北京科技大学
University of Science and Technology Beijing

人工智能基础 II——人工智能与机器学习

2022-2023 学年第一学期课程考核报告

题　　目：基于机器学习的 Al-Co-Cr-Cu-Fe-Ni 系高熵合金硬度预测

小组成员：杨小文 M202211401、景思睿 M202211359
高景鑫 M202210294、何佳璇 M202211348
金显哲 M202221438、王洪港 M202221592

授课教师：班晓娟、罗熊

2022 年 12 月 16 日

基于机器学习的 Al-Co-Cr-Cu-Fe-Ni 系高熵合金硬度预测

摘要

机器学习在材料研发方面发挥着越来越重要的作用，材料研究模式也由“经验指导实验”向“理论预测和实验验证相结合”转变，高熵合金凭借着这样的独特的设计理念赋予了其众多优异的性能。本文我们提出了一种结合机器学习模型算法进行材料设计的策略，在 Al-Co-Cr-Cu-Fe-Ni 体系中搜索大硬度的高熵合金。本文首先通过收集到的 212 条 Al-Co-Cr-Cu-Fe-Ni 系高熵合金成分数据集进行特征计算、筛选得到需要的 11 个性能特征组合。其次，基于以上数据，选择了支持向量机、弹性回归、LASSO 回归、岭回归、神经网络回归和随机森林回归 6 种不同模型对数据集进行训练，并得到最佳的拟合模型是随机森林回归模型。最后，使用随机森林回归模型进行训练、验证和预测，在搜索空间中进行硬度预测和搜索，得到最高预测硬度的高熵合金为：$Al_{0.23}Co_{0.09}Cr_{0.19}Cu_{0.05}Fe_{0.12}Ni_{0.32}$，预测硬度为 621.55HV。

关键词： 高熵合金，机器学习，材料基因组计划，硬度预测

Abstract

Machine learning plays an increasingly important role in material research and development, and the material research mode has been transformed from experience-guided experiment to the combination of theoretical prediction and experimental verification. With such a unique design concept, high entropy alloy has been endowed with many excellent properties. In this paper, we propose a material design strategy combining machine learning model algorithm to search for high hardness and high entropy alloys in Al-Co-Cr-Cu-Fe-Ni system. In this paper, 212 pieces of Al-Co-Cr-Cu-Fe-Ni high entropy alloy composition data set were collected for feature calculation and screening to obtain the required 11 performance characteristic combinations. Secondly, based on the above data, six different models including support vector machine, elastic regression, LASSO regression, ridge regression, neural network regression and random forest regression were selected to train the data set, and the best fitting model was random forest regression. Finally, the use of random forests regression model for training, validation, and prediction of hardness prediction and search in the search space, get the highest hardness high entropy alloys for: $Al_{0.23}Co_{0.09}Cr_{0.19}Cu_{0.05}Fe_{0.12}Ni_{0.32}$, the predicted hardness is 621.55HV.

Keyword: **High entropy alloys, Machine learning, Materials genome initiative, Hardness prediction**

1 项目背景

高熵合金是近十几年来出现的一类新型金属材料，通常由 5 种或 5 种以上的元素以等原子比或近等原子比构成，形成以固溶体相为主的组织结构，高熵合金各种元素的的质量分数在 5%~35%之间。由于高熵合金具有大晶格畸变、高混合熵、原子缓慢扩散和“鸡尾酒”效应等多重效应，所以它显示出高强度、高韧性、高硬度、异常优异的低温韧性、优异的耐腐蚀和抗辐照等独特性能。

目前研究表明，高熵合金中面心立方和体心立方固溶体种类、原子半径差异造成的晶格畸变、纳米或 Laves 等增强相形成是决定性能的主要原因。且高熵合金中由于元素较多且比例较为平均，因此元素的种类和含量的改变就会引起材料组织和性能的较大改变。所以，通过选择性的添加某些合金元素并控制合金元素的比例，就可以改变合金的晶体结构和相组成，从而实现合金性能的突破。

由于高熵合金具有多种组元成分，存在着巨大的合金成分设计空间，因此，继续使用传统的试错法来寻找新的高熵合金成分组合无异于大海捞针且十分费时费力。而随着第一原则计算方法和计算工具的飞速发展，目前也有了很多使用计算机模拟的方法预测合金成分，但是这样的方法仍然有一定的局限性且更适用于小数据集测试获得预测的良好性能。

但随着大数据人工智能技术的发展，学者们开始使用机器学习对已有的材料数据构建描述符模型，从而有效找到材料微观结构与性能之间的复杂关系。目前已经有大量的工作证实，使用机器学习可以更高效地寻找高熵合金的成分配比，且研发成本更低。

2 实验方法

2.1 数据收集

本项目的实验数据集主要来自于文献[1]，共采集了 151 条关于该 Al-Co-Cr-Cu-Fe-Ni 系高熵合金的成分信息。为了提高模型预测的精确度，本文又调研了其它文献[2-4]中的数据来进行补充，最后得到了 212 条 Al-Co-Cr-Cu-Fe-Ni 系高熵合金成分数据集。

2.2 建立特征数据集

本实验方案通过文献调研并收集了 16 个高熵合金系的物理特征，包括原子半径差（δr）、电负性差（$\Delta\chi$）、价电子浓度（VEC）、混合焓（ΔH）、构型熵（ΔS）、Ω 参数（Ω）、Λ 参数（Λ）、局部电负性失配（D.χ）、流动电子

