

基于分类算法的笑话偏好特征挖掘

ACM1501 李培昊 U201514616

摘要

信息时代中，用户的浏览、评分记录清晰反映了用户的行为特征和习惯偏好，这对于用户行为的研究具有广泛的社会学意义和商业价值。如何高效的从用户数据中选择具有特征性的数据组，如何根据数据进行准确的行为预测是研究用户行为的一个基本任务。文中使用了 Jester 在线笑话推荐系统的部分匿名数据集，通过数据挖掘常用的分类算法和神经网络方法，分析用户对特定笑话的偏好问题，即分辨一个笑话是否被指定用户所喜欢。通过多种分类算法进行建模，尝试减少样本空间的特征列数，从不同算法和不同样本空间维度两个方向上分别进行对比

1 数据介绍

1.1 数据背景

数据来源于从 1999 年四月到 2003 年五月收集的，共计 73421 个用户提供的对于 100 个笑话的超过 410 万条连续评分（-10.0~+10.0）记录。

1.2 数据特点

1. 数据包含 73421 名匿名用户的评分记录。
2. 数据以 Excel(.xls)格式进行存储，以.zip 的形式进行压缩。
3. 评分为连续实数，范围在-10.0~+10.0 分之间，而值 99 意味着空值或未评分。
4. 数据每行代表一个用户。
5. 每行第一列代表用户对 100 个笑话评价的个数，之后 100 列代表此用户对 1~100 号笑话的评分。
6. 子列{5, 7, 8, 13, 15, 16, 17, 18, 19, 20}是密集的，几乎所有用户都对这些笑话进行了评分。

1.3 分析目标

获取多个用户在某些的笑话上的评分，并将其当作一个数据的特征向量，判断用户对于一个指定的笑话的偏好程度，进行二分类，即评分大于等于 0 记为 1（喜欢），评分小于 0 记为-1（不喜欢），通过不同的分类算法实现不同的模型，使用同样的数据集对模型进行训练并验证，分析不同模型的验证准确性。

定义：大众性，为特征列的属性，表现为全部样本在此特征列的取值相同或近似，无法在此特征列上无法差异化样本，从而可能导致样本的分类预测准确度下降。

定义：样本空间特征化，通过高斯训练模型得到的特征的方差，选取其中最小的一部分列将其删除，使得样本降维。

在朴素贝叶斯算法高斯模型中，训练模型得到了全部的特征列的均值和方差，而对于某些大众列（大家都很喜欢的笑话或大家都不喜欢的笑话），其方差较小，这意味着对于不同标签的样本，在此列上的区别不明显，或许有可能因为大众列降低了不同标签的区别度，从而使预测成功的概率降低。因此使用样本空间特征化后的样本，重新训练模型，有可能使得模型的准确度上升，因此分析不同模型对于特征的独特性（区别于大众性）的依赖程度。

1.4 数据清洗

对于问题研究的共计 19763 个用户中，存在大量用户未能对全部的 100 个笑话进行评分。统计对全部 100 个笑话进行评分的用户数共计 2344 个，结合问题研究的重点，只选择此类用户的数据作为模型的训练集和验证集。

2 算法描述

2.1 总体描述

本实验重点测试了四种分类算法和一种神经网络算法实现的分类器模型。

1. 朴素贝叶斯分类器(Naïve Bayes Classifier, NBC)

朴素贝叶斯法是基于贝叶斯定理和特征属性条件独立假设的分类方法，其发源于古典数学理论，有着坚实的数学基础和稳定的分类效率。同时朴素贝叶斯分类器所需估计的参数很少，对缺失数据不太敏感，算法也比较简单。理论上 NBC 与其他分类

华中科技大学课程设计报告

方法相比有最小的误差率，但实际并非如此，这是因为 NBC 模型假设属性之间相互独立，这个假设在实际应用中往往不成立，给 NBC 的正确分类带来了一定影响。

2. 逻辑回归 (Logistic Regression, LR)

逻辑回归是一种广义的线性回归分析模型，其结果可以是二分类的也可以是多分类的，但二分类更为常用。其特征变量可以是连续的也可以是离散的，通过逻辑回归得到特征的权重，从而预测某一特征变量下的标签值。

3. 支持向量机 (Support Vector Machine, SVM)

支持向量机被视为一中对感知机模型的扩展的线性分类器模型。感知机找到一个超平面使得数据线性可分，而支持向量机找到使得特征空间上间隔最大化的超平面作为分类器。通过支持向量机模型训练得到的分类器，预测输入的某一特征变量的分类标签。

4. K 邻居分类器(K Neighbors Classifier)

K 邻近分类器是使用了 KNN 算法，即 K 最近邻分类法，顾名思义，算法的思想就是每个样本点都可以由它最近的 K 个邻居代表。通过获取其 K 个邻居的标签值，预测输入的某一个特征变量的分类标签。由于 KNN 方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于类域的交叉或重叠较多的待分样本集来说，KNN 方法较其他方法更为适合。

5. 神经网络(Neural Network)

神经网络是一种模仿动物神经网络行为特征，进行分布式并行信息处理的算法数据模型。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的。

2.2 朴素贝叶斯分类器

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

贝叶斯分类法的基础是基于条件概率下的贝叶斯定理

$$p(\text{类别}|\text{特征}) = \frac{p(\text{特征}|\text{类别})p(\text{类别})}{p(\text{特征})}$$

也即，通过贝叶斯定义，我们可以通过较为容易求解的 $P(A|B)$ 获得不易求解的 $P(B|A)$ 。

朴素贝叶斯分类的正式定义如下：

1. 设 $x = \{a_1, a_2, \dots, a_m\}$ 为一个待分类的数据项的特征向量，而 a_i 是 x 的第 i

个特征属性。

2. 有标签 $C = \{y_1, y_2, \dots, y_n\}$ 。
3. 计算 $P(y_1|x), P(y_2|x), \dots, P(y_n|x)$ 。
4. 求解 k , 使得 $P(y_k|x) = \max\{P(y_1|x), P(y_2|x), \dots, P(y_n|x)\}$ 。

基于朴素贝叶斯的特征属性条件独立的假设，有以下公式推导

$$P(y_i|x) = \frac{P(x|y_i) \cdot P(y_i)}{P(x)} \propto P(y_i) \cdot P(x|y_i) = P(y_i) \cdot \prod_{j=1}^m P(a_j|y_i)$$

