

哈尔滨理工大学

实 验 报 告

课程名称： 深度学习与目标检测 I

实验名称： 利用 SVM 预测交通流量

预习 报告		过程及 记录		结果 分析		实验 总结		总评	
教师签名：									

班 级

学 号

姓 名

指导教师

教务处 印制

一、实验预习（准备）报告

1、实验目的

- ①利用支持向量机预测体育场馆周边交通流量。样本特征分别为：星期、时间、对手球队、棒球比赛是否正在进行、通行汽车数量。
- ②利用 Sklearn 中函数划分训练集、测试集，其中参数 `test_size=0.25`, `random_state=5`
- ③选择基于径向基核函数的支持向量机回归器模型初始参数为 `C=10`, `epsilon=0.2`
- ④打印 R2 分数
- ⑤预测样本为: "Tuesday", "13:35", "San Francisco", "yes"
- ⑥打印预测结果
- ⑦对预测结果进行分析讨论

2、实验相关原理及内容

支持向量机是一种基于二分类的机器学习算法也就是一种基于最大间距的线性分类器，其最大间距使得其与感知机不同；支持向量机也包含了核技术，使得其本质上是一个非线性分类。支持向量机的学习策略是最大区间，它可以被形式化成一个凸二次规划问题，并且可以将其等价于正则化的关键损耗函数最小化。支持向量机的学习算法是一种最优的凸二次规划。数学上，一个支持向量机在一个高维或有限维空间构造了一个或一组超平面，这些超平面被用作分类、回归或其它任务。本质上，由超平面实现的最优分割，即是这个超平面到任何类的最近的训练数据点的距离是最大的。通常来说，边界越大，分类器的泛化误差就越低。

本实验是选择基于径向基核函数的支持向量机回归器模型来预测体育场馆周边交通流量的，并对某个给定的样本预测，同时对预测结果进行分析讨论。

3、实验方法及步骤设计

- ①数据预处理：从文本文件中提取数据，并处理可能存在的缺失值，选择合适的方法进行填充或处理。对分类特征进行编码，确保模型能够正确处理这些特征。
- ②模型选择：选择支持向量机回归作为预测模型，采用基于高斯径向基的 SVM 回归。确定模型的超参数，例如 `C` 和 `epsilon`。
- ③数据集划分：将准备好的数据集划分为训练集和测试集，确保在模型训练和评估过程中有独立的数据集。
- ④模型训练与评估：使用训练集对支持向量机回归模型进行训练。使用测试集评估模型的性能，主要关注 R2 值。
- ⑤预测分析：使用训练好的模型对新数据进行预测，确保预测特征经过正确的编码。分析预测结果，包括合理性分析、特征重要性分析等。
- ⑥实验总结与改进空间：总结实验结果，强调模型的优点和局限性。提出可能的改进空间，包括调整超参数、增加特征或尝试其他回归算法等。
- ⑦文档记录与报告：记录实验过程中的关键步骤、参数设置和实验结果。撰写实验报告，包括问题定义、数据收集、模型设计、实验步骤、结果分析和结论等部分。

4、实验设备仪器选择及材料

一台笔记本电脑、PyCharm、A4 记录纸、截屏软件、键盘、鼠标等。

5、实验注意事项（包括实验过程、仪器设备、个人操作等方面）

①处理预测特征时，确保使用的编码器能够将其转化为模型可理解的形式。这涉及对数据进行适当的转换和标准化，以确保模型能够准确解读输入。

②确保实验环境的稳定性和一致性，包括数据处理软件、模型训练工具和硬件设备等。这有助于消除外部因素对实验结果的影响。

③详细记录实验过程、参数设置、数据处理步骤以及模型训练结果。这有助于实验结果的复现和分析，并在必要时进行修改和改进。

二、实验过程及记录

2.1 数据预处理

通过对本实验的数据集“traffic.txt”仔细查看，可以发现特征存在字符串形式的数据以及存在缺失值的情况，因此在模型训练之前需要处理一下。对于字符串形式数据处理的办法，我采用的是使用标签编码，直接对类别特征进行大量映射，有多少类别那么取值就代表了多少。对于存在缺失值的情况，我发现只有通行汽车数量存在‘-1’，由此可以看出为‘-1’的汽车数量数据是为缺失的，对于数据缺失的情况我们有好几种处理办法，常见的有简单删除法、权重法和统计补全法。通过把全部数据提取到一个列表中，查看数据集的 shape 为(17568, 4)，可以发现数据集数量比较庞大，因此可以采用对存在缺失值的样本“删除”，具体代码中表现为在提取样本时检测是否含有“-1”项数据，若有则不读取到代码中的数据集中。

2.2 模型训练和评估

根据题目要求使用基于径向基核函数的支持向量机回归器模型，并初始化参数为 $C=10$, $\epsilon=0.2$ 。再将训练集送进模型里去预测，并打印 R2 分数如图 1 所示，得到 R2 分数为 0.53436384983032。同时由于数据集已经被划分成了训练和测试集，于是我又拿测试集对模型进行了预测，并将前 500 个预测结果和真实结果可视化出来了，结果如图 2 所示。

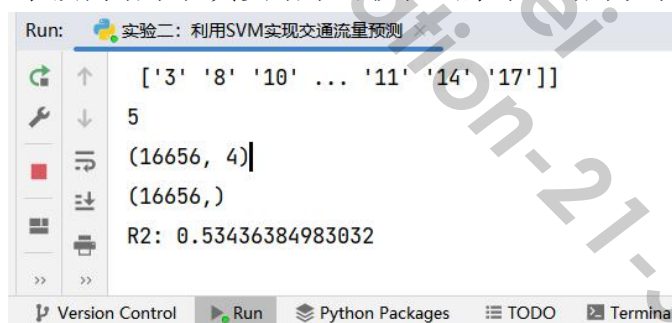


图 1 支持向量机下的 R2 分数

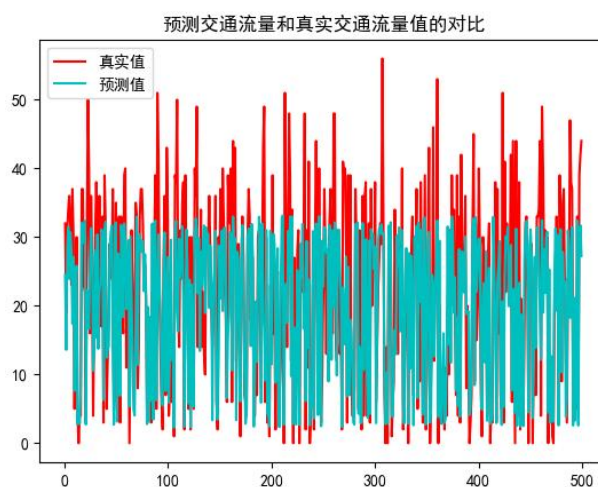
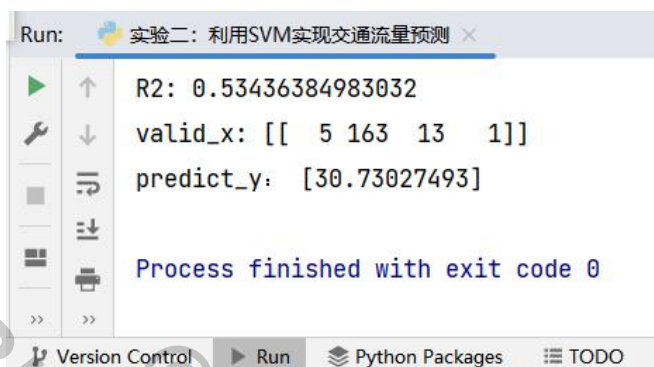