数目（*e/a*）、内聚能（*Ec*）、能量项（A）、纳巴罗系数（F）、功函数（W）、剪切模量（G）、剪切模量差（δG）、晶格畸变能（μ），计算公式如下。

$$\delta r = \sqrt{\sum_{i=1}^{n} C_i \bullet (1 - \frac{r_i}{r})^2} \tag{2-1}$$

$$\Delta\chi = \sqrt{\sum_{i=1}^{n} C_i \bullet (\chi - \chi_i)^2} \tag{2-2}$$

$$VEC = \sum_{i=1}^{n} C_i \bullet VEC_i \tag{2-3}$$

$$\Delta H = \sum_{i=1, j>1}^{n} 4C_i C_j \bullet \Delta H_{ij}^{mix} \tag{2-4}$$

$$\Delta S = -R\sum_{i=1}^{n} C_i \bullet \ln C_i \tag{2-5}$$

$$\Omega = Tm \bullet \frac{\Delta S}{|\Delta H|} \tag{2-6}$$

$$\Lambda = \frac{\Delta S}{\delta r \bullet \delta r} \tag{2-7}$$

$$D.\chi = \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} C_i C_j \bullet |\chi - \chi_i| \tag{2-8}$$

$$e/a = \sum_{i=1}^{n} C_i \bullet (e/a)_i \tag{2-9}$$

$$Ec = \sum_{i=1}^{n} C_i \bullet (Ec)_i \tag{2-10}$$

$$A = G \bullet \delta r \bullet (1+\mu)(1-\mu) \tag{2-11}$$

$$F = \frac{2G}{1-\mu} \tag{2-12}$$

$$W = (\sum_{i=1}^{n} C_i \bullet w_i)^6 \tag{2-13}$$

$$G = \sum_{i=1}^{n} C_i \bullet G_i \tag{2-14}$$

$$\delta G=\sqrt{\sum_{i=1}^{n}C_i\bullet(1-\frac{G_i}{G})^2} \tag{2-15}$$

$$\mu=\frac{1}{2}E\delta r \tag{2-16}$$

以上 2-1 到 2-16 十六个公式为高熵合金 Al-Co-Cr-Cu-Fc-Ni 体系的物理特征的公式。其中 C_i、r_i、Xi、$VECi$、$(e/a)i$、Eci、Gi、wi 分别代表成分比、原子半径、鲍林电负性、价电子浓度、电子密度、内聚能、剪切模量和第 i 个元素的功函数；T、r、X、G、E、μ 分别代表熔点的均值、原子半径的均值、鲍林电负性的均值、剪切模量的均值、杨氏模量的均值以及泊松比的均值；ΔH_{ij}^{mix} 代表第 i 和第 j 个元素的混合焓。经过调研找到了 16 种特征计算所需要的所有变量，通过 Excel 计算得到了 212 种成分配比下的 16 种物理特征。

在确定 16 个物理特征后，本文采用皮尔逊相关性分析方法对 16 个物理特征进行了进一步的筛选，确定了的 11 个主要特征。

2.3 模型选择与构建

基于以上数据，选择了支持向量机、弹性回归、LASSO 回归、岭回归、神经网络回归和随机森林回归 6 种不同模型对数据集中 212 条数据进行训练，并使用决定系数 R^2 找出训练效果最优的模型。选出最优的模型后，本文将 212 条数据按 70%和 30%的比例划分为训练集与测试集，并构建模型在搜索空间中进行硬度预测和搜索，在预测搜索过程中找到的硬度较高的高熵合金。

3 实验结果

本文通过皮尔孙相关性分析确定的 11 个主要特征如图 3-1 所示，11 个主要特征分别为：原子半径差（δr）、混合焓（ΔH）、构型熵（ΔS）、Ω 参数（Ω）、Λ 参数（Λ）、局部电负性失配（D.χ）、内聚能（Ec）、能量项（A）功函数（W）、剪切模量差（δG）、晶格畸变能（μ）。

接下来，本文选用了回归神经网络、支持向量机回归 SVR、岭回归、弹性回归、LASSO 回归和随机森林回归对以上数据进行了训练，并通过 R^2 来比较模型的拟合程度，确定训练效果最佳的模型，具体 Python 代码如附录 1 所示，将六个模型的决定系数 R^2 的值绘制成如下表格 3-1，R^2 结果保留四位有效数字。

