《人工智能在医学影像技术中的应用》实验报告

姓名：杨浩然	学号：221180053

一、 实验目的

1.通过选择有监督学习：

（1）logistic回归

（2）K近邻算法（KNN）

（3）支持向量机（SVM）

（4）随机森林（Random Forest）

无监督学习：

（1）K平均算法（K-Means）

来比较分析不同机器学习算法在sklearn的breast_cancer数据集上的表现。

2.通过使用三种不同的超参数调优方法：

（1）网格搜索法（Grid Research）

（2）随机搜索法（Random Research）

（3）贝叶斯优化（Bayesian Optimization）

来比较其调优速度和调优效果。

3.通过五种数据预处理方法：

（1）数据集划分——十折交叉验证（分层划分）；随机划分；

（2）标准化——(logictic回归，SVM，KNN，K-means)

（3）归一化——(logictic回归，SVM，KNN，K-means)

（4）数据清洗——(去除缺失值和异常值)

（5）高维数据进行降维——(主成分分析法（PCA），变分编码器（VAE），特征提取等)

总结预处理对于最后训练结果和训练速度的影响。

4.通过三种方法

（1）绘制ROC曲线

（2）计算AUC面积

（3）打印混淆矩阵分析准确率、精确率、召回率和F1分数

来对模型性能进行评估

二、 实验原理与方法

2.1 数据集特征

2.1.1 精简数据集Prostate_Cancer

该数据集共有9个维度的特征，其中diagnosis是分类问题的标签，后面8个维度是分类或者是预测问题中所输入的特征。

diagnosis_result：诊断结果

radius：半径

texture：纹理

perimeter：周长

area：面积

smoothness：光滑度

compactness：紧凑度

symmetry：对称性

fractal_dimension：分形维度

2.1.2 原数据集Sklearn_BreastCancereWisconsinDataSet

该数据集共有30个维度的特征，是一个高维度的数据集，可以分为三组：

（1）Mean（平均值）：表示肿瘤的常规统计信息，如平均半径、平均纹理等。

（2）Error（误差）：表示每个特征的标准误差（即变化程度）。

（3）Worst（最差）：表示肿瘤在所有测量中的最大值或最差表现，如最差半径、最差纹理等。

通过输入特征，进行回归就可以预测患者的良性或者恶性。

具体特征列举如下：

mean radius：肿瘤的平均半径

mean texture：肿瘤的平均纹理

mean perimeter：肿瘤的平均周长

mean area：肿瘤的平均面积

mean smoothness：肿瘤的平均光滑度

mean compactness：肿瘤的平均紧凑度

mean concavity：肿瘤的平均凹陷度

mean concave points：肿瘤的平均凹点数

mean symmetry：肿瘤的平均对称性

mean fractal dimension：肿瘤的平均分形维度

radius error：半径的标准误差

texture error：纹理的标准误差

perimeter error：周长的标准误差

area error：面积的标准误差

smoothness error：光滑度的标准误差

compactness error：紧凑度的标准误差

concavity error：凹陷度的标准误差

concave points error：凹点数的标准误差

symmetry error：对称性的标准误差

fractal dimension error：分形维度的标准误差

worst radius：肿瘤的最差半径

worst texture：肿瘤的最差纹理

worst perimeter：肿瘤的最差周长

worst area：肿瘤的最差面积

worst smoothness：肿瘤的最差光滑度

worst compactness：肿瘤的最差紧凑度

worst concavity：肿瘤的最差凹陷度

worst concave points：肿瘤的最差凹点数

worst symmetry：肿瘤的最差对称性

worst fractal dimension：肿瘤的最差分形维度

2.2 有监督学习和无监督学习

2.2.1 算法特点

有监督学习：输入和输出已知，数据集中每个输入样本都对应一个已知的标签（输出）。目标是通过学习输入和输出之间的关系来预测新的、未见过的数据的标签。算法通过使用标注数据集（输入与输出对）来“训练”，并通过优化模型使预测输出尽可能接近实际标签。最后可以通过误差度量（如准确率、精度、ROC曲线，AUC）来评估模型的性能。

无监督学习：与有监督学习不同，无监督学习的数据集没有标签（即没有预定义的输出）。目标是发现数据中的模式、结构或规律。算法根据输入数据本身的特征（而不是标签）来进行学习，尝试将数据聚类、降维或提取特征。模型的评估常通过聚类效果、可视化效果等间接评估。

2.2.2 适用场景

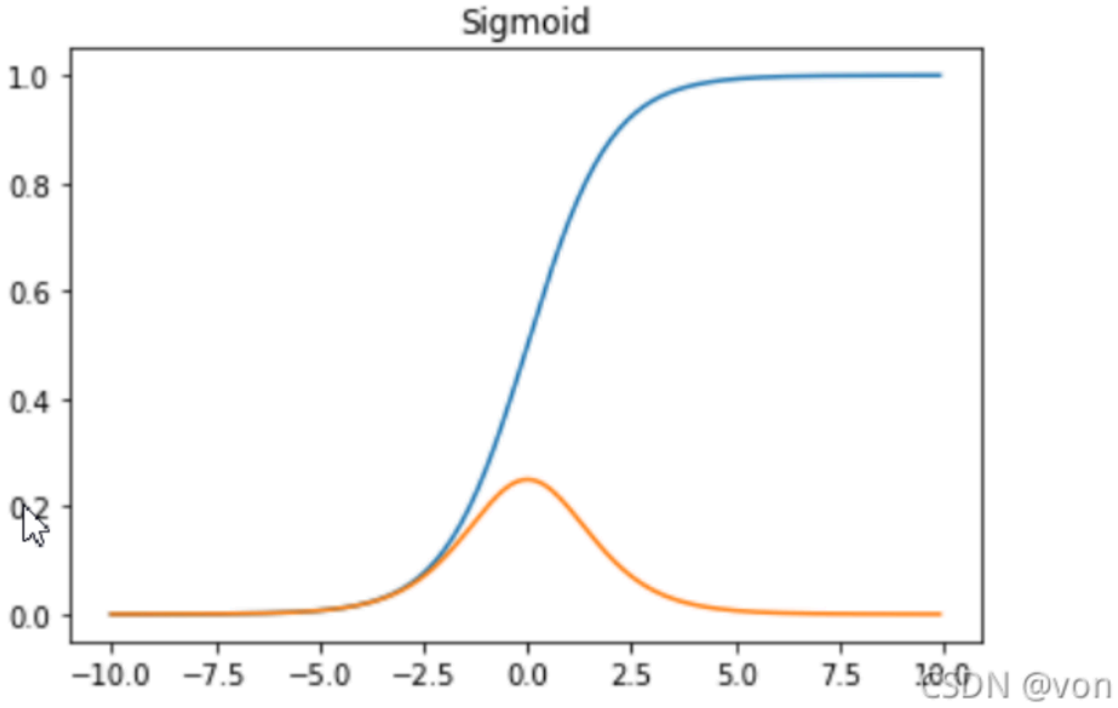
有监督学习：(1)分类问题：例如垃圾邮件分类（是否是垃圾邮件），病人是否患有某种疾病（疾病诊断），图片的物体分类(猫、狗等); (2）回归问题：例如预测房价（基于特征如面积、位置等），预测股票价格，天气预测等。

无监督学习：(1)聚类问题：当没有标签时，希望根据样本的特征进行分组，发现自然聚类;(2)降维问题：当数据维度过高时，可以用降维技术将数据映射到低维空间，保留数据的关键信息。(3)关联规则：购物篮分析、推荐系统、基于用户行为的广告推送。

2.3 算法基本原理

2.3.1 Logistic回归（Logistic Regression）

基本原理： Logistic回归是一种用于二分类问题的广泛使用的统计学模型，其基本思想是通过一个**Sigmoid函数**（也叫逻辑函数）将线性回归的输出映射到概率值上。它用于预测某个事件发生的概率。



线性回归模型：首先，假设我们通过线性回归得到一个输出：

$$z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

其中，$w_1, w_2, \ldots, w_n$ 是模型的权重，$x_1, x_2, \ldots, x_n$ 是特征(二分类问题中X取值为$0/1$)，b 是偏置。Z的值表示了X取1的值在W和b的映射下的输出。

Sigmoid函数：Logistic回归通过将线性输出 z 传入Sigmoid函数将其转换为概率值：

$$p = \frac{1}{1 + e^{-z}}$$

其中，p 是类别1的预测概率，$1 - p$ 就是类别0的预测概率。

模型训练：使用**最大似然估计**（MLE）来训练模型，通过最小化代价函数（交叉熵损失函数）来优化模型参数。

输出：根据计算出的概率值 p，如果 $p \geq 0.5$，预测类别为1；否则，预测类别为0。

APPENDIX2.3.1：最大似然估计与交叉熵

TIP1：最大似然估计

最大似然估计是一种统计方法，用于根据观察到的数据推断模型参数。简单来说，MLE的目标是找到一组参数，使得在这些参数下，观测到的数据的出现概率（或似然）最大。具体步骤如下：

定义似然函数：首先，根据所用的统计模型，定义似然函数。似然函数通常表示为 $L(\theta|x)$，其中 θ 是模型参数，x 是观察到的数据。似然函数实际上表示在参数为 θ 时，观测到数据 x 的概率。

最大化似然函数：然后，通过选择合适的参数 θ\thetaθ，使得似然函数的值最大化。这通常是通过对似然函数取对数（得到对数似然函数）来简化计算。

对数似然函数通常写作：$\ell(\theta|x) = \log L(\theta|x)$

通过最大化对数似然函数，可以得到估计的参数值。

TIP2：交叉熵（Cross-Entropy）

交叉熵是一种衡量两个概率分布之间差异的常用指标，尤其在机器学习中，常用于分类问题的损失函数。交叉熵衡量的是实际分布和预测分布之间的“距离”，它越小，表示模型预测的概率分布与真实分布越接近。

交叉熵可以表示为：

$$H(p,q) = -\sum_{x} p(x) \log q(x)$$

其中：

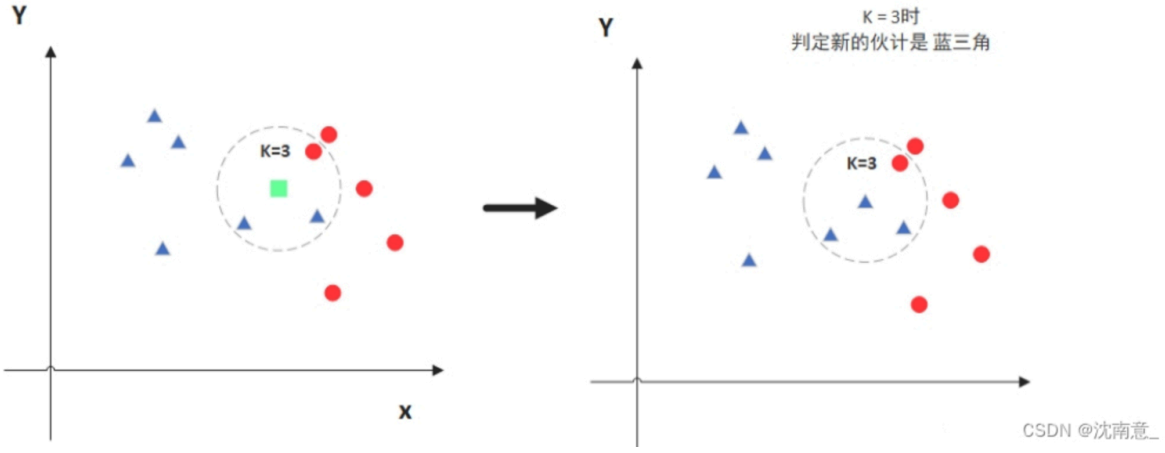
$p(x)$ 是实际分布（通常是标签的真实分布）。

$q(x)$ 是预测分布（通常是模型的输出概率）。

交叉熵可以理解为：若你的模型预测概率分布是 $q(x)$，则交叉熵度量了你模型预测的分布和真实分布 p(x)p(x)p(x) 之间的差异。如果预测完全正确（即 $q(x) = p(x)$），交叉熵的值会非常小（最小为 0）。如果预测不准确，交叉熵则会较大。

2.3.2 K近邻算法（K-Nearest Neighbors, KNN）

基本原理： K近邻算法是一种基于实例的监督学习算法，其基本思想是：给定一个新的样本，通过找到训练集中与该样本最接近的K个邻居，并根据这些邻居的类别信息来决定新样本的类别。



计算距离：通过一定的距离度量（如欧氏距离、曼哈顿距离等）计算新样本与训练集中每个样本之间的距离。

选择K个邻居：选择距离新样本最近的K个样本。

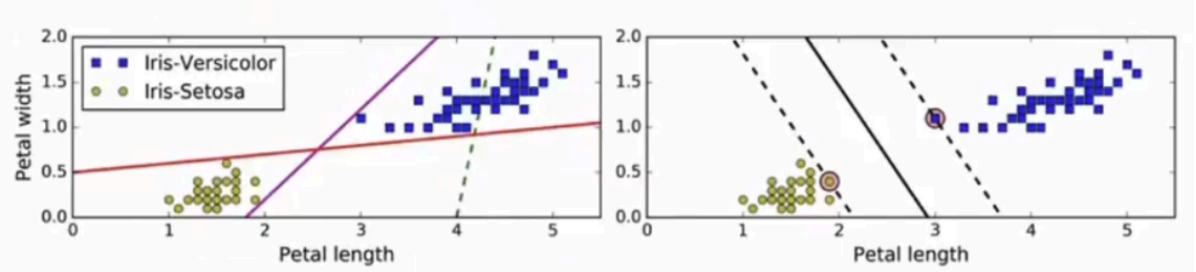
投票决定类别：通过多数投票法（分类问题）或平均值法（回归问题）确定新样本的类别或值。

