1. 特征提取 2
1.1 用户评论词频矩阵 2
1.2 用户头像主题 2
1.3 用户名数据 2
1.4 用户评论语气词 2
2. 特征处理部分要点 2
2.1 数据降维 2
2.2 词汇相似度 4
2.2.1 基于 WordNet 的方法 4
2.2.2 基于语料统计的方法 5
2.2.3 基于检索页面数量的方法 5
3. 分类器 5
3.1 Logistic Regression 5
3.1.1 简述 5
3.1.2 参数 6
3.2 SVM 6
3.2.1 简述 6
3.2.2 参数 6
3.3 Random Forest 8
3.3.1 简述 8
3.3.2 参数 9
3.4 Multi-layer Perceptron classifier 10
3.4.1 简述 10
3.4.2 参数 10
4. 分类器部分要点 12
4.1 分类器的评价指标 12
4.2 归一化？标准化？ 18

1. 特征提取

根据提供的点评原始数据集以及已经提取的特征数据集，将用户特征分为四大类分别处理。

1.1 用户评论词频矩阵

用户评论词频矩阵直接使用助教提供的 dianping_user_vec.csv 数据。该数据是基于用户评论词频特征提取的 200 维特征。将 200 维特征数据进一步降维。

1.2 用户头像主题

特征来自用户头像 dianping_avatar_themes.txt 数据。根据该数据集提供的头像主题，构造主题（词）相似度矩阵。将矩阵进行 PCA 降维。

1.3 用户名数据

该特征取自 dianping_username.csv。从该数据集中共提取出三个特征——用户名长度、用户名中使用的特殊符号占比、用户名中使用英文符号占比。

1.4 用户评论语气词

该特征来自 dianping_figuration_vecs.csv 和 dianping_user_words_count.csv。从前者中提取出用户评论使用的 emoj 表情、语气词等总数，从后者提取出评论的长度。得到用户评论使用语气词等的比例特征。

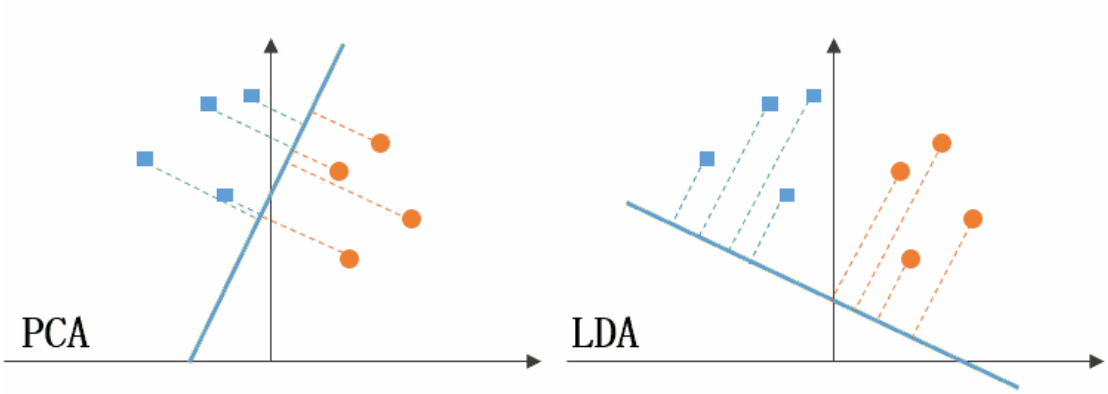
2. 特征处理部分要点

2.1 数据降维

在特征处理过程中，我主要在两个步骤中进行了数据降维。两个步骤分别是 200 维词向量降维，以及头像主题相似度矩阵降维。降维的主要考虑有两点。首先是因为我提取的用户特征来自不同的数据集且维度相差悬殊。如果维度很大的特征不进行降维很容易将会掩盖其他特征的发挥作用。其次，高维数据容易造成维度灾难，大大降低算法的效率。总之所以最好将高维特征降到可以接受的范围不要掩盖其他特征的作用，同时也要尽可能减少降维带来的信息损失。

在实验过程中，我主要考虑的降维方法有两种——PCA 和 LDA。PCA 实际是对数据在高维空间下的一个投影转换。给定原始空间，PCA（无监督）会找到一个到更低维度空间的线性映射。因为需要使所有样本的投影尽可能分开，则需要**最大化投影点的方差**。而 LDA