表 3-1 不同回归模型的比较

模型	神经网络	SVR	岭回归	弹性回归	LASSO	随机森林
R^2	0.8217	0.6257	0.8561	0.7719	0.8463	0.9854

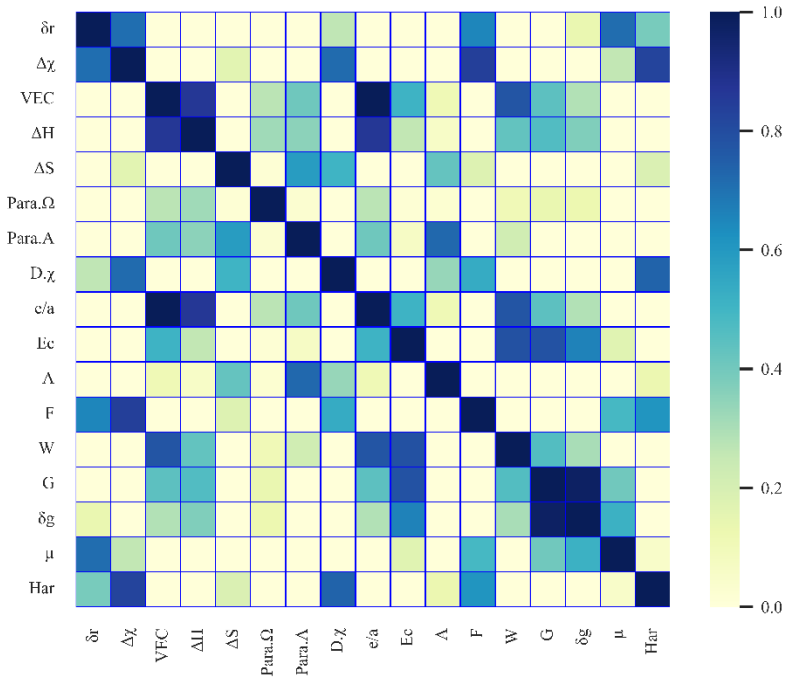

图 3-1 皮尔孙热图相关性分析结果

通过该表 3-1 可以发现，随机森林回归模型的决定系数值最高，R^2 为 0.9854，其次是岭回归模型，R^2 值为 0.8561。决定系数值最低的为高斯核的支持向量回归模型，R^2 值为 0.6257。R^2 的值越接近 1，说明回归模型对测试集的拟合程度越好；反之，R^2 的值越小，说明回归直线对观测值的拟合程度越差。由此判断出，随机森林回归模型为训练效果最优的模型。图 3-2 至 3-7 为以上六种不同回归模型的拟合结果图。

