工业过程智能优化技术课程报告

题目 针对风电功率预测与优化的研究

姓　　名：　王愉杰

学　　号：　20211001578

班　　级：　231216

二〇二四年七月

摘　要

风能是一种重要的可再生能源，风电功率预测可以为电网调度、发电计划制定以及风电场运行维护提供科学依据，降低风电并网对电力系统的冲击，提高电力系统的安全性和经济性。同时，风电功率优化能够有效提高风能利用效率，最大化风电场的发电量，减少弃风现象，进一步推动风电产业的可持续发展。因此，研究风电功率预测和优化不仅具有重要的理论意义，还具有显著的实际应用价值。

本文结合工业过程智能优化技术课程的要求，系统地研究了风电功率预测与优化模型的建立。

第一部分是预测模型的建立，涵盖了数据预处理、相关性分析、回归分析等环节。在常见方法的基础上，本文引入了工况识别的内容，通过聚类算法得到三种工况，并分别对不同工况建立预测模型，在实际应用中，工况识别具有非常重要的作用。预测算法使用的是支持向量回归（SVR），采用交叉验证和网格搜索的方法训练，最后与最小二乘法和随机森林预测模型进行对比，发现 SVR 的预测效果更优。

第二部分是功率优化模型的建立，这是本文的第二大核心。前期预测模型的建立和工况识别为优化打下了基础。在得到预测效果良好的模型后，本文采用了模拟退火算法，以功率最大化为目标，使用罚函数法考虑风速、风向和温度的不等式约束条件。分别在三种工况对应的风速下，寻找最优风向和温度组合，以最大化风电功率输出。

通过本次实验，不仅加深了对预测与优化方法的理解，也为提升风电场运行效率和经济效益提供了一些理论研究，贡献了自己的力量。

关键词：风电功率，支持向量回归，模拟退火，工况识别，罚函数

目录

一、绪论

1.1 研究背景

随着全球能源需求的不断增长和环境问题的日益突出，清洁能源的开发和利用已经成为全球关注的焦点。风能作为一种重要的可再生能源，具有储量丰富、环境友好等优点，逐渐在全球范围内得到广泛应用。然而，风能发电具有明显的随机性和波动性，风速的变化直接影响到风电场的功率输出，这对电网的稳定运行提出了严峻挑战。因此，准确的风电功率预测和优化显得尤为重要。

1.2 研究意义

风电功率预测可以为电网调度、发电计划制定以及风电场运行维护提供科学依据，降低风电并网对电力系统的冲击，提高电力系统的安全性和经济性。同时，风电功率优化能够有效提高风能利用效率，最大化风电场的发电量，减少弃风现象，进一步推动风电产业的可持续发展。因此，研究风电功率预测和优化不仅具有重要的理论意义，还具有显著的实际应用价值。

1.3 研究现状

目前，风电功率预测方法主要分为物理模型法、统计模型法和混合模型法。物理模型法基于风能转换的物理过程，适用于短期预测；统计模型法通过对历史数据的分析，利用时间序列模型、神经网络等进行预测；混合模型法则结合了物理模型和统计模型的优点，具有更高的预测精度和稳定性。然而，现有方法仍然存在预测精度不足、计算复杂度高等问题，需要进一步优化和改进。

调研发现，已有大量研究采用了多种算法，包括线性回归、时间序列分析、人工神经网络、决策树等。然而，这些方法在处理复杂非线性关系和高维数据时，往往存在一定的局限性。相比之下，支持向量回归算法以其良好的泛化能力和对高维数据的处理能力，成为一种有效的预测工具。

在风电功率优化方面，常见的方法包括风电场布局优化、风电机组运行优化和风电场经济调度等。风电场布局优化主要通过合理布局风电机组，提高风能利用效率；风电机组运行优化则侧重于根据风速变化调整风电机组的运行状态，包括风向、温度等因素，以实现最大功率输出。常用的优化方法包括粒子