（有监督，需要提供类别信息）则更多的是考虑了分类标签信息，寻求**投影后不同类别之间数据点距离更大化**以及同一类别数据点距离最小化，即选择分类性能最好的方向，使得数据更易区分。[1]



我选择的降维方法是 PCA 主成分分析法。因为 LDA 降维后最多可生成 C-1 维子空间（分类标签数-1），我做的是一个二分类问题，所以实际只能投影到 1 维。虽然能使数据更容易区分，但是不可避免损失大量信息，而且很容易造成过拟合。

以下是 PCA 降维时的输出。

1）这是头像主题矩阵降维结果（10000+->16)），最终我选择把它降到 8 维已经能保存 90%+的信息

```
[0.30433537 0.24567504 0.1422722  0.10384453 0.03342513 0.03137308
 0.02540474 0.01460213 0.01391349 0.0109829  0.00866998 0.00779173
 0.00571905 0.00554101 0.00440036 0.00425481]
16
```

2）这是词向量降维结果（200->16）

```
loaded success
[0.31429148 0.10922536 0.10470477 0.04821978 0.03911721 0.03408994
 0.02659451 0.021724   0.01615985 0.0136253  0.01197288 0.01030142
 0.00897735 0.00873349 0.00827047 0.00741306]
16
```

能保存大约 80%的信息量。

降维过程中的一些小插曲。首先是在对主题矩阵降维时出现了一个这样的结果。降到 1 维就能保存 100%的信息。振奋人心的消息，又好像有什么地方不对。于是我调试发现，它在载入数据时，把用户 ID 当成特征拿去降维。而 ID 数据都是动辄 7，8 位数，其他特征都是归一化的小于 1 的结果，当然就造成如下让人哭笑不得的结果。

```
[1.00000000e+00 9.22840628e-15 7.17423093e-15 4.48104567e-15
 3.15425593e-15 1.03647283e-15 9.21889429e-16 8.02616314e-16
 4.80626815e-16 4.37323576e-16 3.44446905e-16 2.70194920e-16
 2.43390426e-16 1.81051699e-16 1.77557841e-16 1.44467651e-16]
16
```

另一个问题是，之前提到主题矩阵是一个 10000+的方阵。因为对数据量没什么概念，

[1] https://blog.csdn.net/dongyanwen6036/article/details/78311071

我初期直接使用普通的 PCA 方法，将这个巨无霸一次性执行 PCA 过程。马上报应就出现了，电脑 CPU 预警，内存告急。即使这样仍然花了 20min 才完成整个降维过程。于是之后我就放弃了直接降维的打算。使用另一种 incremental 的 PCA 方式[2]，分成多个 batch 进行降维，虽然牺牲了一些降维的精度，但是也不用再折磨电脑了。

2.2 词汇相似度

之前提到我在提取特征时使用到了主题相似度的概念（虽然之后的实验结果证明这个特征颇有负优化的嫌疑，因为图片提取主题再计算主题的相似度再降维，显然这样的不知道转了几手的特征不知还残留多少信息量）。

这里就谈一下用到的几种计算词汇相似度的方法。

2.2.1 基于 WordNet 的方法[3]

- Path similarity
- Leacock-Chodorow Similarity
- Wu-Palmer Similarity
- Resnik Similarity
- Jiang-Conrath Similarity
- Lin Similarity

这种方法主要基于 WordNet 提供的一个词义树。简单的说就是比较两个词义在整个 WordNet 树的相对位置。如果树上查无此词，那就无可奈何了。（这也是 WordNet 方法只接受动词和名词的相似度度量的原因，没有其他词的词义树其他也就无从下手）。而不同的度量其实也就是在不同的距离度量标准（前三种 metric，结果大同小异），或者结合一些语料库中的数据联合分析上下点文章（后三种 metric，与使用的语料密切相关）。[4]通常使用语料库的指标效果好过单纯使用词义树的指标。