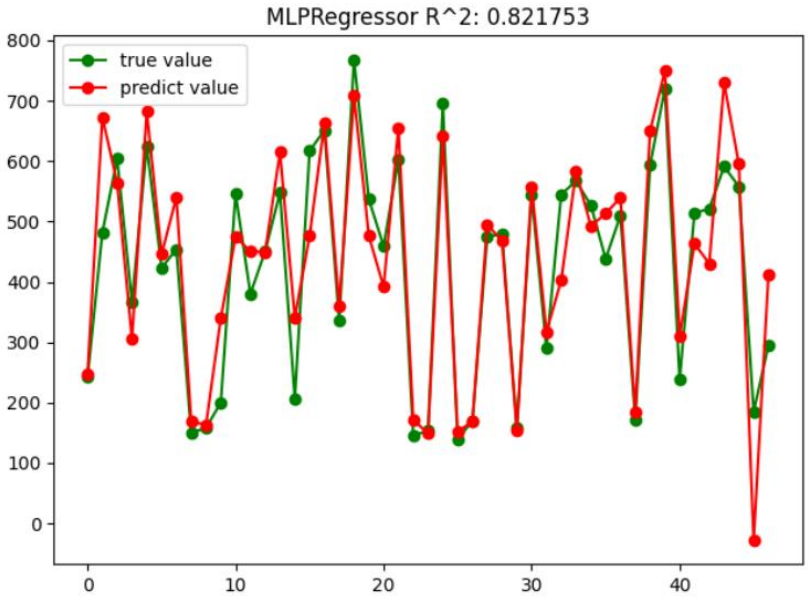

图 3-2 神经网络回归模型的拟合结果图

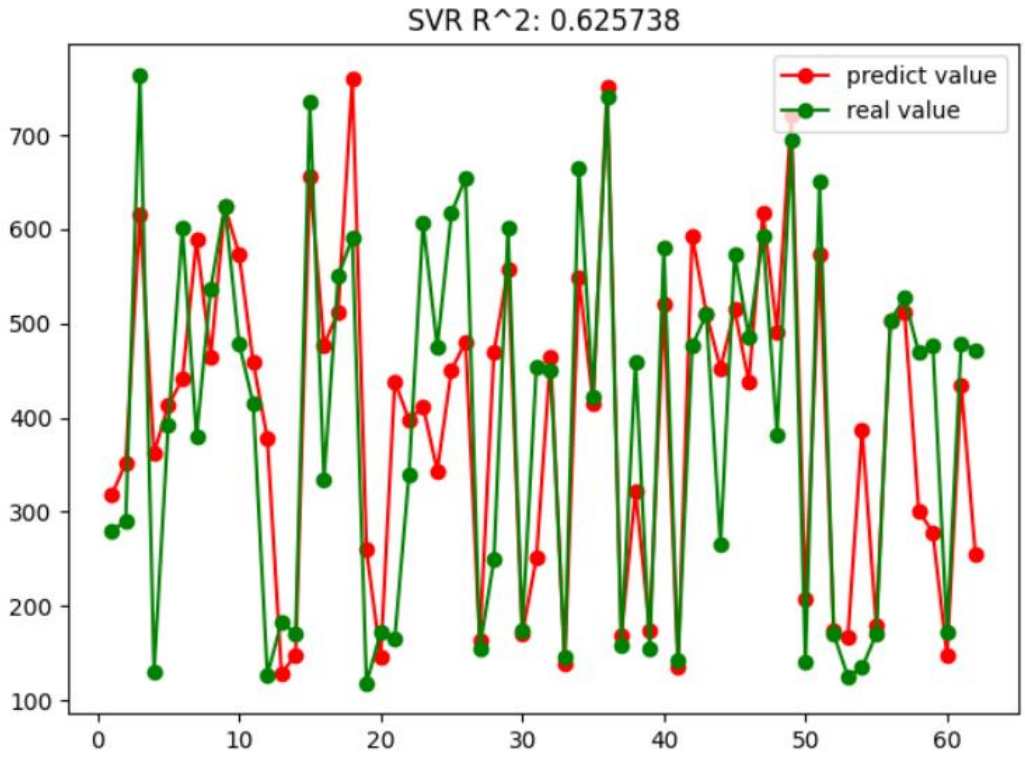

图 3-3 支持向量回归模型的拟合结果图

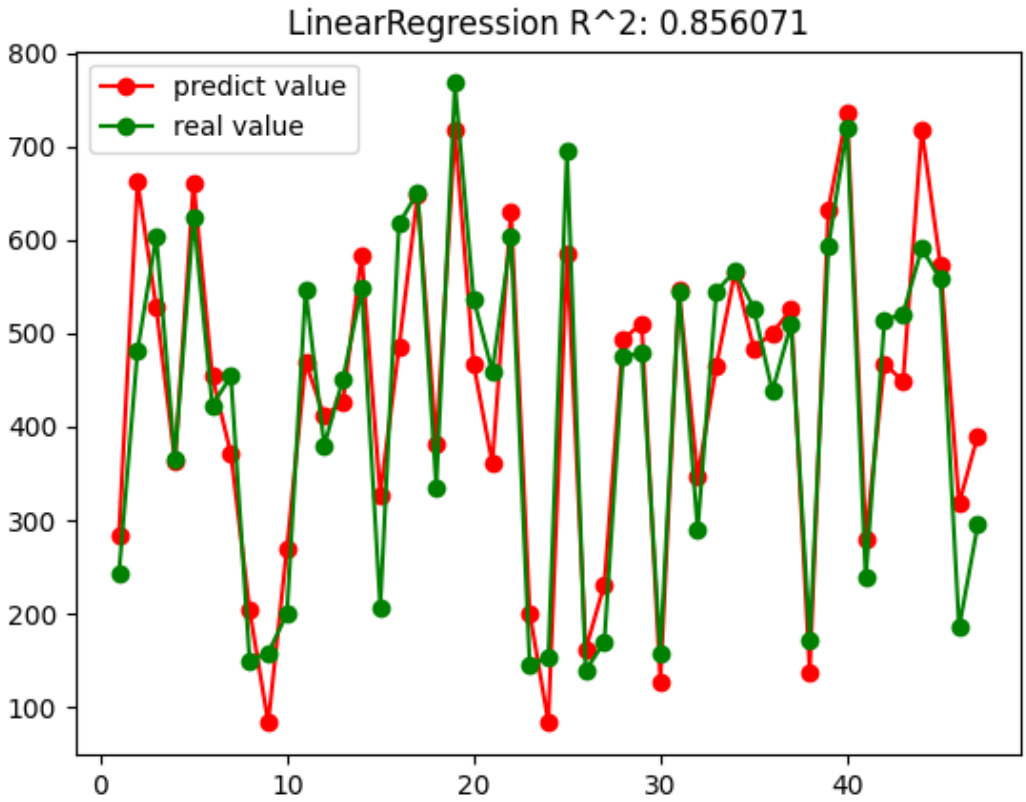

图 3-4 岭回归模型的拟合结果图

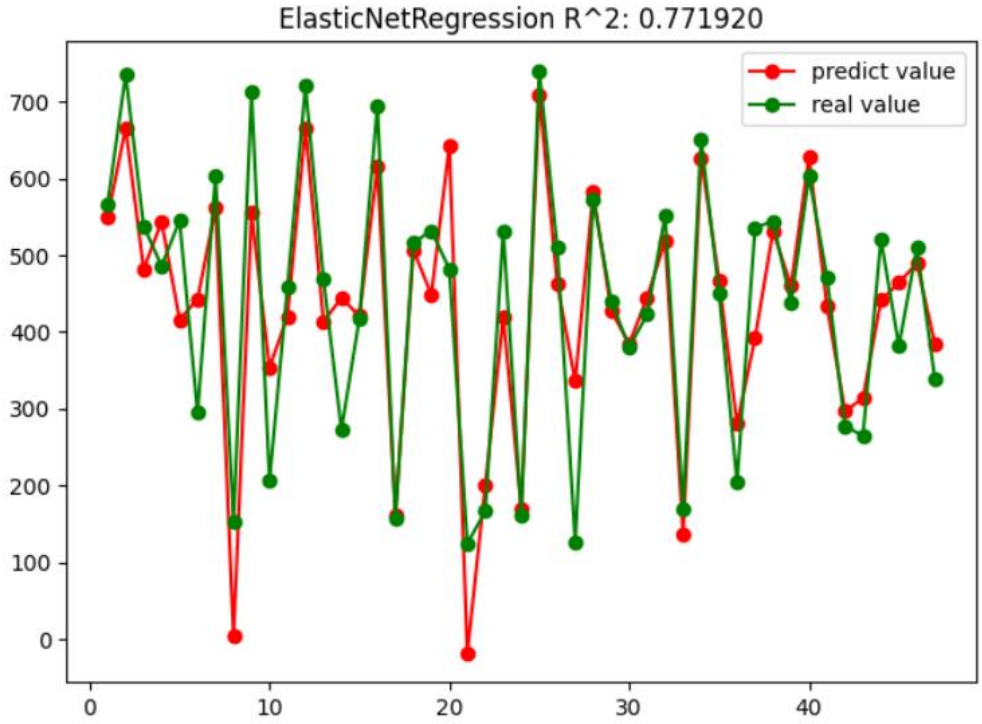

图 3-5 弹性回归模型的拟合结果图

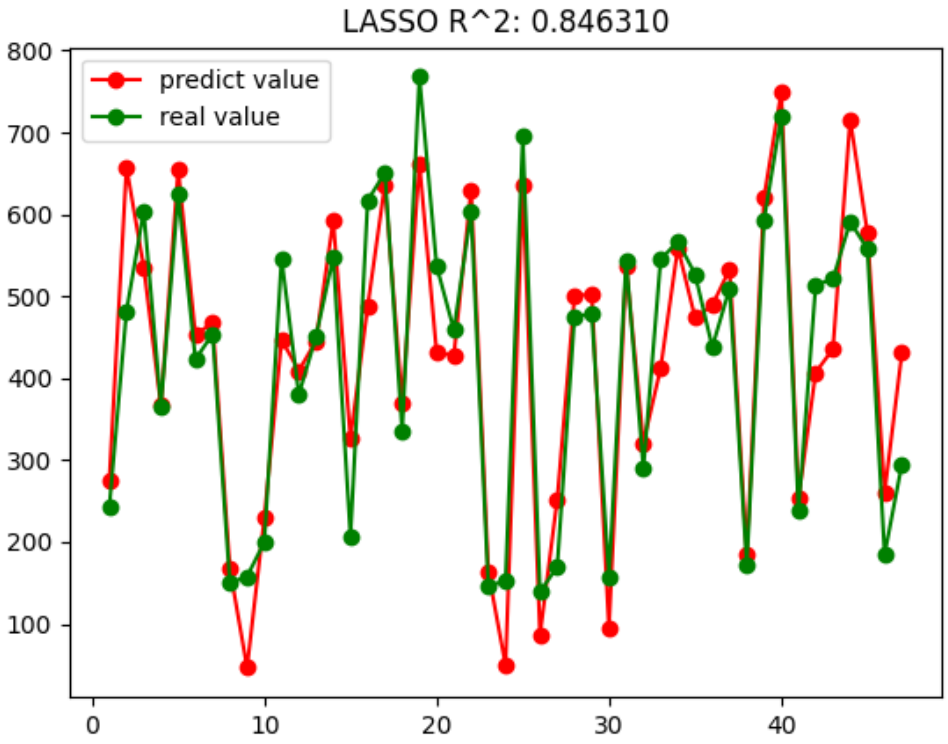

图 3-6 LASSO 回归模型的拟合结果图

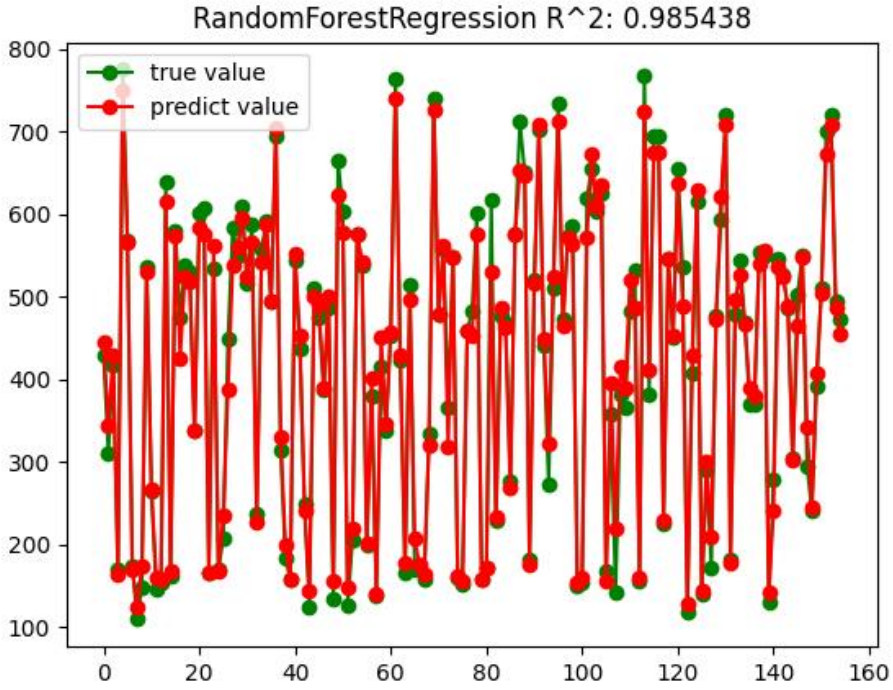

图 3-7 随机森林回归模型的拟合结果图

确定最优模型后，本文使用随机森林回归模型进行训练与预测，将数据集按 70%和 30%划分为训练集和测试集，构建模型在搜索空间中进行硬度预测和搜索。最终得到如下图 3-8 所示，以及高熵合金硬度预测如表 3-2 所示。