至此，朴素贝叶斯分类器的表示被推导如下：

$$f(x) = \operatorname{argmax}_i P(y_i) \prod_{j=1}^m P(a_j|y_i)$$

关于 $P(y_i)$ 和 $P(x_j|y_i)$ 的求解，因为特征变量是连续性变量，所以选用高斯模型处理连续的特征变量，假设其值服从高斯分布（也称正态分布），即

$$P(x_i|y_k) = \frac{1}{\sqrt{2\pi\sigma_{y_k,i}^2}} e^{-\frac{(x_i - \mu_{y_k,i})^2}{2\sigma_{y_k,i}^2}}$$

$\mu_{y_k,i}$ 表示类别为 y_k 的样本中，第 i 维特征的均值。

$\sigma_{y_k,i}^2$ 表示类别为 y_k 的样本中，第 i 维特征的方差。

在 python 实现中，使用 sklearn 库中的 GaussianNB 模型进行实现，模型获取训练样本和训练样本标签，训练得到每一个标签类每一维特征的均值和方差，并据此对输入的测试点进行预测。

```
from sklearn.naive_bayes import GaussianNB
```

```
theta_ : array, shape (n_classes, n_features)
```

```
mean of each feature per class
```

```
sigma_ : array, shape (n_classes, n_features)
```

```
variance of each feature per class
```

2.3 逻辑回归分类器

逻辑回归就是一种减小预测范围，将预测值限定为 $[0,1]$ 间的一种回归模型。首先

要构造预测函数，即 Sigmoid 函数，函数形式为 $g(z) = \frac{1}{1+e^{-z}}$ ，函数图像如图 2-1 所示。

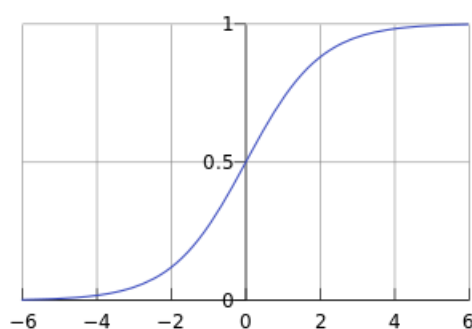


图 2-1 Sigmoid 函数图像

套用 Sigmoid 函数，对于标签为 $C=\{0,1\}$ 的分类，有公式：

$$\begin{cases} p(y=1 | x, \theta) = \frac{1}{1 + e^{-\theta^T x}} & (3) \\ p(y=0 | x, \theta) = \frac{1}{1 + e^{-\theta^T x}} = 1 - p(y=1 | x, \theta) = p(y=1 | x, -\theta) & (4) \end{cases}$$

$$\text{令: } h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}; \quad g(z) = \frac{1}{1 + e^z}$$

$$p(y | x, \theta) = (h_{\theta}(x))^y (1 - h_{\theta}(x))^{1-y}$$

构建逻辑回归模型，对于单个样本，其后验概率为 $p(y | x, \theta)$ 其中 $y=1$ (或 0)，使用极大似然估计方法得：

$$\begin{aligned} L(\theta | x, y) &= \prod_{i=1}^m p(y^{(i)} | x^{(i)}; \theta) \\ &= \prod_{i=1}^m (h_{\theta}(x))^{y^{(i)}} (1 - h_{\theta}(x))^{1-y^{(i)}} \end{aligned}$$

对其求 log 函数，不改变其单调性：

$$\begin{aligned} l(\theta) &= \log(L(\theta | x, y)) \\ &= \sum_{i=1}^m y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \end{aligned}$$

采用梯度下降的方法：

$$\begin{aligned}
 \frac{\partial}{\partial \theta_j} (l(\theta)) &= \frac{\partial}{\partial \theta_j} \left(\sum_{i=1}^m y^{(i)} \log(h(x^{(i)})) + (1-y^{(i)}) \log(1-h(x^{(i)})) \right) \\
 &= \left(\frac{y^{(i)}}{h(x^{(i)})} - (1-y^{(i)}) \frac{1}{1-h(x^{(i)})} \right) \frac{\partial}{\partial \theta_j} (h(x^{(i)})) \\
 &= \left(\frac{y^{(i)}}{g(\theta^T x^{(i)})} - (1-y^{(i)}) \frac{1}{1-g(\theta^T x^{(i)})} \right) \frac{\partial}{\partial \theta_j} (g(\theta^T x^{(i)})) \\
 &= \left(\frac{y^{(i)}}{g(\theta^T x^{(i)})} - (1-y^{(i)}) \frac{1}{1-g(\theta^T x^{(i)})} \right) g(\theta^T x^{(i)}) (1-g(\theta^T x^{(i)})) \frac{\partial \theta^T x^{(i)}}{\partial \theta_j} \\
 &= (y^{(i)} (1-g(\theta^T x^{(i)})) - (1-y^{(i)}) g(\theta^T x^{(i)})) x_j \\
 &= (y^{(i)} - h_\theta(x^{(i)})) x_j
 \end{aligned}$$

迭代使 θ 收敛即可 $\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$ ，其中 α 为学习速率。

python 实现中调用 sklearn 库中的 LogisticRegression 模型，使用训练样本和训练样本标签对模型进行训练，训练得到样本的特征系数，并据此对测试样本进行预测。

```
from sklearn.linear_model import LogisticRegression
```

```
coef_ : array, shape (1, n_features) or (n_classes, n_features)
```

Coefficient of the features in the decision function.

coef_ is of shape (1, n_features) when the given problem is binary.

2.4 支持向量机

和感知机模型一样，SVM 求出 n 维空间中的超平面将数据按标签二分。区分与感知机的是 SVM 使得两类样本点距离超平面的最近距离最大。

给定一个特征空间上大小为 N 的训练数据集 $T = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_N, y_N)\}$ $x_i \in X = R^n$, $y_i \in Y = \{+1, -1\}$, $i = 1, 2, 3, \dots, N$, x_i 为样本的第 i 个特征向量, y_i 为第 i 个特征向量的标签，目标是找到一个超平面使得正负类分到平面的两侧，当数据集线性可分，这样的超平面存在无数个，SVM 要求间隔最大化，解唯一。超平面如图 2-2 所示。

