郑 重 声 明

本人呈交的课程设计实验报告，是在导师的指导下，独立进行课程设计工作所取得的成果，所有数据、图片资料真实可靠. 尽我所知，除文中已经注明引用的内容外，本课程设计实验报告不包含他人享有著作权的内容. 对本课程设计实验报告所涉及的研究工作做出贡献的其他个人和集体，均已在本文中以明确的方式标明. 本课程设计实验报告的知识产权属于培养单位.

本人签名：_____　　日期：_____

摘要

本文讨论了基于机器学习算法的心脏病预测模型.本文简要介绍了应用 K 最邻近(KNN,K-Nearest Neighbor)，决策树，随机森林，支持向量机(SVM, Support Vector Machines)实现的心脏病数据处理与预测，探讨了不同参数的选取对于算法的影响，并用图像的形式直观呈现.最后比较了几种算法的预测结果，探讨和比较不同算法适合的应用场景，并就不足之处讨论可采取的改进措施.

关键词：KNN;决策树;随机森林;SVM;机器学习;心脏病预测

ABSTRACT

This paper discusses the prediction model of heart disease based on machine learning algorithm. This paper briefly introduces the application of K-Nearest Neighbor, Decision Tree, Random Forest, Support Vector Machine to realize the heart disease data processing and prediction, discusses the influence of different parameter selection on the algorithm, and visually presents it in the form of images. Finally, the prediction results of several algorithms are compared, and the suitable application scenarios of different algorithms are discussed and the improvement measures are discussed.

Key words: KNN; Decision Tree; Random Forest; SVM; machine learing; prediction of heart disease

目录

第 1 章 引言

1.1 项目简介

心脏病是一类比较常见的循环系统疾病，多发于中老年人群体。但随着现代生活节奏加快，工作学习的压力过大，生活不规律，饮食不科学等因素致使心脏病年轻化的出现。这一现象警示我们不可趁年轻肆意妄为，而要时刻注意自己的身体状况。

因此本项目旨在建立一个模型，该模型可以根据描述疾病的特征组合预测心脏病发生的概率。为了实现这一目标，我从 Kaggle 上下载并使用了瑞士 Cleveland Clinic Foundation 收集的数据集。

该项目中使用的数据集包含针对心脏病的 13 个特征。数据集显示不同水平的心脏病，用数字 1-4 表示其心脏病的严重程度，另外用 0 表示没有心脏病。我们近 300 行数据，13 个连续观察不同的症状。此项目研究了不同的经典机器学习模型，以及它们在疾病风险中的发现。

1.2 实验流程

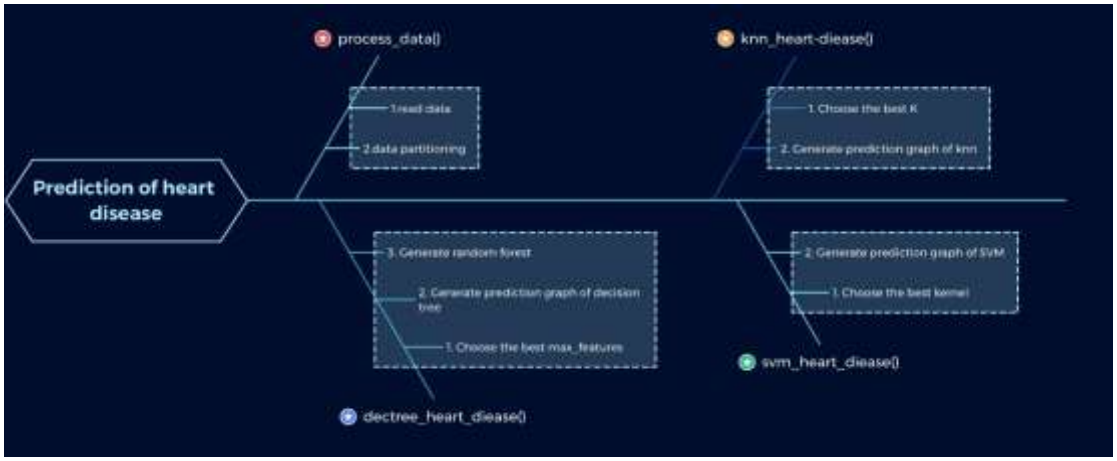

图 1.1 实验流程图

实验主要分为四个模块，分别是数据加工模块，以及三个算法实现类模块，包括 KNN 实现，决策树与随机森林实现，SVM 实现。

1.3 实验环境与配置

Operation System	Windows11
Python Version	Python 3.10.8 64-bit
Dependency Packages	pandas, numpy, matplotlib, seaborn, sklearn

第 2 章 数据加工

2.1 数据预处理

```
    # 添加列名
      header_row = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope',
                    'ca','thal', 'target']
      # 载入数据
      heart = pd.read_csv('heart.csv', names = header_row)
      # 心脏病（0=否，1=是）
      heart["target"] = np.where(heart["target"] != 0, 1, 0)
```

从下载好的表格中读取数据，为简化实验数据集，我们对 target 做处理，不区分心脏病的程度，只标记是否患病，0 代表无心脏病，1 代表有心脏病

```
    #划分特征值和目标值
      x = heart[
          ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca',
           'thal']]
      label = heart.target
```

对数据集进行划分，除 target 一列外均为特征值，target 一列为目标值

2.2 查看数据集

利用 Pandas 中的 info 和 describe 函数对数据集进行查看

```
dataset.describe() # 描述统计相关信息
```

结果如下：

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 297 entries, 0 to 296
Data columns (total 14 columns):
age         297 non-null int64
sex         297 non-null int64
cp          297 non-null int64
trestbps    297 non-null int64
chol        297 non-null int64
fbs         297 non-null int64
restecg     297 non-null int64
```

```
thalach     297 non-null int64
exang       297 non-null int64
oldpeak     297 non-null float64
slope       297 non-null int64
ca          297 non-null int64
thal        297 non-null int64
target      297 non-null int32
dtypes: float64(1), int32(1), int64(12)
memory usage: 31.4 KB
```

简单介绍一下这些特征的意义：