图 2 预测交通流量和真实交通流量值的对比

2.3 模型预测

预测样本为: "Tuesday", "13:35", "San Francisco", "yes", 首先对要预测的样本特征进行编码, 再进行预测, 最后将预测结果打印, 得到的结果为: `predict_y: [30.73027493]`, 如下图 4 所示。



```
Run: 实验二: 利用SVM实现交通流量预测 x
R2: 0.53436384983032
valid_x: [[ 5 163 13 1]]
predict_y: [30.73027493]
Process finished with exit code 0
Version Control Run Python Packages TODO
```

图 4 预测样本的预测结果

三、实验结果分析

(1) 模型训练结果分析

模型最开始训练得到的 R^2 值为 0.5343, 表示模型在测试集上的解释方差相对较高, 但仍有一定的误差。这说明模型对数据的一部分变异性能够进行较好的解释, 但并非全部。这样的 R^2 值表明模型能够解释测试集上 53.43% 的目标变量的变异性。这并不是一个很高的 R^2 值, 但也不是低到完全没有解释能力。

这表示模型在测试集上的拟合效果一般。可能的原因有模型过于简单或者过于复杂, 如果有优化需求的话, 那么需要进一步调整正则化参数 C 和 ϵ 。同时也有可能是因为模型的性能受到了特征的质量和数量的影响。

通过对图 2 预测交通流量和真实交通流量值的对比图分析可以发现, 观察到实际交通流量值具有很大的起伏性和波动, 这表明在现实数据中存在着较大的变化和不确定性。可能的情况是, 实际的交通流量受到多种影响因素的复杂组合, 包括天气、事件、节假日等, 这些因素可能导致交通流量有很大的波动性。

而预测曲线相对平缓, 没有捕捉到实际数据中的急剧波动和噪声。可能是因为 SVM 回归器对于训练数据中的噪声敏感, 面对真实结果的突然变化, 预测结果不能准确预测可能是因为模型对于这种突变的情况不够灵活。径向基核函数在局部区域内具有较强的拟合能力, 可能存在局部区域过度拟合的情况, 但它对数据中的快速变化不够敏感。

当然也有可能是因为样本不平衡, 如果实际数据中存在较大的起伏性, 但是在训练数据中未能充分涵盖这些起伏, 模型可能无法捕捉到这些特征。我们需要确保训练数据充分代表实际数据的多样性。

在实验过程中有一个小插曲, 我在标签编码的时候不小心也对标签值进行了编码, 导致标签值错乱, 得到的 R^2 值如图 5 所示, 为 -0.04159282632686989, 这显然有问题, R^2 值为负数通常表示模型的性能比直接使用平均值来预测目标变量的效果还要差。我排查了半天错误, 最后才发现是对标签值进行编码导致的问题, 因为这样会掩盖它的真实值, 模型可能无法正确理解标签之间的关系, 从而导致性能下降。

```
Run: 实验二：利用SVM实现交通流量预测 ×
[ '3' '8' '10' ... '11' '14' '17' ]
5
(16656, 4)
(16656,)
R2: -0.04159282632686989
Version Control Run Python Packages TODO
```

图 5 代码有 BUG 情况下的 R2 分数

(2) 模型预测结果分析

根据给定的特征，得到的预测值为 30.73027493，由于预测值表征的是通行汽车数量，取整为 31。根据问题的背景，31 辆车可能是一个合理的交通流量估计。如果这个值与实际情况相符，那么模型的预测就显得可信。如果需要进一步考察其合理性，考虑查看模型对特征的重要性。对于这个问题，星期、时间、对手球队、棒球比赛是否正在进行、通行汽车数量都是特征。检查哪些特征对预测结果的影响最大，这可以帮助理解模型的决策过程。

(3) 模型改进分析

结合以上分析和数据特点，我尝试了使用随机森林模型来查看实验结果表现如何，先通过网格搜索法找到适合的参数 $\{\text{max_depth}=15, \text{n_estimators}=200, \text{min_samples_split}=5\}$ ，然后打印 R2 分数为 0.8095354500555578，可以发现 R2 分数较高，说明随机森林模型可能较为适合本实验的模型，下图 6 是两个模型得到的预测值与真实值的对比图。可以发现随机森林模型可以更好地去追踪噪声点。

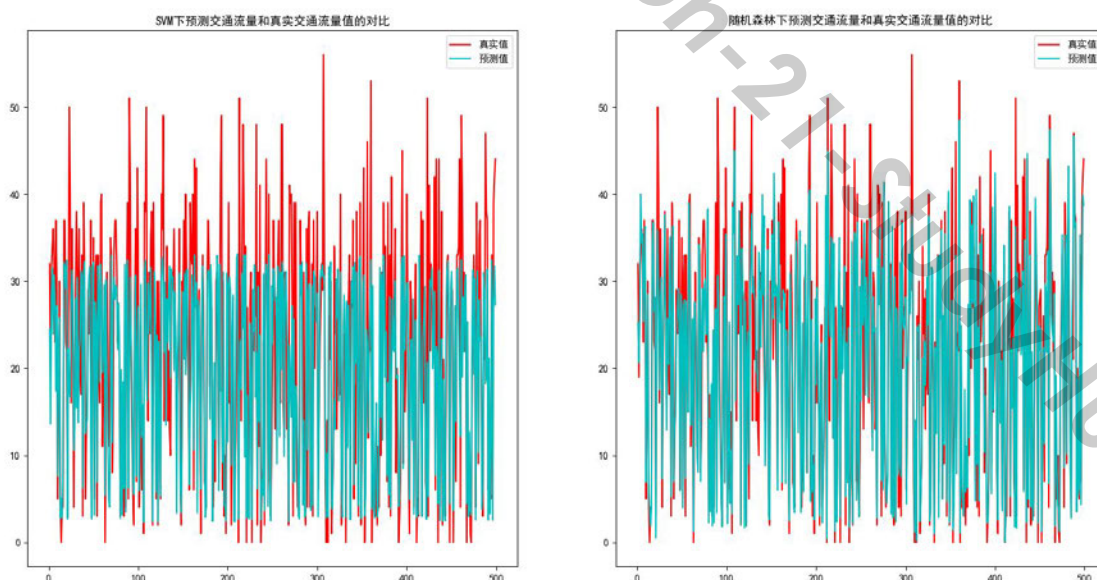


图 6 两个模型预测值与真实值的对比

并且拿预测样本进行预测，得到的结果和支持向量机的结果很接近，说明之前用支持向量机预测的结果较为准确。


```
test × 实验二：利用SVM实现交通流量预测 ×
5
(16656, 4)
(16656,)
随机森林-R2: 0.8095354500555578
SVM-R2: 0.53436384983032
valid_x: [[ 5 163 13 1]]
SVM-predict_y: [30.73027493]
随机森林-predict_y: [30.64354926]
```