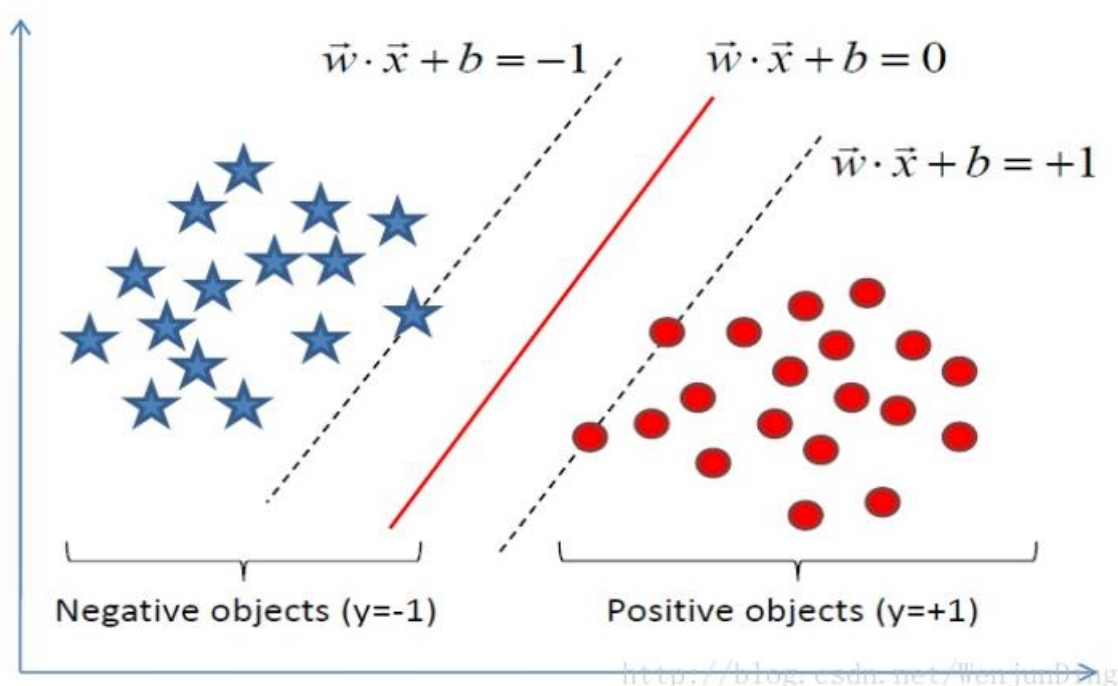


图 2-2 支持向量机的超平面示意图

SVM 的目的在于最优化几何间隔最大的超平面，在样本是线性可分时，间隔最大化被称为硬间隔，在样本近似可分时，被称为软间隔。SVM 学习算法可以被表示为约束最优化问题：

$$\max_{w,b} \Upsilon = \frac{w \cdot x_i + b}{\|w\|}$$

$$s.t. \ y_i(w \cdot x + b) \geq 1, i = 1, 2, 3, \dots, N$$

在进行放缩变化(w, b)改变函数间隔的大小，但超平面不会改变，使得问题简化为：

$$\max_{w,b} \Upsilon = \frac{1}{\|w\|}$$

$$s.t. \ y_i(w \cdot x + b) \geq 1, i = 1, 2, 3, \dots, N$$

继续进行转化可得：

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

$$s.t. \ y_i(w \cdot x + b) \geq 1, i = 1, 2, 3, \dots, N$$

在 python 实现中, 使用 `sklearn.svm` 库中的 `LinearSVC` 模型进行实现, 模型获取训练样本和训练样本标签, 训练得到每一特征的权重, 并据此对输入的测试点进行预测。

```
from sklearn.svm import LinearSVC
```

```
coef_ : array, shape = [n_features] if n_classes == 2 else [n_classes, n_features]
```

Weights assigned to the features (coefficients in the primal problem). This is only available in the case of a linear kernel.

`coef_` is a readonly property derived from `raw_coef_` that follows the internal memory layout of liblinear.

2.5 K 邻居分类器

KNN 算法的核心思想是如果一个样本在特征空间中的 k 个最相邻的样本中的大多数属于某一个类别, 则该样本也属于这个类别, 并具有这个类别上样本的特性。该方法在确定分类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。KNN 方法在类别决策时, 只与极少量的相邻样本有关。

其中, 在计算距离矩阵的方法上, 一般选用 Minkowski Distance 方法:

$$dist(x, z) = \left(\sum_{i=1}^d |x_i - z_i|^p \right)^{\frac{1}{p}}$$

其中 d 为特征维度, p 为可变参数, 通常选 $p=2$ 欧式距离, 或 $p=\infty$ 最大向量有效。

算法描述为:

1. 计算测试数据与各个训练样本点之间的距离。
2. 按照距离增大的顺序进行排序。
3. 选取前 K 个点。
4. 确定前 K 个点对应标签出现的频率。
5. 将最大频率的标签作为测试数据的预测标签。

在 python 实现中, 使用 `sklearn.neighbors` 库中的 `KNeighborsClassifier` 模型进行实

现，模型获取训练样本和训练样本标签，并据此对输入的测试点进行预测。

```
from sklearn.neighbors import KNeighborsClassifier
```

2.6 神经网络

神经网络由多个神经元构成，神经元的模型如图 2-3 所示。

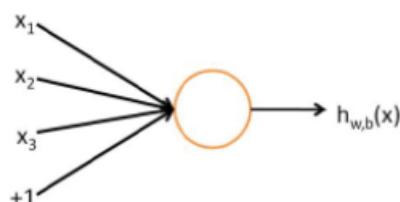


图 2-3 神经元模型

这个神经元以 x_1, x_2, x_3 以及截距+1 作为输入值的运算待援，其输出为：

$$h_{w,b}(x) = f(W^T x) = f(\sum_{i=1}^3 W_i x_i + b)$$

其中 $f(\cdot)$ 被称为激活函数。

神经网络就是将许多个单一的神经元联结在一起，这样，一个神经元的输出就可以使另一个神经元的输入。列如，图 2-4 就是一个简单的神经网络：

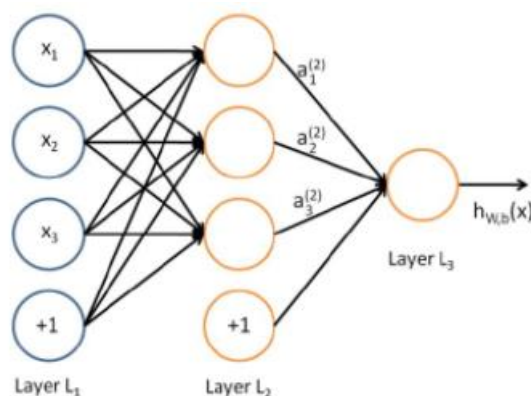


图 2-4 神经网络示意图

以上述模型为例，计算过程如下：

$$\begin{aligned} a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\ a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\ a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\ h_{w,b}(x) &= a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)}) \end{aligned}$$

也可以表示为：

$$\begin{aligned}z^{(2)} &= W^{(1)}x + b^{(1)} \\a^{(2)} &= f(z^{(2)}) \\z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\h_{w,b}(x) &= a^{(3)} = f(z^{(3)})\end{aligned}$$