```
age：年龄
sex：性别（1 代表男性，0 代表女性）
cp：胸部疼痛情况（1：典型心绞痛；2：非典型心绞痛；3：没有心绞痛；4：无症状）
trestbps：静息血压
chol：胆固醇
fbs：空腹血糖>120mg/dl （1：true；2：false）
restecg：静息心电图测量（0：普通；1：ST-T 波异常；2：左心室肥大）
thalach：最高心跳率
exang：运动诱发心绞痛（1：yes；2：no）
oldpeak：运动相对于休息引起的 ST 抑制
slope：运动 ST 段的峰值斜率（1：上坡；2：平的；3：下坡）
ca：主要血管数目（0、1、2、3、4）
thal：一种血液疾病（1：正常；2：固定缺陷；3：可逆的缺陷）
target：是否患病（1：yes；2：no）
```

describe 的特征结果如下：

	age	sex	cp	trestbps	chol	fbs	\
count	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	
mean	54.542088	0.676768	3.158249	131.693603	247.350168	0.144781	
std	9.049736	0.468500	0.964859	17.762806	51.997583	0.352474	
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000	
50%	56.000000	1.000000	3.000000	130.000000	243.000000	0.000000	
75%	61.000000	1.000000	4.000000	140.000000	276.000000	0.000000	
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	

	restecg	thalach	exang	oldpeak	slope	ca	\
count	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	
mean	0.996633	149.599327	0.326599	1.055556	1.602694	0.676768	
std	0.994914	22.941562	0.469761	1.166123	0.618187	0.938965	
min	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000	
25%	0.000000	133.000000	0.000000	0.000000	1.000000	0.000000	
50%	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000	
75%	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000	
max	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000	

	thal	target
count	297.000000	297.000000
mean	4.730640	0.461279
std	1.938629	0.499340
min	3.000000	0.000000
25%	3.000000	0.000000
50%	3.000000	0.000000
75%	7.000000	1.000000
max	7.000000	1.000000

图 2.1 describe 函数结果

2.3 数据特征图

利用 matplotlib 和 seaborn 提供的绘图函数进行各特征与目标间的特征图绘制

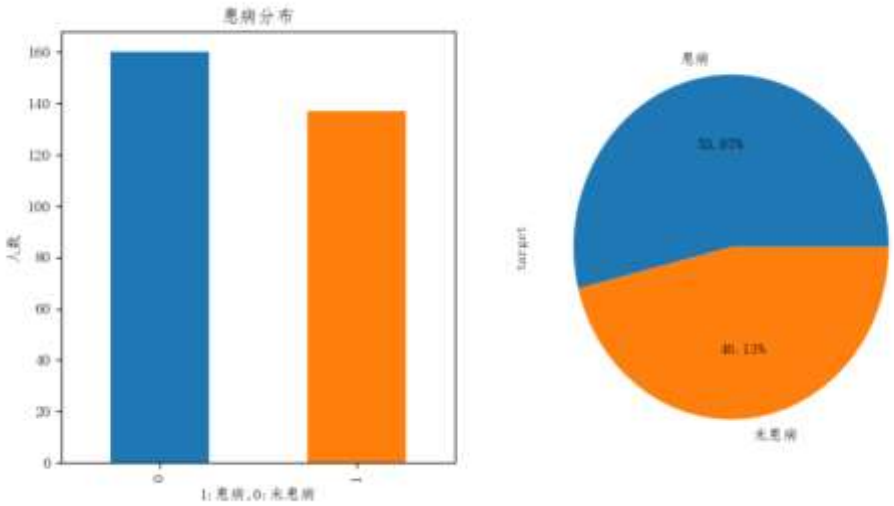

图 2.2 是否患病分布

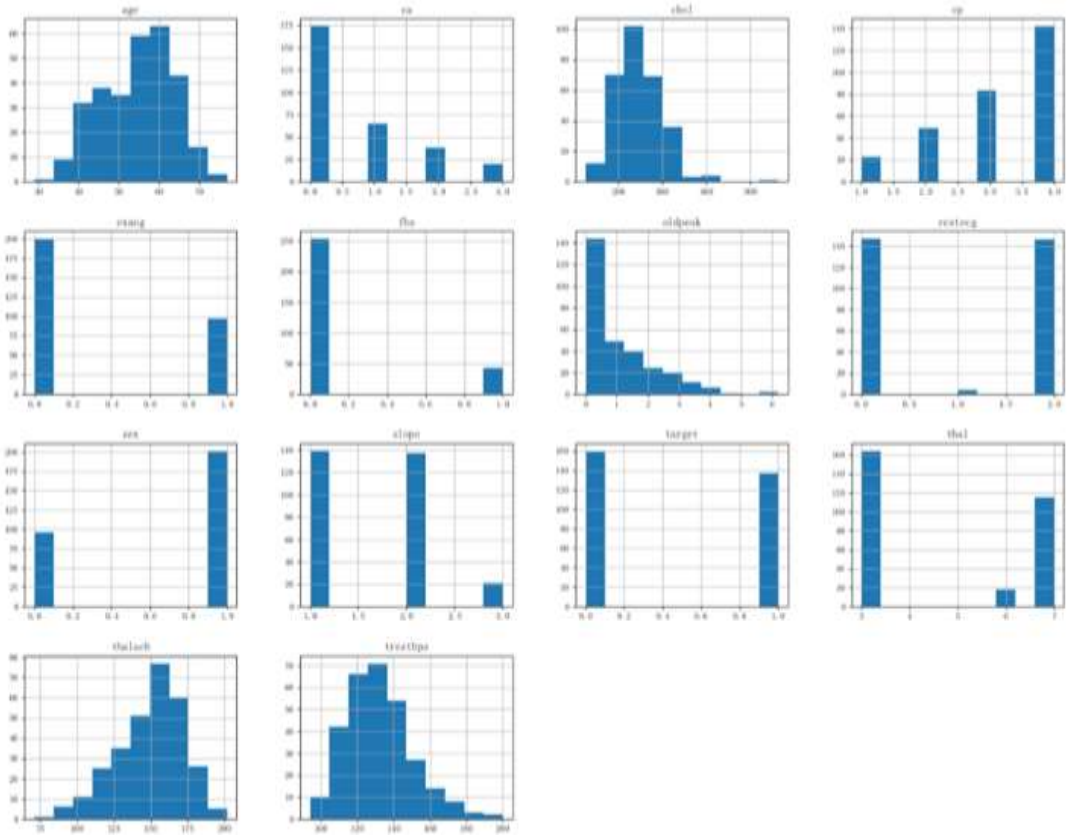

图 2.3 各特征的数据分布柱形图

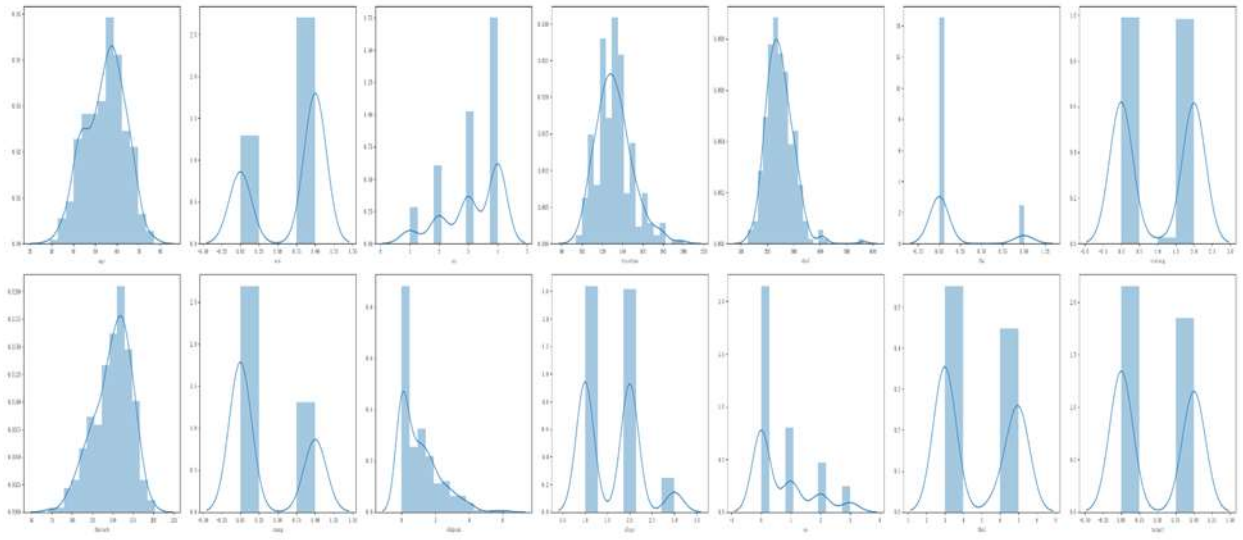

图 2.4 各特征的数据分布折线图

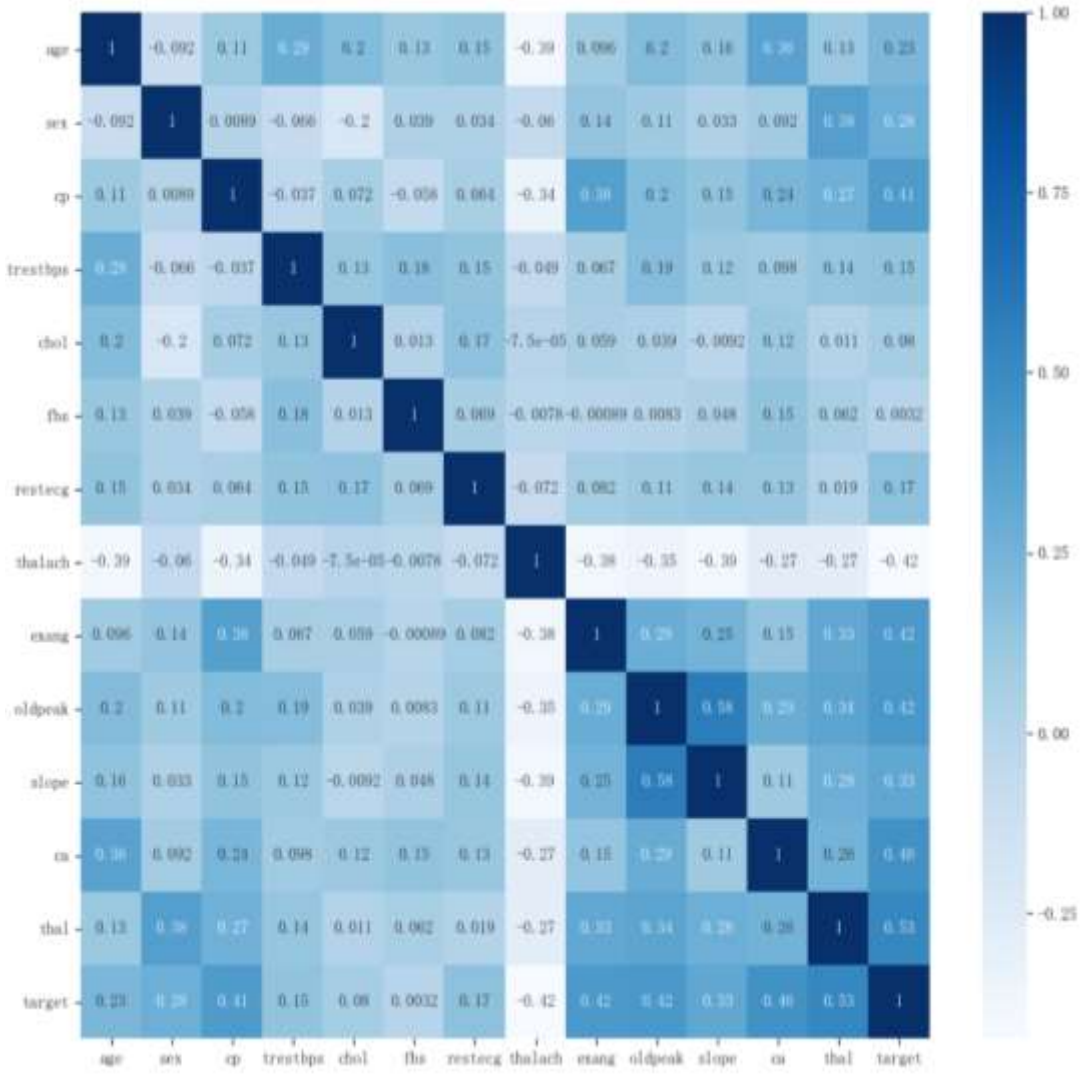

图 2.5 各特征相关性热力图

从上面的图形中我们可以发现，thalach 与 age 呈高度负相关(-0.39)。这意味着如果年纪增大，心率将降低，反之亦然。Target 与 thal(一种血液疾病)呈正相关最高，为 0.53，其次是 ca（主要血管数目），为 0.46

2.4 划分数据集

将已经处理好的特征值 x，和目标值 label 传入划分函数，以 30% 测试数据，70% 训练数据划分训练集和测试集。由于要比较几种算法，因此我们对数据集进行统一划分

