机器学习导论大作业报告

面向电信行业用户的智能套餐个性化匹配

万之凡 2016K8009929049
李金洋 2016K8009929033
熊可欣 2016K8009929021
钟　赟 2016K8009915009

一、任务描述

1、赛题描述

电信产业作为国家基础产业之一，覆盖广、用户多，在支撑国家建设和发展方面尤为重要。随着互联网技术的快速发展和普及，用户消耗的流量也成井喷态势，近年来，电信运营商推出大量的电信套餐用以满足用户的差异化需求，面对种类繁多的套餐，如何选择最合适的一款对于运营商和用户来说都至关重要，尤其是在电信市场增速放缓，存量用户争夺愈发激烈的大背景下。针对电信套餐的个性化推荐问题，通过数据挖掘技术构建了基于用户消费行为的电信套餐个性化推荐模型，根据用户业务行为画像结果，分析出用户消费习惯及偏好，匹配用户最合适的套餐，提升用户感知，带动用户需求，从而达到用户价值提升的目标。

套餐的个性化推荐，能够在信息过载的环境中帮助用户发现合适套餐，也能将合适套餐信息推送给用户。解决的问题有两个：信息过载问题和用户无目的搜索问题。各种套餐满足了用户有明确目的时的主动查找需求，而个性化推荐能够在用户没有明确目的的时候帮助他们发现感兴趣的新内容。

2、任务分析

此题利用已有的用户属性(如个人基本信息、用户画像信息等)、终端属性(如终端品牌等)、业务属性、消费习惯及偏好匹配用户最合适的套餐，对用户进行推送，完成后续个性化服务，是一个多分类任务。

因此我们在具体实现的过程中需要向处理分类问题的模型上考虑，包括向量机、决策树等。

二、实施方案

在拿到题目之后的基本思路如下:

首先对赛题进行标签化的分析: 数据挖掘、分类预测、回归分析, 根据这些我们能够大致预测需要使用的模型与相关算法。我个人开始较早, 使用 SVM 进行测试, 测试结果不佳的情况下转用决策树。后用随机森林和 LightGBM 进行优化, 最终获得结果。在实现不同模型的过程中我们分别进行各自的探究, 小课题分别有: 对特征选择的思考, 相关系数在这次的任务中是不是一个合适的特征呢? 按照不同 service_type 为什么预测结果会有那么大的差距呢? 这些在我们之后的报告中会有详细描述。

三、具体实现

1、数据预处理

首先对数据进行分析。经查，没有重复的数据。对于\N 数据的，查找发现有 4 个数据，因为数据量足够多、有 70 多万条，所以我们剔除掉这几处\N（但是后来加入的复赛数据并没有\N 类型的数值缺失）；为了方便对套餐进行分类，我们将 11 个类别的套餐分别标记为 1-11 号；为方便后续处理，在查阅资料之后我们选用 python pandas 库函数，将数据读取到 DataFrame 中。

2、数据可视化

为了大致明确各个标签下数据的分布情况，对数据进行可视化。

使用 seaborn 绘制 current_service（标签）与其余特征的箱线图，查看数值分散情况、平均值和是否有过于异常的值。

使用 seaborn 绘制 current_service（标签）与其余特征的关系图，查看 current_service 在其余特征上的分布情况和其余特征自身的分布情况。

计算 current_service（标签）与其余特征的相关系数——相关系数能在某种程度上表明二者之间联系的强弱（但无法作为判断特征重要性的强依据，具体分析见后（Li 的大作业反思部分））。绘制了相关系数热度图，让相关性更加直观。

可以根据上述这些可视化考虑是否弃用特征，或在后续的特征工程中做其它的组合处理。

3、数据分析

不论哪个模型，都会涉及到特征的选取问题，因此需要对各个特征进行分析，剔除贡献不大的特征，并对已有特征之间进行组合。

由于对电信领域知识有限，我们采用的方法是假定所有的特征都可能会帮助分类，再对有足够把握去除的特征进行剔除，类似假设检验的思想。上一步得到的数据分布可视化作图可以帮助判断。此外由于数据已经变成数值，我们采用特征和套餐类别之间的相关性来辅助判断。相关性可以用相关系数来衡量，进一步我们对相关系数矩阵制作热力图，以方便对特征进行选择。判断上，可以先查看特征与最终套餐类别的相关程度，相关程度高的优先保留；也可以查看特征之间的相关程度，相关程度高的可以考虑进行选择性保留或者进行组合。