上述计算步骤被称为前向传播，给定第 1 层的激活值 a^1 后，第 1+1 层的激活值就可以通过下面的计算步骤得到：

$$\begin{aligned}z^{(l+1)} &= W^{(l)}a^{(l)} + b^{(l)} \\a^{(l+1)} &= f(z^{(l+1)})\end{aligned}$$

在 python 实现中，使用 sklearn.neighbors 库中的 KNeighborsClassifier 模型进行实现，模型获取训练样本和训练样本标签，训练得到每层的权重矩阵和偏置矢量，并据此对输入的测试点进行预测。

```
from sklearn.neural_network import MLPClassifier
```

classes_ : array or list of array of shape (n_classes,)

Class labels for each output.

loss_ : float

The current loss computed with the loss function.

coefs_ : list, length n_layers - 1

The ith element in the list represents the weight matrix corresponding to layer i.

intercepts_ : list, length n_layers - 1

The ith element in the list represents the bias vector corresponding to layer i + 1.

n_iter_ : int,

The number of iterations the solver has ran.

n_layers_ : int

Number of layers.

n_outputs_ : int

Number of outputs.

out_activation_ : string

Name of the output activation function.

3 实现过程

3.1 总体分析

实现过程被分为两个模块,第一个模板是 `jester_lib.py` 文件,第二个文件是 `run.py`。

`jester_lib` 中主要包括但不限于数据读取,数据预处理,训练模型,预测标签,验证数据,写入文件等功能。

`run` 中主要包括运行测试和通过 IO 进行模型选择功能。

3.2 函数设计和实现

为完成 3.1 小节中描述的功能函数,定义并分析函数设计和具体功能如下:

`read_file(file_path)`: 函数获取文件的路径名,打开文件读取数据,将数据转为 `numpy` 格式后返回。

`spilt_data(data, verification_proportion=0.1, label_col=-1)`: 函数获取数据,可选参数包括训练集和验证集比例(默认 9:1),标签列的选择(默认最后一列)。函数选取全部评价了 100 个笑话的用户行作为数据集,根据训练集和验证集比例分割,将其余数据作为测试集一起返回。

`def get_model(model, X_train, Y_train)`: 函数获取模型和训练集的数据及标签,返回经过训练的模型。

`def prediction(model, X_point, Y_point=0, carry_Point_func=(lambda x: x.reshape(1, -1)))`: 函数获取训练好的预测模型、测试点,可选参数为测试点的实际标签(默认为 0),对测试点的预处理函数。函数对测试点进行预处理后,通过模型进行预测,获取返回值,判断预测值与实际值是否相同并返回。

`def write_to_file(data)`: 函数获取数据,写入指定的 `xls` 文件中并保存。

`run` 模块下主要定义运行函数,运行函数循环获取用户输入进行不同模型的验证并输出结果,代码如下:

```
model_dict = {'0': GaussianNB(),
              '1': MLPClassifier(),
              '2': LogisticRegression(),
```

华中科技大学课程设计报告

```
        '3': LinearSVC(),
        '4': KNeighborsClassifier(n_neighbors=3),
        '5': GaussianNB()}

model_string_dict = {'0': 'GaussianNB() to cutdown 1/4 feature',
                    '1': 'MLPClassifier()',
                    '2': 'LogisticRegression()',
                    '3': 'LinearSVC()',
                    '4': 'KNeighborsClassifier(n_neighbors=3)',
                    '5': 'GaussianNB()'}

if __name__ == "__main__":
    # testAll()
    data = jester.read_all_file()
    while 1:
        try:
            print("please choose model")
            print(model_string_dict)
            k = input()
            X_test, Y_test, X_veri, Y_veri, X_train, Y_train = jester.split_data(data)
            print(model_string_dict.get(k))
            model = jester.get_model(model_dict.get(k), X_train, Y_train)
            if k == '0':
                jestersort = jester.sigma(model)
                data = jester.futureData(jestersort, data)
            jester.verify(model, X_veri, Y_veri)
        # while 1:
        #     try:
        #         print("please choose label col in data")
        #         ind = int(input())
```

```
#           X_test, Y_test, X_veri, Y_veri, X_train, Y_train =
jester.split_data(jester.read_all_file(), label_col=ind)

#           model = jester.get_model(model_dict.get(k), X_train,
Y_train)

#           jester.verify(model, X_veri, Y_veri,
carry_point_func_dict.get(k))

#       except EOFError or ValueError:

#           break

except EOFError:

    break
```

3.3 功能补充

为了实现对本实验更为细致具体的功能，需在 `jester_lib` 中添加或修改函数，描述如下。

1. 为了进行测试结果的随机性，在对数据进行预处理的过程中，不失一般性的对数据的行列进行随机化。修改 `read_file` 函数和 `split_data` 函数代码如下：

```
# 行随机化
temp_matrix = data_matrix.copy()
random_index = list(range(data_matrix.shape[0]))
random.shuffle(random_index)
for cur in range(data_matrix.shape[0]):
    temp_matrix[random_index[cur], :] = data_matrix[cur, :]
data_matrix = temp_matrix.copy()

# 列随机化
temp_matrix = data.copy()
random_index = list(range(data.shape[1]))
random.shuffle(random_index)
for cur in range(data.shape[1]):
    data[:, random_index[cur]] = temp_matrix[:, cur]
```

华中科技大学课程设计报告

2. 为了便捷的验证全部验证集并返回验证成功率，设计实现函数 `verify`，代码如下：

```
def verify(model, X_veri, Y_veri, carry_Point_func=(lambda x: x.reshape(1, -1))):
    """
    :param model: 训练好的样本
    :param X_veri: 验证集
    :param Y_veri: 验证标签
    :param carry_Point_func: 对验证坐标点处理
    :return:
    """
    assert model is not None
    # since = time.time()
    count = 0
    sum = X_veri.shape[0]
    for cur in range(sum):
        point = X_veri[cur, :]
        predict_label, predict_succ = prediction(model, point, Y_veri[cur][0],
        carry_Point_func)
        if predict_succ:
            count += 1
    print("test num : %d, succ num : %d, succ precent: %f\n" % (sum, count, count *
    1.0 / sum))
    return sum, count, count * 1.0 / sum
    # time_elapsed = time.time() - since
    # print('Training complete in {:.0f}m {:.0f}s'.format(time_elapsed // 60,
    time_elapsed % 60))
```

3. 在朴素贝叶斯算法高斯模型中，训练模型得到了全部的特征列的均值和方差，选取其中方差最小的一部分列删除，以提高样本特征的独特性。代码如下：

```
def sigma(model):
```

```
assert model is not None


t = model.sigma_
t = numpy.sum(t, axis=0)
print(t)
sortInd = t.argsort()
return sortInd[:int(len(sortInd) / 4)]
```

```
def futureData(sortInd, data: object) -> object:
    sortInd = numpy.sort(sortInd)
    sortInd = sortInd[::-1]
    # print(sortInd)
    for cur in range(len(sortInd)):
        data = numpy.delete(data, sortInd[cur], axis=1)
    return data
```