```
#数据集的划分
    x_train, x_test, y_train, y_test = train_test_split(x, label, test_size=0.3)
```

由于划分函数中数据的选取是随机的 3:7,因此每次实验运行结果并不一定相同，本文以一次较好的实验数据为基础撰写。

第 3 章 不同算法对心脏病的预测

3.1 KNN

KNN（K-Nearest Neighbor)是数据挖掘分类技术中最简单的方法之一。所谓 K 最近邻，就是 k 个最近的邻居的意思，说的是每个样本都可以用它最接近的 k 个邻近值来代表。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。

3.1.1 数据预处理

标准化：在多指标评价体系中，由于各评价指标的性质不同，通常具有不同的量纲和数量级。当各指标间的水平相差很大时，如果直接用原始指标值进行分析，就会突出数值较高的指标在综合分析中的作用，相对削弱数值水平较低指标的作用。因此，为了保证结果的可靠性，需要对原始指标数据进行标准化处理。

```
01      transfer = StandardScaler()
02       # 标准化实验集
03       x_train = transfer.fit_transform(x_train)
04       # 标准化测试集
05       x_test = transfer.transform(x_test)
```

3.1.2 KNN 算法实现

K 值是 KNN 算法的一个超参数，K 的含义即参考“邻居”标签值的个数。 有个反直觉的现象，K 取值较小时，模型复杂度（容量）高，训练误差会减小，泛化能力减弱；K 取值较大时，模型复杂度低，训练误差会增大，泛化能力有一定的提高。

因此在实验中，我们选择通过交叉验证不断尝试最优的 K 值，从选取一个较小的 K 值开始，不断增加 K 的值，然后计算验证集合的方差，最终找到一个比较合适的 K 值。