距离计算：对于两个样本 $x = (x_1, x_2, \ldots, x_n)$和$y = (y_1, y_2, \ldots, y_n)$，欧氏距离公式为：

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

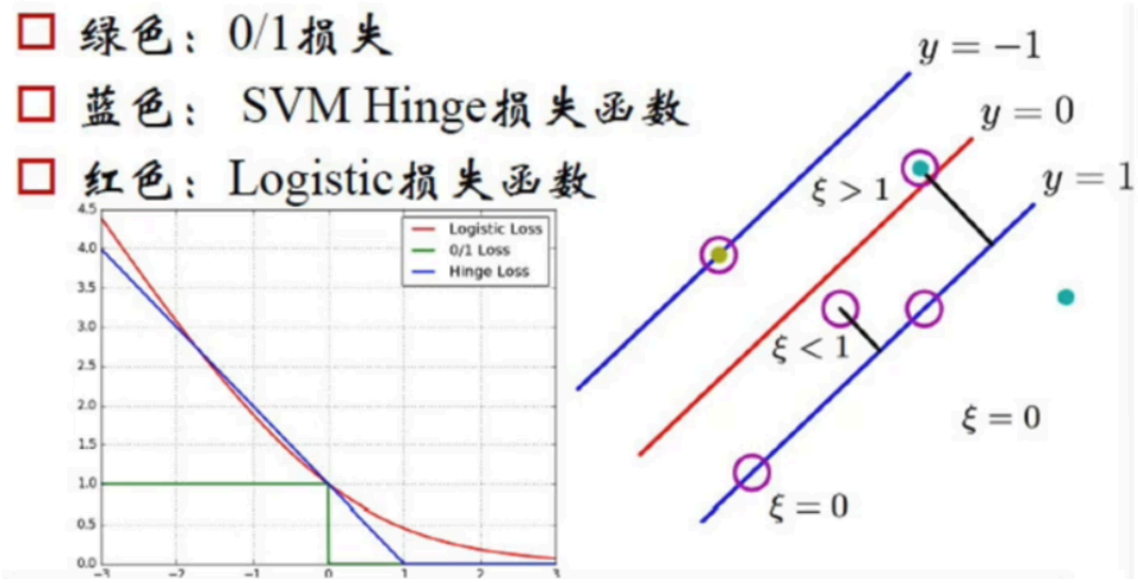
2.3.3 支持向量机（Support Vector Machine, SVM）

基本原理： 支持向量机是一种强大的监督学习算法，常用于二分类问题。其基本思想是通过寻找一个最佳的超平面来最大化分类边界，以实现最佳的分类效果。



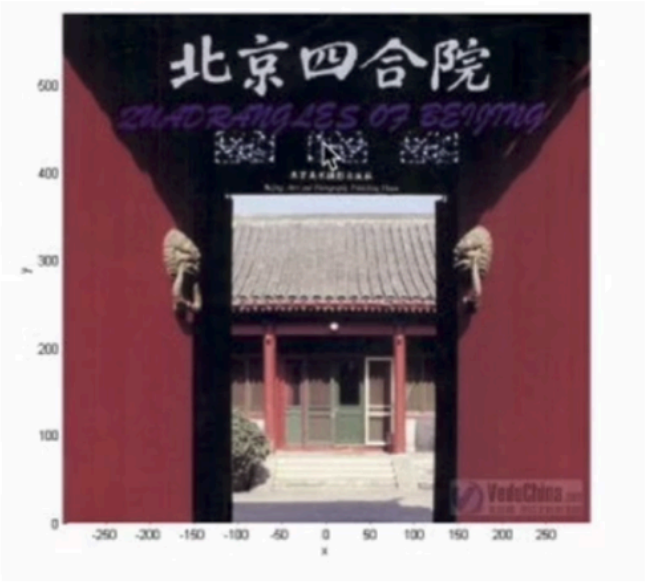
线性可分问题：对于线性可分的样本，SVM会寻找一个超平面将两类样本分开，并最大化该超平面到各类样本的最小距离。这个最小距离称为**间隔**（Margin）。

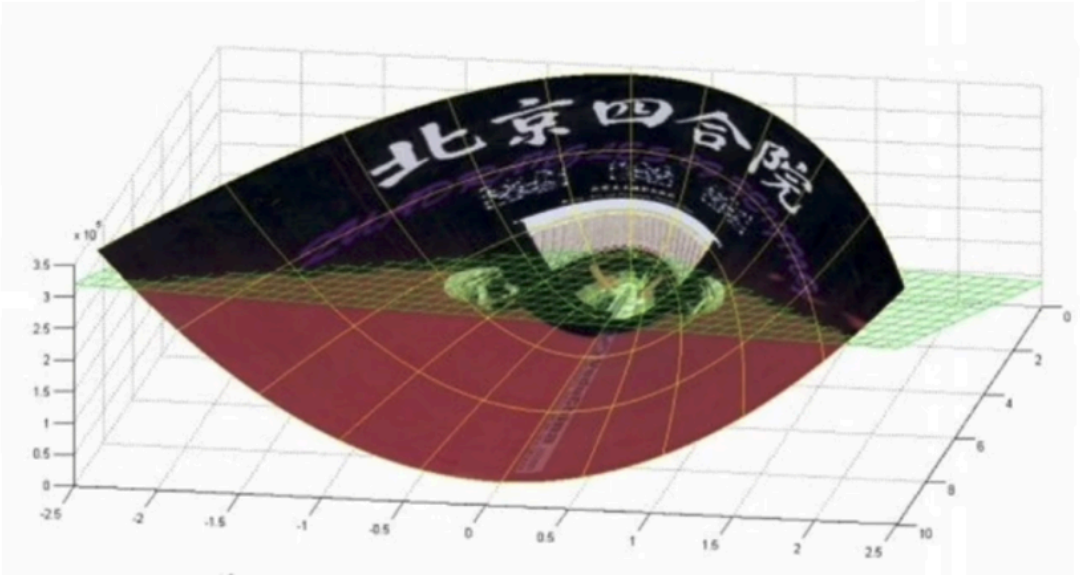
损失函数：主要讨论三种损失函数：（1）0/1损失（若分类正确损失为0，若分类错误损失为1）（2）SVM Hinge损失函数（3）Logistic损失函数【（2），（3）方法相比于（1）更加具有精确性，所有的点距离超平面越近则损失越大，距离超平面越远则损失越小】

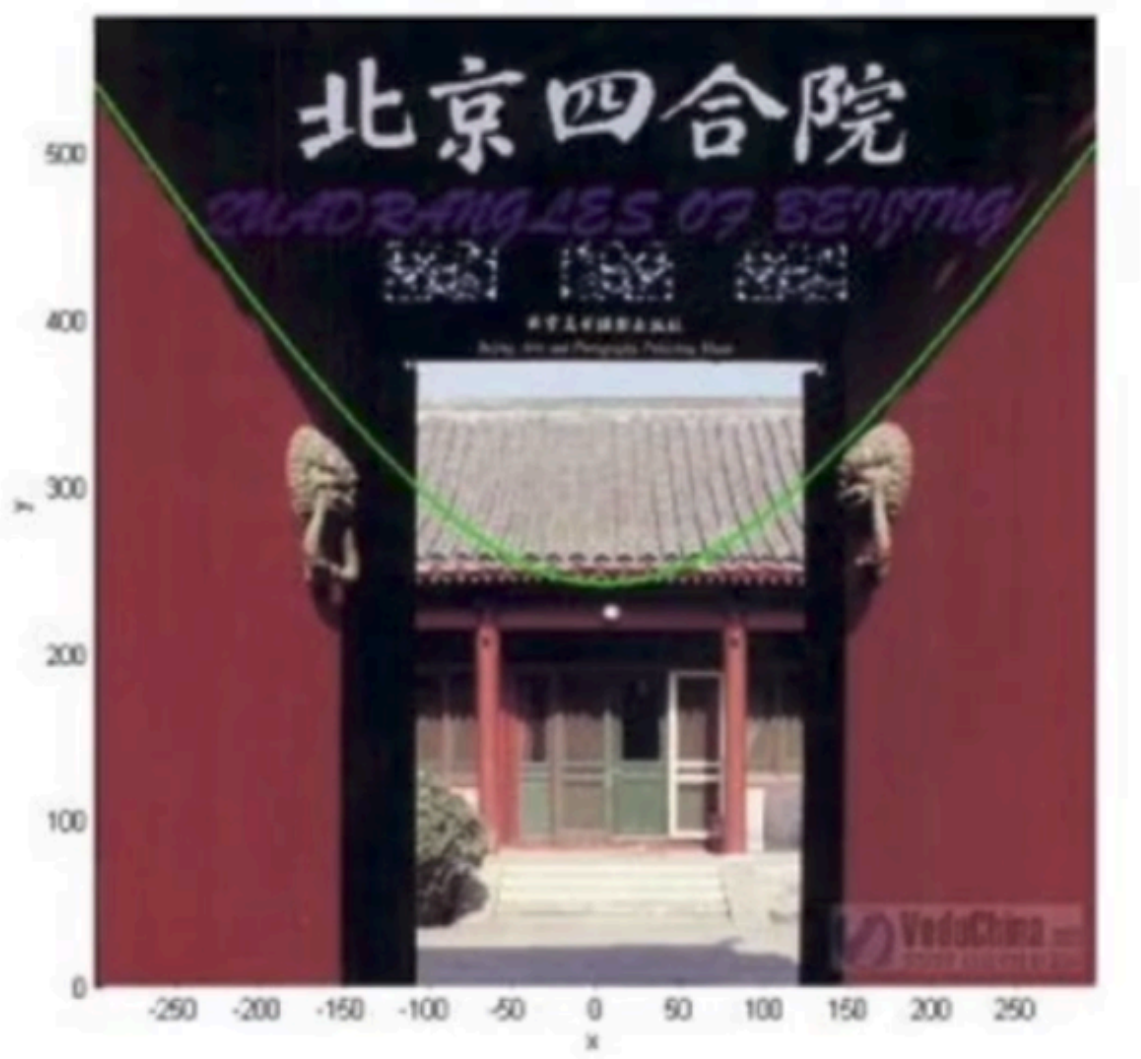


支持向量：在寻找最优超平面的过程中，最靠近超平面的样本点被称为**支持向量。**这些支持向量对确定分割超平面起决定性作用。

核函数：当数据不是线性可分时，SVM通过引入**核函数**（如高斯核、线性核、多项式核等）将数据映射到高维空间，使其在高维空间中线性可分。（important）


北京四合院
QUADRANGLES OF BEIJING
500
400
300
200
100
0
y
-250 -200 -150 -100 -50 0 50 100 150 200 250
x

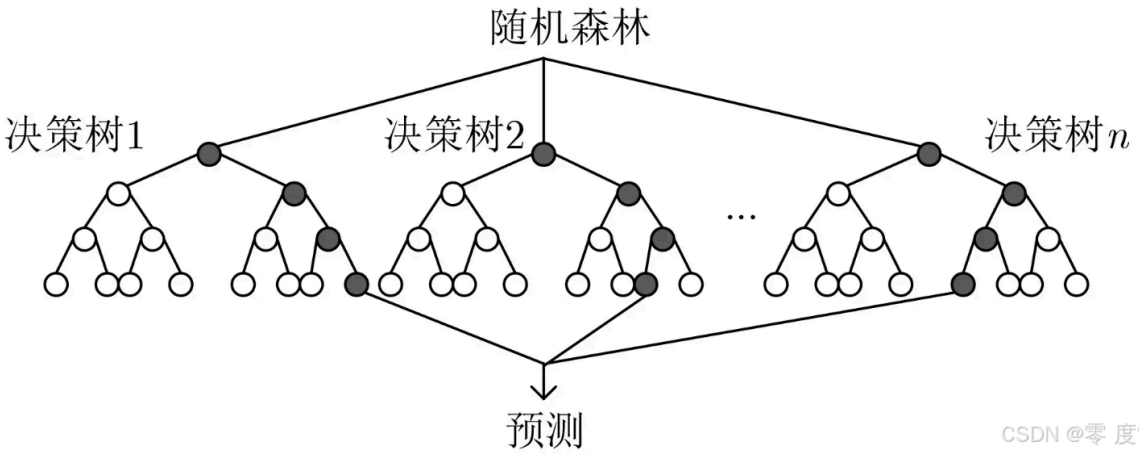

北京四合院
x 10^5
3.5
3
2.5
2
1.5
1
0.5
0
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
0 2 4 6 8 10