4 结果展示

4.1 运行演示

使用 python 运行 run.py，运行结果如图 4-1 所示。



```
Running D:/Text/数据挖掘/Jester/src/run.py
reading file ...
please choose model
['0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier()', '2': 'LogisticRegression()', '3': 'LinearSVC()', '4': 'KNeighborsClassifier(n_neighbors=3)', '5': 'GaussianNB()']
```

图 4-1 运行演示

输入 1，选择神经网络模型进行测试，基础的特征列数都是 99，测试结果如图 4-2 所示。

```
please choose model
{'0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier()',
 0 1
current label col = 99
MLPClassifier()
test num : 1411, succ num : 933, succ precent: 0.661233
```

图 4-2 神经网络模型 99 特征列测试

输入 2，选择逻辑回归模型进行测试，特征列数为 99 个，测试结果如图 4-3 所示。

```
please choose model
{'0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier()',
 0 2
current label col = 99
LogisticRegression()
test num : 1411, succ num : 1118, succ precent: 0.792346
```

图 4-3 逻辑回归模型 99 特征列测试

输入 3，选择 SVM 模型进行测试，特征列数为 99 个，测试结果如图 4-4 所示。

```
please choose model
{'0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier()',
 0 3
current label col = 99
LinearSVC()
test num : 1411, succ num : 1156, succ precent: 0.819277
```

图 4-4 SVM 模型 99 特征列数测试

输入 4，选择 KNN 模型进行测试，特征列数为 99 个，测试结果如图 4-5 所示。

```
please choose model
{'0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier()',
 0 4
current label col = 99
KNeighborsClassifier(n_neighbors=3)
test num : 1411, succ num : 1035, succ precent: 0.733522
```

图 4-5 KNN 模型 99 特征列数测试

输入 5，选择朴素贝叶斯高斯模型进行测试，特征列数为 99 个，测试结果如图 4-6 所示。


```
please choose model
{'0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier'}
5
current label col = 99
GaussianNB()
test num : 1411, succ num : 887, succ precent: 0.628632
```

图 4-6 朴素贝叶斯高斯模型 99 特征列数测试

输入 0，选择朴素贝叶斯高斯模型进行测试，特征列数为 99 个，测试结果如图 4-7 所示，测试后去除最大众性的 1/4 特征列。

```
please choose model
{'0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier()', '2': 'MLPClassifier'}
0
current label col = 99
GaussianNB() to cutdown 1/4 feature
[73.48032807 61.05623904 68.36462757 63.01079058 60.43182487 78.45499465
 61.04371252 66.87808218 69.88481711 55.83469028 67.3333131 68.12475605
 57.9919694 75.11930021 57.0470231 77.54843357 71.02117403 72.78882563
 71.68621908 57.4251121 94.88578785 58.6033809 66.67538952 69.42144485
 63.02166244 58.31850266 75.43724133 72.459393 58.03124025 70.68325323
 63.21171995 66.3986555 66.01376893 77.02361426 58.71384985 85.91047205
 62.57930157 64.20487597 59.56669235 73.43397116 61.27144974 56.46617088
 63.50577546 61.55323161 69.37536506 80.81559653 60.04143596 64.92139521
 76.98559304 79.35241289 62.66146644 57.91926877 71.02194127 66.42199316
 57.40624991 78.96858077 54.55317889 63.89288512 73.02062385 71.21857866
 71.7206689 60.95984411 70.92514144 65.24362798 81.16899201 72.56176597
 73.23706883 66.39869748 65.1787353 57.97457946 76.55220461 63.55994661
 60.38827617 53.40374862 57.38710152 82.56767977 81.21719707 66.28870325
 60.46147509 63.81995754 65.9089011 63.9896821 61.4165814 60.67988635
 67.05240295 77.17094802 72.66391106 57.66449128 56.23771437 61.768475
 71.13007835 85.65641689 79.86267227 72.66579965 61.19607001 61.62988465
 85.56762948 60.23017483 75.95141568]
test num : 1411, succ num : 882, succ precent: 0.625089
```

图 4-7 朴素贝叶斯高斯模型去除 1/4 特征列数测试

输入 1，选择神经网络模型进行测试，特征列数为 75 个，测试结果如图 4-8 所示。

```
please choose model
{'0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier'}
1
current label col = 75
MLPClassifier()
test num : 1411, succ num : 900, succ precent: 0.637845
```

图 4-8 神经网络模型 75 特征列数测试

输入 0，选择朴素贝叶斯高斯模型进行测试，特征列数为 75 个，测试结果如图 4-

9 所示，测试后去除最大众性的 1/4 特征列。

```
please choose model
{'0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier()', '2':
0
current label col = 75
GaussianNB() to cutdown 1/4 feature
[61.61062288 63.96875491 55.75048169 68.5713351 59.22992385 56.40594333
57.76492176 65.37717806 57.48838054 61.43555394 67.70958018 57.36907849
84.75510389 53.28287212 73.55447604 69.31462844 61.0993736 88.84786492
55.77832418 63.25030603 50.58840393 73.96659352 78.59743807 59.83281696
68.54884081 65.04470888 59.49132516 64.78455151 65.02510991 71.64744176
58.32412757 57.21460219 80.39835381 70.8845628 53.50905724 67.00920379
65.55769205 68.58125231 61.14778671 79.31850083 49.86892293 65.0051556
67.32904113 58.16886885 63.6024581 46.37592388 63.80153914 54.08871045
62.37507131 69.78139643 64.73995525 61.15118667 60.8269654 81.6650426
66.74547505 56.18830094 71.12743885 73.60328469 68.74356592 78.45713947
52.90769451 58.50956742 69.82122729 62.83298823 58.43441252 67.35972471
74.28825457 83.13631352 68.18432112 71.46174611 59.96521943 64.92144215
64.42213445 71.58360361 78.73245163]
test num : 1411, succ num : 914, succ precent: 0.647768
```

图 4-9 朴素贝叶斯高斯模型去除 1/4 特征列数测试

输入 2，选择逻辑回归模型进行测试，特征列数为 57 个，测试结果如图 4-10 所示。

```
please choose model
{'0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier()', '2':
2
current label col = 57
LogisticRegression()
test num : 1411, succ num : 1079, succ precent: 0.764706
```