```
01       krange = range(1, 10)
02       for i in krange:
03           #KNN 算法预估器
04           clf = neighbors.KNeighborsClassifier(n_neighbors=i, metric=
   "euclidean")
```

```
            clf = clf.fit(x_train,y_train)
            score.append(clf.score(x_test, y_test))
        bestK = krange[score.index(max(score))]
        print(bestK)
        print(score)
        print(max(score))
```

实验 K 值范围选取 1-10，经计算取其中最优值作为 K 值

3.1.3 预测结果可视化

K 值选取：

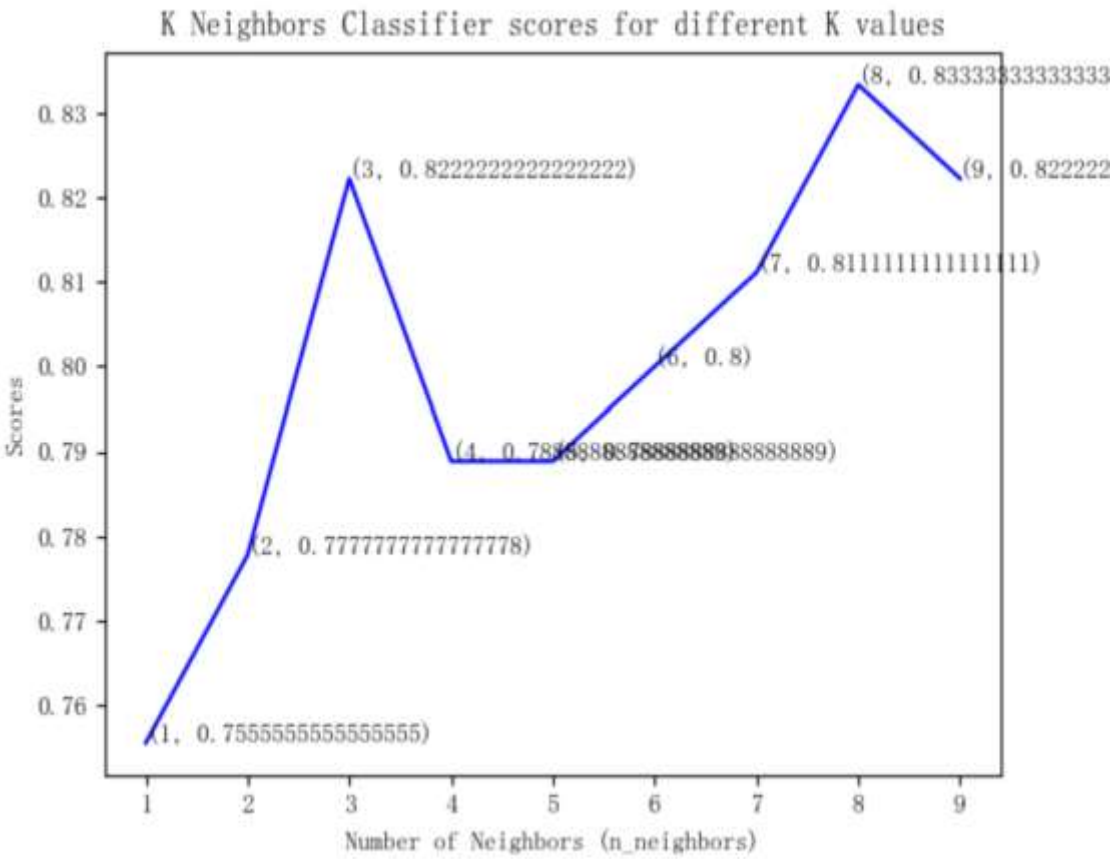

图 3.1 K 值分布图

由图中数据得，最好的 K 值选为 8，最好的正确预测率可达 83.3%

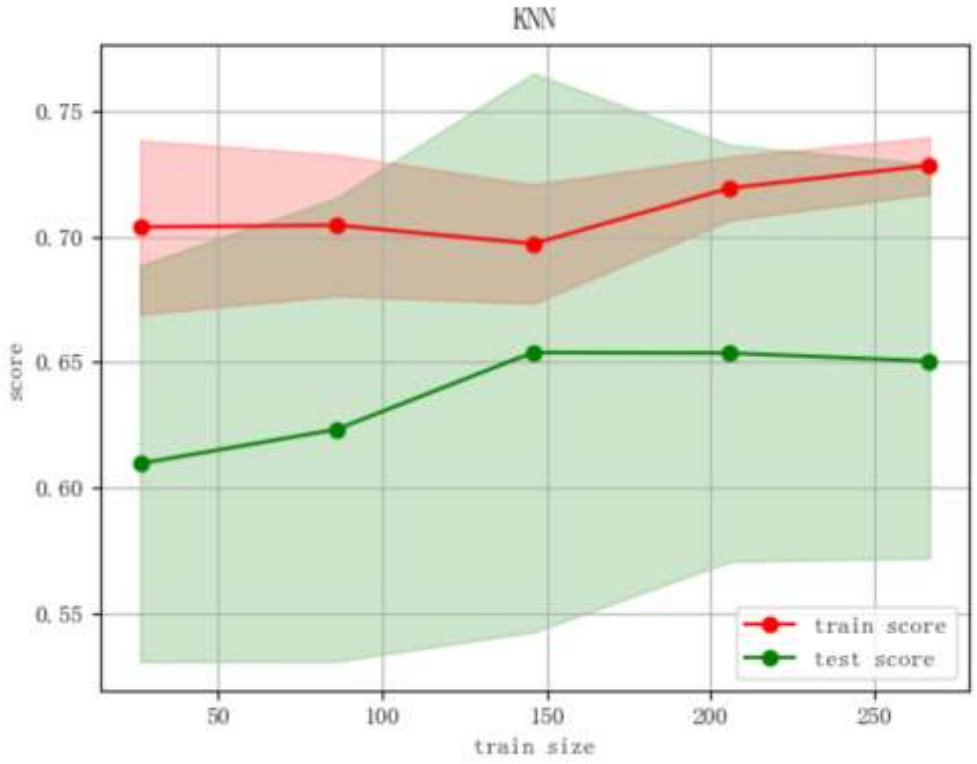

图 3.2 KNN 训练集与测试集拟合效果图

3.2 决策树与随机森林

决策树(Decision Tree) 又称为判定树，是数据挖掘技术中的一-种重要的分类与回归方法。决策树学习的算法通常是一个递归地选择最优特征，并根据该特征对训练数据进行分割，使得对各个子数据集有一个最好的分类的过程。这一过程对应着特征空间的划分，也对应着决策树的构建。

3.2.1 数据预处理

独热编码(One-Hot Encoding): 独热编码又称一位有效编码，其方法是使用 N 位状态寄存器来对 N 个状态进行编码，每个状态都有它独立的寄存器位，并且在任意时候，其中只有一位有效。即，只有一位是 1，其余都是零值。独热编码 是利用 0 和 1 表示一些参数，使用 N 位状态寄存器来对 N 个状态进行编码。

使用独热编码，将离散特征的取值扩展到了欧式空间，离散特征的某个取值就对应欧式空间的某个点。将离散型特征使用独热编码，会让特征之间的距离计算更加合理。