4、超参选择与特征工程

在特征选择部分：首先进行一批特征的删除。不论从理解上去分析，还是从特征统计来分析，“is_mix_service”对于结果的判断贡献不大，可以舍弃。“complaint_level”“former_complaint_num”“former_complaint_fee”“net_service”“pay_times”特征在热力图上和套餐类别的相关系数非常低，和其他特征之间的相关性也不大，再结合对这些特征在日常生活中的理解，决定进行剔除。

我们发现“total_fee”系列数据和套餐类别的相关系数较高，而且他们彼此之间具有很高的相关性，因此我们对他们进行选择和重组。基于相关系数，我们最终选择四列数据的 min 和 mean 作为特征，并去除原有的四列数据。

```
#2个连续float
#total_fee系列，pattern相似
#all_data['1_total_fee'].dtypes#float64
#此处对total_fee系列数据进行基于相关系数的提取与重组
all_data.iloc[:,2:6]#确认目标数据
print(all_data.dropna()['1_total_fee'].corr(all_data.dropna()['current_service']))
print(all_data.dropna()['2_total_fee'].corr(all_data.dropna()['current_service']))
print(all_data.dropna()['3_total_fee'].corr(all_data.dropna()['current_service']))
print(all_data.dropna()['4_total_fee'].corr(all_data.dropna()['current_service']))
print(all_data.iloc[:,2:6].mean(axis=1).corr(all_data.dropna()['current_service']))
print(all_data.iloc[:,2:6].std(axis=1).corr(all_data.dropna()['current_service']))
print(all_data.iloc[:,2:6].max(axis=1).corr(all_data.dropna()['current_service']))
print(all_data.iloc[:,2:6].min(axis=1).corr(all_data.dropna()['current_service']))
```

```
0.5152339722718543
0.497141808554911
0.5094022427430654
0.5000643678789383
0.5462751177169615
0.21413992546539737
0.4703718595573495
0.5928537502350928
```

尽管看上去组合后的特征在相关关系上有了较好的提升，不过相关性是否可以作为一个

优良标准存疑，因为相关系数对套餐类型的序关系做了一个假定，而实际上这个序关系是不存在的。详细可以看后面的讨论部分。

在基础的筛选之后，对特征进行组合，在特征分析中，我们发现在 4 中没有进行处理的 traffic 系列,Caller_time 系列和 Pay 系列特征,与 current_service 的相关性整体较小。于是做了下面这些尝试：

traffic 系列（包含 month traffic, last_month_traffic, local_trafffic_month）

month traffic, last_month_traffic 求和，形成 traffic_month_sum；

找出最大值，整合成 traffic_month_max

删除 month traffic, last_month_traffic

Caller_time 系列（包含 service1_caller_time, service2_caller_time,）：

求最大值，形成特征 call_time_max

求最小值，形成特征 call_time_min

求和，形成特征 call_time_local

Pay 系列（包含 pay_times, pay_num）：

相除计算出 pay_num_pertimes 这一新特征

删除 pay_times 这一原特征

事实上这些尝试基于后续的训练结果，根据训练的效果，特征工程会不断地进行调整。

5、模型设计与选择

四个组员分别对实现的模型进行讲解，并对其中的思考进行阐释。

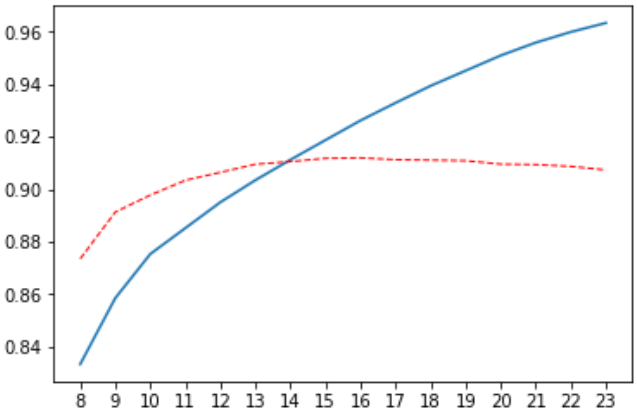
1 SVM

根据 csdn 上的博文进行模仿实现，替换其中的一部分输入，将我们的测试数据进行比对，主要按照 total_fee 系列数据为主要的参考数据进行学习分析。最终测试预测的准确率约为 63%，获得的 f1_score 为 0.502。经过大量的调参，修改函数之后还是没有取得很好的效果，采用前 60 万行数据学习预测后 10 万行的准确率没有超过 68%，因此判断出 SVM 并不是一个很适合这个问题的解决方法。这个是由其原理决定的，它适合的是线性程度较高的回归学习，但是这次任务并不适合简单的线性回归，原因在最后的反思中有提到，这次的套餐类型、用户画像并不是一个稳定的数值，而是离散型关联程度较低的字符变量，因此相关系数等并不能作为一个很好的解决方法。我们也因为这次失败将关注点转移到了决策树上面去。