图 4-10 逻辑回归模型 57 特征列数测试

输入 0，选择朴素贝叶斯高斯模型进行测试，特征列数为 57 个，测试结果如图 4-11 所示，测试后去除最大众性的 1/4 特征列。

```

please choose model
{'0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier()', '2': '
0
current label col = 57
GaussianNB() to cutdown 1/4 feature
[69.48392862 65.46989398 80.7616912 50.38136581 73.2837123 65.9708085
77.58212033 55.93813532 72.87315319 56.87406508 58.42184602 63.58490452
67.39270243 62.48853181 64.27960775 65.9116702 64.78717236 66.51262958
63.37289315 83.10288628 72.44309605 65.48967239 64.70411742 59.03569841
61.7857038 51.85791586 63.20530253 60.40309559 71.7711146 56.65673982
53.84734893 69.93982538 66.35460223 59.19101815 64.30713302 72.71185816
63.49372172 67.84234202 59.46313429 65.60411208 70.7128183 59.59208602
80.86624431 63.98386507 62.94288431 57.12910955 59.45748301 64.17290612
60.18123895 66.63740583 56.46465455 59.59972871 67.36185194 64.51185355
60.29176782 66.08413885 70.37671861]
test num : 1411, succ num : 964, succ precent: 0.683203

```

图 4-11 朴素贝叶斯高斯模型去除 1/4 特征列数测试

输入 4，选择 SVM 模型进行测试，特征列数为 43 个，测试结果如图 4-12 所示。

```

please choose model
{'0': 'GaussianNB() to cutdown 1/4 feature', '1': 'MLPClassifier()', '2': '
3
current label col = 43
LinearSVC()
test num : 1411, succ num : 969, succ precent: 0.686747

```

图 4-12 SVM 模型 43 特征列数测试

4.2 结果记录

分别对于 5 个不同的模型，对于三个不同特征列数取值进行 10 次测试，计算测试成功率的最值，均值和方差。结果见下表

表 4-1 测试结果记录表					
	feature	99			
	count	MAX	MIN	方差	均值
Name	Method				
neural_network	MLPClassifier	0.771793	0.593196	0.002731	0.71318
linear_model	LogisticRegression	0.796598	0.686038	0.001214	0.76258
svm	LinearSVC	0.809355	0.55776	0.006436	0.70666

华中科技大学课程设计报告

neighbors	KNeighborsClassifier	0.728561	0.579731	0.003145	0.68448
naive_bayes	GaussianNB	0.733522	0.605245	0.001648	0.65811
	feature	75			
	count	MAX	MIN	方差	均值
Name	Method				
neural_network	MLPClassifier	0.716513	0.700921	0.65769	0.66052
linear_model	LogisticRegression	0.753366	0.808646	0.819986	0.6995
svm	LinearSVC	0.743444	0.641389	0.692417	0.81999
neighbors	KNeighborsClassifier	0.781006	0.705882	0.595322	0.64989
naive_bayes	GaussianNB	0.632884	0.654855	0.717222	0.6832
	feature	57			
	count	MAX	MIN	方差	均值
Name	Method				
neural_network	MLPClassifier	0.822821	0.581148	0.672573	0.61233
linear_model	LogisticRegression	0.722183	0.785259	0.783133	0.7888
svm	LinearSVC	0.557052	0.691708	0.712261	0.65131
neighbors	KNeighborsClassifier	0.804394	0.65769	0.731396	0.59391
naive_bayes	GaussianNB	0.693125	0.561304	0.664777	0.69809
	feature	43			
	count	MAX	MIN	方差	均值
Name	Method				
neural_network	MLPClassifier	0.716513	0.815025	0.647059	0.59249
linear_model	LogisticRegression	0.728561	0.734231	0.771793	0.69738
svm	LinearSVC	0.623671	0.677534	0.815734	0.63785
neighbors	KNeighborsClassifier	0.681786	0.804394	0.635011	0.61233
naive_bayes	GaussianNB	0.638554	0.705882	0.67399	0.70872

4.3 数据分析

根据上表中的数据，生成验证成功率（此后简称“值”）数据统计图表如图 4-13~图 4-16 所示。对每个图以 1.对比不同模型的差距。2.对比不同特征列数（此后简称“列数”）的差距。 3.对比不同列数对于同一模型的影响。 4.对比不同模型对于同一列数的影响。

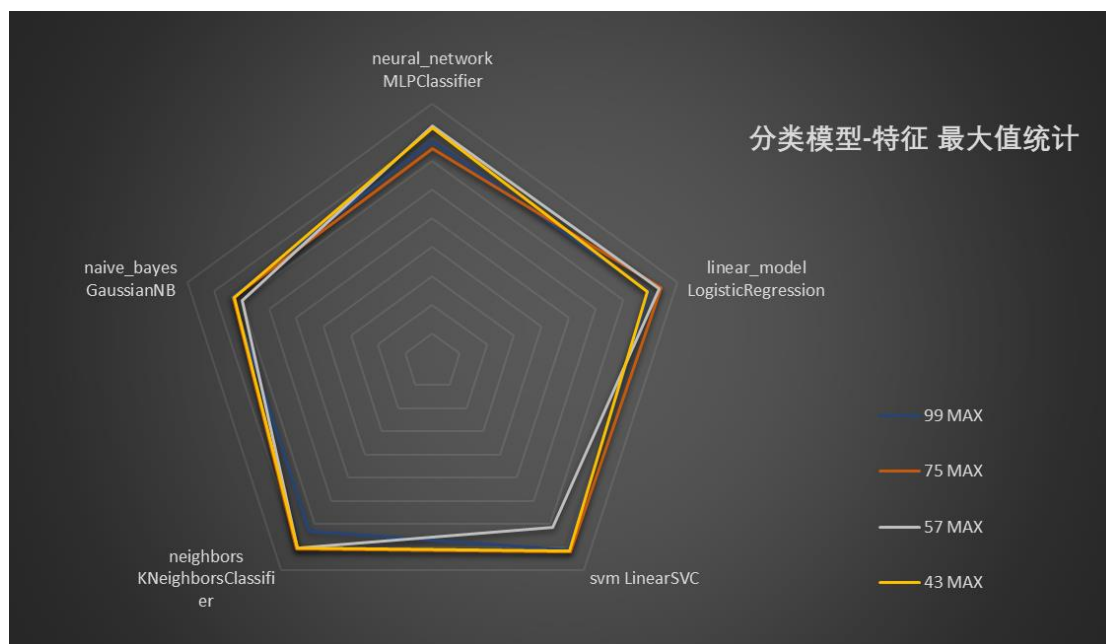


图 4-13 分类模型-特征 预测成功率最大值统计图

分析图 4-13 预测成功率最大值统计图：

1. 对比不同模型，除朴素贝叶斯模型的最大值较小外，其余四种模型的最大值基本相同。
2. 对比不同列数，可以发现去除超过一半的特征列后，特征列数为 43 时，最大值基本为每个模型不同列数的最大值中最大的。而列数为 99、75、57 时，都在某一种模型中属于最大值最小的列数值。
3. 对比不同列数对同一模型的影响，发现对于朴素贝叶斯模型和逻辑回归模型中，不同列数对其影响较小，而剩余三者中模型预测成功率的最大值在不同列数上偏差较大。
4. 对比不同模型对于同一列数的影响，可以发现对于列数为 43 和 75 的情况下，不同模型的影响较小，而对于列数为 99 和 57 的情况下，不同模型影响较大，在 SVM 模型下，列数为 99 取得最大值，列数为 57 取得最小值。

