df))
```

- 使用pandas库的read_table函数读取一个特征文件的内容，其中指定分隔符为逗号、缺失值为问号且文件不包含表头行。
- 使用Imputer函数，通过设定strategy参数为‘mean’，使用平均值对缺失数据进行补全。fit()函数用于训练预处理器，transform()函数用于生成预处理结果。
- 将预处理后的数据加入feature，依次遍历完所有特征文件

数据导入函数

```
30        for file in label_paths:
31            # 读取标签数据，文件中不包含表头
32            df = pd.read_table(file, header=None)
33            # 将新读入的数据合并到标签集合中
34            label = np.concatenate((label, df))
35        # 将标签归整为一维向量
36        label = np.ravel(label)
37        return feature, label
```

- 遵循与处理特征文件相同的思想，我们首先使用pandas库的read_table函数读取一个标签文件的内容，其中指定分隔符为逗号且文件不包含表头行。
- 由于标签文件没有缺失值，所以直接将读取到的新数据加入label集合，依次遍历完所有标签文件，得到标签集合label。
- 最后函数将特征集合feature与标签集合label返回。

主函数-数据准备

```
41  if __name__ == '__main__':
42      # 设置数据路径
43      feature_paths = ['A/A.feature', 'B/B.feature', 'C/C.feature', 'D/D.feature', 'E/E.feature']
44      label_paths = ['A/A.label', 'B/B.label', 'C/C.label', 'D/D.label', 'E/E.label']
45
46      # 将前4个数据作为训练集读入
47      x_train, y_train = load_dataset(feature_paths[:4], label_paths[:4])
48      # 将最后1个数据作为测试集读入
49      x_test, y_test = load_dataset(feature_paths[4:], label_paths[4:])
50
51      # 使用全量数据作为训练集，借助train_test_split函数将训练数据打乱
52      x_train, x_, y_train, y_ = train_test_split(x_train, y_train, test_size = 0.0)
```

- 设置数据路径feature_paths和label_paths。
- 使用python的分片方法，将数据路径中的前4个值作为训练集，并作为参数传入load_dataset()函数中，得到训练集合的特征x_train，训练集的标签y_train。
- 将最后一个值对应的数据作为测试集，送入load_dataset()函数中，得到测试集合的特征x_test，测试集的标签y_test。

主函数-数据准备

```
41  if __name__ == '__main__':
42      # 设置数据路径
43      feature_paths = ['A/A.feature', 'B/B.feature', 'C/C.feature', 'D/D.feature', 'E/E.feature']
44      label_paths = ['A/A.label', 'B/B.label', 'C/C.label', 'D/D.label', 'E/E.label']
45
46      # 将前4个数据作为训练集读入
47      x_train, y_train = load_dataset(feature_paths[:4], label_paths[:4])
48      # 将最后1个数据作为测试集读入
49      x_test, y_test = load_dataset(feature_paths[4:], label_paths[4:])
50
51      # 使用全量数据作为训练集，借助train_test_split函数将训练数据打乱
52      x_train, x_, y_train, y_ = train_test_split(x_train, y_train, test_size = 0.0)
```

使用train_test_split()函数，通过设置测试集比例test_size为0，将数据随机打乱，便于后续分类器的初始化和训练。

3-主函数-knn

```
54        # 创建k近邻分类器，并在测试集上进行预测
55        print("Start training knn")
56        knn = KNeighborsClassifier().fit(x_train, y_train)
57        print("Training done!")
58        answer_knn = knn.predict(x_test)
59        print("Prediction done!")
```

- 使用默认参数创建K近邻分类器，并将训练集x_train和y_train送入fit()函数进行训练，训练后的分类器保存到变量knn中。
- 使用测试集x_test，进行分类器预测，得到分类结果answer_knn。

3-主函数-决策树

```
61        # 创建决策树分类器，并在测试集上进行预测
62        print("Start training DT")
63        dt = DecisionTreeClassifier().fit(x_train, y_train)
64        print("Training done!")
65        answer_dt = dt.predict(x_test)
66        print("Prediction done!")
```

- 使用默认参数创建决策树分类器dt，并将训练集x_train和y_train送入fit()函数进行训练。训练后的分类器保存到变量dt中。
- 使用测试集x_test，进行分类器预测，得到分类结果answer_dt。

3-主函数-贝叶斯