```
01        # 进行处理（特征工程）one hot 编码
02        dict = DictVectorizer(sparse=False)
```

```
        x_train = dict.fit_transform(x_train.to_dict(orient="records"))
        print(dict.get_feature_names())
        x_test = dict.transform(x_test.to_dict(orient="records"))
        print(x_train)
```

3.2.2 决策树算法实现

选择最优考虑最大特征数：不输入则默认全部特征，本算法检查了 1 到全部特征数，找寻其中最优的去构造决策树

```
        dtc_scores = []
        for i in range(1, len(x.columns) + 1):
            dtc_classifier = DecisionTreeClassifier(max_features=i, ran
    dom_state=0)
            dtc_classifier.fit(x_train, y_train)
            dtc_scores.append(dtc_classifier.score(x_test, y_test))

        print(dtc_scores)
        best_feature = dtc_scores.index(max(dtc_scores)) + 1
        print(best_feature)
        print(max(dtc_scores))
```

在本实验应用的划分数据集下得到结果为 9，在最优条件下构造决策树算法如下

```
        #找到最优参数，生成最优决策树
        dec = DecisionTreeClassifier(max_features=best_feature)
        # 用决策树进行预测
        dec.fit(x_train, y_train)
        # 预测准确率
        print("预测的准确率：", dec.score(x_test, y_test))
        plt_show(dec, x, y,'决策树')
```

3.2.3 预测结果可视化

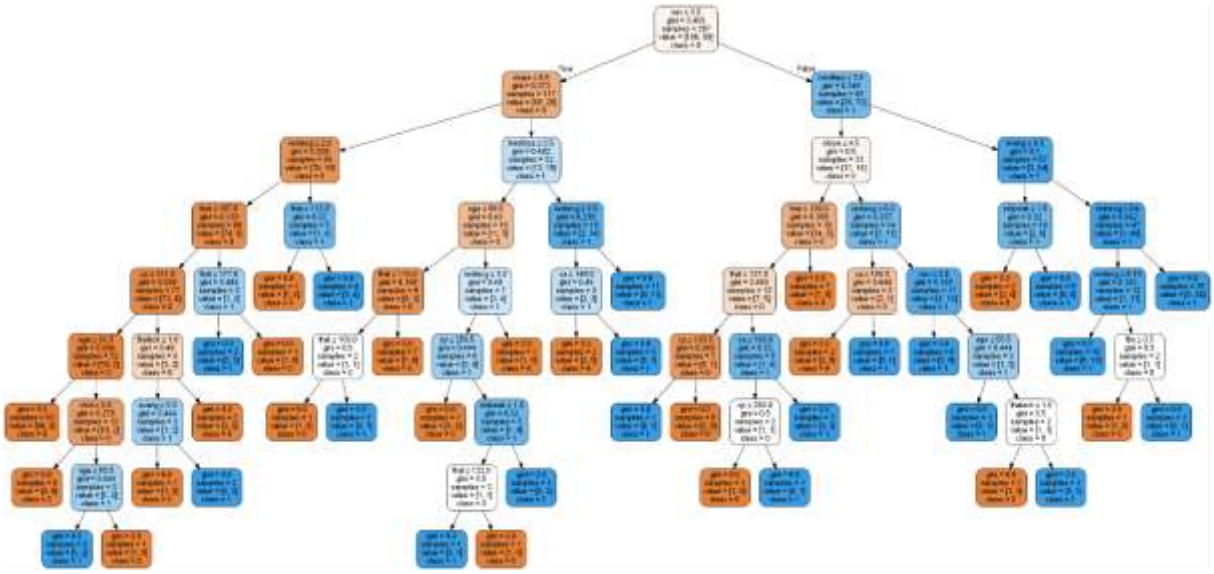

图 3.3 决策树可视化

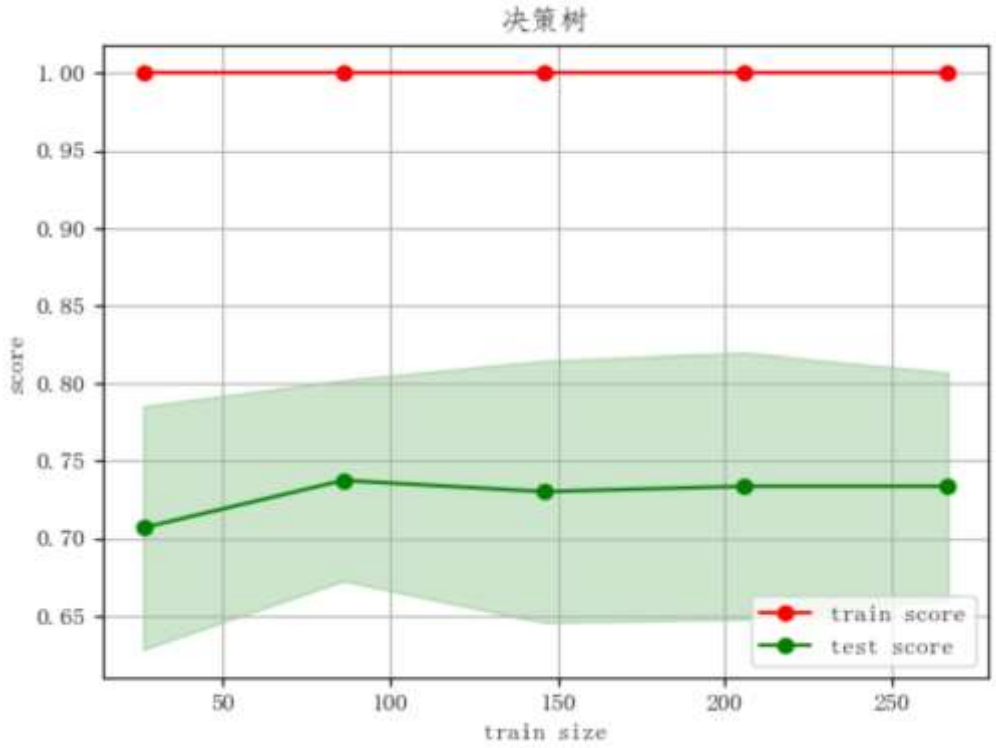

图 3.4 决策树训练集与测试集拟合效果图

如图所示，决策树在训练集上表现出高度的过拟合性，在测试集上的表现远不如训练集

3.2.4 算法优化：随机森林算法实现

随机森林本质上是许多决策树的集合，其中每棵树都和其他树略有不同。随机森林背后的思想是，每棵树的预测可能都相对较好，但可能对部分数据过拟合。如果构造很多树，并且每棵树的预测都很好，但都以不同的方式过拟合，那么我们可以对这些树的结果取平均值来降低过拟合。既能减少过拟合又能保持树的预测能力。