图 7 两个模型预测值对比

四、实验总结

这次实验使用了基于径向基核函数的支持向量机回归器模型（SVM）来预测体育场馆周边的交通流量。在数据处理阶段，我首先从文本文件中提取数据。我使用了标签编码器来对特征进行编码，确保其能够被模型准确辨识。这种处理方式为模型提供了可靠的输入，从而提高了预测的准确性。我注意到文本文档中存在一些通行汽车数量的缺失值，因此我采用了简单删除法，将这些缺失值去掉了，以确保数据的完整性。

在选择模型时，本实验采用了基于高斯径向基（RBF）的 SVM 回归。最开始我误用了 SVC 分类法，导致产生了一个较差的预测效果。在进行预测时，模型的 R2 分数并不高，我猜想可能是因为选取的模型不太合适，R2 值的分析结果提示模型在解释目标变量方面存在一定的不足，可能需要进一步优化或考虑其他预测模型以提高预测性能。通过查看样本真实值的曲线可以发现曲线波动比较大，也就是说存在一些噪声点，而经过课堂上的理论学习和实践，我们知道支持向量机对于噪声非常敏感，当大量的噪声时，它会对支持向量机模型产生很大的影响。并且这次的数据集的样本量并不小，有 17568 个样本，由于数据量较大，因此支持向量机的训练时间也相对较长。这个结果为未来的实验提供了有益的指导，帮助我更好地理解模型的局限性，并采取进一步的改进措施。

在实验中还出现了一个小 BUG，就是我对标签值进行编码导致了一些问题，导致模型性能非常差，后来通过仔细分析和检查终于找到原因了，这也让我对标签编码有了很深刻的认识。

最后我尝试改进模型，选用了课堂上学过的随机森林模型，效果比支持向量机好一些，但是，随机森林模型求得的 R2 值也仅为 0.80 左右。看来也不算是能够解决目标问题的适合模型，希望在以后可以学习到更多的算法模型，来解决更多地实际问题，从而更好地帮助人们。

实验二附录源码:

```

1. import sklearn.preprocessing as sp      # 便签编码
2. import numpy as np
3. from sklearn.model_selection import train_test_split
4. import matplotlib.pyplot as plt
5. import sklearn.svm as svm
6. import sklearn.metrics as sm  # 用来性能评价
7. import sklearn.ensemble as se  # 集成学习的模块
8. import sklearn.model_selection as ms
9.
10. plt.rcParams["font.sans-serif"] = "SimHei"  # 显示中文
11. #STEP 1  读取样本并对样本进行标签编码
12. raw_samples = []  # 用来临时保存样本列表 数据集有缺省值, 需要直接删除
13. with open("traffic.txt", "rt") as f:
14.     for line in f.readlines():
15.         if '-1' not in line:  # 判断每一行的数据中是否缺失值
16.             line = line.replace("\n", '')
17.             raw_samples.append(line.split(","))
18.         else:
19.             continue  # 有缺失值的处理情况
20. x = np.array(raw_samples).T  # 转化成数组, 并且转置, 那么每一行代码的就是全部的特
    征
21. print(x)
22. print(len(x))
23. # 记录编码规则 (用标签编码器来记录)
24. encoders = []  # 存放所有的标签编码器, 所有的编码规则
25. train_x = []  # 存放编码后的输入
26. train_y = []
27. # 对样本进行遍历, 并执行标签编码
28. for row in range(len(x)) :
29.     encoder = sp.LabelEncoder()  # 创建标签编码器, 相当于从本子上撕下来一个标签
        纸
30.     encoders.append(encoder)
31. # 前4行是特征数据值, 最后一行是标签值
32.     if row < len(x) - 1 :  # 不是最后一行就拿过来进行标签编码
33.         lbl_code = encoder.fit_transform(x[row])  # 产生一个编码规则并且进行
            编码, 相当于思考怎么安排, 然后对标签纸进行填写信息
34.         # fit 表示制定规则(思考), transform 表示进行编码(写标签)
35.         train_x.append(lbl_code)  # 二维的, lbl_code 是编码之后的序列
36.     else:  # 是最后一行(是数据的标签值)不需要标签编码
37.         train_y = [int(y) for y in x[row]]  # 把标签值转化为 int 型
38.         # train_y = encoder.fit_transform(x[row])  # 一维的
39. train_x = np.array(train_x).T

```

```

40. train_y = np.array(train_y).T
41. print(train_x.shape)
42. print(train_y.shape)
43. train_x, test_x, train_y, test_y= train_test_split(train_x,train_y,test_size=0.25, random_state=5)
44. model = svm.SVR(kernel="rbf", #径向基核函数
45.                  epsilon=0.2, #核函数的误差系数
46.                  C=10) #概率强度, 影响分类边界
47. model_tree = se.RandomForestRegressor(
48.     max_depth=15,
49.     n_estimators=200,
50.     min_samples_split=5 # 最小划分样本, 防止过拟合
51. )
52. model_tree.fit(train_x,train_y)
53. predict_test_tree_y = model_tree.predict(test_x)
54. r2 = sm.r2_score(test_y, predict_test_tree_y)
55. print("随机森林-R2:",r2)
56. model.fit(train_x,train_y) #训练
57. predict_test_y = model.predict(test_x)
58. r2 = sm.r2_score(test_y, predict_test_y)
59. print("SVM-R2:",r2)
60. plt.subplot(1,2,1)
61. plt.title("SVM 下预测交通流量和真实交通流量值的对比")
62. plt.plot(range(1,500),test_y[:499],color="r",label="真实值")
63. plt.plot(range(1,500),predict_test_y[:499],color="c",label="预测值")
64. plt.subplot(1,2,2)
65. plt.title("随机森林下预测交通流量和真实交通流量值的对比")
66. plt.plot(range(1,500),test_y[:499],color="r",label="真实值")
67. plt.plot(range(1,500),predict_test_tree_y[:499],color="c",label="预测值")
68. plt.savefig("对比曲线图.jpg")
69. plt.legend()
70. valid_data = [["Tuesday", "13:35", "San Francisco", "yes"]] #待测试的数据
71. valid_data = np.array(valid_data).T
72. valid_x = []
73. for row in range(len(valid_data)):
74.     data_encode = encoders[row].transform(valid_data[row])
75.     valid_x.append(data_encode)
76. valid_x = np.array(valid_x).T
77. print("valid_x:",valid_x)
78. predict_y = model.predict(valid_x) #解码
79. print("SVM-predict_y: ",predict_y)
80. predict_y = model_tree.predict(valid_x)
81. print("随机森林-predict_y: ",predict_y)
82. plt.show()