优化问题：SVM通过求解一个优化问题来最大化间隔，通常通过拉格朗日乘数法来求解。

2.3.4 随机森林（Random Forest）

基本原理： 随机森林是一种集成学习方法，通过构建多个决策树并对其结果进行投票或平均来提高模型的预测精度。



决策树：随机森林的基本单元是决策树。每个决策树通过递归地选择特征进行数据划分，直到满足停止条件（如节点纯度或最大深度）。

随机性：

（1）**数据集随机性**：在训练每一棵决策树时，随机森林从训练集中**有放回地抽取样本**（即引入Bootstrap抽样）。

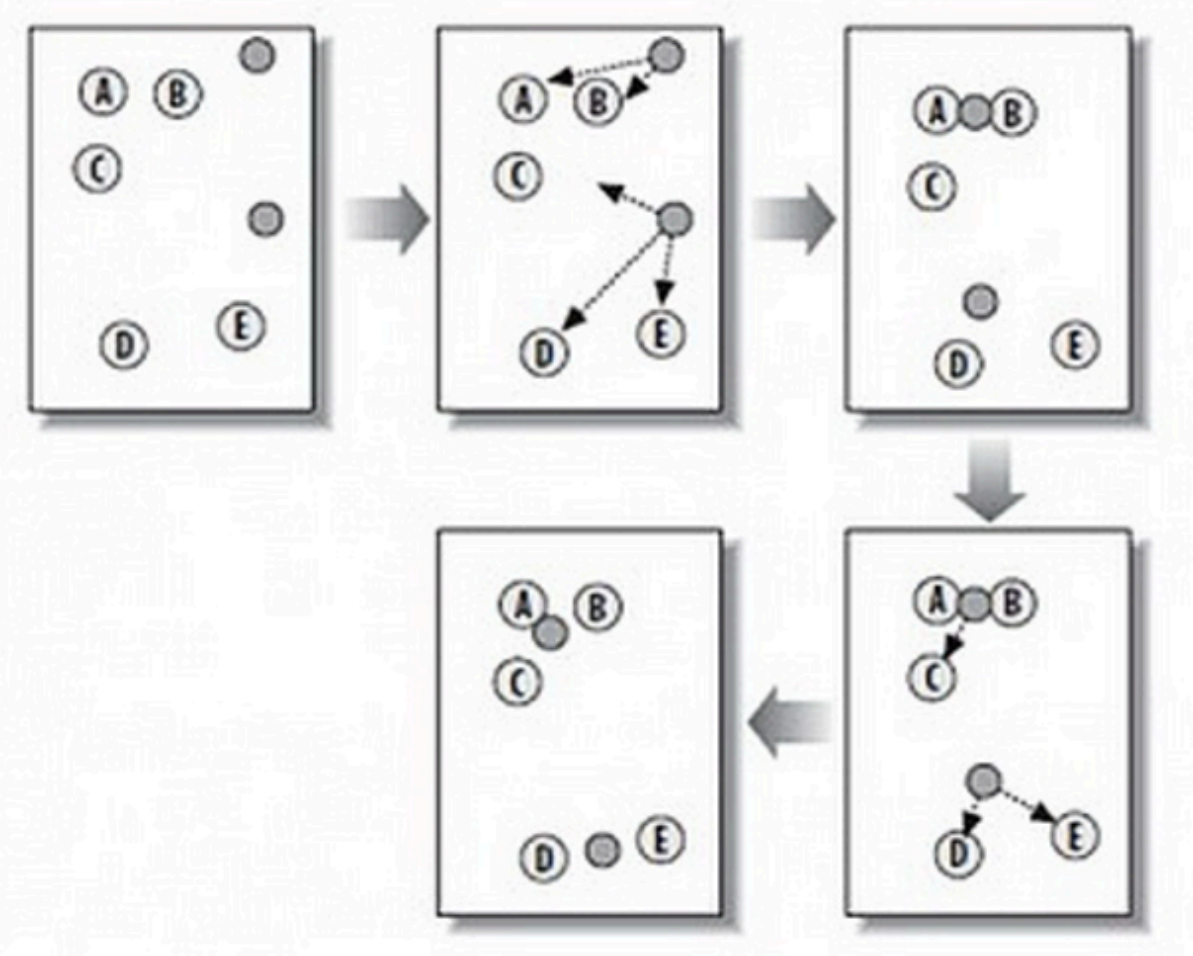
（2）**特征随机性**：每个决策树在划分节点时，并不是考虑所有特征，而是从所有特征中**随机选择一部分**特征进行分裂。

分类问题：通过投票的方式对每棵树的预测结果进行投票，得到最终的分类结果。

回归问题：通过对每棵树的预测结果取平均来获得最终结果。

2.3.5 K均值算法（K-Means）

基本原理： K均值算法是一种无监督学习算法，主要用于数据聚类（Clustering）。其目标是将数据集划分为K个簇，每个簇中的数据点具有相似性。



初始化：随机选择K个初始聚类中心（也叫质心）。

分配数据点：将每个数据点分配给距离最近的聚类中心，形成K个簇。

更新质心：对于每个簇，计算簇中所有数据点的均值，将这个均值作为新的聚类中心。

迭代：重复上述步骤，直到质心不再变化或者变化很小为止，算法收敛。

损失函数：K均值算法的目标是最小化簇内平方误差（SSE, Sum of Squared Errors），即最小化数据点到其所属聚类中心的距离之和。

$$J = \sum_{i=1}^{K} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

其中，μ_i 是第i个簇的聚类中心，x_j 是簇中数据点。

三、 实验步骤与结果

3.1 数据预处理

3.1.1 数据集划分

（1）logistic回归

Logistic回归模型的训练集和测试集划分是通过随机选择的。

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
random_state=42)
```

（2）K近邻算法（KNN）

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
random_state=42)
```

KNN是一个基于实例的学习方法，它通过计算样本之间的距离来进行分类或回归。KNN模型的训练和测试过程通常也遵循相同的训练集和测试集划分。

（3）随机森林（Random Forest）

```
clf = DecisionTreeClassifier(criterion = "entropy", random_state=90)
```

在训练时，每棵树会使用数据集的不同子集（通常是通过“自助采样法”或Bootstrapping采样生成），在模型评估时，将数据集划分为训练集和测试集。

（4）K平均算法（K-Means）

```
kms = KMeans(n_clusters=3, n_init='auto')
```

初始化中心点，然后通过迭代调整簇中心来优化聚类结果。

3.1.2 标准化与归一化

```
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

3.1.3 降维

```
pca = decomposition.PCA()

pca.fit(X)

explained_variance = pca.explained_variance_ratio_

cumulative_variance = np.cumsum(explained_variance)
```

初始化PCA模型并拟合数据X，计算解释方差比和累计解释方差比

```
plt.figure(figsize=(8, 6))

plt.plot(range(1, len(explained_variance) + 1), explained_variance, 'o-',
label='Individual Explained Variance')

plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance, 's-',
label='Cumulative Explained Variance')
```

绘制陡坡图，确定索要选取的主成分的数量

```
pca = decomposition.PCA(n_components=2)

reduced_X = pca.fit_transform(X)
```

对数据进行降维

3.1.4 radiomics影像组学特征提取

特征提取

```
featureVector = extractor.execute(imgname,maskname)
```

执行统计检验（验证所提取的特征是否可以有效的反应输出结果并做特征选择）

```
index=[]

for col in columns_:

  if levene(data_A_[col],data_B_[col])[1] > 0.05:  # 方差齐性，选择T检验

     if ttest_ind(data_A_[col],data_B_[col])[1] < 0.05:

        index.append(col)

  else:

     if ttest_ind(data_A_[col],data_B_[col],equal_var=False)[1] < 0.05:
           # 方差不齐，使用Welch T检验

        index.append(col)
```

APPENDIX3.1 假设检验

TIP1. T检验（Student's t-test）

T检验是用于比较两个独立样本的均值差异是否显著的经典统计方法，特别适用于样本量较小且假设数据符合**正态分布**的情况。T检验有几个重要的假设：

正态性假设：数据应该符合正态分布。

方差齐性假设：两个样本的方差应该相等。

基于这两个假设，T检验的计算公式如下：

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

其中：

$\bar{X}_1, \bar{X}_2$：两个样本的均值。

s_1^2, s_2^2：两个样本的方差。

n_1, n_2：两个样本的样本量。

TIP2. Welch检验（Welch's t-test）

Welch检验是T检验的一个改进版本，专门用于处理两个样本方差不相等的情况。相比传统的T检验，Welch检验对方差齐性的要求较低，适用于方差不相等的情况。

Welch检验没有要求两个样本具有相同的方差，因此它更为通用，特别是在处理不满足方差齐性假设的实际数据时。

Welch检验的t统计量计算公式与T检验类似，但它对方差不等的情况进行了调整，公式为：

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

但是，Welch检验使用的是**调整后的自由度**，自由度的计算公式为：

$$df = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{1}{n_1-1}\left(\frac{s_1^2}{n_1}\right)^2 + \frac{1}{n_2-1}\left(\frac{s_2^2}{n_2}\right)^2}$$

这里的自由度通常是一个浮动值，不像传统T检验那样使用固定的整数值。

3.2 模型选择与建立

3.2.1 logistic回归

```
model = LogisticRegression(max_iter=1000)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

3.2.2 K近邻算法（KNN）

```
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
```

3.2.3 支持向量机（SVM）

```
class SVM:

  def __init__(self, learning_rate=0.001, lambda_param=0.01, n_iters=1000):

```

```
    self.lr = learning_rate

    self.lambda_param = lambda_param

    self.n_iters = n_iters

    self.w = None

    self.b = None

  def fit(self, X, y):

    n_samples, n_features = X.shape

    y_ = np.where(y <= 0, -1, 1)

    self.w = np.zeros(n_features)

    self.b = 0

    for _ in range(self.n_iters):

      for idx, x_i in enumerate(X):

        condition = y_[idx] * (np.dot(x_i, self.w) - self.b) >= 1

        if condition:

          self.w -= self.lr * (2 * self.lambda_param * self.w)

        else:

          self.w -= self.lr * (2 * self.lambda_param * self.w - np.dot(x_i,
y_[idx]))

          self.b -= self.lr * y_[idx]

  def predict(self, X):

    approx = np.dot(X, self.w) - self.b

    return np.sign(approx)
```

`n_samples, n_features = X.shape`：获取训练数据的样本数和特征数。

`y_ = np.where(y <= 0, -1, 1)`：将标签转换为 -1 和 1，以便于 SVM 算法处理。

然后，代码进入一个循环，迭代 `self.n_iters` 次：

`condition = y_[idx] * (np.dot(x_i, self.w) - self.b) >= 1`：检查当前样本是否被正确分类。

如果 `condition` 为真，则权重向量 `self.w` 减去学习率 `self.lr` 乘以 `2 * self.lambda_param * self.w`。（防止过拟合）

如果 `condition` 为假，则权重向量 `self.w` 减去学习率 `self.lr` 乘以 `(2 * self.lambda_param * self.w - np.dot(x_i, y_[idx]))`，并且偏置项 `self.b` 减去学习率 `self.lr` 乘以 `y_[idx]`。（权重，偏置更新）

3.2.4 随机森林（Random Forest）

生成决策树

```
lf = DecisionTreeClassifier(criterion = "entropy", random_state=90)

clf.fit(data.data, data.target)
```

生成随机森林

```
rfc = RandomForestClassifier(n_estimators=100, random_state=90)
```

3.2.5 K平均算法（K-Means）

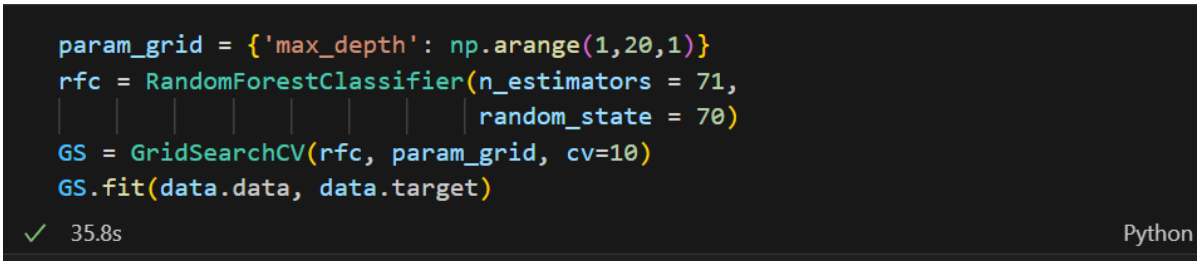
```
kms = KMeans(n_clusters=3, n_init='auto')

cluster = kms.fit_predict(reduced_X)
```

3.3 超参数优化

以随机森林，最大深度参数调优为例，通过实验对比发现网格搜索法的时间花费大于随机搜索，但是两种方法有时会得到不同的超参数，最后平均来说效果上网格搜索所得的超参数的准确率略大于随机搜索。

3.3.1 网格搜索法（Grid Research）

```
param_grid = {'max_depth': np.arange(1,20,1)}
#交叉验证
rfc = RandomForestClassifier(n_estimators = 71,
                             random_state = 70)
GS = GridSearchCV(rfc, param_grid, cv=10)
GS.fit(data.data, data.target)
```



```
GS.best_params_
```

✓ 0.0s Python

```
{'max_depth': 7}
```

```
GS.best_score_
```

✓ 0.0s Python

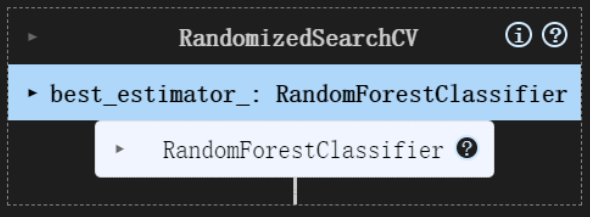
```
0.9666666666666666
```

3.3.2 随机搜索法（Random Research）

```
1  lasso = Lasso()
2
3  param_dist = {'max_depth': np.arange(1, 20, 1) }
4
5  rfc = RandomForestClassifier(n_estimators = 71,
6
7                  random_state = 70)
8
9  GS = RandomizedSearchCV(rfc,param_dist,cv=10)
10
11 GS.fit(data.data, data.target)
```

```
lasso = Lasso()
param_dist = {'max_depth': np.arange(1, 20, 1) }
rfc = RandomForestClassifier(n_estimators = 71,
                             random_state = 70)
GS = RandomizedSearchCV(rfc,param_dist,cv=10)
GS.fit(data.data, data.target)
```