群算法、遗传算法、禁忌搜索算法等。

1.4 研究内容和方法

在众多预测算法中，支持向量回归算法以其良好的泛化能力和对高维数据的处理能力，成为一种有效的预测工具。而在优化算法方面，模拟退火算法以其模拟物理退火过程的全局优化能力，在解决复杂优化问题时表现出色，特别是在风电功率优化方面，具有显著的应用价值。

本文旨在利用支持向量回归算法（Support Vector Regression, SVR）对风电功率进行预测，并通过模拟退火算法（Simulated Annealing, SA）对功率进行优化，找到在指定风速条件下最优的风向和温度设置。通过这种方法，不仅可以提高风电功率预测的准确性，还能在实际应用中最大化风力发电的功率输出，从而提升风能的利用效率。

1.5 论文结构

本论文将分为以下几个章节：

(1) 绪论：介绍研究背景、研究目的、研究现状以及本文的研究方法。

(2) 数据预处理：对获得的原始数据集进行可视化、统计分析、数据去噪、数据变化等内容。

(3) 相关性分析：采用皮尔逊相关系数来分析变量之间的相关性，根据分析结果确定预测模型的预测变量和响应变量。

(4) 工况识别：通过 K-means 聚类方法来实现风电场的工况识别，在不同工况的基础上分别建立预测模型。

(5) 风电功率预测模型：在 Kmeans 聚类分析的基础上，在训练集中的三个簇内各自训练回归模型。在测试集中，我们先根据欧式距离将数据点划分到对应的簇中，然后根据训练集中各个簇建立的回归模型，预测训练集中该簇的结果，并对模型的性能进行评价。

(6) 风电功率优化：通过模拟退火算法对功率进行优化，找到在指定风速条件下最优的风向和温度设置。

(7) 结论与展望：总结研究成果，提出进一步研究的方向。

通过上述研究，本文希望能够为风电功率预测与优化提供新的思路和方法，为提升风力发电的利用效率和经济效益贡献力量。

二、数据预处理

2.1 数据特征分析

原始数据集中包含六个特征，风速、风向、温度、湿度、气压以及风电功率。本节对风电场的五种天气指标进行特征分析，包括正态性检验、数据趋势分析、数据分布以及统计分析。使用到了直方图、Q-Q 图、散点图、玫瑰图、箱线图以及常见的统计量如均值、极差和标准差，流程图如下图所示。

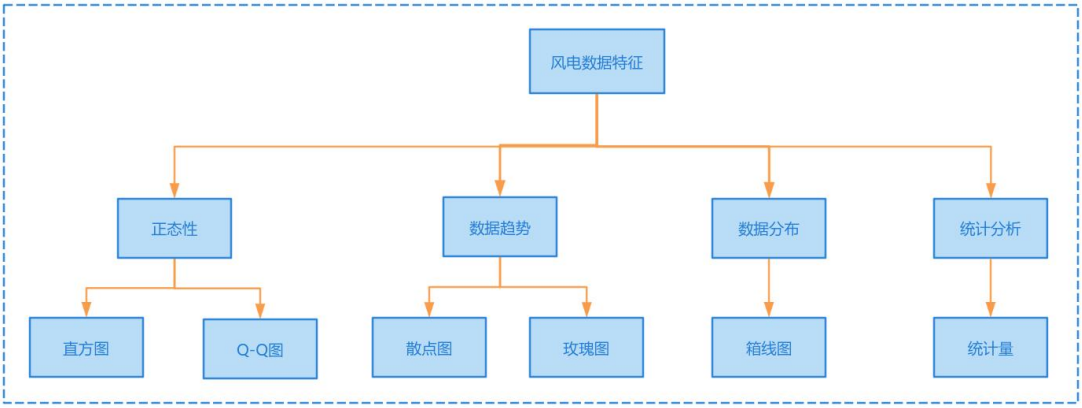

图 1 数据特征分析流程图

2.1.1 正态性检验

本文最终需要对风电功率进行优化，在此之前需要建立天气变量对功率的预测模型，很多机器学习预测算法要求特征的数据服从正态分布，因此在预处理环节需要进行数据正态性检验。