这里主要详细说明下我采纳的两种 metric。

1）Wu-Palmer Similarity

先展示下计算公式：

[2] https://scikit-learn.org/stable/modules/decomposition.html#incrementalpca

[3] http://www.nltk.org/howto/wordnet.html

[4] https://linguistics.stackexchange.com/questions/9084/what-do-wordnetsimilarity-scores-mean

$$Sim_{wup}(c1, c2) = (2* Dep(LCS(c1, c2))) / (Len(c1, c2) + 2*dep(LCS(c1, c2)))$$

$$LCS(c1, c2) = \text{Lowest node in hierarchy that is a hypernym of c1, c2.}$$

LCS 是 c1,c2 的最近的公共父节点，它主要考虑了 c1,c2 的距离（连接的 edge 数目）以及 LCS 的深度。

2） RESNIK SIMILARITY

$$Sim_{Resnik}(c1, c2) = IC(LCS(c1, c2))$$

$$LCS(c1, c2) = \text{Lowest node in hierarchy that is a hypernym of c1, c2.}$$

$$IC(c) = -\log P(c)$$

这里仍然使用了公共父节点的概念，但是不再根据词义树计算距离，而是根据语料库中出现的频率当作评价指标。

其他指标的详细内容。[5]

2.2.2 基于语料统计的方法

根据调查的结果这种方法效果是最好的。[6]原因可能是它突破了 WordNet 的词量限制（一些词语（如 CD 等）没有被收录到语义词典中，而且收录的词语不同词性之间也无法计算语义相似度）

2.2.3 基于检索页面数量的方法

它的基本思想是通过搜索引擎提供的语料库。根据单独搜索和联合搜索的搜索结果条数，构建相似度。

3. 分类器

3.1 Logistic Regression

3.1.1 简述

逻辑回归实际是对线性回归的扩展，利用 Logistic 函数（或称为 Sigmoid 函数）将值域限制在 0-1 之间。经常用于二分类问题。（在此基础上衍生出来的 softmax 可以用于多分类）

---

[5] https://www.3pillarglobal.com/insights/measures-of-semantic-similarity

[6] https://blog.csdn.net/Garfy_/article/details/68485604

其优点是计算代价不高，易于理解和实现；缺点是容易欠拟合，分类精度可能不高。

3.1.2 参数

注：sklearn 实现的 LR 算法自动完成了 regularization 的过程。

class_weight
这个参数实际是为每个分类赋予一个系数，默认为不赋权（权重为 1）．如果给定参数 'balanced'，则使用 y 的值自动调整与输入数据中的类频率成反比的权重。即类别数越多惩罚项越小，某一类的输入样本数越多，这一类的惩罚项越小，这样就能很好的平衡输入样本不均衡带来的学习偏移问题。

惩罚项 C 会相应的放大或者缩小某一类的损失，如果某一类 C 越大，这一类的损失也被（相对于其他类来说）放大，那么系统会把本次学习重点放在这一类上，使得系统尽可能的预测对这一类的输入，所以惩罚项 C 不会影响计算的损失，但反向学习时会相应的放大或缩小损失，间接影响学习的方向。

下图是添加该参数前后的变化，效果提升很显著
= None

```
[LibLinear][LibLinear][LibLinear][LibLinear]LR:  0.7425289393130446   0.007509583590827075
[0.7438901908269167, 0.7361901573485102, 0.7542684968195514, 0.7357669122572003]
```

="balanced"

```
[LibLinear][LibLinear][LibLinear][LibLinear]LR:  0.7614454442122607   0.010261262087586064
[0.7602946099765651, 0.7572815533980582, 0.7780381653833277, 0.7501674480910918]
```

3.2 SVM

3.2.1 简述

支持向量机（英语：support vector machine，常简称为 SVM，又名支持向量网络）是在分类与回归分析中分析数据的监督式学习模型与相关的学习算法。SVM 模型是将实例表示为空间中的点，这样映射就使得单独类别的实例被尽可能宽的明显的间隔分开。然后，将新的实例映射到同一空间，并基于它们落在间隔的哪一侧来预测所属类别。

3.2.2 参数

scale or not
scale 不是一个参数，只是对数据一个预处理的过程。因为 SVM 通常使用高斯核函数，假设

样本服从高斯分布。所以 scale 这个步骤对提升准确率有很大帮助。

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.7810324702422263   0.0059341809643529385
[0.7860729829260127, 0.7743555406762638, 0.7877469032474054, 0.775954454119223]
```

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.8043020929497795   0.003934267132606078
[0.8085035152326749, 0.7994643454971543, 0.8078339471041178, 0.8014065639651708]
```

class_weight

= None

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.8043020929497795   0.003934267132606078
[0.8085035152326749, 0.7994643454971543, 0.8078339471041178, 0.8014065639651708]
```

= balanced

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.7644598181761512   0.004801946921394351
[0.7716772681620355, 0.7606293940408436, 0.759625041848008, 0.7659075686537173]
```

效果反而有显著下降?