```
68        # 创建贝叶斯分类器，并在测试集上进行预测
69        print("Start training Bayes")
70        gnb = GaussianNB().fit(x_train, y_train)
71        print("Training done!")
72        answer_gnb = gnb.predict(x_test)
73        print("Prediction done!")
```

- 使用默认参数创建贝叶斯分类器，并将训练集x_train和y_train送入fit()函数进行训练。训练后的分类器保存到变量gnb中。
- 使用测试集x_test，进行分类器预测，得到分类结果answer_gnb。

3-主函数-分类结果分析

```
75      # 计算准确率与召回率
76      print("\n\nThe classification report for knn:")
77      print(classification_report(y_test, answer_knn))
78
79      print("\n\nThe classification report for dt:")
80      print(classification_report(y_test, answer_dt))
81
82      print("\n\nThe classification report for gnb:")
83      print(classification_report(y_test, answer_gnb))
```

使用classification_report函数对分类结果，从精确率precision、召回率recall、f1值f1-score和支持度support四个维度进行衡量。

分别对三个分类器的分类结果进行输出

结果展示-k近邻

Classification report for knn:

	precision	recall	f1-score	support
0.0	0.56	0.60	0.58	102341
1.0	0.92	0.93	0.93	23699
2.0	0.94	0.78	0.85	26864
3.0	0.83	0.82	0.82	22132
4.0	0.85	0.88	0.87	32033
5.0	0.39	0.21	0.27	24646
6.0	0.77	0.89	0.82	24577
7.0	0.80	0.95	0.87	26271
12.0	0.32	0.33	0.33	14281
13.0	0.16	0.22	0.19	12727
16.0	0.90	0.67	0.77	24445
17.0	0.89	0.96	0.92	33034
24.0	0.00	0.00	0.00	7733
avg / total	0.69	0.69	0.68	374783

结果展示-决策树

Classification report for dt:

	precision	recall	f1-score	support
0.0	0.55	0.83	0.66	102341
1.0	0.79	0.96	0.87	23699
2.0	0.91	0.84	0.87	26864
3.0	0.93	0.73	0.82	22132
4.0	0.63	0.95	0.76	32033
5.0	0.73	0.50	0.59	24646
6.0	0.05	0.01	0.02	24577
7.0	0.32	0.14	0.20	26271
12.0	0.60	0.66	0.63	14281
13.0	0.67	0.48	0.56	12727
16.0	0.57	0.07	0.13	24445
17.0	0.86	0.85	0.86	33034
24.0	0.37	0.30	0.33	7733
avg / total	0.61	0.64	0.60	374783

结果展示-贝叶斯

Classification report for gnb:

	precision	recall	f1-score	support
0.0	0.62	0.81	0.70	102341
1.0	0.97	0.91	0.94	23699
2.0	1.00	0.65	0.79	26864
3.0	0.60	0.66	0.63	22132
4.0	0.91	0.77	0.83	32033
5.0	1.00	0.00	0.00	24646
6.0	0.87	0.72	0.79	24577
7.0	0.31	0.47	0.37	26271
12.0	0.52	0.59	0.55	14281
13.0	0.61	0.50	0.55	12727
16.0	0.89	0.72	0.79	24445
17.0	0.75	0.91	0.82	33034
24.0	0.59	0.24	0.34	7733
avg / total	0.74	0.68	0.67	374783

结果对比

模型	K近邻	决策树	贝叶斯
准确度	0.69	0.61	0.74
召回率	0.69	0.64	0.68
F1值	0.68	0.60	0.67

结论：

- 从准确度的角度衡量，贝叶斯分类器的效果最好
- 从召回率和F1值的角度衡量，k近邻效果最好
- 贝叶斯分类器和k近邻的效果好于决策树

课后思考

- 在所有的特征数据中，可能存在缺失值或者冗余特征。如果将这些特征不加处理地送入后续的计算，可能会导致模型准确度下降并且增大计算量。
- 在特征选择阶段，通常需要借助辅助软件（例如Weka）将数据进行可视化并进行统计。
- 请大家可以通过课外学习思考如何筛选冗余特征，提高模型训练效率，也可以尝试调用sklearn提供的其他分类器进行数据预测。