2 决策树

首先，选择 sklearn 库中的 tree.DecisionTreeClassifier 进行决策树上的尝试，在初赛训练集上进行 train_test_split 并进行训练与预测，设置深度为 8，达到了 86%的预测准确率，f1_score 达到 0.832。可以看出，决策树模型在这个问题上表现优秀，在一开始就达到较好的成果。逐渐提高深度，可以看出 f1_score 呈现先上升后下降的趋势，并且在训练数据和测试数据上出现较大差异，呈现过拟合的现象。这一部分在汇报进行讲解的时候放错了图，下图中分别展示出测试集和训练集中预测趋势的不同，这才是过拟合应有的表现。

	depth	train	tesst	f1_train	f1_test
0	8	0.842824	0.883325	0.833314	0.873439
1	9	0.861754	0.895592	0.858464	0.891260
2	10	0.875752	0.900410	0.875288	0.897734
3	11	0.885475	0.905825	0.885156	0.903354
4	12	0.895377	0.908321	0.895018	0.906403
5	13	0.903750	0.910937	0.903426	0.909468
6	14	0.911294	0.911916	0.911210	0.910502
7	15	0.918847	0.913128	0.918700	0.911732
8	16	0.926308	0.913107	0.926162	0.911922
9	17	0.933023	0.912487	0.932879	0.911211
10	18	0.939508	0.911994	0.939364	0.911040
11	19	0.945293	0.911964	0.945140	0.910828
12	20	0.951072	0.910753	0.950923	0.909523
13	21	0.955979	0.910368	0.955841	0.909339
14	22	0.960049	0.909802	0.959887	0.908673
15	23	0.963485	0.908449	0.963323	0.907312



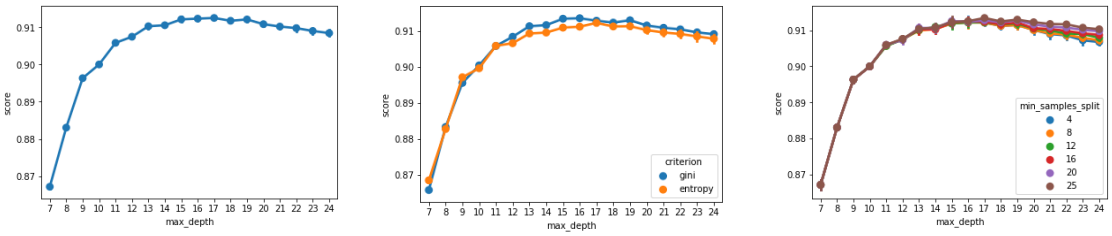
得分趋势图

（蓝线表示训练集得分，红线表示验证集得分）

Depth=16 达到最优

(max_depth = i, criterion = 'gini', splitter = 'b' ,min_samples_split = 8)

为了增强优度，采用 GridSearchCV 进行模型的重新建立。这里不再采用 train_test_split，而是选用初赛 train_all 作为已知数据进行训练、复赛 train_all 作为验证集、不进行训练。