cache_size

主要用于提升速度，空间换时间。

C

C 理解为调节优化方向中两个指标（间隔大小，分类准确度）偏好的权重

soft-margin SVM 针对 hard-margin SVM 容易出现的过度拟合问题，适当放宽了 margin 的大小，容忍一些分类错误（violation），把这些样本当做噪声处理，本质上是间隔大小和噪声容忍度的一种 trade-off，至于具体怎么 trade-off，对哪个指标要求更高，那就体现在 C 这个参数上了。[7]

= 1

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.8043020929497795   0.003934267132606078
[0.8085035152326749, 0.7994643454971543, 0.8078339471041178, 0.8014065639651708]
```

= 0.5 增加泛化能力

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.8036330573812711   0.008653086509037717
[0.8125209240040174, 0.804820890525611, 0.7894208235687982, 0.8077695914266577]
```

= 0.1 大大增加泛化能力

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.7827913479061196   0.006183178172091822
[0.7850686307331771, 0.7740207566119852, 0.7810512219618346, 0.7910247823174816]
```

= 2 增加过拟合几率

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.7998657836559452   0.004719610081611226
[0.7947773685972548, 0.8068295949112823, 0.8014730498828255, 0.796383121232418]
```

= 10

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.7864734961120874   0.005208501749329003
[0.7947773685972548, 0.7854034147974557, 0.7803816538332775, 0.7853315472203617]
```

[7] https://www.zhihu.com/question/40217487

可以看出还是=1 时效果最好，而且训练数据集体现不出过拟合和欠拟合的效果。这可能是因为 SVC 本身训练数据时已经对训练集进行随机取样进行拟合，所以体现不出测试集随机取样的效果。

kernel
核函数选择
在 SVM 中核函数的作用就是隐含着一个从低维空间到高维空间的映射，而这个映射可以把低维空间中线性不可分的两类点变成线性可分的。

= linear

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.7994472194871682   0.005639254268511427
[0.8091730833612321, 0.796786072982926, 0.7964512889186475, 0.7953784326858674]
```

= rbf

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.8043020929497795   0.003934267132606078
[0.8085035152326749, 0.7994643454971543, 0.8078339471041178, 0.8014065639651708]
```

= sigmoid

```
[LibSVM][LibSVM][LibSVM][LibSVM]SVM:  0.7388473516966019   0.010135092573599448
[0.7308336123200536, 0.7495815199196518, 0.7268162035487111, 0.7481580709979906]
```

可以看出符合一般规律，即 rbf 是最普适的核函数。

gamma
gamma 是高斯核函数的参数，隐含地决定了数据映射到新的特征空间后的分布，gamma 越大，支持向量越少，gamma 值越小，支持向量越多。支持向量的个数影响训练与预测的速度。

这里我使用了 sklearn 推荐的 grid search 方法寻找最优参数（C 和 gamma）,下面是最终结果，果然还是 default 大法好。默认参数效果最好。

```
The best parameters are {'C': 1, 'gamma': 'auto'} with a score of 0.80
```

3.3 Random Forest

3.3.1 简述

随机森林实际上是一种特殊的 bagging 方法，它将决策树用作 bagging 中的模型。首先，用 bootstrap 方法生成 m 个训练集，然后，对于每个训练集，构造一颗决策树，在节点找特征进行分裂的时候，并不是对所有特征找到能使得指标（如信息增益）最大的，而是在特征中随机抽取一部分特征，在抽到的特征中间找到最优解，应用于节点，进行分裂。随机森林的方法由于有了 bagging，也就是集成的思想在，实际上相当于对于样本和特征都进行了采样（如果把训练数据看成矩阵，就像实际中常见的那样，那么就是一个行和列都进行采样的过程），所以可以避免过拟合。

3.3.2 参数

class_weight

= 'balanced' ， 以整个森林为基准计算

```
RF:  0.7701512874162676   0.004183216247829162
[0.7766990291262136, 0.7656511550050218, 0.7706729159691998, 0.7675820495646349]
```

= 'balanced_subsample'，是以决策树为基准计算

```
RF:  0.7713230877001949   0.0032553844801699707
[0.7766990291262136, 0.7693337797120857, 0.7710077000334784, 0.768251841929002]
```

= None

```
RF:  0.7753402722357274   0.008308511570560246
[0.7693337797120857, 0.7864077669902912, 0.780046869768999, 0.7655726724715338]
```

有意思的是添加参数后效果反而有所下降。

n_estimators

用来指定决策树的数目，默认为 10. 一般来说这个参数越大越好，但是超过一定界限不会再有明显提升效果。

= 150