✓ 16.8s Python




```
GS.best_params_
```

✓ 0.0s Python

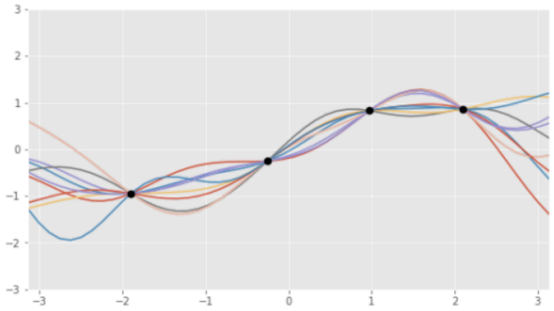
```
{'max_depth': 14}
```

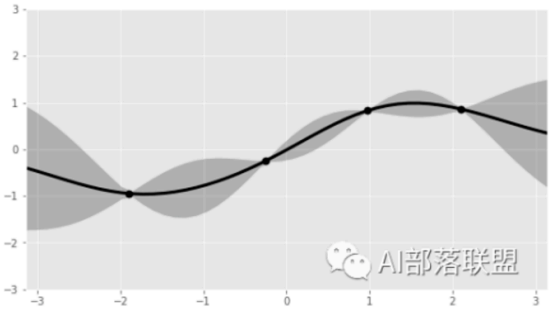
```
GS.best_params_
✓ 0.0s                                                    Python
{'max_depth': np.int64(11)}

GS.best_score_
✓ 0.0s                                                    Python
np.float64(0.9649122807017543)
```

APPENDIX3.3 贝叶斯优化与算法优化

TIP1：贝叶斯优化

基本原理： 贝叶斯优化是一种基于概率模型的全局优化方法，尤其适用于高计算成本的黑箱优化问题。在贝叶斯优化中，优化目标函数通常是未知的，且计算代价很高，因此它通过建立目标函数的概率模型来指导搜索过程，从而减少函数评估的次数。





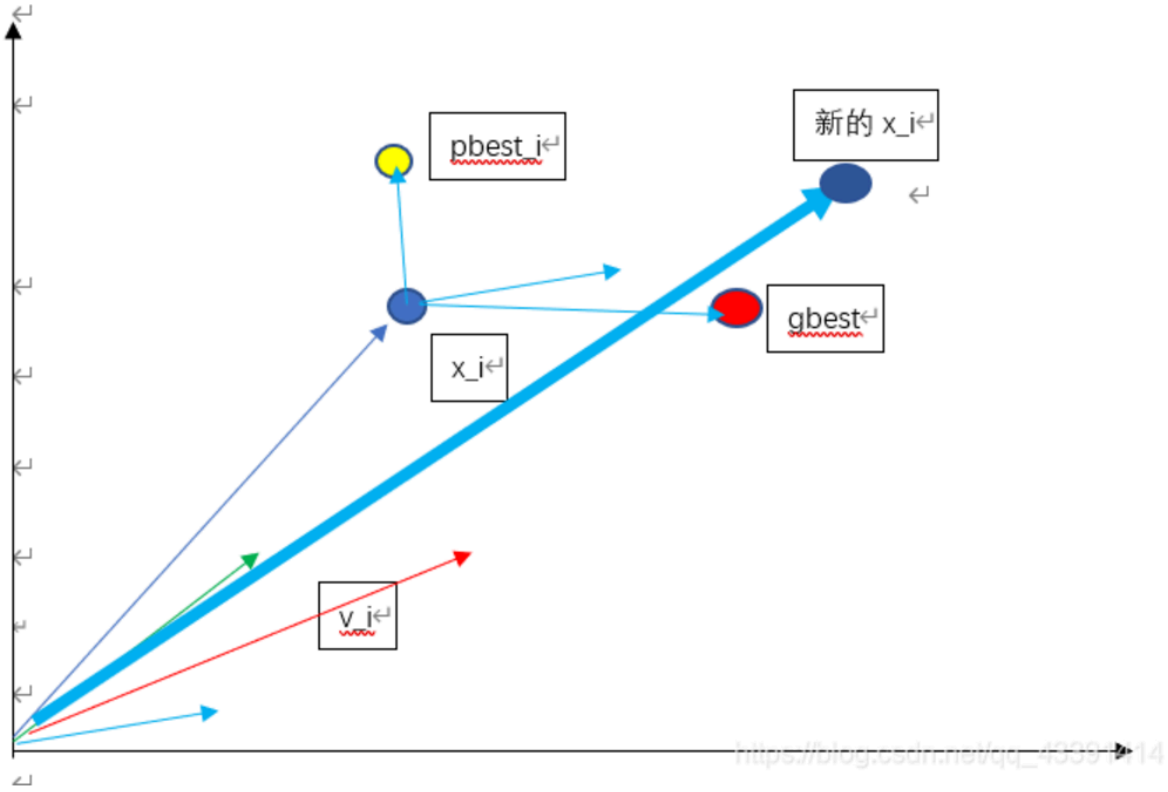
贝叶斯优化的流程如下：

1. **选择代理模型（通常为高斯过程）**：使用代理模型来近似目标函数。高斯过程可以通过给定数据点来预测目标函数的值，并为每个预测提供一个置信度（即预测的不确定性）。
2. **获取目标函数的预测分布**：根据代理模型预测未评估点的目标函数值，同时评估不确定性。
3. **选择优化策略**：通常使用 **采集函数**（acquisition function）来选择下一个评估点。常见的采集函数有期望改进（EI）、概率改进（PI）等，这些函数基于当前代理模型评估哪些点最可能带来较大的改进。
4. **更新模型**：在选择一个新的点后，使用该点的目标函数值更新代理模型，进行下一轮优化。

贝叶斯优化的优势是通过模型的反馈逐步逼近最优解，且相较于传统的网格搜索或随机搜索，能够在较少的评估次数下找到全局最优解。

TIP2：粒子群算法

基本原理： 粒子群算法（PSO）是一种模拟群体智能的优化方法，灵感来源于鸟群觅食行为。在PSO中，一组“粒子”代表潜在的解，每个粒子的位置代表一个参数的组合，而粒子的速度决定了该解在搜索空间中移动的方式。



PSO的流程如下：

1. **初始化**：随机生成一组粒子，每个粒子的位置和速度都初始化。每个粒子有一个当前位置（解）和一个速度（方向与步长），并根据当前粒子的位置评估目标函数的值。
2. **更新粒子的速度和位置**：每个粒子的运动由两个部分决定：

   个体经验：粒子会向其历史最佳位置（个体最佳位置）靠近。

   群体经验：粒子会向整个群体中表现最好的粒子（全局最佳位置）靠近。

   速度更新公式为：

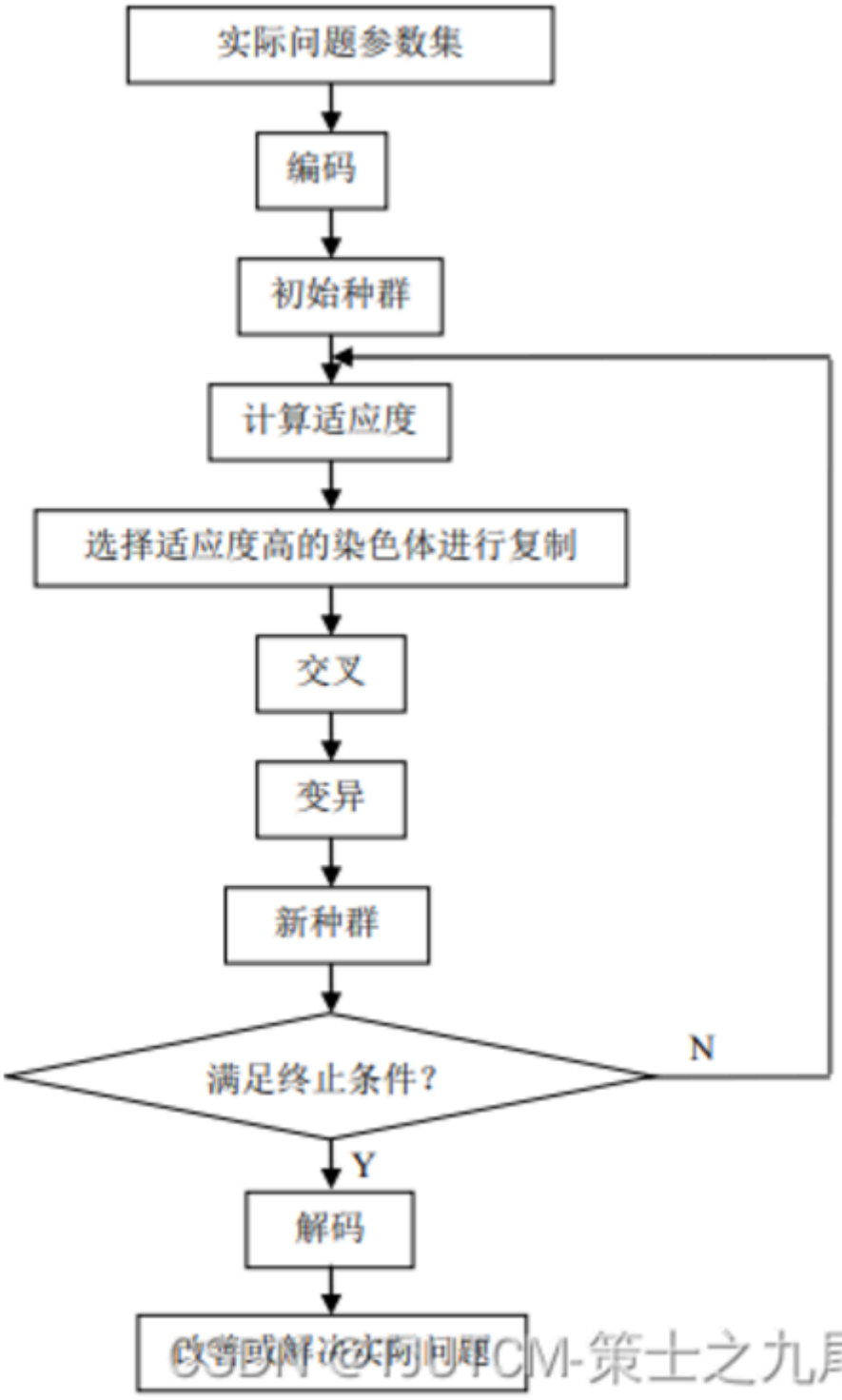
   $$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pbest_i - x_i) + c_2 \cdot r_2 \cdot (gbest - x_i)$$

   其中，w 是惯性权重，c_1 和 c_2 是学习因子，r_1 和 r_2 是随机数，$pbest_i$ 和 gbestgbestgbest 分别是粒子个体的历史最佳位置和全局最佳位置，x_i 是粒子当前的位置。
3. **评估与更新**：在每一步，粒子的当前位置会根据目标函数值进行评估，如果当前位置比之前的最优解好，就更新粒子的历史最佳位置。

PSO通过全局和局部的搜索策略，不断调整粒子的速度和位置，逐步逼近最优解。

TIP3：遗传算法

基本原理： 遗传算法（GA）是一种模拟自然选择和遗传学原理的优化方法，基于生物进化过程中的“适者生存”思想。GA通过模拟自然选择中的交叉、变异、选择和遗传等过程，寻找问题的最优解。



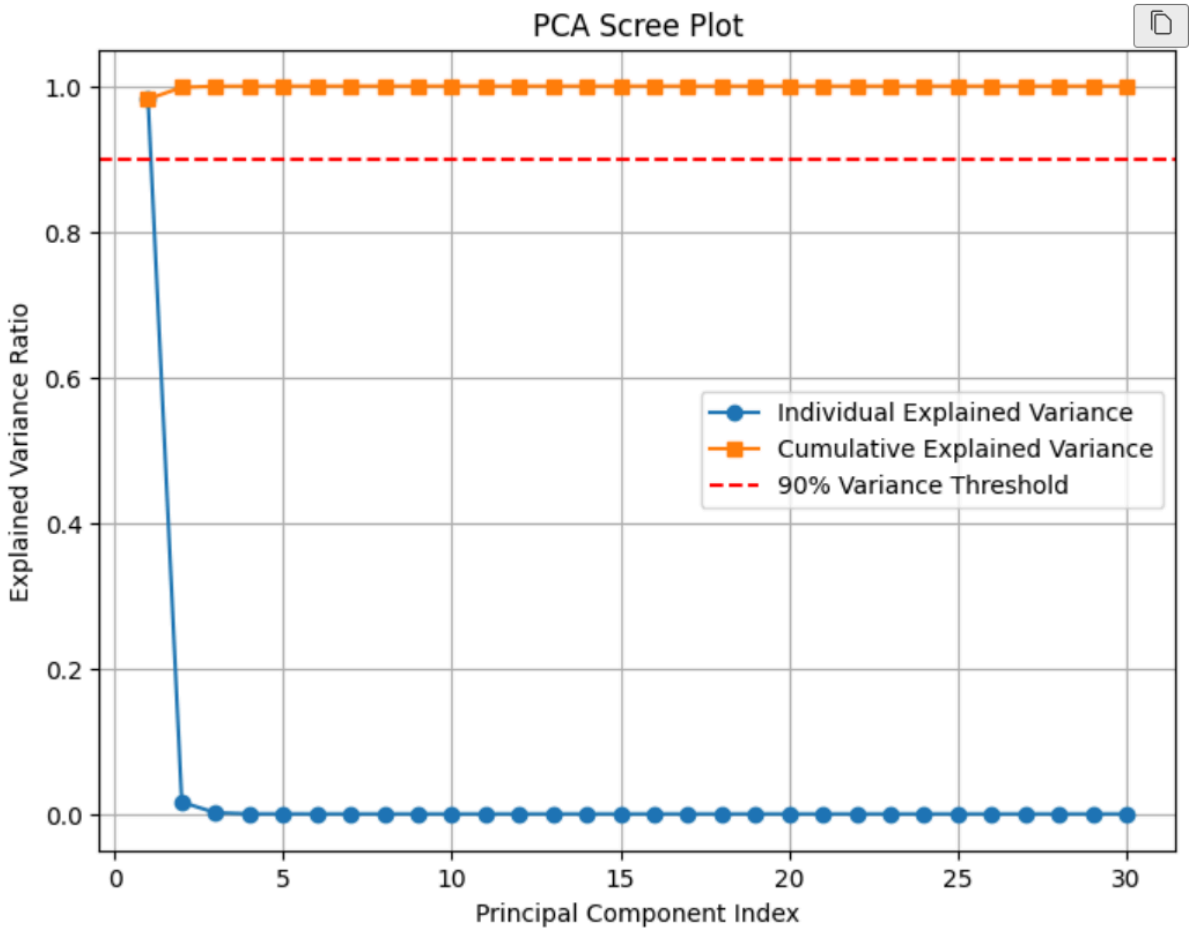
1. **初始化种群**：随机生成初始解（个体），每个解称为“染色体”，通常用二进制编码表示，每个染色体对应一个参数组合。
2. **选择操作**：根据适应度函数（目标函数的值）选择优秀的个体（父代），较优的个体更容易被选中。常用的选择方法包括轮盘赌选择、锦标赛选择等。
3. **交叉操作**：从父代中选择两个个体，进行交叉操作（即交换部分基因），生成两个子代个体。交叉操作模拟了生物的遗传交换过程。
4. **变异操作**：对子代个体的基因进行变异操作，通常是随机改变某些基因位的值。变异操作保证了遗传过程中的多样性。
5. **适应度评估**：计算每个个体的适应度，即目标函数的值。适应度高的个体会在下一代中得到更高的繁殖机会。
6. **终止条件**：算法在经过一定代数的演化后，或者当找到满意的解时，停止搜索。