```
# 用随机森林进行预测(超参数调优)
rf = RandomForestClassifier()
# 参数准备
param = {"n_estimators": [120, 200, 300, 500, 800, 1200], "max_depth": [5, 8, 15, 25, 30]}
# 网格搜索与交叉验证
rf = GridSearchCV(rf, param_grid=param, cv=2)
rf.fit(x_train, y_train)
print("随机森林调优结果如下：")
print("准确率:", rf.score(x_test, y_test))
print("选择的参数模型：", rf.best_params_)
```

运行结果如下：

```
随机森林调优结果如下:
准确率: 0.8222222222222222
选择的参数模型:  {'max_depth': 25, 'n_estimators': 200}
```

图 3.5 随机森林算法运行结果

由运行结果可得，运用随机森林算法可以优化决策树，随机森林中的每个决策树可以分布式的训练，解决了单棵决策树在数据量大的情况下预算量大的问题。当训练样本中出现异常数据时，决策树的抗干扰能力差，对于随机森林来说也解决了模型的抗干扰能力。

3.3 SVM

支持向量机（SVM）是一类按监督学习方式对数据进行二元分类的广义线性分类器，其决策边界是对学习样本求解的最大边距超平面，可以将问题化为一个求解凸二次规划的问题。

具体来说就是在线性可分时，在原空间寻找两类样本的最优分类超平面。在线性不可分时，加入松弛变量并通过使用非线性映射将低维度输入空间的样本映

射到高维度空间使其变为线性可分，这样就可以在该特征空间中寻找最优分类超平面。

3.3.1 核函数

支持向量机算法分类和回归方法的中都支持线性性和非线性类型的数据类型。非线性类型通常是二维平面不可分，为了使数据可分，需要通过一个函数将原始数据映射到高维空间，从而使得数据在高维空间很容易可分，需要通过一个函数将原始数据映射到高维空间，从而使得数据在高维空间很容易区分，这样就达到数据分类或回归的目的，而实现这一目标的函数称为核函数。

工作原理：当低维空间内线性不可分时，可以通过高位空间实现线性可分。但如果在高维空间内直接进行分类或回归时，则存在确定非线性映射函数的形式和参数问题，而最大的障碍就是高维空间的运算困难且结果不理想。通过核函数的方法，可以将高维空间内的点积运算，巧妙转化为低维输入空间内核函数的运算，从而有效解决这一问题。

常见核函数：
线性核（Linear Kernel）：

$$k(x_i, x_j) = x_i^T x_j \tag{3.1}$$

多项式核（Polynomial Kernel）：

$$k(x_i, x_j) = (x_i^T, x_j)^d \tag{3.2}$$

高斯核（Gaussian Kernel）：

$$k(x_i, x_j) = \exp(-\frac{\| x_i - x_j \|^2}{2\sigma^2}) \tag{3.3}$$

拉普拉斯核（Laplacian Kernel）：

$$k(x_i, x_j) = \exp(-\frac{\| x_i - x_j \|}{2\sigma}) \tag{3.4}$$

Sigmoid 核（Sigmoid Kernel）：

$$k(x_i, x_j) = \tanh(\beta x_i^T x_j + \theta) \tag{3.5}$$

3.3.2 SVM 算法实现

算法验证了四种核函数对预测结果的影响：

```
  svc_scores = []
      kernels = ['linear', 'poly', 'rbf', 'sigmoid']  # 'precomputed'

      for i in range(len(kernels)):
          svc_classifier = SVC(kernel=kernels[i])
          svc_classifier.fit(x_train, y_train)
          svc_scores.append(svc_classifier.score(x_test, y_test))