```
RF:  0.7916639945232646   0.007454420078493075
[0.7941078004686977, 0.7904251757616337, 0.7807164378975561, 0.8014065639651708]
```

= 10

```
RF:  0.7713230877001949   0.0032553844801699707
[0.7766990291262136, 0.7693337797120857, 0.7710077000334784, 0.768251841929002]
```

max_features

因为随机森林属于特殊的 bagging。随机的切分样本以及特征。由于子样本集的相似性以及使用的是同种模型，因此各模型有近似相等的 bias 和 variance（事实上，各模型的分布也近似相同，但不独立）。bagging 后的 bias 和单个子模型的接近，一般来说不能显著降低 bias，甚至因为只考虑部分特征所以 bias 会有上升。但是换来的效果是因为它取得是全部决策树的评价结果，所以它的 variance 会下降。当然通常来说 variance 下降的优势会抵消 bias 上升的劣势，产生更好的结果。

所以调整这个参数实际上是在做 bias 和 variance 的 trade-off 的过程[8]。

= log2

---

[8] （https://zhuanlan.zhihu.com/p/60649636）理解机器学习中的 Bias(偏差)，Variance(方差)
（https://www.zhihu.com/question/26760839）为什么说 bagging 是减少 variance，而 boosting 是减少 bias?

```
RF:  0.7898214208433015   0.007731200562581179
[0.8024774020756612, 0.7817207900903917, 0.788751255440241, 0.7863362357669123]
```

= sqrt（默认）

```
RF:  0.7916639945232646   0.007454420078493075
[0.7941078004686977, 0.7904251757616337, 0.7807164378975561, 0.8014065639651708]
```

= None(实际是完全采样的单个分类器)

```
RF:  0.7844643432547963   0.0031610823822314053
[0.7854034147974557, 0.788751255440241, 0.783729494476063, 0.7799732083054253]
```

bootstrap

是否采用 bootstrap 的采样方式构造决策树。bootstrap 在样本数目较少时能发挥很大作用[9]，但是因为我的样本数量有 10000+。所以我尝试关闭 bootstrap。

= False

```
RF:  0.7923334505339167   0.010000862273384708
[0.7803816538332775, 0.8041513223970539, 0.7847338466688986, 0.8000669792364367]
```

= True

```
RF:  0.7930035512225224   0.009022408354459256
[0.7820555741546702, 0.7884164713759625, 0.7951121526615333, 0.8064300066979236]
```

可以看到影响不明显。

3.4 Multi-layer Perceptron classifier

3.4.1 简述

多层感知器（Multilayer Perceptron,缩写 MLP）是一种前向结构的人工神经网络，映射一组输入向量到一组输出向量。MLP 可以被看作是一个有向图，由多个的节点层所组成，每一层都全连接到下一层。除了输入节点，每个节点都是一个带有非线性激活函数的神经元（或称处理单元）。一种被称为反向传播算法的监督学习方法常被用来训练 MLP。

3.4.2 参数

hidden_layer_sizes

隐藏层数目

(100,) default

```
MLP:  0.8077341901981594   0.007114255654537294
[0.8021426180113826, 0.8192166052895882, 0.8014730498828255, 0.8081044876088412]
```

[9] (https://blog.csdn.net/edogawachia/article/details/79357844) 随机森林（Random Forest）算法原理

(100,100,)增加隐藏层

```
MLP:  0.762618477793143   0.014494469707911201
[0.7830599263475059, 0.7592902577837295, 0.7425510545698025, 0.7655726724715338]
```

结果是训练的 loss 的确有很大提升，但是过拟合现象很严重，总体效果反而更差

(50,50,)增加隐藏层，减少神经元

```
MLP:  0.7715743158957851   0.006345406490412634
[0.7619685302979579, 0.7783729494476063, 0.7760294609976565, 0.7699263228399196]
```

仍然不如单隐藏层结果

（200,)
增加单层神经元数

```
MLP:  0.7798617070489199   0.012088271637363358
[0.7954469367258119, 0.7700033478406428, 0.7663207231335788, 0.7876758204956463]
```

减少单层神经元数
(50,)

```
MLP:  0.7975216224985654   0.008356733935488848
[0.7900903916973552, 0.804820890525611, 0.8068295949112823, 0.7883456128600134]
```

（80,)