```


哈尔滨理工大学

实 验 报 告

课程名称：深度学习与目标检测 I

实验名称：对共享单车使用量进行预测

预习 报告		过程及 记录		结果 分析		实验 总结		总评	
教师签名：									

班 级

学 号

姓 名

指导教师

教务处 印制

一、实验预习（准备）报告

1、实验目的

- 掌握数据处理与预处理：通过完成划分数据集为训练集和测试集和将数据随机打乱，我们将学会基于天和基于小时的数据集处理。
- 掌握不同回归模型的训练与预测：本实验是回归问题，我们将通过课堂上学过的几种深度学习模型来进行训练，并根据不同的数据集进行共享单车使用量的预测。
- 掌握性能评估与可视化：我们将了解如何使用 R^2 分数来评估模型的性能，同时学习如何可视化特征重要性，以深入了解模型对预测的贡献。
- 掌握调整超参数与比较：通过尝试不同的深度学习模型以及调整模型参数，我们将分析不同模型、不同参数对预测结果的影响，最终选择出最优的模型和参数组合。

2、实验相关原理及内容

在本次实验中，我会使用随机森林回归、Lasso 回归、岭回归这三种回归模型来训练，因此以下分别介绍它们的原理和内容。

①随机森林回归

原理：随机森林是一种基于集成学习的监督式机器学习算法，集成学习是一种学习类型，可以多次加入不同类型的算法或相同算法，以形成更强大的预测模型。随机森林结合了多个相同类型的算法，即多个决策的树木，故名“随机森林”。在回归问题中，每个决策树对目标变量进行预测，最终的预测结果是所有决策树预测结果的平均值。

实验内容：使用随机森林回归模型对共享单车使用量进行预测，利用网格搜索法或者是验证曲线，来寻找适合的最优超参数，如树的数量、最大深度等，观察对预测性能的影响。

②Lasso 回归（L1 正则化）

原理：Lasso 回归通过在损失函数中添加 L1 正则化项，促使模型的权重参数趋向于零，实现特征的稀疏性，某些特征的权重会变成精确的零。从逻辑上说，Lasso 回归可以理解为通过调整损失函数，减小函数的系数，从而避免过于拟合于样本，降低偏差较大的样本的权重和对模型的影响程度。

实验内容：使用 Lasso 回归模型对共享单车使用量进行预测。调整正则化参数，观察对特征选择和模型性能的影响。

③岭回归（L2 正则化）

原理：岭回归是一种线性回归模型，通过在损失函数中引入 L2 正则化项来改进普通的最小二乘法。L2 正则化项是参数的平方和的倍数，用以对参数进行惩罚，防止过拟合。通过引入 L2 正则化项，岭回归对于参数的估计不仅要拟合训练数据，还要保持参数尽可能小。这有助于减小参数的方差，提高模型的泛化性能。

实验内容：使用岭回归模型对共享单车使用量进行预测。调整正则化参数，观察对模型性能的影响。

3、实验方法及步骤设计

①获取数据集：使用 pandas 库中的 read_csv 来读取数据集 bike_day.csv 和 bike_hour.csv，并将相应的特征与标签分开存储。

②对实验数据进行预处理：在训练前对数据进行随机打乱，以确保模型在不同样本上的泛化能力，避免模型学到数据中的某种顺序性。

③数据集划分：将数据集划分为训练集和测试集，采用 80-20 的比例。确保训练集用于模型训练，测试集用于性能评估。

④模型选择：分别建立基于随机森林、Lasso 回归和岭回归回归的三种模型。⑤模型训练

与预测：对每个选择的回归模型，分别在基于天和基于小时的数据上进行训练，利用网格搜索法给定参数取值范围，然后在对应的测试集上进行预测。

⑥性能评估：使用 R^2 分数来评估模型在测试集上的性能，通过查看将测试集送到模型当中测评得到的分数。

⑦调优与比较：以岭回归模型举例，通过调整不同的正则化参数，观察对模型性能的影响。可绘制正则化参数与性能的关系曲线，选择合适的参数值。比较随机森林、Lasso 回归和岭回归在性能上的差异，来评估哪个模型更适合用于对共享单车使用量进行预测，从而选择出最优模型和最优参数。

⑧特征重要性可视化：对于随机森林模型，可视化各特征的重要性，了解模型对不同特征的关注程度。这有助于解释模型的预测过程。

⑨讨论与总结：讨论不同模型对预测的影响，分析不同参数的作用。总结各模型的优劣势，适用场景，以及在实验中的表现。

4、实验设备仪器选择及材料

一台笔记本电脑、PyCharm、A4 记录纸、截屏软件、键盘、鼠标等。

5、实验注意事项（包括实验过程、仪器设备、个人操作等方面）

①注意过拟合：需要注意模型是否存在过拟合的情况。

②注意划分数据集：合理划分训练集和测试集，确保模型在未见过的数据上进行测试，以评估其泛化性能。

③实验文档记录：记录实验过程中的关键步骤、参数设置和实验结果，以建立清晰的实验文档，方便后续分析和报告。

④避免偶然性：实验结果具有偶然性，编写完程序后应该进行多次的实验。

⑤及时保存文件：及时保存编写的程序，以防电脑突然死机。

二、实验过程及记录

2.1 数据预处理

实验给定的数据集是 csv 格式，所以这里应用了 pandas 库来对 csv 文件读取。同时由于数据集当中的数据并不都是有效特征，故采用切片操作来读取相应有关的特征。相关代码如下：

```
1. raw_data = [] # 用来临时保存样本列表
2. raw_data = pd.read_csv("bike_day.csv")
3. # 只提取有用的特征，对数据集切片，使用 iloc 函数
4. x_data = raw_data.iloc[:, 2:13] # 把有用的特征即从 season-windspeed, 行全取，列只取 2-12
5. x_data = np.array(x_data) # 转换成二维数组
6. y_data = raw_data.iloc[:, -1] # 提取最后一列也就是总使用量即可
7. y_data = np.array(y_data)
```

2.2 模型选择与评价

①随机森林回归

对于随机森林来说，决策树的数量变化和影响的范围更大，因此，我先根据经验设定最大深度 $\text{max_depth}=10$ ，最小划分样本数 $\text{min_samples_split}=5$ ，然后用验证曲线来测试模型在不同决策树数量下的不同表现，并可视化出来。如图 1 所示。经过综合考虑，我发现当决策树数量为 130 以上时，训练集分数基本保持在 0.968 而测试集分数却开始下降，说明模型存在过拟合现象了，因此我选择 120 作为比较理想的决策树数量值。