首先，确定可变参数为最大树深度 max_depth、最小节点分割条件 min_samples_split、信息增益标准 criterion，结果如下：

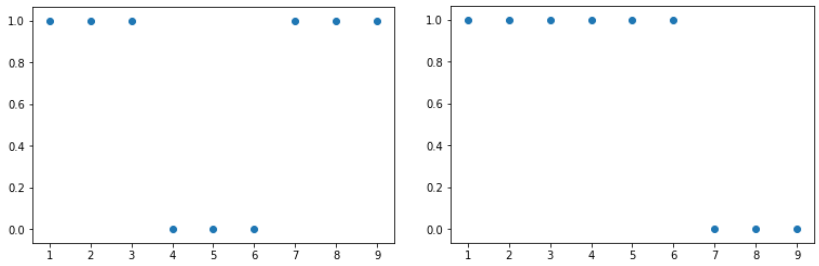
根据图，细化 max_depth 区间为 13~17、min_samples_split 区间为 20~40，由于 criterion 中 gini 指数普遍好于 entropy，采用 gini 参数。此时再加入_samples_leaf 可变参数，进行新的 GridSearchCV 训练，进一步细化区间。最终确定最佳参数为{'max_depth': 22, 'min_samples_leaf': 18, 'min_samples_split': 65}。得分 f1_score: 0.91770295。可以看出这里找到的最佳参数是局部最优值，通过类似梯度下降的想法进行搜索，找到了一个足够好的分类模型。

在具体实现的过程中有如下思考：

在前文中我们提到了用相关系数作为特征的参考标准，做出了很漂亮的热力图。参考正式比赛其他组的经验总结，也有很多组都采用了热力图。然而，相关系数真的能作为相关关系的良好参考依据吗？我觉得不能。因为在我们的问题中，预测套餐类别是一个多分类的问题，这些分类之间是没有序关系的，然而在相关系数的计算过程中，对套餐类别或多或少做了一个序关系的假设。换言之，因为计算的是线性相关系数，结果高说明特征和套餐类别线性相关性强，但这不一定意味着相关。

反例轻而易举。比如对于 x=1,2,3,4,5,6,7,8,9 y=1,1,1,0,0,0,1,1,1，计算得相关

系数为 4.0539612963254114 e-17。数据分布如下图左 1 所示。然而，如果对 x 的顺序进行调整，则数据分布如图左 2 所示，相关系数绝对值变为 0.821584，提升了 11 个数量级。所以相关系数本身作为参考的依据是有限的。



那么能用什么作为更好的判据吗？也许可以通过调整套餐与序号的序关系，但序关系本身如何确定却难以找到。或者使用别的标准，比如用特征的信息增益来确定特征的重要性，而这恰好就回到了决策树的思想中来。

3 随机森林

（1）模型原理

随机森林由 Leo Breiman（2001）提出的一种分类算法，它通过自助法（bootstrap）重采样技术，从原始训练样本集 N 中有放回地重复随机抽取 n 个样本生成新的训练样本集合训练决策树，然后按以上步骤生成 m 棵决策树组成随机森林，新数据的分类结果按分类树投票多少形成的分数而定。

大致过程如下：

1）从样本集中有放回随机采样选出 n 个样本；

2）从所有特征中随机选择 k 个特征，对选出的样本利用这些特征建立决策树（一般是 CART，也可是别的或混合）；

3）重复以上两步 m 次，即生成 m 棵决策树，形成随机森林；

4）对于新数据，经过每棵树决策，最后投票确认分到哪一类。

（2）实现过程：

选择 sklearn 库中的 RandomForestClassifier 进行随机森林的尝试，在初赛训练集上进行 train_test_split 并进行训练与预测。数将前 600000 行作为训练数据训练随机森林模型，600000 行之后作为测试数据进行测试，得到的结果（准确率）为 92.6827235602967 %。若按照 service_type 分开训练、检测，那么 service_type1 的准确率为 99.2409 %，service_type4 的准确率为 86.7679%。

可以看出，随机森林模型在这个问题上的表现优总体优于决策树，在一开始就达到很好的成果。分析原因，应该是每棵树都选择部分样本及部分特征，一定程度避免了过拟合现象；同时，每棵树随机选择样本并随机选择特征，使得具有很好的抗噪能力，性能稳定。

使用复赛数据作为测试数据后，得到的结果是：

Accuracy of pure RandomForest classifier: 92.10767239193392 %

F1_score： 0.918518079023627

但很快就发现了新的问题，若按照 service_type 分开训练、检测，那么 service_type1 的准确率有 99.19785988857603 %，得分 F1 Score 是 0.9919613701846962；而 service_type4 的准确率只有 77.97543279022403 %，得分 F1 Score 是 0.7719350752865586，可以看到效果变差。

为了分析效果变差的原因，随后对 service_type4 这一部分的数据进行调参训练。

设置参数：对于最小叶子节点的参数取值，从 1 到 500 每隔 3 取 1 个；对于决策树个数

的参数取值，从 1 到 1000 每隔 5 取一个。由于随机森林的模型训练和预测都很慢，再加上机器限制，所以这里实际上只跑完了 60 多组，用了大约 20h。

模型遍历所有参数组合，并分别输出当前组合的准确率和 f1_score。在遍历结束后后输出 f1_score 最高的参数组合。目前得到的最高准确率是 0.7823956211812627。