```
MLP:  0.8057255699009169   0.004082664100002695
[0.7987947773685973, 0.8068295949112823, 0.8081687311683964, 0.8091091761553918]
```

activation
logistic

```
MLP:  0.8022933325051557   0.008725000614207981
[0.805490458654168, 0.7894208235687982, 0.813525276196853, 0.8007367716008037]
```

tanh

```
MLP:  0.8035493893946777   0.006052791472616458
[0.8024774020756612, 0.7941078004686977, 0.8095078674255105, 0.8081044876088412]
```

relu

```
MLP:  0.8058918968129588   0.005617706147812054
[0.8088382992969535, 0.8074991630398393, 0.8108470036826247, 0.796383121232418]
```

identity

```
MLP:  0.7989454077739415   0.001160073169006491
[0.7971208570472046, 0.8001339136257114, 0.7987947773685973, 0.7997320830542531]
```

激活函数的改变对结构并没有太大影响，identity 即线性激活函数效果更差。

early_stopping

这是一个很有趣的参数，当设置为 True 时，能自动分隔一部分数据用于测试，提前发现过拟合问题终止训练

True （100，100,）网络

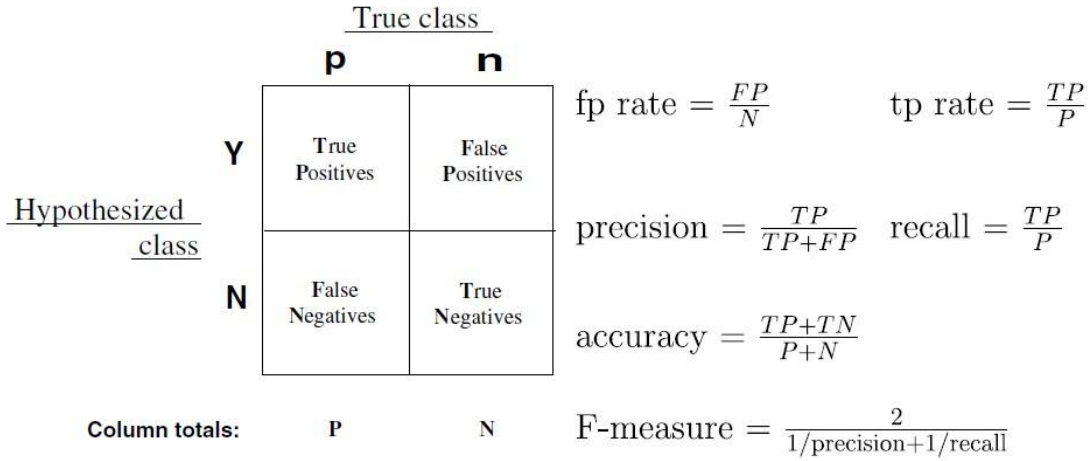
```
MLP:  0.8018753849848562   0.004677106972197523
[0.805490458654168, 0.7994643454971543, 0.7954469367258119, 0.8070997990622907]
```

的确防止了过拟合现象

4. 分类器部分要点

4.1 分类器的评价指标

在之前的调参过程中，只使用了准确率作为唯一的指标。但实际上分类器的结果有很多其他指标可以使用。[10]



精确率 precision = TP / (TP+FP)

精确率容易和准确率被混为一谈。其实，**精确率只是针对预测正确的正样本**而不是所有预测正确的样本。它可以由预测正确的正样本数除以模型所有预测为正样本的数目之比来计算出来。表现为**预测出是正的里面有多少真正是正的**。
召回率 recall = TP / (TP+FN)

召回率是由预测正确的正样本数目除以测试集中真正的实际正样本数目之比计算得出。表现出**所有真正是正样本中分类器能召回多少**。

还有一个指标是 F 值，它是精确率和召回率的调和值。

[10] https://blog.csdn.net/xmu_jupiter/article/details/48270503

那现在让我们看看各个分类器在这些指标上的表现（分成四份交叉验证）。

LR

```
11947  selected, manpercent =  0.2574704946848581
             precision    recall  f1-score   support

          0       0.52      0.72      0.61       774
          1       0.89      0.77      0.83      2213

avg / total       0.79      0.76      0.77      2987

             precision    recall  f1-score   support

          0       0.53      0.72      0.61       770
          1       0.89      0.78      0.83      2217

avg / total       0.80      0.76      0.77      2987

             precision    recall  f1-score   support

          0       0.52      0.74      0.61       771
          1       0.89      0.76      0.82      2216

avg / total       0.80      0.76      0.77      2987

             precision    recall  f1-score   support

          0       0.50      0.73      0.59       761
          1       0.89      0.75      0.81      2225

avg / total       0.79      0.74      0.76      2986
```

SVM:

```
11947  selected, manpercent =  0.2574704946848581
             precision    recall  f1-score   support

       male       0.74      0.41      0.53       776
     female       0.82      0.95      0.88      2211

avg / total       0.80      0.81      0.79      2987

             precision    recall  f1-score   support

       male       0.70      0.37      0.48       758
     female       0.82      0.95      0.88      2229

avg / total       0.79      0.80      0.78      2987

             precision    recall  f1-score   support

       male       0.74      0.39      0.51       775
     female       0.82      0.95      0.88      2212

avg / total       0.80      0.81      0.78      2987

             precision    recall  f1-score   support

       male       0.72      0.35      0.47       767
     female       0.81      0.95      0.88      2219

avg / total       0.79      0.80      0.77      2986
```

Random Forest

```
11947  selected, manpercent =  0.2574704946848581
             precision    recall  f1-score   support

       male       0.72      0.33      0.45       766
     female       0.81      0.96      0.87      2221

avg / total       0.78      0.80      0.77      2987

             precision    recall  f1-score   support

       male       0.74      0.35      0.47       782
     female       0.80      0.96      0.87      2205

avg / total       0.79      0.80      0.77      2987

             precision    recall  f1-score   support

       male       0.70      0.32      0.44       753
     female       0.81      0.95      0.87      2234

avg / total       0.78      0.79      0.76      2987

             precision    recall  f1-score   support

       male       0.68      0.33      0.45       775
     female       0.80      0.94      0.87      2211

avg / total       0.77      0.79      0.76      2986
```

MLP

```
11947  selected, manpercent =  0.2574704946848581
             precision    recall  f1-score   support

       male       0.66      0.50      0.57       789
     female       0.83      0.91      0.87      2198

avg / total       0.79      0.80      0.79      2987

             precision    recall  f1-score   support

       male       0.71      0.43      0.54       783
     female       0.82      0.94      0.88      2204

avg / total       0.79      0.80      0.79      2987

             precision    recall  f1-score   support

       male       0.68      0.49      0.57       759
     female       0.84      0.92      0.88      2228

avg / total       0.80      0.81      0.80      2987

             precision    recall  f1-score   support

       male       0.68      0.41      0.51       745
     female       0.83      0.93      0.88      2241

avg / total       0.79      0.80      0.79      2986
```

可以看出四种模型在预测过程中都存在对男性预测准确率远低于女性准确率的问题。但是倾向性又略有不同。LR 的男性 precision 很低，recall 较高，说明它倾向于发现更多的男性，不惜牺牲预测的准确率。而其他三种分类器相反，它们保证了较高的预测准确率，但是代价是只发现了 50%不到的男性。

这其实也和数据集的选取有关。因为我们的数据集中只有 25.7%的男性用户。换句话说即使分类器把所有用户都预测为女性也能有接近 75%的正确率。即使我在分类器调参上下再多功夫，数据集的倾向性还是会有很大的误导性。所以很自然我们想到，如果保证训练数据男女比例 1：1 会产生什么样的结果。

下面是男女比分占一半的结果

LR

```
6152  selected, manpercent =  0.5
             precision    recall  f1-score   support

          0       0.76      0.72      0.74       799
          1       0.71      0.75      0.73       739

avg / total       0.74      0.73      0.73      1538

             precision    recall  f1-score   support

          0       0.78      0.72      0.75       763
          1       0.74      0.80      0.77       775

avg / total       0.76      0.76      0.76      1538

             precision    recall  f1-score   support

          0       0.74      0.73      0.74       747
          1       0.75      0.75      0.75       791

avg / total       0.74      0.74      0.74      1538

             precision    recall  f1-score   support

          0       0.75      0.73      0.74       767
          1       0.74      0.76      0.75       771

avg / total       0.75      0.75      0.75      1538
```

RF

```
6152  selected, manpercent =  0.5
             precision    recall  f1-score   support

       male       0.73      0.76      0.74       731
     female       0.77      0.74      0.76       807

avg / total       0.75      0.75      0.75      1538

             precision    recall  f1-score   support

       male       0.73      0.78      0.76       754
     female       0.78      0.72      0.75       784