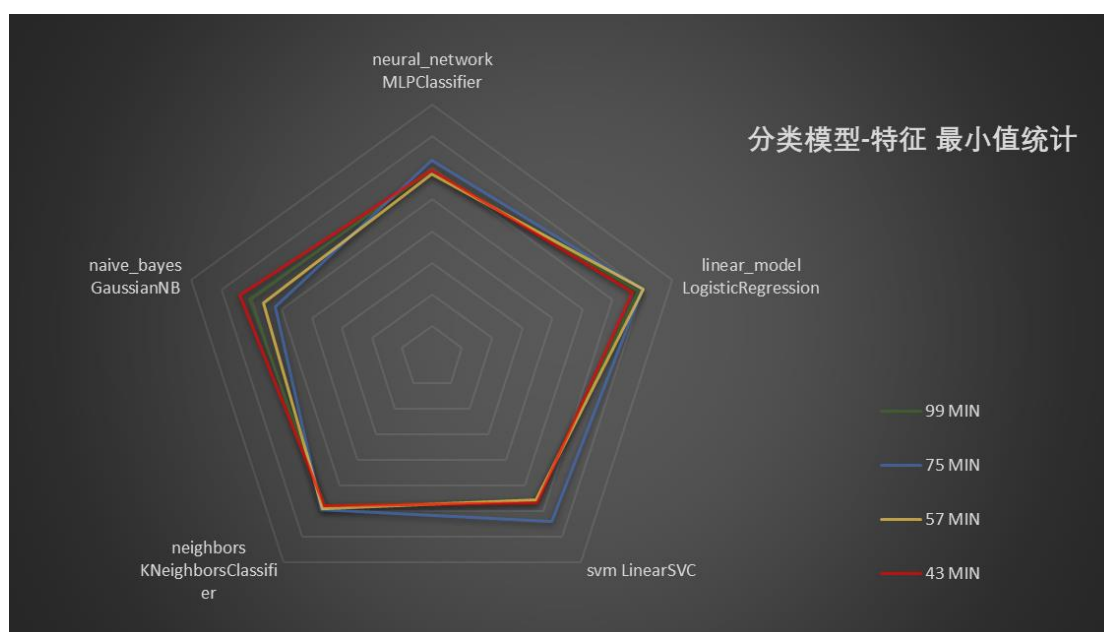


图 4-14 分类模型-特征 预测成功率最小值统计图

分析图 4-14 分类模型-特征 预测成功率最小值统计图：

1. 对比不同模型，除逻辑回归模型的最小值较大外，其余四种模型的最小值基本相同。
2. 对比不同列数，可以发现列数为 99 的情况下，最小值较为稳定。而列数为 75、57、43 的情况下，最小值波动较为严重。
3. 对比不同列数对同一模型的影响，发现对于逻辑回归、KNN 模型和神经网络模型，不同列数对模型最小值的影响较小，对于朴素贝叶斯来说，列数为 43 时，预测成功率最小值最大，列数为 75 时，最小值最小。而列数为 75 时，SVM 模型最小值最大。
4. 对比不同模型对于同一列数的影响，可以发现逻辑回归模型对在每一个列数的情况下，最小值普遍较大。在朴素贝叶斯模型上，最小值普遍较小。其余三种模型下，不同列数的情况最小值基本一致。

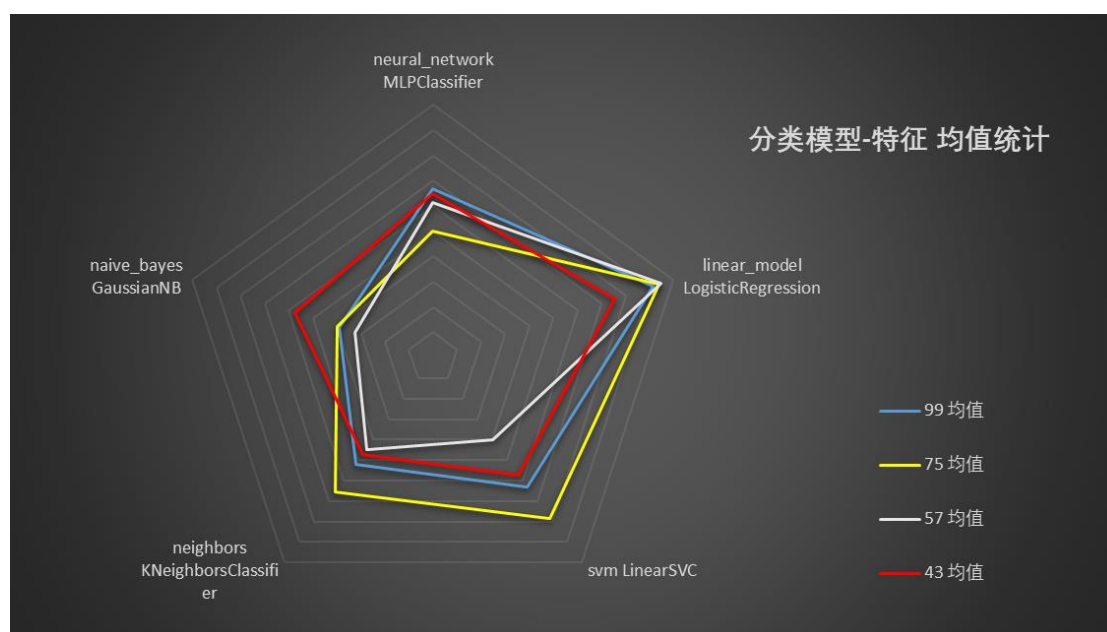


图 4-15 分类模型-特征 预测成功率均值统计图

分析图 4-15 分类模型-特征 预测成功率均值统计图

1. 对比不同模型，显著观察值逻辑回归模型的均值最大，SVM 次之，朴素贝叶斯模型均值最小，其余二者相近。
2. 对比不同列数，可以看到列数为 75 的模型均值较大，列数为 99 的模型次之，列数为 43 的模型较为稳定，列数为 57 的模型均值最小。
3. 对比不同列数对同一模型的影响，发现对于朴素贝叶斯模型和 SVM 模型中，不同列数对其均值的影响十分大，对于神经网络和 KNN 模型，不同列数对其影响较大，不同列数对逻辑回归模型有一定影响。
4. 对比不同模型对于同一列数的影响，可以看到列数为 75 时，在逻辑回归、SVM、KNN 模型下均值都取得最大值。而列数为 57 时，在朴素贝叶斯、KNN、SVM 模型下均值取得最小值。列数为 43 的模型最为稳定，列数为 57 的模型最不稳定，列数为 99 和 75 的模型稳定性较差。