本小节采用直方图和 Q-Q 图两种方式进行综合分析。

直方图通过将数据分组成连续的区间并展示每个区间的频数或频率，可以直观看到数据的分布形状。对于大型数据集，直方图提供了一种快速的方法来初步判断数据是否接近正态分布。Q-Q 图将样本数据的分位数与正态分布的理论分位数进行比较，如果数据点在 Q-Q 图上接近于一条直线，则数据接近于正态分布。

直方图和 Q-Q 图相结合，直方图通过展示数据的频率分布并与理论正态分布曲线进行比较，初步判断数据的正态性。QQ 图则通过比较样本分位数和理论分位数，提供更精确的正态性检验。综合使用这两种方法，可以更全面地理解

数据的分布特征。5 个变量数据的直方图和 Q-Q 图分别如图 2 和图 3 所示。

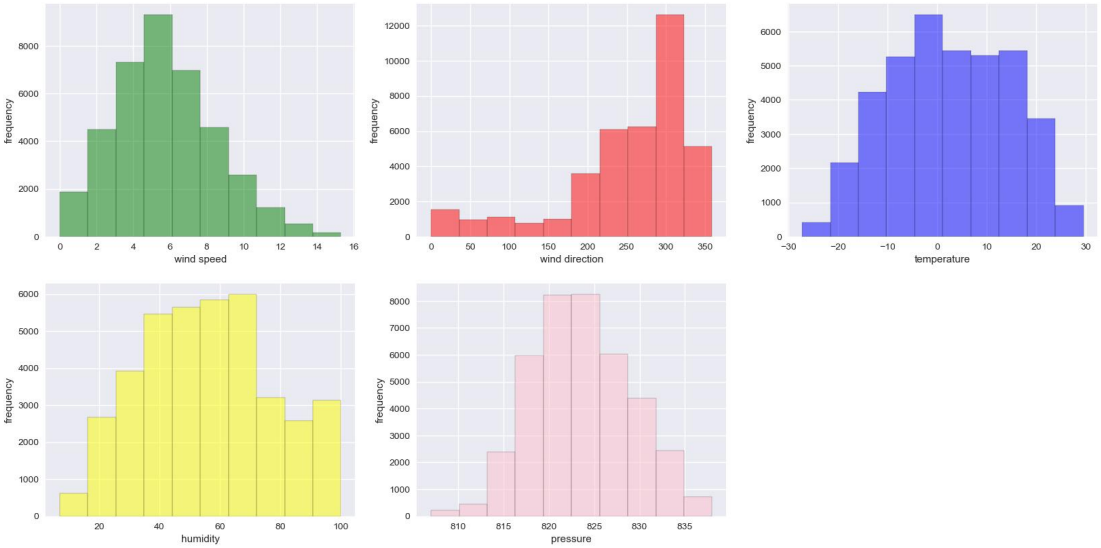

图 2 五个天气变量的直方图

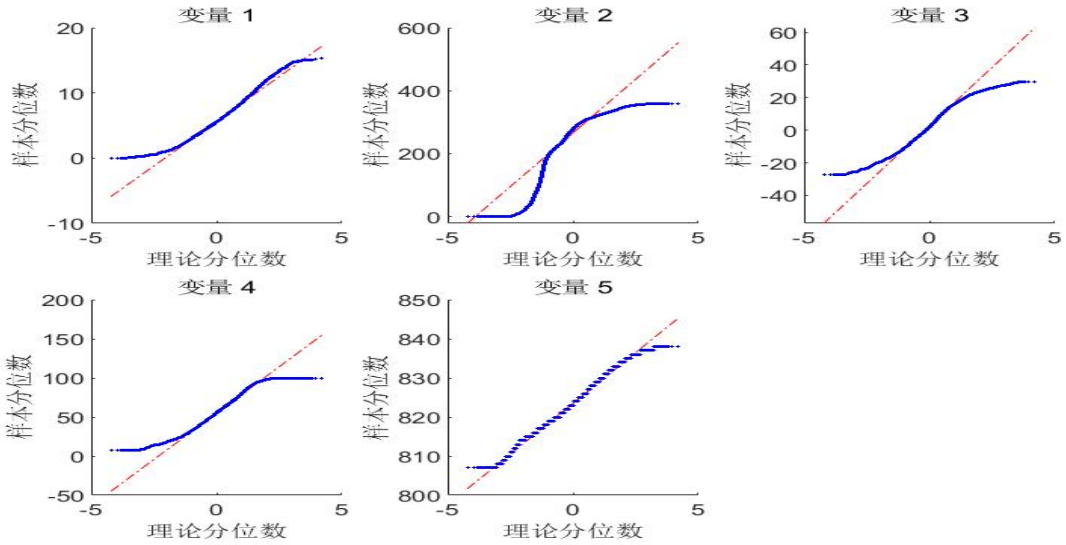

图 3 五个天气变量的 Q-Q 图

综合观察上图，可以发现直方图中 5 个天气变量基本呈现中间向两边递减的趋势，Q-Q 图中 5 个变量近似呈现出一条直线。结合两种图形可近似将天气数据看做正态分布。