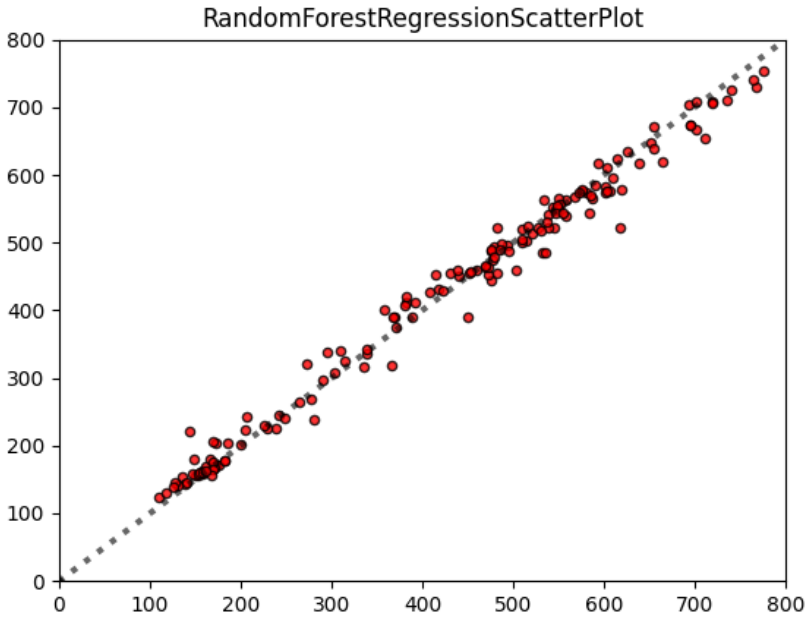

图 3-8 测试集中高熵合金实际硬度与预测硬度的回归曲线图

表 3-2 该五元系高熵合金硬度预测结果（按硬度由大到小排列）

Al/%	Co/%	Cr/%	Cu/%	Fe/%	Ni/%	Hardness/HV
23	**9**	**19**	**5**	**12**	**32**	**621.55**
26	16	25	9	16	8	617.89
22	5	21	5	20	27	612.63
15	17	24	8	11	25	611.89
17	10	19	13	12	29	597.77
15	6	5	32	10	32	588.88
19	18	18	18	9	18	580.73
19	7	31	22	7	14	577.57
19	8	19	20	18	16	572.40
20	10	5	19	14	32	572.17