GA通过模拟自然选择中的“优胜劣汰”过程，在搜索空间中不断进行局部探索和全局探索，最终找到全局最优解。

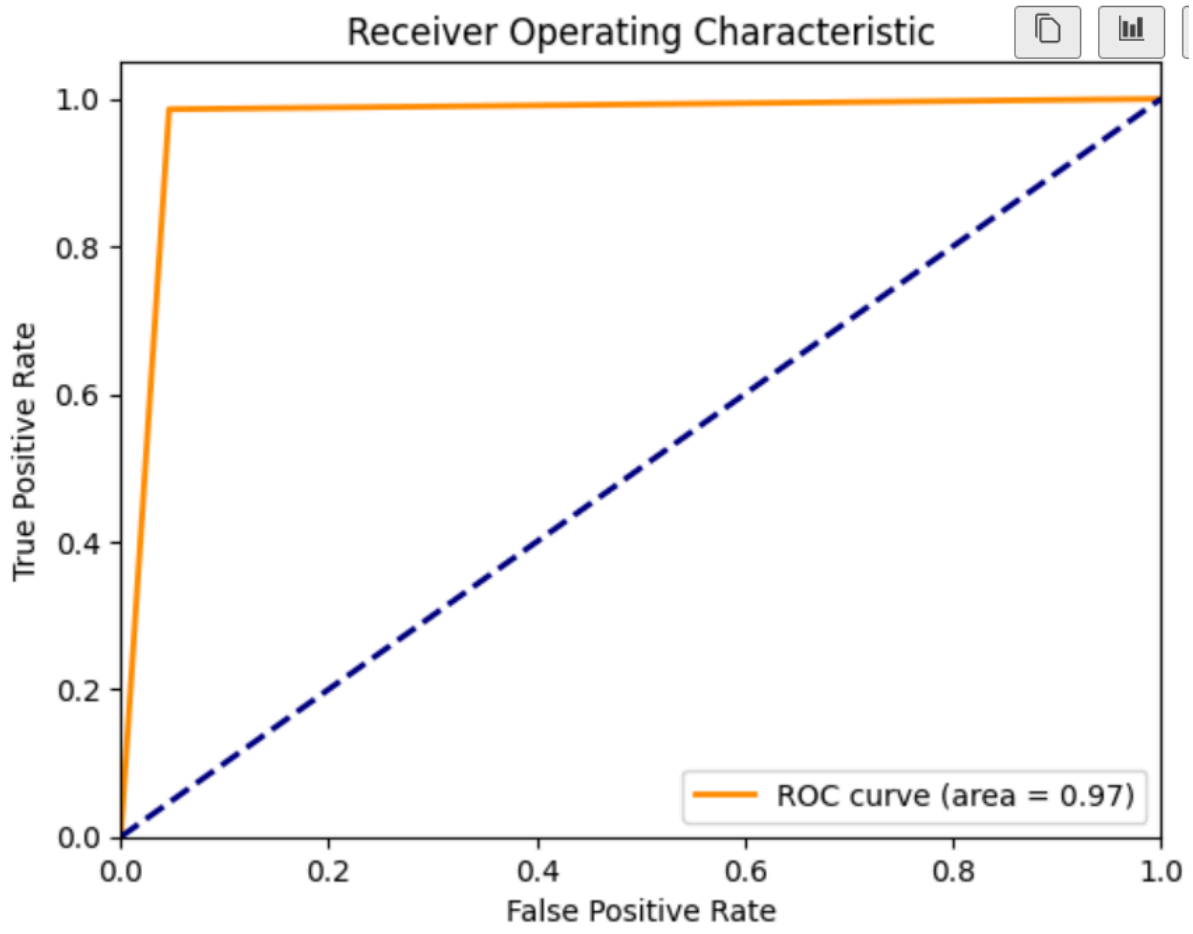
3.4 模型性能的结果分析

3.4.1 logistic回归

对数据进行降维

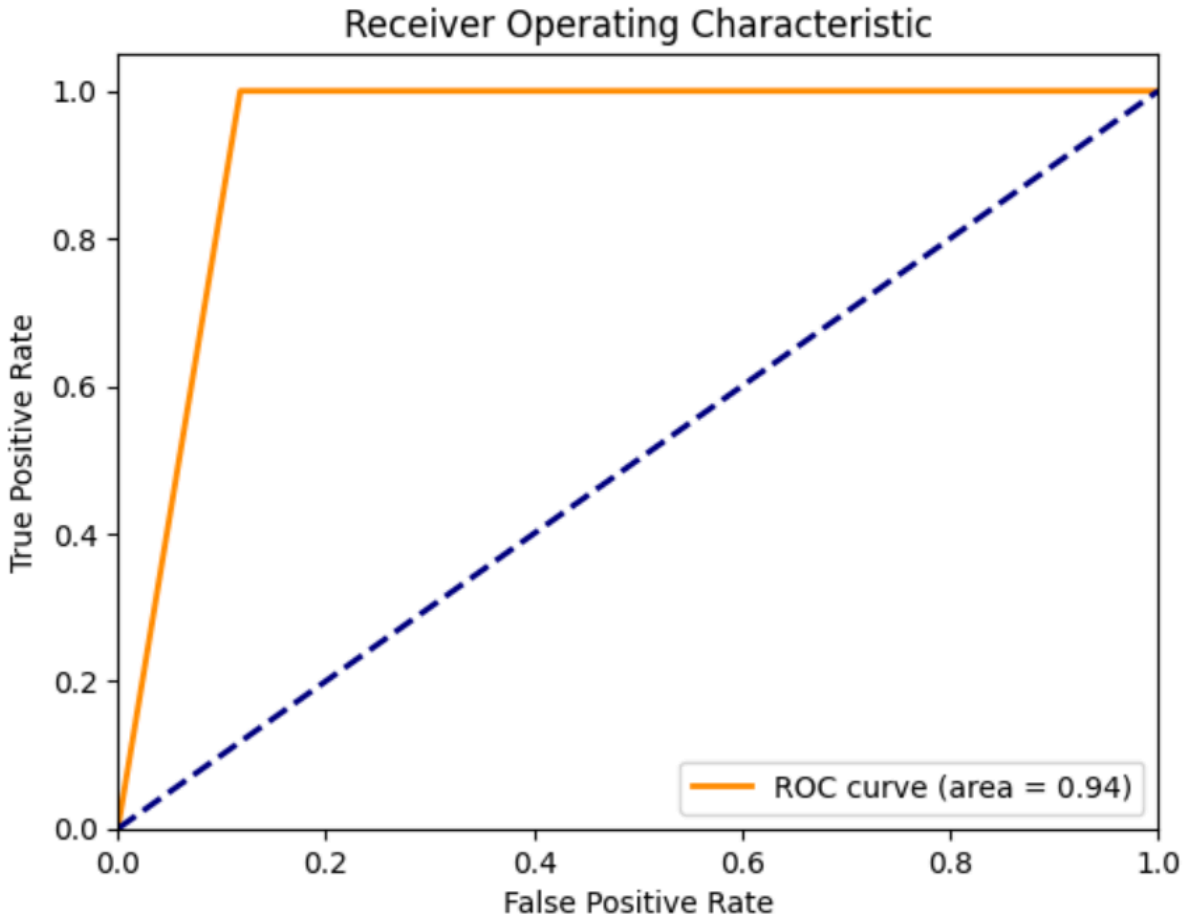


原始ROC，AUC以及精确度：




```
Accuracy: 0.9824561403508771
/nConfusion Matrix:
[[16  1]
 [ 0 40]]
/nClassification Report:
              precision    recall  f1-score   support

           0       1.00      0.94      0.97        17
           1       0.98      1.00      0.99        40

    accuracy                           0.98        57
   macro avg       0.99      0.97      0.98        57
weighted avg       0.98      0.98      0.98        57
```