2.1.2 散点图

散点图可以展现出两个变量直接的关系和变化趋势，例如风速功率散点图，可直观地表现出风速与功率之间的分布情况和功率随风速变化的大致趋势。5 个

天气变量与功率的散点图如图 4 所示。

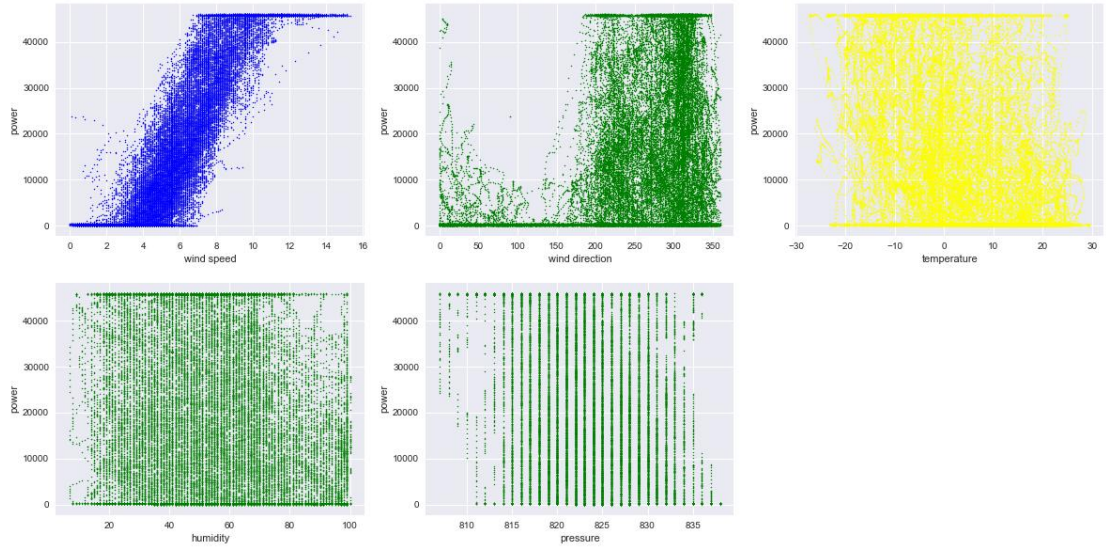

图 4 天气变量与功率的散点图

2.1.3 风向玫瑰图

需要注意的是风向变量的量纲为角度，直方图和散点图并不能形象地展现出不同风向的情况，此处使用玫瑰图，可直观地表现出风场风向的分布情况。

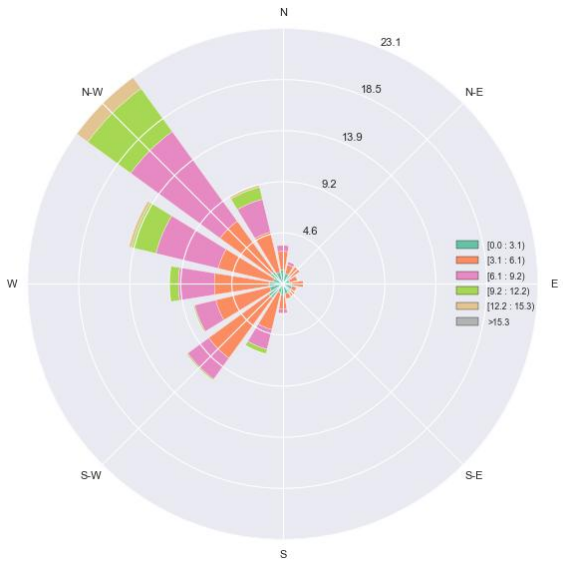

图 5 风速-风向玫瑰图

风向玫瑰图如上图所示，包括风速和风向两个变量，360° 的圆环代表风向，其中不同颜色的色块则表示不同范围的风速，生动形象地展现出不同风向上的风速分布情况。

2.1.4 箱线图

在数据统计分析环节进行箱线图分析是非常重要的，因为箱线图能够提供数据集的直观概览，帮助我们快速识别和理解数据的特征及其分布情况。具体而言，箱线图分析具有以下几个优势：