训练过程中，可以看到，不同参数组合的得分并没有过大的波动，始终在 0.77 至 0.79 之间。所以之前“效果下降的原因是过拟合”的猜测不成立。我们猜想，service_type4 的预测效果不尽如人意的原因可能与该种类本身的特性有关。

在前文中提到的通过遍历不同的参数组合来观察随机森林模型预测结果的变化，一定程度上排除了参数的特殊性影响模型效果的可能性。但是能完全排除这样的可能性吗？我觉得不一定。随机森林模型的一个缺点就是参数较复杂，模型训练和预测都比较慢。所以在我们有限的时间内，能够遍历的参数组合是很少的。所以还是不能排除其余参数和现处理参数的特殊性对实验造成影响的可能。

4 LightGBM

首先使用 sklearn 包对测试集和训练集进行划分，使用 lightgbm 库中的 LGBMClassifier 分类器搭建 LightGBM 模型。

构建参数词典：

```
lgb_params = {
    'boosting_type': 'gbdt',
    'objective': 'multiclass',
    'silent':0,
    'learning_rate': 0.05,
    'num_leaves': 50,           # should < 2^(max_depth)
    'max_depth': -1,            # no limit
    'min_child_samples': 15,    # Minimum number of data need in a child(min_data_in_leaf)
    'max_bin': 200,             # Number of bucketed bin for feature values
    'subsample': 0.8,           # Subsample ratio of the training instance.
    'subsample_freq': 1,        # frequence of subsample, <=0 means no enable
    'colsample_bytree': 0.5,    # Subsample ratio of columns when constructing each tree.
    'min_child_weight': 0,      # Minimum sum of instance weight(hessian) needed in a child(leaf)
    'subsample_for_bin': 200000,   # Number of samples for constructing bin
    'min_split_gain': 0,        # lambda_l1, lambda_l2 and min_gain_to_split to regularization
    'reg_alpha': 2.0,          # L1 regularization term on weights
    'reg_lambda': 1.0,          # L2 regularization term on weights
    'verbose': 0,
}
```

评价函数采用 F1-macro 宏平均：

```
print("F1_score", f1_score(test_data[:,21], pred, average = 'macro'))
```

采用 5-fold 交叉验证的方式进行训练，即 4 个训练集，1 个测试集：

```
num_round = 10
lgb.cv(param, train_data, num_round, nfold=5)
```

该模型将开始训练，直到验证得分不再提高：

```
lgb.cv(param, train_data, num_round, nfold = 5)
bst = lgb.train(param, train_data, num_round, valid_sets = valid_sets, early_stopping_rounds = 10)
```

最终选择的一组参数如下：

```
clf = lgb.LGBMClassifier(
                objective = 'multiclass',
                boosting_type = 'gbdt',
                num_class = 15,
                lambda_l1 = 0.1,          #权重的L1正则化项
                lambda_l2 = 0.1,          #权重的L2正则化项
                num_leaves = 50,
                max_depth = 6,            #树的最大深度。这个值是用来控制过拟合的
                learning_rate = 0.1,      #每一步迭代步长
                random_state = 2018,
                colsample_bytree = 0.8,   #每棵随机采样的列数的占比(每一列是一个特征)
                subsample = 0.9,          #每棵树随机采样的比例
                n_estimators = 380,       #迭代次数
                n_jobs = 10,
                silent = True
)
```

预测准确率及 F1-weighted score 得分为：

```
Accuracy of LGB Classifier:      92.74208400840244 %
F1_score 0.9249306831900227
```

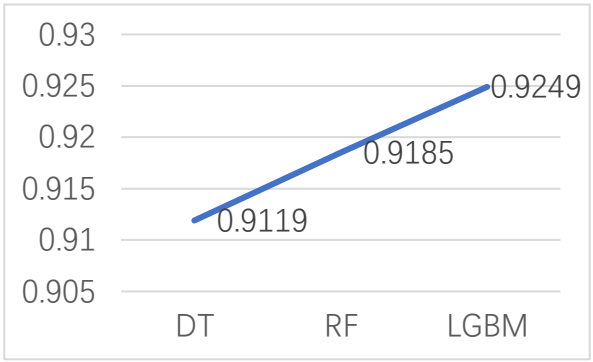
预测准确率及 F1-macro score 得分为：

```
Accuracy of LGB Classifier:      92.74208400840244 %
F1_score 0.8213284672984759
```