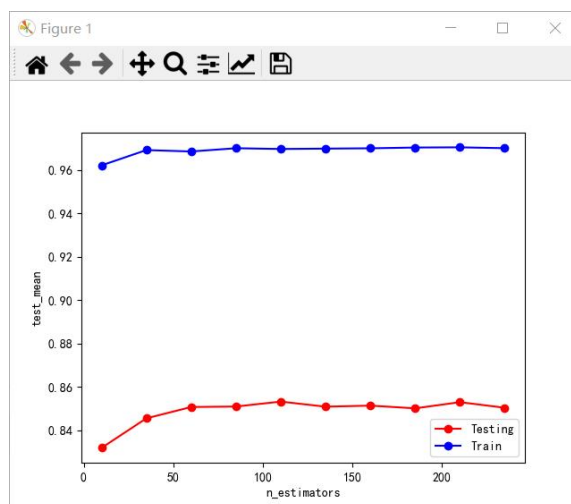


图1 随机森林回归不同决策树数量下的分数

在此基础上，再使用网格搜索法，来确定随机森林回归模型当中另外两个比较常用和重要的参数：最大深度和最小分割样本数。我设置的最大深度可选参数为'max_depth':[7,8,9,10]，最小分割样本数为'min_samples_split':[2,5,8]，经过网格搜索得到了其最优参数{'max_depth': 9, 'min_samples_split': 5, 'n_estimators': 120}和最佳得分：

```
Run: 实验一：模型选择 ×
D:\softwares\anaconda3\python.exe D:\softwares_saving\pycharm\deep_study_1\深度学习大作业\实验一
model_tree_best_score_: 0.9116160737429262
model_tree_best_params_: {'max_depth': 9, 'min_samples_split': 5, 'n_estimators': 120}
Process finished with exit code 0
```

图2 随机森林回归最优参数和最佳得分

②lasso 回归

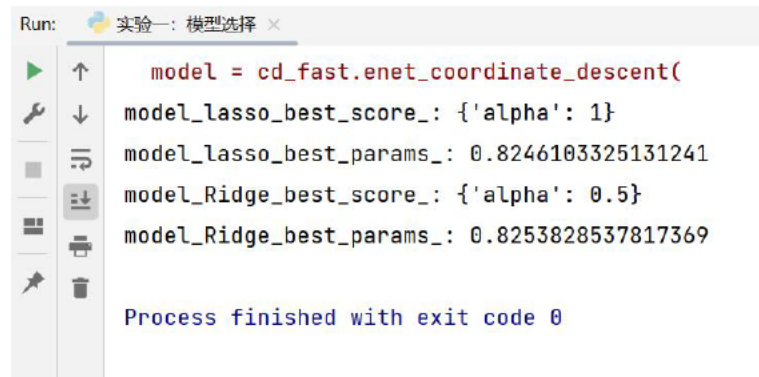
同样利用网格搜索法，alpha 正则项系数选择 [1, 0.5, 0.1, 0.01, 0.0001]，观察模型的表现。训练之后可发现最优参数{'alpha': 1}和最佳得分如下图所示，可以发现 L1 正则化方式较随机森林相比得分差距比较大，不适合本问题进行精度计算。

```
Run: 实验一：模型选择 ×
model = cd_fast.enet_coordinate_descent(
model_lasso_best_score_: {'alpha': 1}
model_lasso_best_params_: 0.8246103325131241
Process finished with exit code 0
```

图3 lasso 回归最优参数和最佳得分

③岭回归

同理可得岭回归的最佳参数{'alpha': 0.5}和最佳得分，可以发现岭回归的得分与 lasso 回归的得分很接近，并且都与随机森林相差较大。



```

Run: 实验一: 模型选择 x
model = cd_fast.enet_coordinate_descent(
model_lasso_best_score_: {'alpha': 1}
model_lasso_best_params_: 0.8246103325131241
model_Ridge_best_score_: {'alpha': 0.5}
model_Ridge_best_params_: 0.8253828537817369

Process finished with exit code 0

```

图4 岭回归最优参数和最佳得分

2.3 打印 R2 分数，可视化特征重要性

在程序里调用 `sklearn.metrics` 下面的 `r2_score` 函数，即可得到对应的 R2 分数。同时为了更加直观地体现各个特征所起的作用，这里通过 `model.feature_importances_` 获取特征的重要性，并且通过 `matplotlib` 可视化出来。在可视化的时候为了显示直观，需要将重要性先进行排序，这样最后就能按大小顺序显示了。相关代码如下：

```

1. # 打印R2 分数
2. r2 = sm.r2_score(test_y, pred_test_y)
3. print("R2:", r2)
4. # 可视化特征重要性
5. fi = model.feature_importances_
6. print("特征重要性: ", fi) # fi 的数据类型是 array
7. plt.figure(figsize=(10, 6), dpi=100)
8. plt.title("特征重要性")
9. plt.xlabel("特征")
10. plt.ylabel("重要性")
11. x = np.arange(fi.size)
12. print(x)
13.
14. # bar 的 x 是数量
15. sorted_idx = fi.argsort()[::-1] # 按照排序返回，下标默认是按从小到大排序
16. # 返回最大值所对应的下标，返回第二个大值对应的下标
17. print(sorted_idx)
18. # 特征重要性按照sorted_idx 顺序排列
19. fi = fi[sorted_idx]
20. print("new:fi", fi)
21.
22. temp_header = []
23. # 读取CSV 文件的表头
24. df_header = pd.read_csv("bike_day.csv", nrows=1)
25. # 打印表头信息
26. header_inf = df_header.columns.tolist()[2:13] # 提取表头标签
27. for i in sorted_idx: # 对header_inf 按照重要性重新排序
28.     temp_header.append(header_inf[i])
29. print("header:", temp_header)

```



```

30. plt.bar(temp_header, fi, width=0.5, color="c", label="feature importance")
31. for a, b in zip(temp_header, fi):
32.     plt.text(x=a, y=b, s=round(b, 4), fontsize=14, ha="center", va="bottom")
33. plt.legend()
34. plt.show()

```

三、实验结果分析

①模型的评估

通过对随机森林、lasso 回归、岭回归三种模型进行训练，并用测试集进行测试，比较它们三者最后的得分。最优模型和最优参数如下表 1 所列：

模型	最优参数	最优得分
随机森林回归	{ <code>'max_depth':10,</code> <code>'min_samples_split':2,</code> <code>'n_estimators': 120</code> }	0.9156
lasso 回归	{ <code>'alpha': 1</code> }	0.8246
岭回归	{ <code>'alpha': 0.5</code> }	0.8253

表 1 三种模型最优模型和最优参数对比

可以发现随机森林效果明显优于 lasso 回归和岭回归。因此在后续的基于天和基于小时的数据训练与预测中，将会采取随机森林作为模型来训练。

对于随机森林回归模型，有几个关键的超参数会对模型的性能产生显著影响。

①树的数量：我发现增加树的数量可以提高模型的性能，因为随机森林是通过多个决策树的集成来进行预测的。但过多的树可能导致过拟合和增加计算成本。

②每棵树的最大深度：我发现控制每棵树的最大深度有助于防止过拟合。较小的深度通常可以提高模型的泛化性能，但也可能减少模型的表达能力。

③最小分割样本数：控制在进行分割之前节点必须具有的最小样本数。较大的值可以防止模型过度拟合，但也可能导致模型欠拟合。

对于 Lasso 回归模型，主要的超参数是正则化参数 α 。Lasso 回归使用 L1 正则化来惩罚模型的复杂性，而 α 控制了正则化的强度。 α 的增加会增强正则化的效果，促使模型更多地将系数稀疏化，即让一些特征的系数趋近于零。这有助于特征选择，排除对目标变量影响较小的特征。