(1) **检测离群点**：箱线图能够有效地显示数据中的异常值或离群点，这些值可能需要进一步处理以避免影响后续的分析和建模。

(2) **理解数据分布**：通过箱线图可以清晰地看到数据的分布情况，包括中位数、四分位数范围（IQR），以及数据的偏态和变异程度。

(3) **比较数据集**：箱线图可以用于比较不同数据集或同一数据集中不同子集的分布情况，帮助我们发现不同组之间的差异和趋势。

(4) **简化数据呈现**：箱线图以简单的图形方式概括了数据的五个统计量（最小值、第一四分位数、中位数、第三四分位数和最大值），使得数据的概览更加简洁明了。

5 个天气变量的箱线图如图 6 所示：

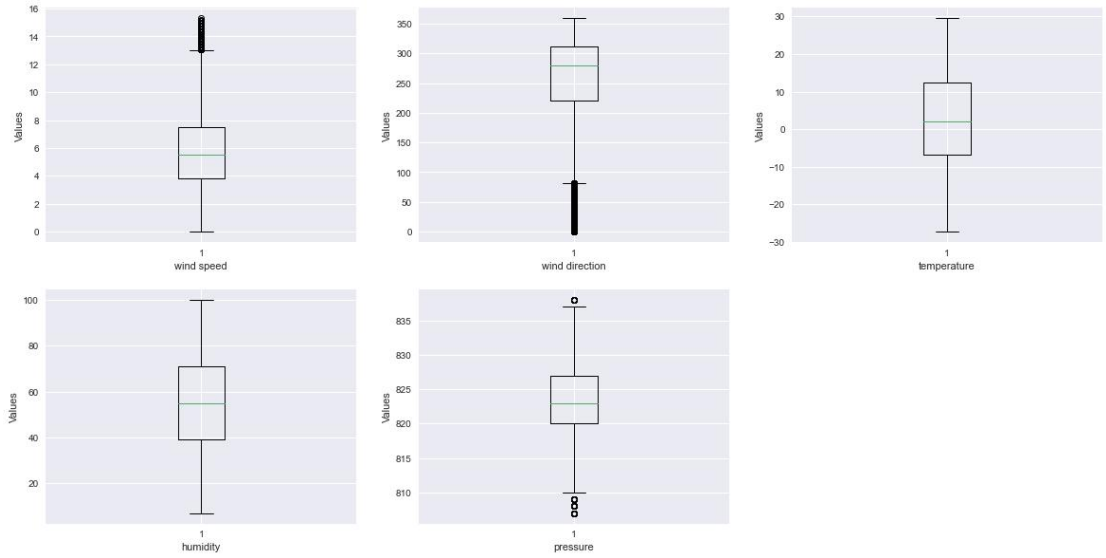

图 6 天气变量箱线图

2.1.5 统计特征

对数据进行可视化分析后，还需要进行定量的统计分析，包括均值、极差和标准差，三个统计量的含义如下。

(1) 均值：均值是数据集之和，代表总体的平均水平。均值在集中数据中为最具代表的统计特征量。

(2) 极差：极差是数据集最大值与最小值之差，说明数据的伸展情况。通常情况下，极差大的一组数据要比极差小的一组数据更为分散。

(3) 标准差：标准差是数据集方差的算术平方根，衡量数据波动性（离散程度）的指标。标准差越大，数值可能变动的程度就越大，稳定度就越小。

经过计算，六个变量的统计量值如下表所示。

表 1 六个变量的统计值

	均值	极差	标准差
功率	16706.772	45994.980	16123.458
风速	5.760	15.300	2.728
风向	252.367	359.000	82.988
温度	2.444	56.800	12.098
湿度	55.973	93.000	21.699
气压	823.610	31.000	5.270

2.2 数据清洗

本节首先对原始数据集进行数据清洗，查找缺失值和重复值。再对数据进行归一化以及数据划分。对划分后的训练集进行 DBSCAN 去噪，并对去噪前后的数据进行对比分析，为后续的相关性分析和模型建立做好准备。

2.2.1 异常数据处理

原始数据集为时间序列数据集，数据较多，可能存在重复时间节点的数据以及缺失数据，因此首先用程序遍历一遍数据集。结果显示不存在缺失值，但存在重复时间的数据，本文只保留该日期第一次采集的数据，删除其余数据。经过数据清洗，39439 条数据缩减为 39072 条。