四、实验结果与分析

1、模型横向结果比较

SVM 作为一个能够较好解决线性分类问题的成熟模型，一开始作为一个测试性质的模型实现，效果不佳（F1score:0.5026），但是决策树(decision tree)更接近人类认知规律，容易理解和实现，鲁棒性较好，对于这种复杂分类的任务更方便，但容易过拟合，对复杂数据和缺失值不敏感。随机森林(random forests)，数据相关性较高，增强泛化能力，以此一定程度上解决过拟合的问题。LightGBM，在包含 XGBoost 优点的同时进一步提升训练速度，优化训练结果，更适合我们硬件设施较弱的情况。



从结果上来看，SVM 的 f1_score 最低，仅有 0.5026，其余三个的得分依次增长，效果不断提升。虽然几个模型之间的提升效果并不明显，但是整体呈现一个上升的趋势，能够证明 LightGBM 是更加优秀的分类回归问题的解决模型。

2、赛题完成情况

由于模型搭建与训练的时间持续到了元旦之后，无法于训练赛上进行提交，因此为了避免用同一分布的数据集进行训练和验证，我们采用了这样的方法：以初赛训练集为训练数据，以复赛训练集为测试数据，训练的时候不访问测试集，以此来保证模型的泛化能力。尝试和已经提交的组进行比较的时候由于训练集和测试集之间的较大区别也无法实现；最终的实现结果无法和已经在训练赛上提交的组进行比较，这也是我们的一点不足。

五、总结展望

1、万之凡：

在最初选择模型的时候没有及时关注不同模型的特点，在 SVM 上耗费了很多时间，但是

做出来效果并不好，没能达到预期的结果；在实际编写代码的过程中也因为 python 库函数和语法不熟悉走了很多弯路。汇报的过程中没能检查出很基础的知识错误，成为一个相当严重的失误，这一点和我们的实验过程中基本没有涉及过拟合相关，也和临场发挥相关，关于这一部分的改进我们在之前已经提到了，使用新的示意图就能完整表示出来。在汇报之后继续进行的调参和采样工作也并没有取得突破性的提升，因此这些详细的过程不再赘述。

2、李金洋：

主要困难还是不知道有哪些具体的库、不熟悉函数的具体操作，只有在做这个任务的过程中才了解了一个个的课本上的算法如何实现。也许有更好的特征工程中的类似线性相关系数的参考依据、或者更好的库或函数可以使用，还需要我们进一步学习了解。

3、熊可欣：

有什么更好的方法吗？对于现有的设备和时间限制而言，我认为可以尝试放大训练时参数变化的跨度，让其更加分散，这样能进一步排除参数特殊性对模型的影响。随机森林模型本身的训练成本较大，如果要寻求更好的方法，可能就需要使用别的模型了。

4、钟赟：

本实验最大的困难在于学习使用 python 相关函数与库，以及训练参数时对计算机硬件的要求。另外，可能由于缺少经验，使得作为结果很大影响因子的数据预处理部分，做得稍显不足，包括套餐分类随意和数据分析不足，仅凭肉眼和直觉分辨箱线图和相关系数图等来简单处理数据。

六、分工

1、万之凡

赛题分析、数据分析、论文阅读、SVM 实现与调参、Xgboost 实现（未竟）、汇报 PPT 编写与讲解、报告资料汇总与编写

2、李金洋

赛题分析、数据预处理、基本特征提取、模型原理调研、决策树实现与调参、报告与汇报 PPT 资料提供

3、熊可欣

赛题分析、数据可视化、特征组合、模型原理调研、随机森林实现与调参、报告与汇报 PPT 资料提供

4、钟赟

赛题分析、数据分析、LGBM 实现与调参、模型评估、过程得失分析、报告与汇报 PPT 资料提供

七、参考文献

[1]Quinlan J R . Induction on decision tree[J]. Machine Learning, 1986, 1(1):81-106.

[2] Cutler K , Breiman L . Random forests[J]. Machine Learning, 2004, 45(1):157-176.

[3] Schuldt C , Laptev I , Caputo B . Recognizing human actions: a local SVM approach[C]//

Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. IEEE, 2004.

[4] Chen T , Guestrin C . XGBoost: A Scalable Tree Boosting System[J]. 2016.