降维之后的ROC与AUC：

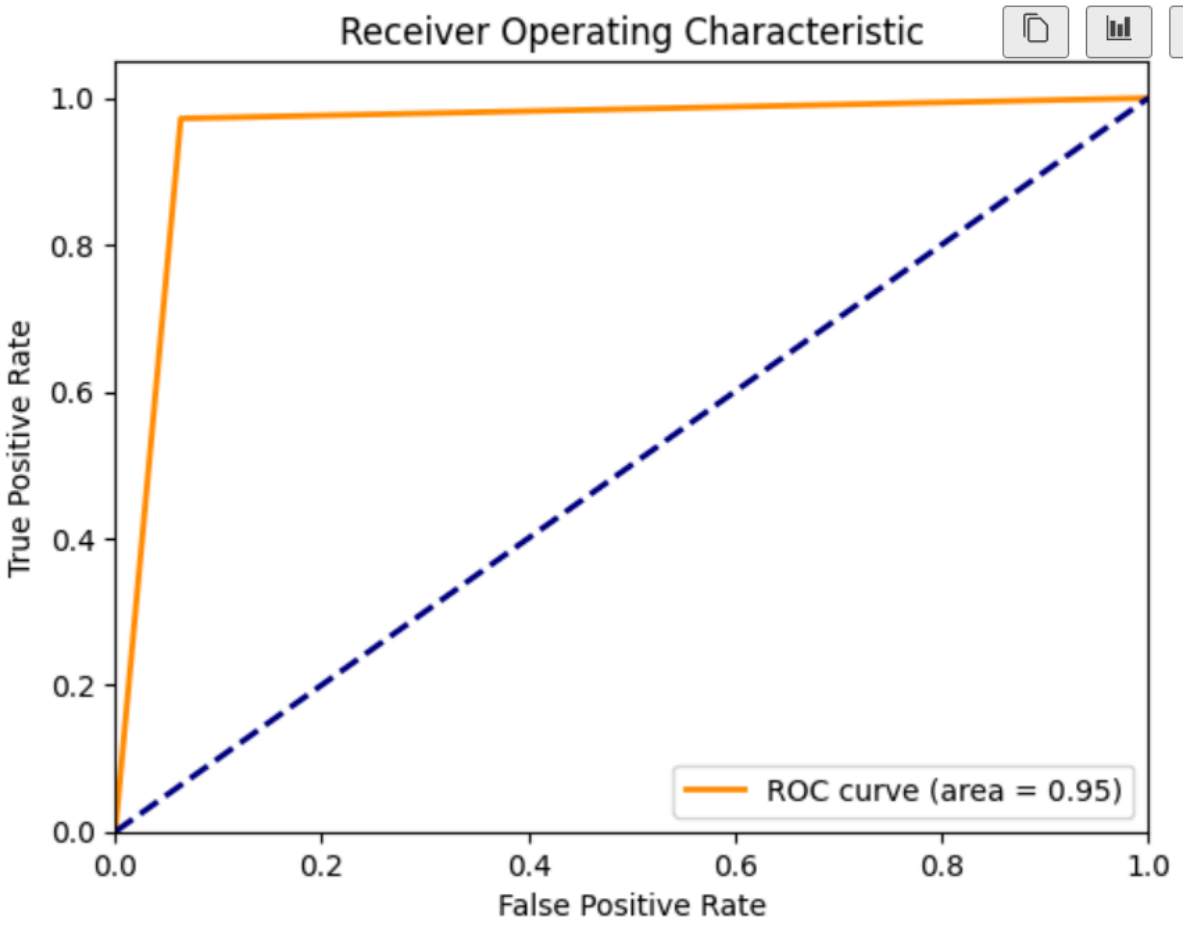


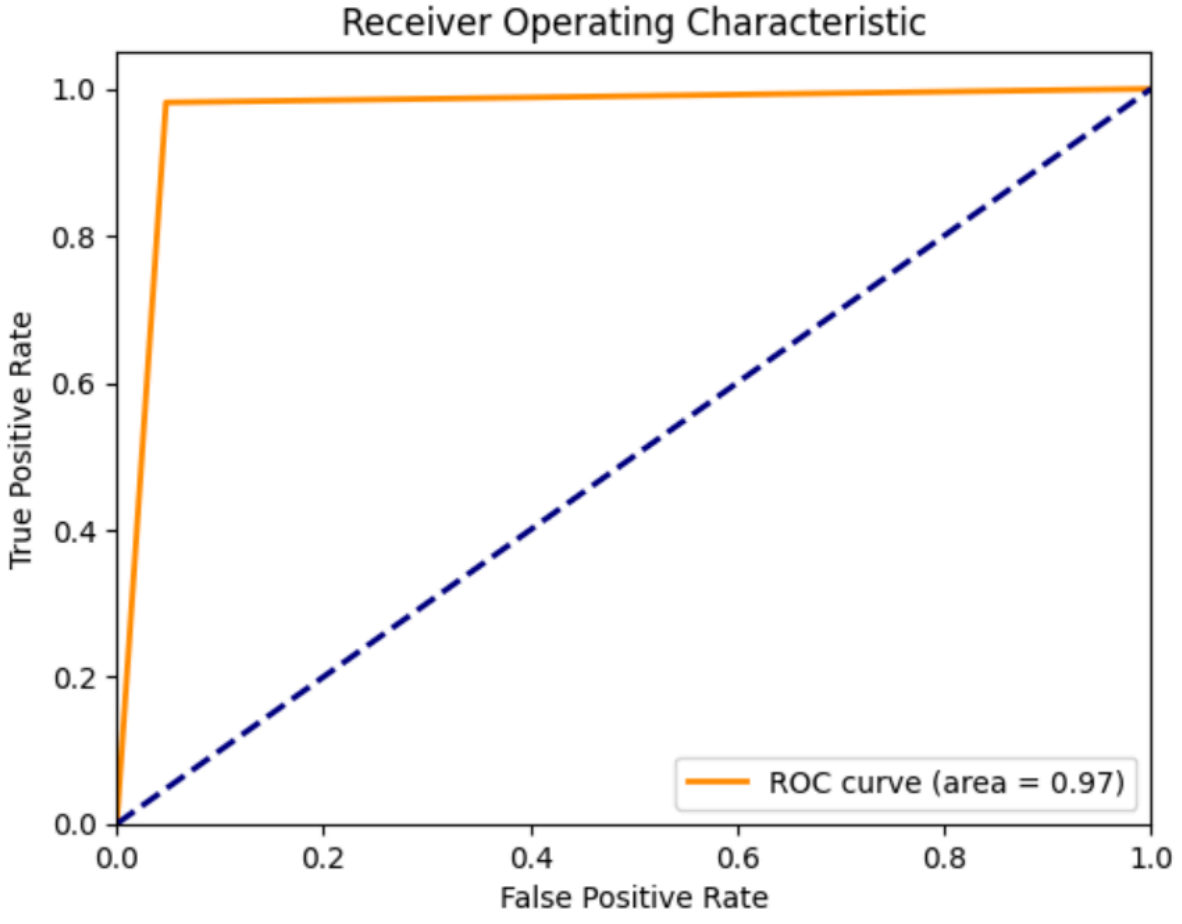

```
Accuracy: 0.96491228070175544
/nConfusion Matrix:
[[15  2]
 [ 0 40]]
/nClassification Report:
              precision    recall  f1-score   support

           0       1.00      0.88      0.94        17
           1       0.95      1.00      0.98        40

    accuracy                           0.96        57
   macro avg       0.98      0.94      0.96        57
weighted avg       0.97      0.96      0.96        57
```

可以明显得出，降维之后的预测精确度和预测能力整体上有所下降，但是计算速度有所提升。

3.4.2 K近邻算法（KNN）

调整K的值，使模型的性能达到最优

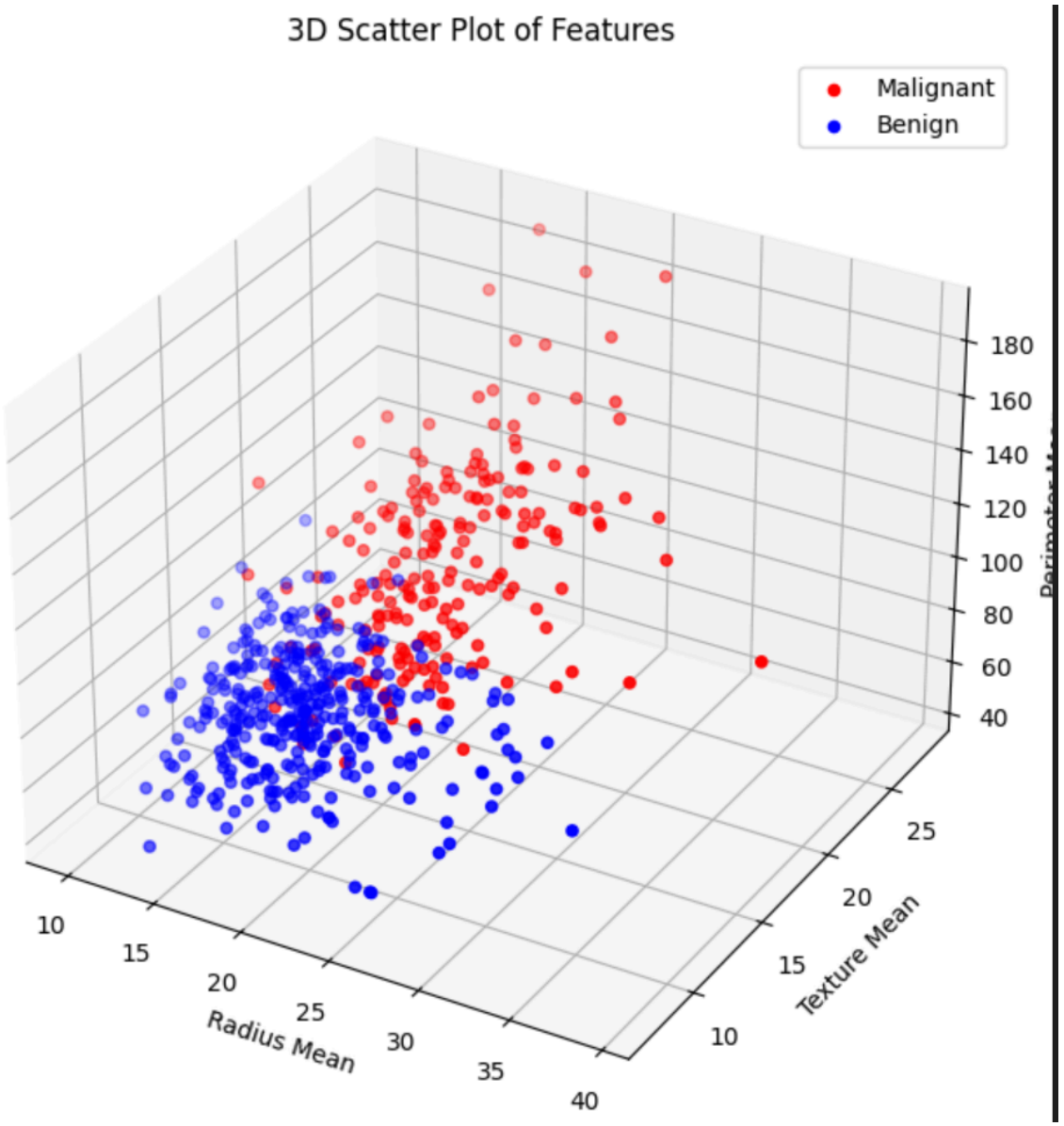




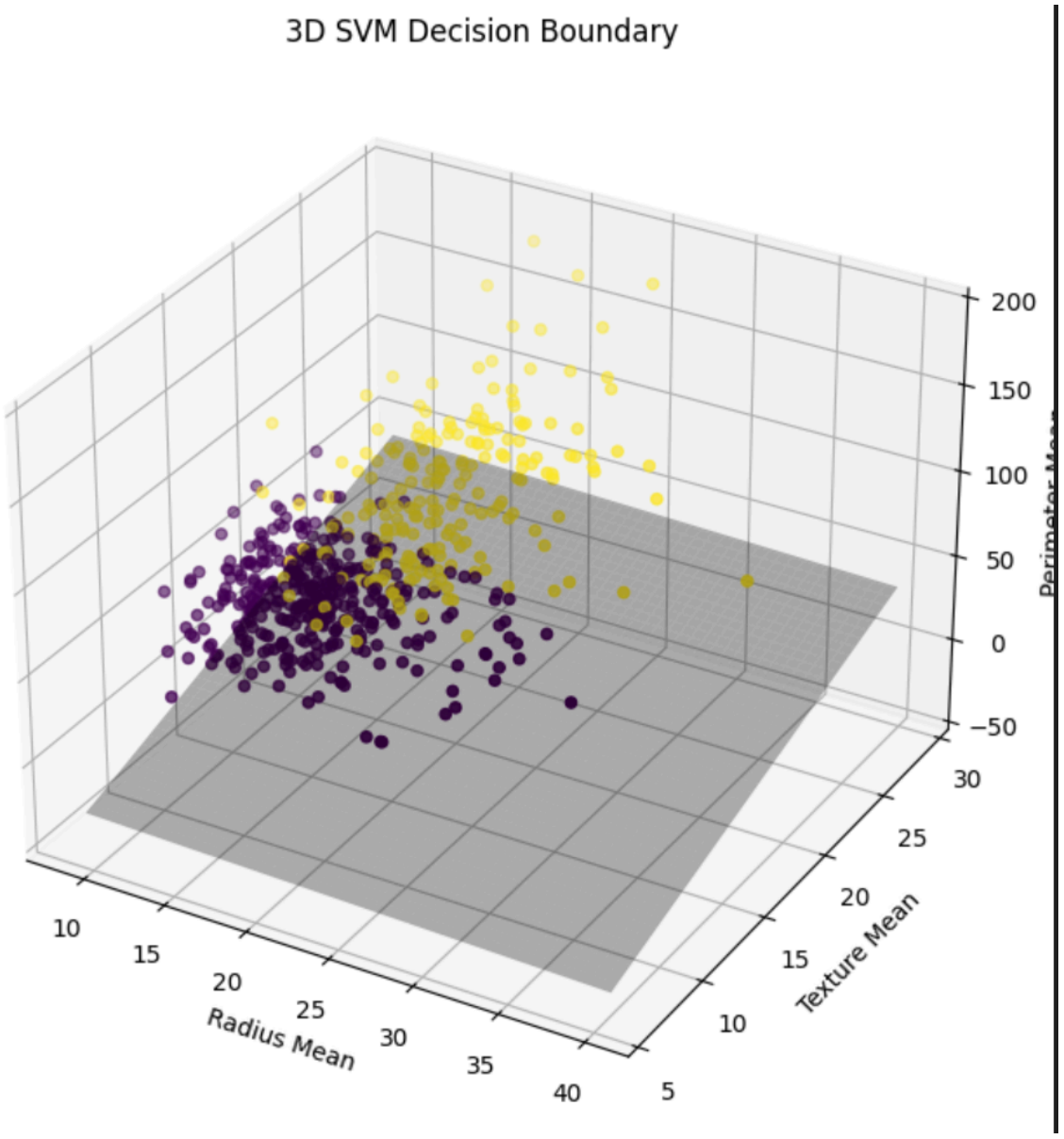
讲K值设为10时，整个算法的预测能力有所上升。

3.4.3 支持向量机（SVM）

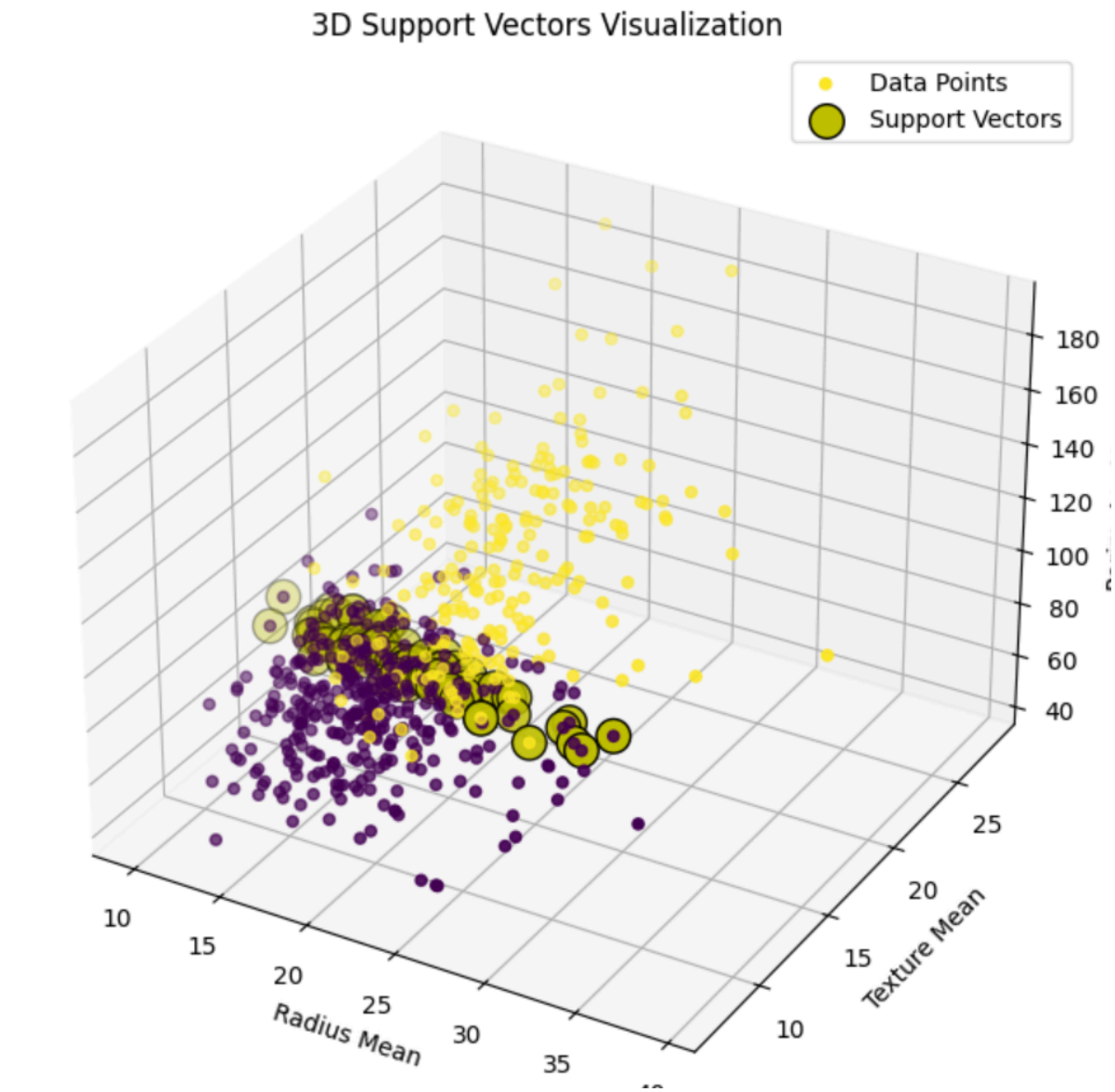
增加一维度特征，进行散点图绘制。



标出超平面的位置：


3D SVM Decision Boundary
Radius Mean
Texture Mean
Perimeter Mean
10
15
20
25
30
35
40
5
10
15
20
25
30
−50
0
50
100
150
200