对于岭回归模型，它的主要超参数也是正则化参数 α （也称为岭系数）。这个参数控制了模型复杂性和正则化的强度。岭回归通过 α 的增加来加强正则化的效果。增大 α 将导致模型的系数向零收缩，减小过拟合的风险。岭回归相对于 Lasso 回归更倾向于对所有特征进行缩放，而不是稀疏化系数。

②基于天的数据训练与预测

(1) 对数据进行预处理后，把 11 个特征当作输入部分，3 个标签作为输出部分。使用随机森林模型，对参数进行初始化的设置分别为：`max_depth=10`，`n_estimators=200`，`min_samples_split=5`，所得到的模型结果来进行预测，所得到的 R2 分数如下图所示：

```
Run: 实验一：对共享单车使用量进行预测
D:\softwares\anaconda3\python.exe D:\softwares_saving\pycharm\d
R2: 0.9155696106237983
特征重要性: [0.06445674 0.2853126 0.027174 0.00364397 0.01339
0.0189978 0.3602524 0.12728318 0.06271621 0.03125284]
[ 0 1 2 3 4 5 6 7 8 9 10]
[ 7 1 8 0 9 10 2 6 4 5 3]
new:fi [0.3602524 0.2853126 0.12728318 0.06445674 0.06271621
```

图 5 基于天数据训练的 R2 分数

对特征重要性进行可视化，可视化的柱状图如下图所示。可以发现在基于天的数据训练与预测当中，温度是最重要的特征，这可能意味着温度对于共享单车使用量有着较大的影响。这与我们的常识相符，因为温度通常与人们对户外活动的偏好相关，可能会影响骑行共享单车的意愿。年份是第二重要的特征，这可能表明共享单车使用量可能受到年份的季节性或趋势性影响。可能的情况包括共享单车系统逐年增长，或者不同年份对于共享单车使用有不同的趋势。并且该天是否是假期是最不重要的特征。这可能表示假期与共享单车使用量之间的关系相对较弱，或者假期并不是决定性的因素。

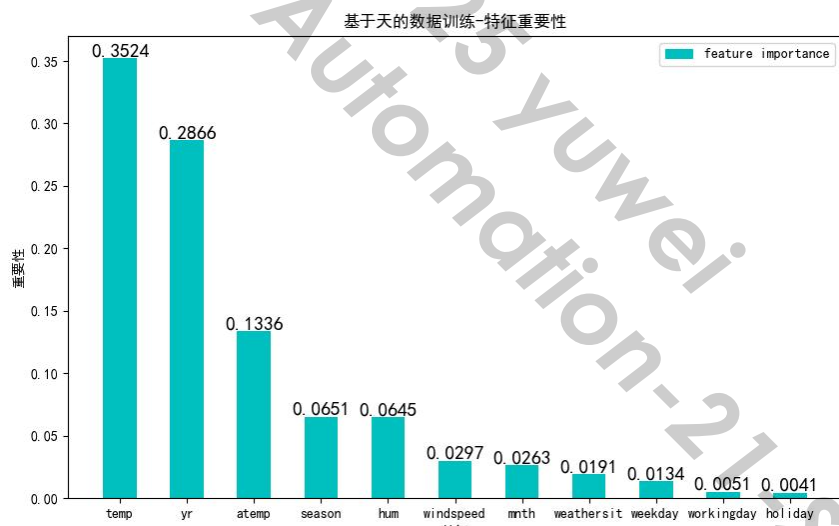


图 6 基于天数据训练的特征重要性

③基于小时的数据训练与预测

(1) 对数据进行预处理后，把 12 个特征当作输入部分，3 个标签作为输出部分。使用随机森林模型，对参数进行初始化的设置分别为：max_depth=10, n_estimators=200, min_samples_split=5，所得到的模型结果来进行预测，所得到的 R2 分数如下图所示：

```
Run: 实验一：基于天的数据训练与预测 实验一：基于小时的数据训练与预测
D:\softwares\anaconda3\python.exe D:\softwares_savi
R2: 0.9196359392001192
特征重要性: [0.02027461 0.08588113 0.00928726 0.64680
0.05769708 0.01676866 0.12140873 0.01409577 0.0153
[ 0 1 2 3 4 5 6 7 8 9 10 11]
[ 3 8 1 6 0 7 10 9 2 5 11 4]
new:fi [0.64680164 0.12140873 0.08588113 0.05769708
```

图 7 基于小时数据训练的 R2 分数

对特征重要性进行可视化，可视化的柱状图如下图所示。结果显示，小时是这 12 个特征中最重要的特征，其重要性程度高达 0.6468。这意味着在基于小时数据的训练中，时间的变化对于共享单车使用量有着很大的影响。这可能反映了每天不同小时的交通模式、人们的工作和休息习惯等因素。尽管在基于天数据训练的模型中，温度是最重要的特征，但在基于小时的数据模型训练中，其重要性程度下降到第二位，为 0.1214。这可能表示在小时级别的分析中，其他与时间相关的特征更为重要。与基于天数据训练的模型相似，结果显示是否是假期是这 12 个特征中最不重要的。这可能表明在基于小时的数据中，假期对于共享单车使用量的影响较小。