      bestkernel = kernels[svc_scores.index(max(svc_scores))]
      print(bestkernel)
      print(svc_scores)
      print(max(svc_scores))
```

结果可视化如下：

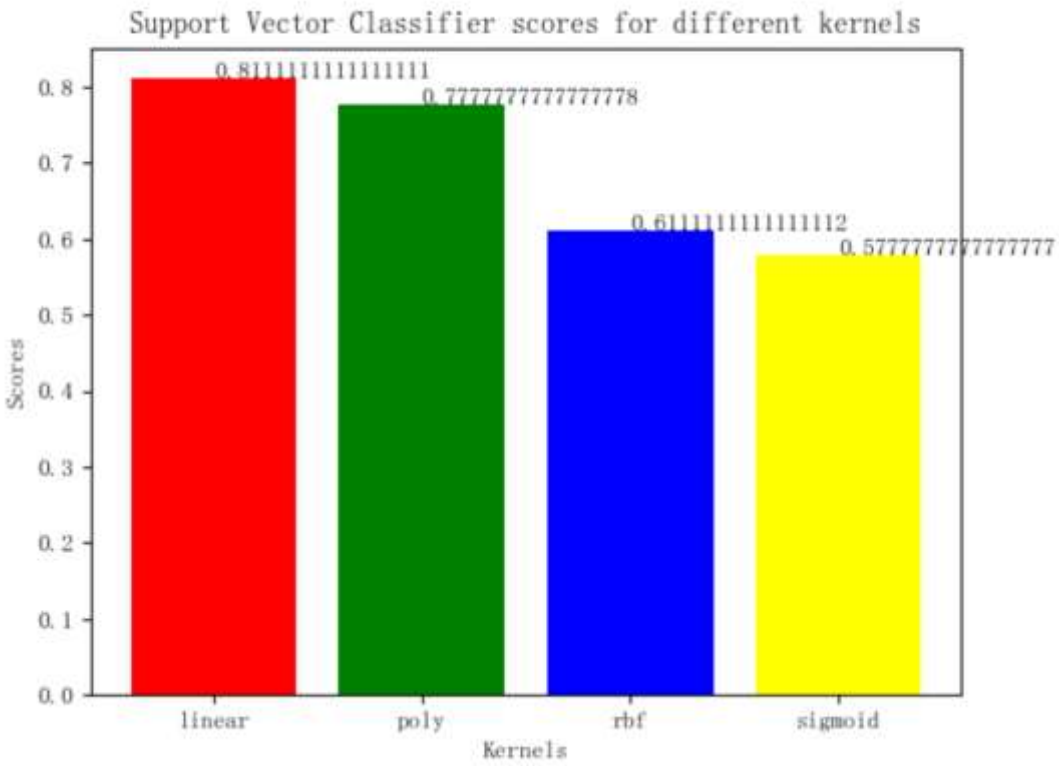

图 3.6 四种核函数的预测准确率

由图可知，本实验中线性核和多项式核的预测效果较好，这与本实验所用数据模型密不可分。

线性核函数主要用于线性可分，其特征空间到输入空间的维度是一样的，参数少速度快，可解释性强。而多项式核函数线性与非线性均适用，通过将低维的输入空间映射到高维的特征空间的方式，使得原本线性不可分的数据线性可分。高斯径向基核函数是一种局部性强的核函数，其可以将一个样本映射到一个更高维的空间内，无论是大样本小样本都有较好的性能。缺点是可解释性差、计算速度比较慢、容易过拟合。而 sigmoid 常用于非线性，神经网络等更复杂的应用场景。

最一般的选择原则是针对数据量很大的时候，可以选择复杂一点的模型。虽然复杂模型容易过拟合，但由于数据量很大，可以有效弥补过拟合问题。如果数据集较小选择简单点的模型，否则很容易过拟合，此时特别要注意模型是否欠拟合，如果欠拟合可以增加多项式纠正欠拟合。

本实验中数据集较小且特征量很少，线性核和多项式核便可达到较好的拟合效果。

3.3.3 预测结果可视化

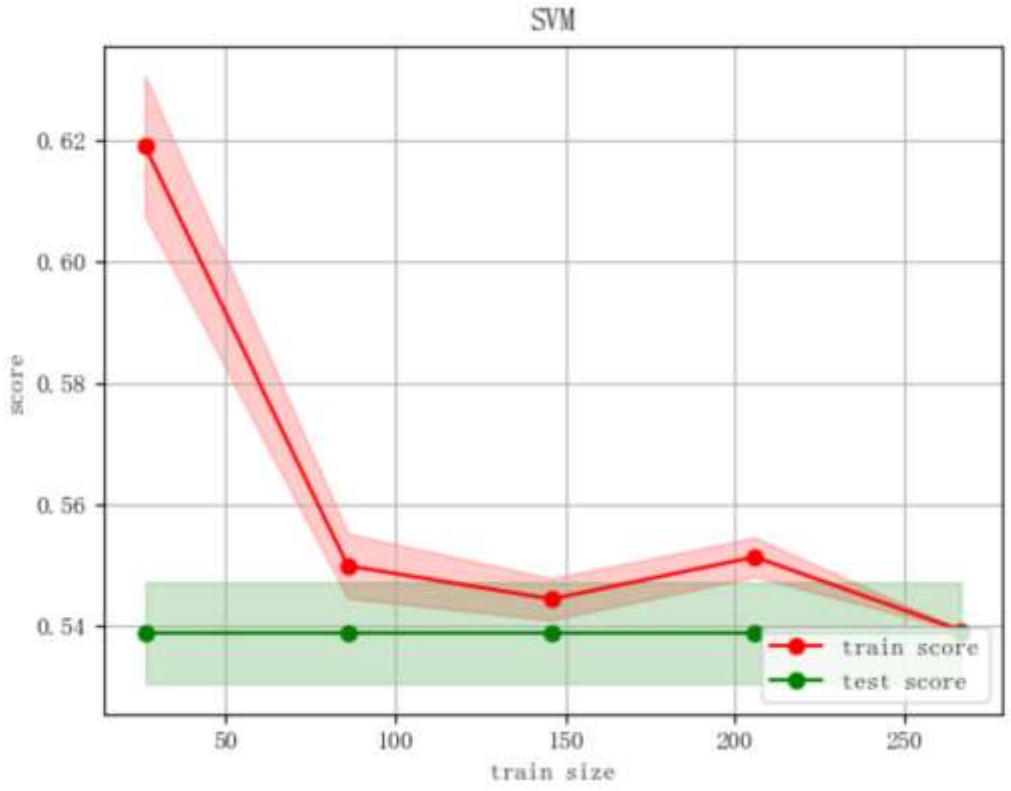

图 3.7 SVM 训练集与测试集拟合效果图

第 4 章 算法总结

4.1 结果对比

我们将本实验四种算法所得预测准确率结果进行对比：