标记支持向量：

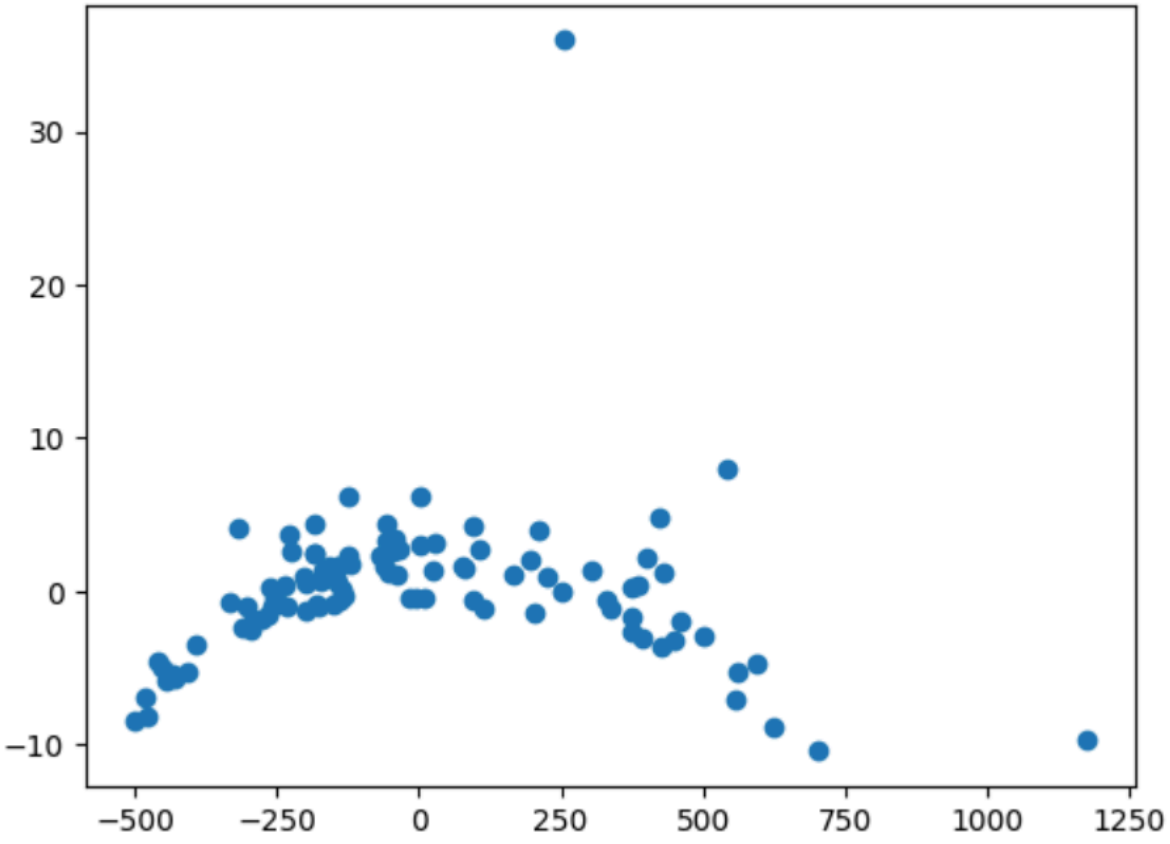


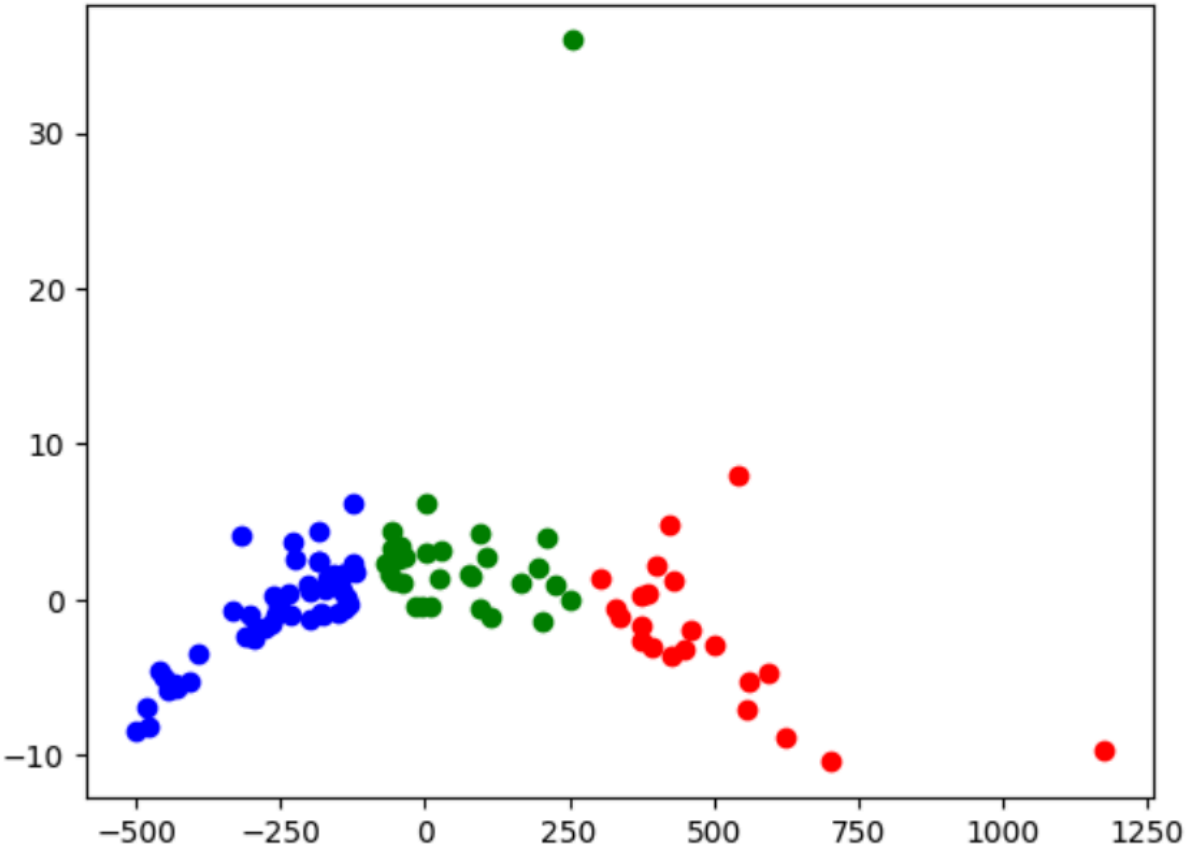
3.4.4 随机森林（Random Forest）

主函数

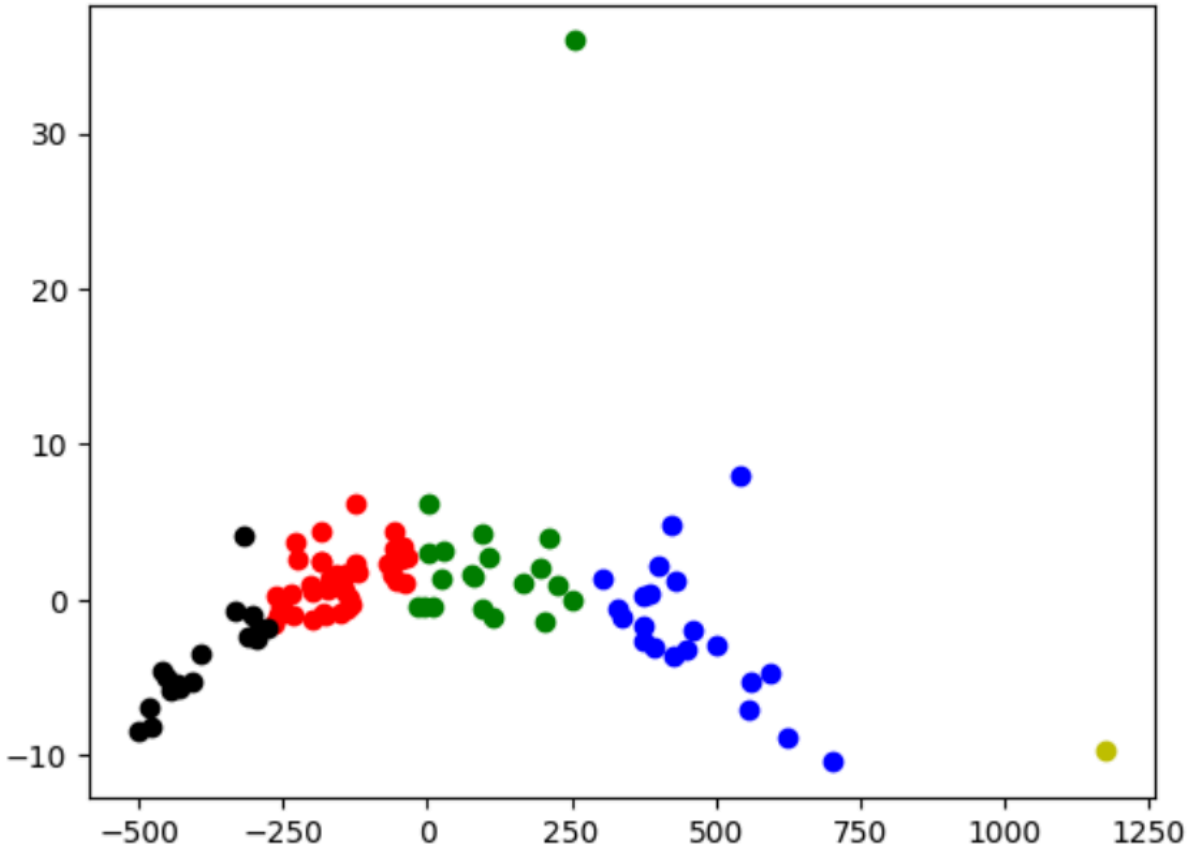
3.4.5 K平均算法（K-Means）

使用更高维的数据进行聚类划分，得到以下结果





最上面的点明显与下面的聚类有区别，应该去除该点，或者增加一类。



	id	diagnosis_result	radius	texture	perimeter	area	smoothness	compactness	symmetry	fr
0	1	M	23	12	151	954	0.143	0.278	0.242	
10	11	M	24	21	103	798	0.082	0.067	0.153	
11	12	M	17	15	104	781	0.097	0.129	0.184	
13	14	M	12	22	104	783	0.084	0.100	0.185	
16	17	M	10	16	95	685	0.099	0.072	0.159	
17	18	M	15	14	108	799	0.117	0.202	0.216	
22	23	M	20	27	103	704	0.107	0.214	0.252	
24	25	M	9	13	110	905	0.112	0.146	0.200	
25	26	M	19	27	116	913	0.119	0.228	0.304	
28	29	M	15	15	102	732	0.108	0.170	0.193	
29	30	M	11	16	115	955	0.098	0.116	0.174	
32	33	M	20	18	113	899	0.120	0.150	0.225	
34	35	M	16	23	107	807	0.104	0.156	0.200	
35	36	M	10	13	110	870	0.096	0.134	0.190	
38	39	M	11	15	96	699	0.094	0.051	0.157	
54	55	M	18	25	97	713	0.091	0.071	0.162	
72	73	M	21	12	114	929	0.107	0.183	0.193	
75	76	M	21	18	104	818	0.092	0.084	0.180	
91	92	M	10	12	100	728	0.092	0.104	0.172	
94	95	M	22	26	100	706	0.104	0.155	0.186	

	id	diagnosis_result	radius	texture	perimeter	area	smoothness	compactness	symmetry	f
1	2	B	9	13	133	1326	0.143	0.079	0.181	
2	3	M	21	27	130	1203	0.125	0.160	0.207	
4	5	M	9	19	135	1297	0.141	0.133	0.181	
6	7	M	16	26	120	1040	0.095	0.109	0.179	
12	13	B	14	15	132	1123	0.097	0.246	0.240	
18	19	M	20	14	130	1260	0.098	0.103	0.158	
23	24	M	19	12	137	1404	0.094	0.102	0.177	
27	28	M	16	24	122	1094	0.094	0.107	0.170	
30	31	M	11	22	125	1088	0.106	0.189	0.218	
33	34	M	11	21	128	1162	0.094	0.172	0.185	
42	43	M	11	11	128	1104	0.091	0.219	0.231	
45	46	M	18	11	124	1076	0.110	0.169	0.191	
53	54	M	14	26	120	1033	0.115	0.149	0.209	
56	57	M	10	19	126	1152	0.105	0.127	0.192	
70	71	M	21	18	124	1130	0.090	0.103	0.158	
77	78	M	11	21	120	1006	0.107	0.215	0.215	
78	79	M	16	18	144	1245	0.129	0.345	0.291	
83	84	M	20	14	129	1132	0.122	0.179	0.163	
85	86	M	14	13	121	1075	0.099	0.105	0.213	
87	88	M	19	11	122	1076	0.090	0.121	0.195	
95	96	M	23	16	132	1264	0.091	0.131	0.210	