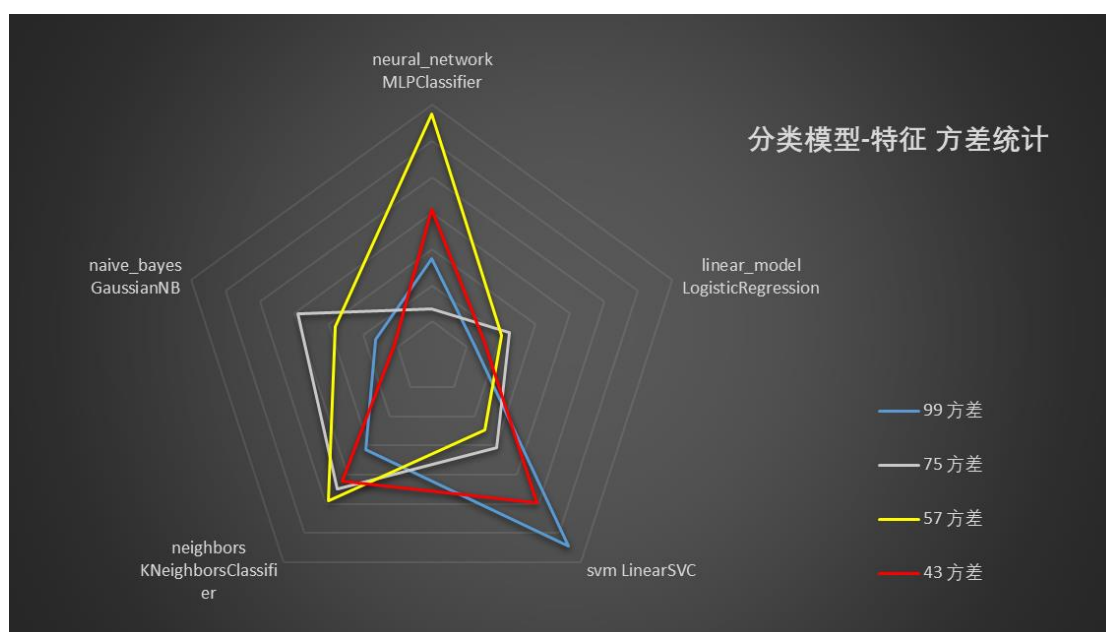


图 4-16 分类模型-特征 预测成功率方差统计图

1. 对比不同模型，除朴素贝叶斯模型的方差很小外，其余四种模型的方差波动性十分大，其中神经网络和 SVM 模型的方差最大。
2. 对比不同列数，可以发现列数为 43 时方差较小，而列数为 57 的方差较大，列数为 75 和 99 的模型预测结果方差稳定性较差。
3. 对比不同列数对同一模型的影响，发现不同列数对逻辑回归和 KNN 模型以及朴素贝叶斯模型的方差影响较小，而对 SVM 和神经网络模型的方差影响十分剧烈。
4. 对比不同模型对于同一列数的影响，可以发现对于列数为 75 的情况，不同模型对其影响较小，列数为 57 的情况，不同模型对其影响巨大。而 SVM 模型对列数为 99 时偏大，神经网络模型对列数为 57 时方差影响十分大，其余模型对列数的影响较小。

4.4 结果评估

总体而言，逻辑回归模型表现最好，其在不同的列数下预测成功率最大值较稳定，最小值较大，均值最高，方差最小，在列数为 99、75、57 时，均值大概在 76%，即使在列数为 43 时，表现最差为 73%，也比第二名神经网络模型的 70% 的均值大了 3%。

不同列数对于 SVM 模型的影响最大，当列数为 75 时，均值为 73% 仅次于逻辑回

归，且较第三名高了 3%，当列数为 57 时，均值为 65%，当列数为 99 时，方差十分大，当列数为 43 时，方差次之。

列数为 43 的模型表现出了令人难以置信的稳定性，对于同一的模型，列数为 43 的情况几乎总能取得最大的最大值和最小值，对于不同模型，列数为 43 时能取得十分稳定且并不小的成功率均值和较为稳定的测试方差。

综上所述，逻辑回归是表现最好的分类模型，但在过小的特征列情况下表现稍差；SVM 模型是最不稳定的模型，能够在较大的特征列数情况下取得最好的均值和最值，在较小的特征列数情况下取得最小的方差；KNN 模型和神经网络模型有相似之处，二者在不同的列数情况下能够取得近似的最值，当相对而言，不同的列数对 KNN 模型中均值的影响较大，对神经网络模型中方差的影响较小，因此可以通过调整参数特征列的数目的方法获取二者的最优模型；朴素贝叶斯算法高斯模型在特征列数最小的情况下表现出了十分优秀的性能，相比其余特征列数，特征列数为 43 时，朴素贝叶斯高斯模型获得最好的最值、均值和方差，因此降低样本空间维度，排除大众性特征列的方法对于朴素贝叶斯模式十分有效。

任何算法都不是一个全能的表现优良的算法，比如从算法的期望来说，本例中逻辑回归算法表现出十分优良的性质，其预测成功率的均值远大于其余算法，但在样本空间维度较小的情况下，其成功率显著下降。因此，在面对不同数据和具体的问题时，我们不应该抱残守缺，而是应该及时尝试不同的算法，这样才能获得更优的结果。

参考文献

- [1] Ken Goldberg ,goldberg at berkeley dot edu ,Prof of IEOR and EECS ,
4135 Etcheverry Hall ,University of California ,Berkeley, CA 94720-1
777 . <http://eigentaste.berkeley.edu/dataset/>
- [2] 周志华,《机器学习》,清华大学出版社
- [3] 数据挖掘概念和技术, Jiawei Han, Micheline Kamber, Jian Pei 著, 范明、孟小
峰 译 2012.7
- [4] 人工神经网络: https://en.wikipedia.org/wiki/Artificial_neural_network