```
01        model = ['KNN', '决策树', '随机森林', 'SVM']
02         score = [score1, score2, score3, score4]
03
04         plt.figure(figsize=(15, 10))
05         sns.barplot(x=score, y=model)
06         plt.show()
```

所得结果如图(4.1)所示

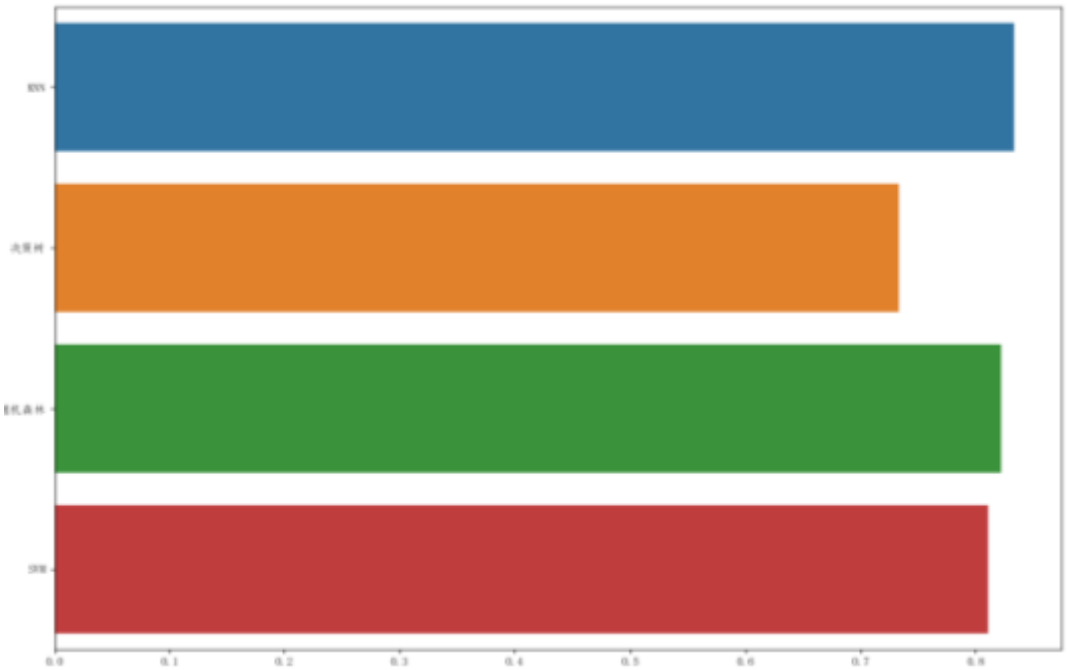

图 4.1 四种算法结果对比

由结果对比可得，本实验使用 KNN 得到最优的预测，并且 KNN 在训练集和测试集上均表现出较优的特性。其次是随机森林，提升了决策树的预测性能，通过分布式训练解决了决策树过拟合的问题。然后是 SVM 支持向量机，通过线性核实现了较好的预测效果。以上三种在本实验数据量不大的基础上均差别不是很大，在时间上，KNN 算法由于其算法简单的特性，效率较高。另外，在 SVM 验证最优核函数上，需要花费较长时间才能得到运行结果。

4.2 算法比较与总结

4.2.1 KNN

优点：

（1）可用于非线性分类

（2）训练时间复杂度为 O(n)

（3）对数据没有假设，准确度高，对 outlier 不敏感

（4）KNN 理论简单，容易实现

缺点：

（1）样本不平衡问题（即有些类别的样本数量很多，而其它样本的数量很少）效果差

（2）需要大量内存

（3）对于样本容量大的数据集计算量比较大（体现在距离计算上）；

（4）KNN 每一次分类都会重新进行一次全局运算

（5）k 值大小的选择没有理论选择最优，往往是结合 K-折交叉验证得到最优 k 值选择

算法应用领域：

文本分类、模式识别、聚类分析，多分类领域

4.2.2 决策树

优点：

（1）决策树易于理解和解释，可以可视化分析，容易提取出规则

（2）可以同时处理标称型和数值型数据

（3）比较适合处理有缺失属性的样本

（4）能够处理不相关的特征

（5）测试数据集时，运行速度比较快

（6）在相对短的时间内能够对大型数据源做出可行且效果良好的结果

缺点：

（1）容易发生过拟合（随机森林可以很大程度上减少过拟合）

（2）容易忽略数据集中属性的相互关联

（3）对于那些各类别样本数量不一致的数据，在决策树中，进行属性划分时，不同的判定准则会带来不同的属性选择倾向；信息增益准则对可取数目较多的属性有所偏好（典型代表 ID3 算法），而增益率准则（CART）则对可取数目较少的

属性有所偏好，但 CART 进行属性划分时候不再简单地直接利用增益率尽心划分，而是采用一种启发式规则）（只要是使用了信息增益，都有这个缺点，如 RF）
（4）ID3 算法计算信息增益时结果偏向数值比较多的特征

算法应用领域：
企业管理实践，企业投资决策，由于决策树很好的分析能力，在决策过程应用较多。

4.2.3 随机森林

优点：
（1）在当前的很多数据集上，相对其他算法有着很大的优势，表现良好
（2）它能够处理很高维度的数据，并且不用做特征选择(因为特征子集是随机选择的)
（3）在创建随机森林的时候，对 generlization error 使用的是无偏估计，模型泛化能力强
（4）训练速度快，容易做成并行化方法(训练时树与树之间是相互独立的)
（5）对于不平衡的数据集来说，它可以平衡误差
（6）如果有很大一部分的特征遗失，仍可以维持准确度

缺点：
（1）随机森林在解决回归问题时，并没有像它在分类中表现的那么好，这是因为它并不能给出一个连续的输出。当进行回归时，随机森林不能够做出超越训练集数据范围的预测，这可能导致在某些特定噪声的数据进行建模时出现过度拟合
（2）对于许多统计建模者来说，随机森林给人的感觉就像一个黑盒子，你无法控制模型内部的运行。只能在不同的参数和随机种子之间进行尝试
（3）可能有很多相似的决策树，掩盖了真实的结果
（4）对于小数据或者低维数据（特征较少的数据），可能不能产生很好的分类。（处理高维数据，处理特征遗失数据，处理不平衡数据是随机森林的长处）

算法应用领域：
既可以用于分类也可以用于回归问题，不适用于需要高实时的场景。

4.2.4 SVM

优点：
（1）可以解决高维问题，即大型特征空间
（2）解决小样本下机器学习问题

（3）能够处理非线性特征的相互作用
（4）无局部极小值问题（相对于神经网络等算法）
（5）无需依赖整个数据
（6）泛化能力比较强

缺点：
（1）当观测样本很多时，效率并不是很高
（2）对非线性问题没有通用解决方案，有时候很难找到一个合适的核函数
（3）对于核函数的高维映射解释力不强，尤其是径向基函数
（4）常规 SVM 只支持二分类
（5）对缺失数据敏感

算法应用领域：
文本分类、图像识别（主要二分类领域）

4.3 总结

在机器学习领域，一个基本的定理就是“没有免费的午餐”。「换言之，就是没有算法能完美地解决所有问题，尤其是对监督学习而言（例如预测建模）」。

举例来说，你不能去说神经网络任何情况下都能比决策树更有优势，反之亦然。它们要受很多因素的影响，比如你的数据集的规模或结构。

其结果是，在用给定的测试集来评估性能并挑选算法时，你应当根据具体的问题来采用不同的算法。

第 5 章 实验总结

5.1 实验优点

本实验基于同一数据集上的相同划分，用于比较各算法之间预测准确度的优劣。既有纵向各个算法的延伸，又有横向不同算法间的比较与联系。

对于 KNN，我们进行最优 K 值的选取，绘制 K 值变化曲线；对于决策树，考虑最大特征数，并选取最优结果构建决策树；随机森林采用网格搜索与交叉验证，在指定的参数范围内，按步长依次调整参数，从所有的参数中找到在验证集上精度最高的参数；SVM 测试了不同的核函数对算法的影响，测试全面而精细。

最后我们横向比较了几个算法，绘制了比较图，得到了较为清晰的比较结果。

5.2 实验缺点

5.2.1 实验集数据过小

本实验选取 297*14 的数据集，数据量较小。模型容易过拟合。模型不仅仅容易在训练集上出现过拟合的问题，而且也可能在验证集上出现过拟合问题，最终造成模型的稳定性降低，这一点在决策树算法中较为明显。

异常值难以避免。异常值可能出现在特征里，也可能出现在响应变量中，对这些异常值进行处理的成本很高；与此同时异常值也可能导致训练样本和测试样本数据分布不一致，降低模型稳定性。好在本实验在数据处理时进行了异常值的查看与处理。

另外数据不平衡的特点导致难以进行模型优化。最终造成模型稳定性降低。由于训练集和实验集的划分随机，有的时候会因为划分的数据不平衡，导致不同划分产生的实验结果有很大差别。

5.2.2 数据处理不足

机器学习里算法固然重要，但好的数据却要优于好的算法，因此对数据进行有效加工尤为重要。

图 5.1 特征预处理

数据预处理方法如图 5.1 所示，本实验中仅采用了简单的数据标准化和独热编码，对于数据量较小的情况，应引入传统统计工具等，使用正则化处理降低数据不平衡。

另外数据划分上，本实验仅选择 sklearn 库中的划分函数，实际应用中可以采取留出（Hold-out）法、交叉验证(cross validation)法、自助法(bootstrap)进行数据划分。

业界广泛流传：数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限而已。因此本实验可以在此基础上对数据和特征进行加工，达到更好的预测效果。