avg / total       0.75      0.75      0.75      1538

             precision    recall  f1-score   support

       male       0.75      0.74      0.74       791
     female       0.73      0.73      0.73       747

avg / total       0.74      0.74      0.74      1538

             precision    recall  f1-score   support

       male       0.75      0.77      0.76       800
     female       0.74      0.72      0.73       738

avg / total       0.74      0.74      0.74      1538
```

SVM

```
6152  selected, manpercent =  0.5
             precision    recall  f1-score   support

       male       0.78      0.73      0.75       779
     female       0.74      0.79      0.76       759

avg / total       0.76      0.76      0.76      1538

             precision    recall  f1-score   support

       male       0.75      0.75      0.75       771
     female       0.75      0.75      0.75       767

avg / total       0.75      0.75      0.75      1538

             precision    recall  f1-score   support

       male       0.77      0.73      0.75       763
     female       0.75      0.78      0.76       775

avg / total       0.76      0.76      0.76      1538

             precision    recall  f1-score   support

       male       0.76      0.73      0.74       763
     female       0.74      0.77      0.76       775

avg / total       0.75      0.75      0.75      1538
```

MLP

```
6152  selected, manpercent =  0.5
             precision    recall  f1-score   support

       male       0.74      0.77      0.75       748
     female       0.77      0.74      0.76       790

avg / total       0.76      0.76      0.76      1538

             precision    recall  f1-score   support

       male       0.80      0.73      0.76       774
     female       0.75      0.82      0.78       764

avg / total       0.78      0.77      0.77      1538

             precision    recall  f1-score   support

       male       0.77      0.70      0.73       746
     female       0.74      0.81      0.77       792

avg / total       0.76      0.75      0.75      1538

             precision    recall  f1-score   support

       male       0.77      0.74      0.76       808
     female       0.72      0.76      0.74       730

avg / total       0.75      0.75      0.75      1538
```

可以看出平衡的样本比例会显著影响分类器的预测结果。在总准确率有所下降的代价下，男女的预测结果有所平衡。可以预见如果再次改变男女比为男性更多，那么一定是男性预测准确率更高。

4.2 归一化？标准化？

在对数据的预处理过程中，我们经常会遇到归一化或标准化的过程。有时这种处理会给实际结果带来很大的影响，有时却又没什么影响。那么为什么要进行这类处理？这些预处理方法又该什么时候拿出来用呢？

首先要搞清楚这些名词分别是什么含义?[11]

"标准化"和"归一化"这两个中文词要指代四种 Feature scaling(特征缩放)方法

1) Rescaling

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

2) Mean normalization

$$x' = \frac{x - \mathrm{mean}(x)}{\max(x) - \min(x)}$$

3) Standardization

$$x' = \frac{x - \bar{x}}{\sigma}$$

4) Scaling to unit length

$$x' = \frac{x}{||x||}$$

可以说明的是 1，2，4 的目的在于对不同特征维度的伸缩变换的目的是使各个特征维度对目标函数的影响权重是一致的。即使得那些扁平分布的数据伸缩变换成类圆形。这也就改变了原始数据的一个分布。

3 则保留了数据的均值和方差信息，对不同特征维度的伸缩变换的目的是使得不同度量之间的特征具有可比性。同时转换为标准正态分布，缩放过程和整体样本分布相关，每个样本点都能对标准化产生影响。[12]

normalization 和 standardization 是差不多的，都是把数据进行前处理，从而使数值都落入

[11] https://www.zhihu.com/question/20467170

[12] https://www.jianshu.com/p/95a8f035c86c

到统一的数值范围，从而在建模过程中，各个特征量没差别对待。normalization 一般是把数据限定在需要的范围，比如一般都是【0，1】，从而消除了数据量纲对建模的影响。standardization 一般是指将数据正态化，使平均值 1 方差为 0. 因此 normalization 和 standardization 是针对数据而言的，消除一些数值差异带来的特种重要性偏见。经过归一化的数据，能加快训练速度，促进算法的收敛。

总之，归一化和标准化过程能带来一定的好处——消除量纲影响、加快收敛速度、减少数值问题。两者又各有优势，归一化在于它能将数据控制在特定范围，但是可能无法保持数据分布特征；标准化胜于他将数据转换为一定范围时又使其在一个正态分布中。而且有些模型原理上对数据有所要求，必须提前对数据进行归一化/标准化。如 SVM、PCA 等。