表 3-2 所示硬度最高的 12 种高熵合金的具体成分与预测硬度中表示的是预测出的高熵合金中每个元素的成分比重，最高预测硬度的高熵合金为：$Al_{0.23}Co_{0.09}Cr_{0.19}Cu_{0.05}Fe_{0.12}Ni_{0.32}$，预测硬度为 621.55 HV。图 3-8 为测试集中高熵合金实际硬度与预测硬度的回归曲线图。

到此为止，本文的工作已完成了整个机器学习建模的流程，包括建立数据集，特征筛选，模型建模，模型预测等工作。

4 结论

本实验首先从不同文献中，总共收集了 212 条成分数据，并通过计算得到了基于成分配比的 16 种物理特征集，之后采用皮尔逊相关性筛选选定其中 11 种主要物理特征。接下来，选用了 sklearn 库中的 6 种不同模型对得到的 212 条特征集进行训练，并通过决定系数 R^2 的值判断回归拟合效果最好的模型，即随机森林模型。在确定训练效果最优的模型后，将数据集按 70%和 30%划分为训练集和测试集预测搜索过程中找到的硬度较高的高熵合金成分，得到最高预测硬度的高熵合金为：$Al_{0.23}Co_{0.09}Cr_{0.19}Cu_{0.05}Fe_{0.12}Ni_{0.32}$，预测硬度为 621.55 HV。

本实验从多种不同来源筛选数据，对原数据集（155 条数据）进行了合理扩充（212 条数据），训练结果更可靠。并使用了多种模型进行比较，找到了最优模型——随机森林模型，并通过随机森林模型得到了 Al-Co-Cr-Cu-Fe-Ni 体系高熵合金的最优硬度合金成分。

但是，由于数据上还存在些不足，特征的初次筛选过程中人为忽略了部分高熵合金物理特征。以及，在使用随机森林模型进行预测和搜索时，只划分了训练集和测试机，没有划分验证集。由于目前高熵合金的实验数据仍然比较缺

乏，所构建的模型的泛化能力仍然需要通过实验进行进一步的验证。在答辩结束后，听取老师的意见进行了模型优化，具体见附录 1。

5 未来展望

高熵合金除 Al-Co-Cr-Cu-Fe-Ni 体系外，还具有广泛的组分空间，且不单单只具有高硬度这一优良力学性能，通过机器学习方法，还可以将高熵合金除硬度外其他的优良性能作为功能目标，利用机器学习方法可以在多目标寻优、大组分空间寻优等方面进一步探索。希望通过机器学习方法，更多具有良好机械性能或能用于特定环境下的高熵合金成分配比能被找到。

参考文献

[1] Wen C, Zhang Y, Wang C, et al. Machine learning assisted design of high entropy alloys with desired property [J]. Acta Materialia, 2019, 170: 109-117.

[2] 赵鼎祺，乔珺威，吴玉程．机器学习辅助高熵合金设计的研究进展[J]．中国材料进展，2021，40(07)：508-517．

[3] Xiong J, Shi S-Q, Zhang T-Y. Machine Learning of Phases and Mechanical Properties in Complex Concentrated Alloys [J]. Journal of Materials Science & Technology, 2021, 87: 133–142.

[4] Yang C, Ren C, Jia Y, et al. A machine learning-based alloy design system to facilitate the rational design of high entropy alloys with enhanced hardness [J]. Acta materialia, 2022, 222: 117431-117440.

[5] PICKERING EJ, JONES NG. High-entropy alloys: a critical assessment of their founding principles and future prospects [J]. International Materials Reviews, 2016, 61(3): 183–202.

附录 1：模型优化

答辩结束，在听取老师的建议后，我们在随机森林的模型中，按照 7:2:1 的比例重新设立了训练集、验证集和测试集，并用五折交叉验证对模型的泛化能力进行评估，具体代码如图附录 2 的随机森林回归所示，发现模型的泛化能力确实存在不足，有待提高。