第二个聚类和第三个聚类相较于之前具有更好的预测能力。

3.4.6 总结

（1）**Logistic回归**适用于特征与目标之间存在线性关系的情况。它的优点是计算速度快、易于理解，特别适用于**特征较少且线性可分**的数据。缺点则在于无法处理复杂的非线性问题，且对异常值敏感。改进方法可以通过特征选择或加入正则化（如L2正则化）来**避免过拟合。**

（2）**K近邻算法（KNN）**适用于非线性分类问题。其主要优点是**无需训练过程**，并且能有效处理复杂的决策边界。缺点是KNN算法计算量大，特别是在数据量较大时，**预测速度较慢**，同时对异常值较为敏感。它适用于小数据集的分类任务，如推荐系统、文本分类等。为了提高KNN的性能，可以通过**选择合适的K值**，并对数据进行标准化或归一化处理。

（3）**支持向量机（SVM）**适用于**高维数据**，能够通过核函数处理非线性问题。SVM的优点是能有效处理高维数据，且在数据分界明显时能提供较好的分类性能。然而，SVM对超参数的选择较为敏感，**训练时间较长，**且在处理大规模数据时会遇到计算瓶颈。适用于高维数据集或特征间有明确分类边界的场景，如文本分类和图像识别等。SVM的性能可以通过**选择合适的核函数和调节超参数**来优化。

（4）**随机森林（Random Forest）**是一种**集成学习**方法，适用于大规模数据集。它通过构建多个决策树并投票决定分类结果，具有很强的**抗过拟合能力**，并且能够自动计算特征的重要性。然而，随机森林的模型较为复杂，训练和预测的速度较慢，而且较难解释。它广泛应用于分类和回归任务，特别是当数据集较大且特征间复杂时，效果较好。可以通过**增加树的数量和进行特征选择**来提升模型的性能。

（5）**K均值算法（K-Means）**优点在于简单高效，适用于大规模数据集，但需要预先指定K值，并且对异常值较为敏感。K均值不适用于分类任务，而是常用于**聚类问题**，如客户分群、市场细分等。通过**选择合适的K值和标准化数据**，可以改善K均值算法的性能。

四、 实验心得与体会

4.1 算法总结

通过实验总结了有监督学习中

（1）logistic回归

（2）K近邻算法（KNN）

（3）支持向量机（SVM）

（4）随机森林（Random Forest）

以及无监督学习中

（1）K近邻算法（K-Means）

五个算法的算法思想，并且针对算法的特性进行了使用场景和模型优缺点的总结。

在此之外，总结了常用的数据预处理的方式方法和超参数调优的常用调优方法。特别是10折交叉验证在对于超参数选取中优化的普遍性的认识。总结各个模型的内容如下：

4.2 超参数调优总结

（1）**网格搜索（Grid Search）**

网格搜索通过遍历设定的超参数空间，进行穷举搜索，找到最佳的超参数组合。但是计算开销大，不利于大规模的超参数优化。

（2）**随机搜索（Random Search）**

随机搜索是从指定的超参数空间中随机选取超参数组合进行训练进行超参数调优的。相对于网格搜索，计算开销较小。但是随机性比较强，且效果不一定有网格搜索好。

（3）**贝叶斯优化（Bayesian Optimization）**

贝叶斯优化是一种基于概率模型（如高斯过程）来选择下一个评估点的优化方法，它通过对历史评估结果进行建模，来指导下一次的搜索方向，能够比网格搜索和随机搜索更高效地寻找最优解。

（4）**其他算法调优**

通过加入粒子群算法和遗传算法等也可实现超参数的调优。在某些多维超参数的调优上也具有很大的优势。

4.3 模型评估总结

这里主要总结了分类问题的模型评估问题。对于分类模型来说，主要有三种评估方法：

（1）绘制ROC曲线

（2）计算AUC面积

（3）打印混淆矩阵分析准确率、精确率、召回率和F1分数

这里只考虑简单的二分类问题的混淆矩阵，多维的混淆矩阵这里暂不做总结。

4.3.1 混淆矩阵（Confusion Matrix）

混淆矩阵 是一种用于评估分类模型性能的工具，尤其适用于 **有监督学习** 中的分类问题。它是一个 **表格**，展示了模型预测结果与实际标签之间的关系，从而帮助我们分析分类模型的表现。

在二分类问题中，混淆矩阵通常是一个 **2x2** 的矩阵，包含四个主要元素：

	预测为正类	**预测为负类**
实际为正类	**真阳性 (TP)**	**假阴性 (FN)**
实际为负类	**假阳性 (FP)**	**真阴性 (TN)**

真阳性 (TP)：预测为正类，且实际为正类的样本数。

假阳性 (FP)：预测为正类，但实际为负类的样本数（模型错误地预测为正类）。

真阴性 (TN)：预测为负类，且实际为负类的样本数。

假阴性 (FN)：预测为负类，但实际为正类的样本数（模型错误地预测为负类）。

通过混淆矩阵中的四个元素，我们可以计算出一些常用的性能指标，帮助更好地评估分类模型。

（1）**准确率 (Accuracy)**：

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

表示分类器正确预测的比例。

（2）**精确率 (Precision)**：

$$\text{Precision} = \frac{TP}{TP + FP}$$

表示所有被预测为正类的样本中，实际为正类的比例。也叫做 **查准率**。

（3）**召回率 (Recall)**：

$$\text{Recall} = \frac{TP}{TP + FN}$$

表示所有实际为正类的样本中，被正确预测为正类的比例。也叫做 **灵敏度** 或 **真阳性率**。

（4）**F1 分数 (F1 Score)**：

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

是精确率和召回率的调和平均数，综合考虑了模型的精确度和召回能力。

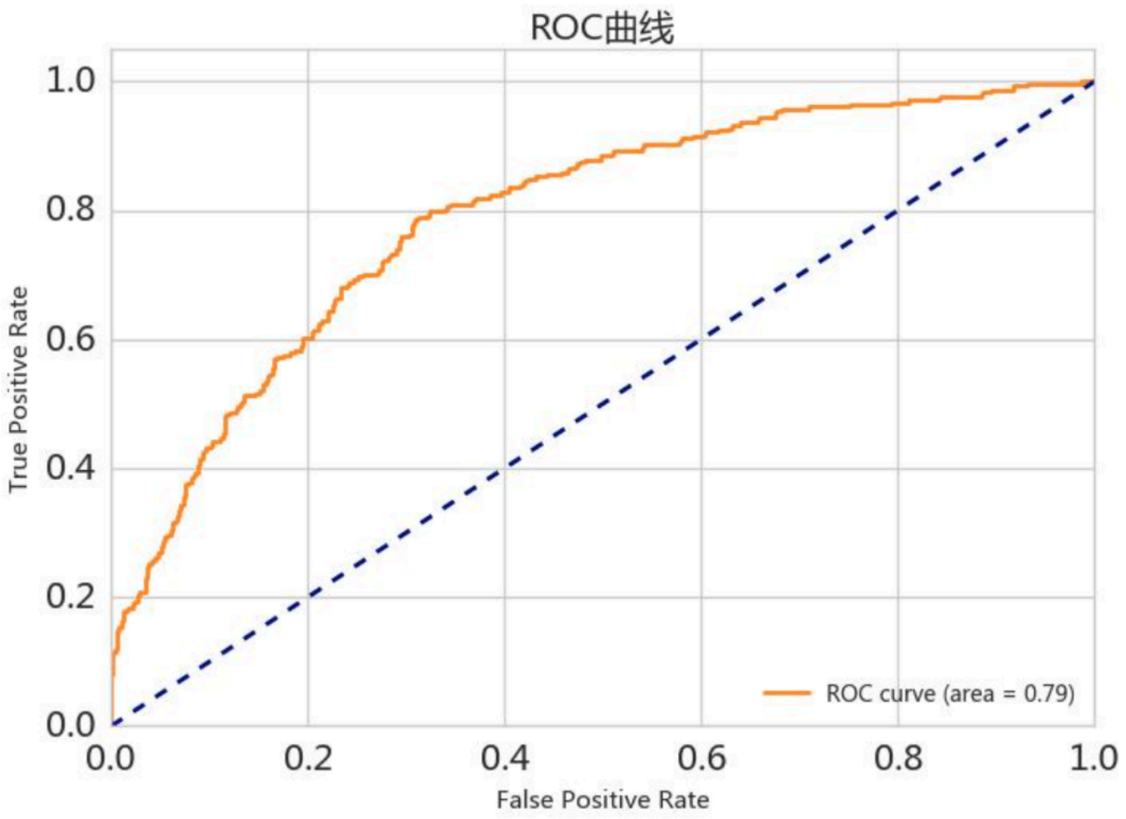
（5）**特异性 (Specificity)**：

$$\text{Specificity} = \frac{TN}{TN + FP}$$

表示所有实际为负类的样本中，正确预测为负类的比例。

4.3.2 绘制ROC曲线

ROC 曲线是 **真阳性率（TPR, True Positive Rate）** 和 **假阳性率（FPR, False Positive Rate）** 的关系图。它显示了分类模型在不同阈值下的表现。



（1）**真阳性率（TPR）**：也叫 **灵敏度** 或 **召回率**，表示实际为正类的样本中，被正确分类为正类的比例：

$$TPR = \frac{TP}{TP + FN}$$

其中，**TP** 是真阳性（被正确预测为正类的样本数），**FN** 是假阴性（实际为正类但被预测为负类的样本数）。

（2）**假阳性率（FPR）**：表示实际为负类的样本中，被错误地预测为正类的比例：

$$\frac{FP}{FP + TN}$$

其中，**FP** 是假阳性（实际为负类但被预测为正类的样本数），**TN** 是真阴性（被正确预测为负类的样本数）。

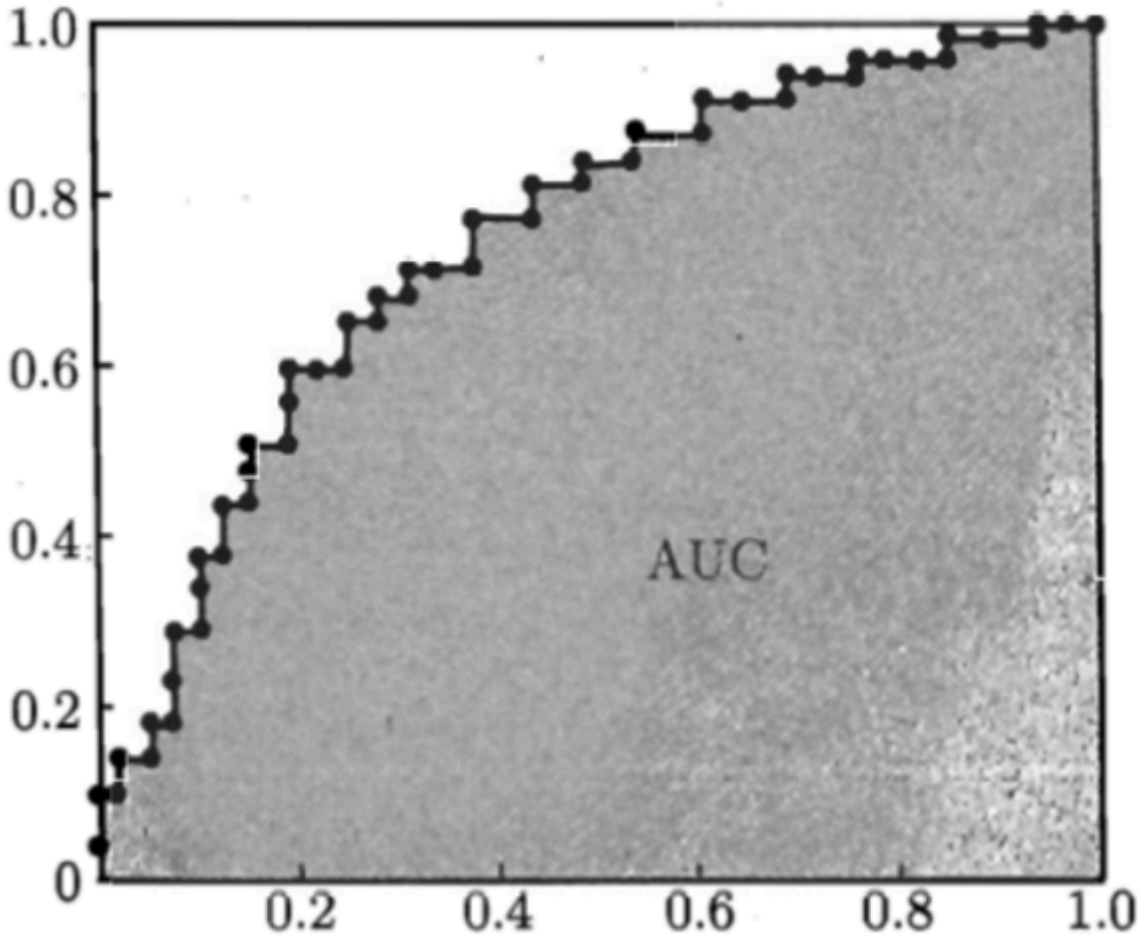
ROC 曲线是以 **FPR** 为横轴，**TPR** 为纵轴，绘制不同分类阈值下模型的表现。

ROC 曲线的理想情况是尽可能接近坐标轴的左上角，意味着高 **TPR** 和低 **FPR**。

曲线越靠近左上角，说明模型的性能越好。

4.3.3 计算ROC曲线下所围面积AUC

AUC 是 **ROC 曲线下面积（Area Under the Curve）** 的简称，衡量的是模型对正类和负类样本的区分能力。AUC 的值介于 0 和 1 之间。



AUC = 1：表示模型具有完美的分类能力，能够完美区分正类和负类。

AUC = 0.5：表示模型的性能与随机猜测相当，没有区分能力。

AUC < 0.5：表示模型表现差，甚至比随机猜测还差，通常意味着模型需要重新训练或调整。