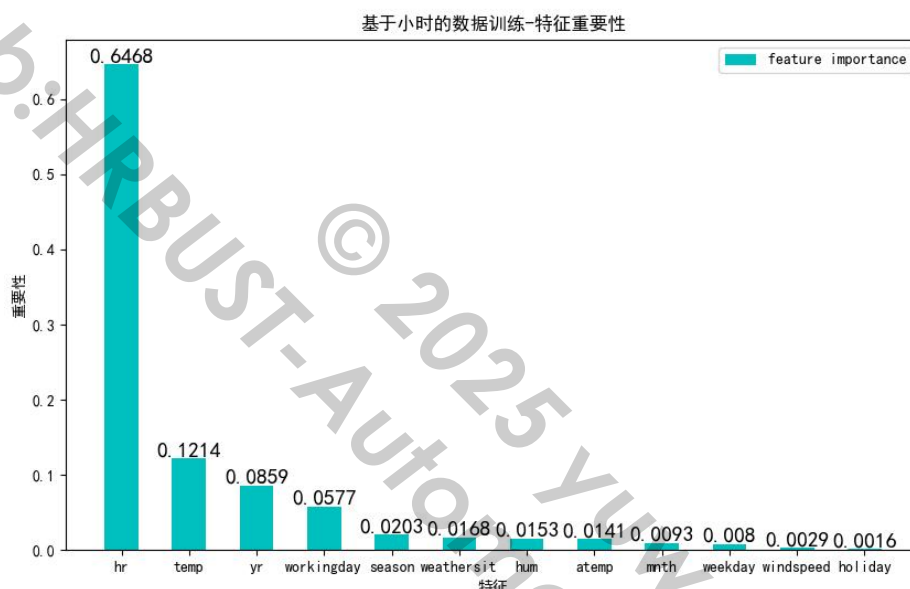


图 8 基于小时的数据训练的特征重要性

④对共享单车使用量进行预测

首先从基于天的数据集中的测试集里随机选取 50 个样本来对其进行预测，并且将共享单车使用量的预测值与真实值画在同一张图里对比观察。

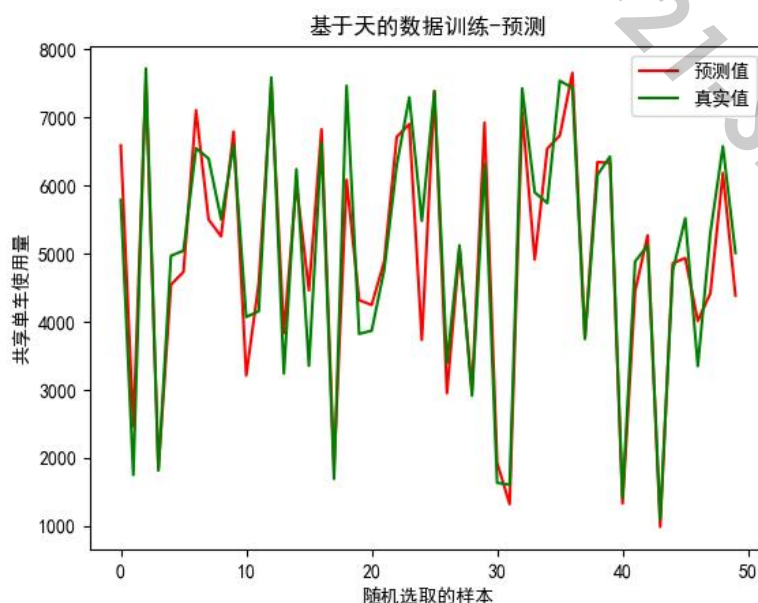


图 9 基于天的数据对共享单车使用量进行预测

然后从基于小时的数据集中的测试集里同样随机选取 50 个样本来对其进行预测，并且画在同一张图里对比观察。可以发现，曲线大体能很好拟合真实曲线。

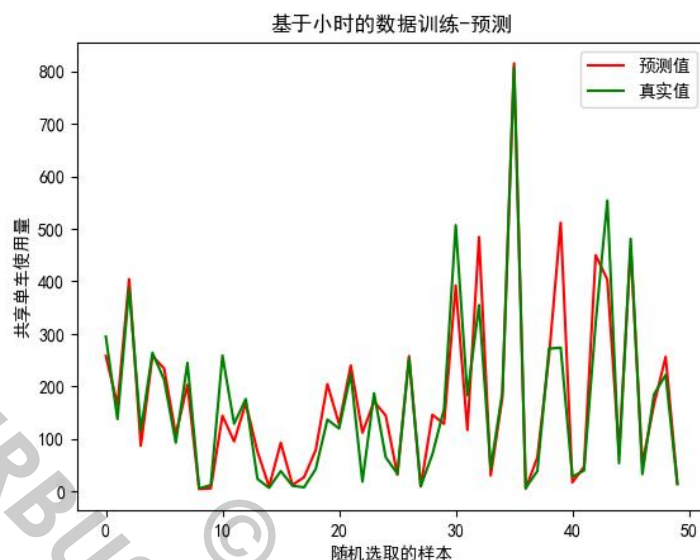


图 10 基于小时的数据对共享单车使用量进行预测

两者的趋势情况大致一样，在随机选取的 50 个测试集样本上，预测值的曲线大致与真实值曲线相吻合，这表明模型能够捕捉到共享单车使用量的整体趋势。这是一个积极的信号，说明模型在一般情况下是有效的。但在真实值突然变化较大的时候，预测值存在较大的误差。这可能是因为模型在处理突然的变化时，需要更多的时间来适应新的趋势。在这种情况下，模型可能会滞后于真实值的变化，导致出现误差。后续可以考虑增加模型的复杂度，或者尝试其他更灵活的模型，以更好地捕捉突然变化的趋势。

总体而言，模型的表现对于整体趋势是令人满意的，但在处理突然变化时存在一些挑战。通过进一步调整模型超参数、增加特征、使用更复杂的模型或者考虑其他改进方法，可能能够提高模型对于变化的适应能力，减小误差。

四、实验总结

在进行共享单车使用量预测的实验中，最开始我讨论了三种模型在这个问题下的不同表现，最后发现随机森林回归模型表现的最出色。然后我对随机森林回归模型在基于天和基于小时的数据集上进行了训练和预测，并进行了特征重要性分析。

首先，对于基于天的数据集，温度和年份被认为是最重要的特征，而假期对于预测共享单车使用量的影响较小。这表明天气和时间趋势是影响共享单车使用的主要因素。在基于小时的数据集中，模型认为小时是最重要的特征，其次是温度。这强调了对于每天不同时间段的小时级别分析的重要性。

在特征重要性分析中，我观察到随机森林模型能够捕捉到整体趋势，但在真实值突然变化较大时存在误差。这可能暗示着模型对于突发事件或异常情况的适应性较差。在这种情况下，感觉以后可以尝试增加模型的复杂度，改进特征工程，或者考虑其他更灵活的模型，以提高模型的性能。综合而言，这个实验为理解共享单车使用量预测提供了有益的见解，并指导了进一步改进模型和优化业务策略的方向。我意识到，在机器学习和数据科学领域，实验的结果常常是一个循环迭代的过程，我们需要耐下性子，通过不断调整和改进，逐步提升模型的性能。

实验一附录源码：

```

1. import numpy as np
2. import matplotlib.pyplot as plt
3. import pandas as pd
4. import sklearn.utils as su # 通用功能模块，主要用到随机化的函数，用来打乱数据集
5. import sklearn.ensemble as se # 集成学习的模块
6. import sklearn.metrics as sm # 用来性能评价
7. import sklearn.tree as st # 决策树所在的模块
8. import sklearn.model_selection as ms
9. import sklearn.linear_model as lm
10.
11. plt.rcParams["font.sans-serif"] = "SimHei" # 显示中文
12. # PART1. 对共享单车基于天的数据训练与预测（读取bike_day数据集）
13. # 数据预处理（数据打乱顺序和数据集的划分）
14. raw_data = [] # 用来临时保存样本列表
15. raw_data = pd.read_csv("bike_day.csv")
16. # print(raw_data)
17. # 只提取有用的特征，对数据集切片，使用iloc函数
18. x_data = raw_data.iloc[:, 2:13] # 把有用的特征即从 season-windspeed, 行全取，列只取 2-12
19. x_data = np.array(x_data) # 转换成二维数组
20. # print(x_data)
21. y_data = raw_data.iloc[:, -1] # 提取最后一列也就是总使用量即可
22. y_data = np.array(y_data)
23. # 样本随机化处理（消除样本顺序对模型的影响）
24. x_data, y_data = su.shuffle(
25.     x_data, # 样本特征
26.     y_data, # 样本标签
27.     random_state=7 # 随机种子
28. )
29.
30. # 样本的划分，划分训练集（80%）和测试集（20%）
31. # split 分割
32. train_size = int(len(x_data) * 0.8) # all_size = 731, train_size = 584
33. train_x = x_data[:train_size] # 训练集的输入部分
34. test_x = x_data[train_size:] # 测试集的输入部分
35. train_y = y_data[:train_size] # 训练集的输出部分
36. test_y = y_data[train_size:] # 测试集的输出部分
37.
38. # 模型试验：随机森林
39. # 先用验证曲线找到最该模型的最优参数，可知n_estimators=250
40. model = se.RandomForestRegressor(
41.     max_depth=10,
42.     n_estimators=200,