附录 2：相关模型的 Python 代码

1. 皮尔逊热图相关性分析

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import seaborn as sns
Input = pd.DataFrame(pd.read_excel(r"C:\Users\y1532\PycharmProjects\untitled\data11\参数.xlsx"))
Input2 = pd.DataFrame(pd.read_excel(r"C:\Users\y1532\PycharmProjects\untitled\data11\硬度.xlsx"))
B = pd.concat([Input, Input2],axis =1)#组合特征列和硬度
#绘制相关系数的热力图
r_pearson = B.corr()
rc = {'font.sans-serif': 'Times New Roman',
      'axes.unicode_minus': False}
sns.set(font_scale=0.7,rc=rc)
sns.heatmap(B.corr(),
            center=0.5,  # 居中
            fmt='.2f',  # 只显示两位小数
            linewidth=0.5,  # 设置每个单元格的距离
            linecolor='blue',  # 设置间距线的颜色
            vmin=0, vmax=1,  # 设置数值最小值和最大值
            xticklabels=True, yticklabels=True,  # 显示 x 轴和 y 轴
            square=True,  # 每个方格都是正方形
            cbar=True,  # 绘制颜色条
            cmap='YlGnBu',  # 设置热力图颜色
            )
plt.savefig("热力图.png",dpi=1800)#保存图片，分辨率为 600
plt.show()
```

2. 随机森林回归

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import math
import xlrd
import xlwt
import random
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
#1.载入数据并且打乱数据集
def load_data(StartPo,EndPo,TestProportion,FeatureNum,Shuffle,FilePath):          #样本起始行数，结束行数，测试集占总样本集比重,特征数，是否打乱样本集      #如果 Testproportion 为 0 或 1 就训练集=测试集
    #打开 excel 文件
    workbook = xlrd.open_workbook(str(FilePath))        #excel 路径
    sheet = workbook.sheet_by_name('Sheet1')              #sheet 表
    Sample = []#总样本集
    train = []#训练集
    test = []#测试集
    TestSetSphere = (EndPo-StartPo+1)*TestProportion  #测试集数目
    TestSetSphere = int(TestSetSphere)#测试集数目
    #获取全部样本集并打乱顺序
    for loadi in range(StartPo-1,EndPo):
        RowSample = sheet.row_values(loadi)
        Sample.append(RowSample)
    if Shuffle == 1:  #是否打乱样本集
        random.shuffle(Sample)  #如果 shuffle=1，打乱样本集
    #如果 Testproportion 为 0 就训练集=测试集
    if TestProportion == 0 or TestProportion == 1:
        TrainSet = np.array(Sample)          #变换为 array
        TestSet = np.array(Sample)
    else:
        #设置训练集
        for loadtraina in Sample[:(EndPo-TestSetSphere)]:
            GetTrainValue = loadtraina
            train.append(GetTrainValue)
        #设置测试集
        for loadtesta in range(-TestSetSphere-1,-1):
            GetTestValue = Sample[loadtesta]
            test.append(GetTestValue)
        #变换样本集
        TrainSet = np.array(train) #变换为 array
        TestSet = np.array(test)
   #分割特征与目标变量
    x1 , y1 = TrainSet[:,:FeatureNum] , TrainSet[:,-1]
    x2 , y2 = TestSet[:,:FeatureNum] , TestSet[:,-1]
```

```
    return x1 , y1 , x2 , y2
#2.回归部分
def regression_method(model):
    model.fit(x_train,y_train)
    score = model.score(x_test, y_test)
    result = model.predict(x_test)
    ResidualSquare = (result - y_test)**2     #计算残差平方
    RSS = sum(ResidualSquare)   #计算残差平方和
    MSE = np.mean(ResidualSquare)       #计算均方差
    num_regress = len(result)   #回归样本个数
    print(f'n={num_regress}')
    print(f'R^2={score}')
    print(f'MSE={MSE}')
    print(f'RSS={RSS}')
############绘制折线图##########
    plt.figure()
    plt.plot(np.arange(len(result)), y_test,'go-
',label='true value')
    plt.plot(np.arange(len(result)),result,'ro-
',label='predict value')
    plt.title('RandomForestRegression R^2: %f'%score)
    plt.legend()        # 将样例显示出来
    plt.show()
    return result
##########3.绘制验证散点图########
def scatter_plot(TureValues,PredictValues):
    #设置参考的 1：1 虚线参数
    xxx = [-0.5,1500]
    yyy = [-0.5,1500]
    #绘图
    plt.figure()
    plt.plot(xxx , yyy , c='black' , linewidth=3 , linestyle=':' , ma
rker='.' , alpha=0.6)#绘制虚线
    plt.scatter(TureValues , PredictValues , s=20 , c='r' , edgecolor
s='k' , marker='o' , alpha=0.8)#绘制散点图，横轴是真实值，竖轴是预测值
    plt.xlim((0,800))   #设置坐标轴范围
    plt.ylim((0,800))
    plt.title('RandomForestRegressionScatterPlot')
    plt.show()
############4.预设回归方法##########
####随机森林回归####
from sklearn import ensemble
model_RandomForestRegressor = ensemble.RandomForestRegressor(n_estima
tors=800)
```

```
85. ########5.设置参数与执行部分##############
86. #设置数据参数部分
87. x_train , y_train , x_test , y_test = load_data(1,155,1,11,0.7, "D:\\
    learning\\y 机器学习\\随机森林.xlsx")   #行数以 excel 里为准
88. #起始行数 2，结束行数 121，训练集=测试集，特征数量 17,不打乱样本集
89. y_pred = regression_method(model_RandomForestRegressor)         #括号内
    填上方法，并获取预测值
90. scatter_plot(y_test,y_pred)  #生成散点图
91. data = pd.read_excel(r'D:\\learning\\y 机器学习\\成分空间预测.xlsx')
92. array = data.values
93. x = array[:, 0:11]
94. reg = model_RandomForestRegressor.predict(x)
95. print(pd.DataFrame(reg))
96. rfc = model_RandomForestRegressor
97. Xt, Xv, Yt, Yv = train_test_split(x_train,y_train,test_size=0.22, ran
    dom_state=420)
98. SUM = 0
99. for i in range(100):
100.       rfc_s = cross_val_score(rfc,Xv,Yv,cv=5)
101.       Mean = sum(rfc_s)/len(rfc_s)
102.       SUM = SUM+Mean
103.   print('五折交叉验证后', SUM/100)
```