2.2.2 数据归一化

处理缺失值和重复值后，需要去除噪声，在应用 DBSCAN 算法进行聚类和去噪之前，通常需要对数据进行归一化。这是因为 DBSCAN 依赖于距离计算，而不同特征的尺度可能会显著影响距离度量，从而影响聚类结果。

考虑到实验中的变量基本服从正态分布（2.3 小节），本文选择使用最常用的 Z 分数归一化，Z 分数归一化是通过将数据转换为均值为 0、标准差为 1 的标准正态分布，公式为：

$$z = \frac{x - \mu}{\sigma}$$

其中，z 表示原始数据，μ 表示数据的均值，σ 表示数据的标准差。z 分数归一化能保留数据的分布形态，不会改变数据的相对关系，适用于数据服从正态分布或近似正态分布的场景。

2.2.3 数据划分

后续训练模型时需要用到训练集，然后用测试集对模型性能进行评价。考虑到实际情形中采集的数据会包括一些噪声，因此接下来的 DBSCAN 去噪只用于训练集以提高模型精度，测试集则保持原本分布，用于真实评价模型。

本文以 8:2 的比例将原数据集划分为训练集和测试集，分别有 31258 条和 7814 条数据。

2.3 DBSCAN 去噪

2.3.1 DBSCAN 算法原理

DBSCAN（Density-Based Spatial Clustering of Applications with Noise）是一种基于密度的聚类算法，在数据挖掘和图像处理等领域广泛应用。除了聚类之外， DBSCAN 还可以用于数据去噪。

DBSCAN 聚类的原理是通过评估样本周围的密度以及被视为相邻的点之间的距离来找出高密度区域（即簇）。每个高密度区域可以视为一个簇，而每个非高密度区域的点可以被视为噪声点。因此，通过聚类操作可以将非噪声点分为不同的簇，并将噪声点识别出来并从数据集中删除，从而实现数据去噪的目的。在

DBSCAN 中，通过以下两个参数来识别密度区域：

(1) eps：指定邻域的距离范围，如果两个点之间的距离小于等于 eps，则将它们视为相邻的点。
(2) min_samples：指定邻域中点的数量，如果一个点周围的相邻点数量超过 min_samples，则它被视为核心点，否则它被视为非核心点。

具体的数据去噪步骤如下：

(1) 将数据集加载到 DBSCAN 算法中，并选择合适的参数。
(2) 根据 DBSCAN 的聚类结果，识别出非噪声点（即属于某个簇的点）和噪声点（即不属于任何簇的点）。