```



```
43.     min_samples_split=5 # 最小划分样本, 防止过拟合
44. )
45. model.fit(train_x, train_y) # 训练模型
46. pred_test_y = model.predict(test_x) # 使用测试集来预测
47. plt.title("基于天的数据训练-预测")
48. plt.xlabel("随机选取的样本")
49. plt.ylabel("共享单车使用量")
50. plt.plot(range(0,50),pred_test_y[0:50],color="r",label="预测值")
51. plt.plot(range(0,50),test_y[0:50],color="g",label="真实值")
52. plt.legend()
53. # 打印R2 分数
54. r2 = sm.r2_score(test_y, pred_test_y)
55. print("R2:",r2)
56. # 可视化特征重要性
57. fi = model.feature_importances_
58. print("特征重要性: ", fi) # fi的数据类型是array
59. plt.figure(figsize=(10, 6), dpi=100)
60. plt.title("基于天的数据训练-特征重要性")
61. plt.xlabel("特征")
62. plt.ylabel("重要性")
63. x = np.arange(fi.size)
64. print(x)
65. # bar 的x 是数量
66. sorted_idx = fi.argsort()[::-1] # 按照排序返回, 下标默认是按从小到大排序
67. # 返回最大值所对应的下标, 返回第二个大值对应的下标
68. print(sorted_idx)
69. # 特征重要性按照sorted_idx 顺序排列
70. fi = fi[sorted_idx]
71. print("new:fi", fi)
72.
73. temp_header = []
74. # 读取CSV 文件的表头
75. df_header = pd.read_csv("bike_day.csv", nrows=1)
76. # 打印表头信息
77. header_inf = df_header.columns.tolist()[2:13] # 提取表头标签
78. print("header_inf:",header_inf)
79. for i in sorted_idx: # 对header_inf 按照重要性重新排序
80.     temp_header.append(header_inf[i])
81. print("header:", temp_header)
82. plt.bar(temp_header, fi, width=0.5, color="c", label="feature importance")
83. for a, b in zip(temp_header, fi):
84.     plt.text(x=a, y=b, s=round(b, 4), fontsize=14, ha="center", va="bottom")
85. plt.legend()
86. plt.show()
```

实验一附录源码：

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. import pandas as pd
4. import sklearn.utils as su # 通用功能模块，主要用到随机化的函数，用来打乱数据集
5. import sklearn.ensemble as se # 集成学习的模块
6. import sklearn.metrics as sm # 用来性能评价
7. import sklearn.tree as st # 决策树所在的模块
8. import sklearn.model_selection as ms
9. import sklearn.linear_model as lm
10.
11. plt.rcParams["font.sans-serif"] = "SimHei" # 显示中文
12. # PART2. 对共享单车基于小时的数据训练与预测（读取bike_hour数据集）
13. raw_data = [] # 用来临时保存样本列表
14. raw_data = pd.read_csv("bike_hour.csv")# 只提取有用的特征，对数据集切片
15. x_data = raw_data.iloc[:, 2:14] # 把有用的特征即从 season-windspeed,
16. x_data = np.array(x_data) # 转换成二维数组
17. y_data = raw_data.iloc[:, -1] # 提取最后一列也就是总使用量即可
18. y_data = np.array(y_data)
19. # 打乱数据
20. # 样本随机化处理（消除样本顺序对模型的影响）
21. x_data, y_data = su.shuffle(
22.     x_data, # 样本特征
23.     y_data, # 样本标签
24.     random_state=7 # 随机种子
25. )
26.
27. # 样本的划分，划分训练集（80%）和测试集（20%）
28. # split 分割
29. train_size = int(len(x_data) * 0.8) # all_size = 731, train_size = 584
30. train_x = x_data[:train_size] # 训练集的输入部分
31. test_x = x_data[train_size:] # 测试集的输入部分
32. train_y = y_data[:train_size] # 训练集的输出部分
33. test_y = y_data[train_size:] # 测试集的输出部分
34.
35. # 模型试验：随机森林
36. # 先用验证曲线找到最该模型的最优参数，可知n_estimators=250
37. model = se.RandomForestRegressor(
38.     max_depth=10,
39.     n_estimators=200,
40.     min_samples_split=5 # 最小划分样本，防止过拟合
41. )
42. model.fit(train_x, train_y) # 训练模型
```

```
43. pred_test_y = model.predict(test_x) # 使用测试集来预测
44. plt.title("基于小时的数据训练-预测")
45. plt.xlabel("随机选取的样本")
46. plt.ylabel("共享单车使用量")
47. plt.plot(range(0,50),pred_test_y[0:50],color="r",label="预测值")
48. plt.plot(range(0,50),test_y[0:50],color="g",label="真实值")
49. plt.legend()
50. # 打印R2 分数
51. r2 = sm.r2_score(test_y, pred_test_y)
52. print("R2:",r2)
53. # 可视化特征重要性
54. fi = model.feature_importances_
55. print("特征重要性:", fi) # fi 的数据类型是 array
56. plt.figure(figsize=(10, 6), dpi=100)
57. plt.title("基于小时的数据训练-特征重要性")
58. plt.xlabel("特征")
59. plt.ylabel("重要性")
60. x = np.arange(fi.size)
61. print(x)
62.
63. # bar 的x 是数量
64. sorted_idx = fi.argsort()[::-1] # 按照排序返回, 下标默认是按从小到大排序
65. # 返回最大值所对应的下标, 返回第二个大值对应的下标
66. print(sorted_idx)
67. # 特征重要性按照 sorted_idx 顺序排列
68. fi = fi[sorted_idx]
69. print("new:fi", fi)
70. temp_header = []
71. # 读取CSV 文件的表头
72. df_header = pd.read_csv("bike_hour.csv", nrows=1)
73. # 打印表头信息
74. header_inf = df_header.columns.tolist()[2:14] # 提取表头标签
75. print("header_inf",header_inf)
76. for i in sorted_idx: # 对header_inf 按照重要性重新排序
77.     temp_header.append(header_inf[i])
78. print("hedar:", temp_header)
79. plt.bar(temp_header, fi, width=0.5, color="c", label="feature importance")
80. for a, b in zip(temp_header, fi):
81.     plt.text(x=a, y=b, s=round(b, 4), fontsize=14, ha="center", va="bottom")
82. plt.legend()
83. plt.show()
```