3. 岭回归

```
1.  from sklearn.model_selection import train_test_split
2.  from sklearn.linear_model import Ridge, Lasso
3.  import pandas as pd
4.  import numpy as np
5.  import matplotlib.pyplot as plt
6.  data = pd.read_excel(r'C:\Users\y1532\PycharmProjects\untitled\data11
    \特征数据.xlsx')
7.  array = data.values
8.  X = array[:,:11]
9.  print(pd.DataFrame(X))
10. y = array[:, 12]
11. print(pd.DataFrame(y))
12. Xtrain, Xtest, Ytrain, Ytest = train_test_split(X,y,test_size=0.3, ra
    ndom_state=420)
13. reg = Ridge(alpha=1).fit(Xtrain,Ytrain)
14. print(reg.score(Xtest,Ytest))
15. Y_pred = reg.predict(Xtest)
16. r = len(Xtest) + 1
17. plt.plot(np.arange(1, r), Y_pred, 'ro-', label="predict value")
18. plt.plot(np.arange(1, r), Ytest, 'go-', label="real value")
```

```
plt.title('LinearRegression R^2: %f'%reg.score(Xtest,Ytest))
plt.legend()
plt.show()
```

4. LASSO 回归

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, Lasso
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_excel(r'C:\Users\y1532\PycharmProjects\untitled\data11\特征数据.xlsx')
array = data.values
X = array[:,:11]
print(pd.DataFrame(X))
y = array[:, 12]
print(pd.DataFrame(y))
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X,y,test_size=0.3, random_state=420)
reg = Ridge(alpha=1).fit(Xtrain,Ytrain)
reg2 = Lasso(alpha=0.0003, max_iter=10000).fit(Xtrain,Ytrain)  #  学习率过小时增加迭代次数
print(reg2.score(Xtest,Ytest))
Y_pred = reg2.predict(Xtest)
r = len(Xtest) + 1
plt.plot(np.arange(1, r), Y_pred, 'ro-', label="predict value")
plt.plot(np.arange(1, r), Ytest, 'go-', label="real value")
plt.title('LinearRegression R^2: %f'%reg2.score(Xtest,Ytest))
plt.legend()
plt.show()
```

5. SVR 回归

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import r2_score

data = pd.read_excel(r'C:\Users\y1532\PycharmProjects\untitled\data11\特征数据.xlsx')
array = data.values
x = array[:,:10]
y = array[:,11]
clf = SVR(kernel='linear', C=1.25)
x_tran, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4)
```

```
clf.fit(x_tran, y_train)
y_hat = clf.predict(x_test)
print("得分:", r2_score(y_test, y_hat))
```

6. 弹性回归

```
from sklearn import metrics
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
#导入数据集
data = pd.read_excel(r'C:\Users\y1532\PycharmProjects\untitled\data11\特征数据.xlsx')
array = data.values
x = array[:,:10]
y = array[:,11]
from sklearn.model_selection import train_test_split #导入数据划分包
#以 20%的数据构建测试样本，剩余作为训练样本
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state =10)
elastic= linear_model.ElasticNet(alpha=0.0001,l1_ratio=0.5)  # 设置 lambda 值,l1_ratio 值
elastic.fit(X_train,y_train)  #使用训练数据进行参数求解
y_hat2 = elastic.predict(X_test)  #对测试集的预测
score = r2_score(y_test, y_hat2)
print(score)
r = len(X_test) + 1
plt.plot(np.arange(1, r), y_hat2, 'ro-', label="predict value")
plt.plot(np.arange(1, r), y_test, 'go-', label="real value")
plt.title('ElasticNetRegression R^2: %f'%r2_score(y_test, y_hat2))
plt.legend()
plt.show()
```

7. 神经网络回归

```
'''''载入数据'''
import pandas as pd
Input = pd.DataFrame(pd.read_excel(r"C:\Users\y1532\PycharmProjects\untitled\data11\组合 1.xlsx"))
Input2 = pd.DataFrame(pd.read_excel(r"C:\Users\y1532\PycharmProjects\untitled\data11\硬度.xlsx"))
B = pd.concat([Input, Input2],axis =1)#组合特征列和硬度
x, y = B.data, B.target
'''''引入标准化函数'''
from sklearn import preprocessing
x_MinMax = preprocessing.MinMaxScaler()
y_MinMax = preprocessing.MinMaxScaler()
  ''''' 将 y 转换成 列 '''
```

```
import numpy as np
  y = np.array(y).reshape(len(y), 1)
'''''标准化'''
x = x_MinMax.fit_transform(x)
y = y_MinMax.fit_transform(y)
from sklearn.model_selection import train_test_split

np.random.seed(2019)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

'''''模型构建'''
from sklearn.neural_network import MLPRegressor
fit1 = MLPRegressor(hidden_layer_sizes=(100, 50),
                    activation='relu',
                    solver='adam',
                    alpha = 0.01,
                    max_iter = 200)
print("fitting model right now")
fit1.fit(x_train, y_train)
pred1_train = fit1.predict(x_train)
'''''计算训练集 MSE'''
from sklearn.metrics import mean_squared_error
mse_1 = mean_squared_error(pred1_train, y_train)
print("Train ERROR = ", mse_1)
'''''计算测试集 mse'''
pred1_test = fit1.predict(x_test)
mse_2 = mean_squared_error(pred1_test, y_test)
print("Test ERROR = ", mse_2)

'''''结果可视化'''
import matplotlib.pyplot as plt
xx = range(0, len(y_test))
plt.figure(figsize=(8, 6))
plt.scatter(xx, y_test, color="red", label="Sample Point", linewidth=3)
plt.plot(xx, pred1_test, color="orange", label="Fitting Line", linewidth=2)
plt.legend()
plt.show()
```