(3) 将非噪声点保留，而过滤掉噪声点。

通过初步的调研，我们发现功率与风速有非常大的关系，因此此处以风速和功率两个特征作为去噪算法的输入。接下来可视化地展示 DBSCAN 的聚类结果，从图 7 可以看出，散点图中的离群点均被 DBSCAN 算法提取了出来，被标记为红色，且去噪后的数据分布更加集中。

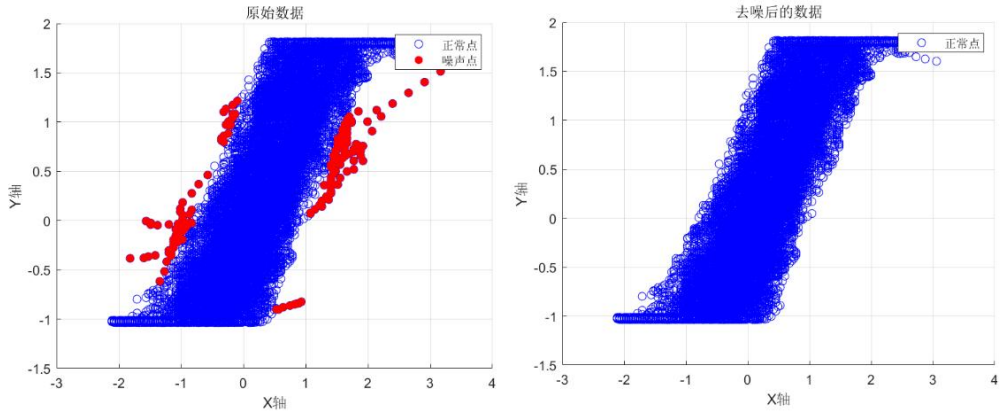

图 7 去噪前后数据可视化对比

2.3.2 去噪前后对比

经过 DBSCAN 去噪，训练集由 31258 缩减为 31059 条，接下来通过统计分析对训练集的 5 个天气变量去噪前后的数据进行对比，包括直方图、散点图等。

(1) 直方图

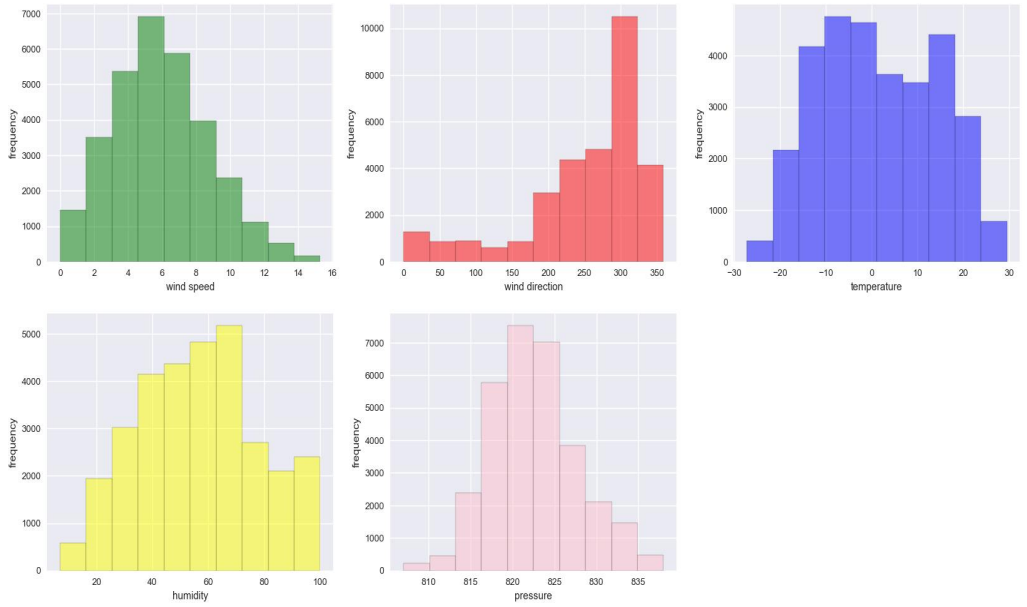

图 8 去噪前训练集天气变量直方图

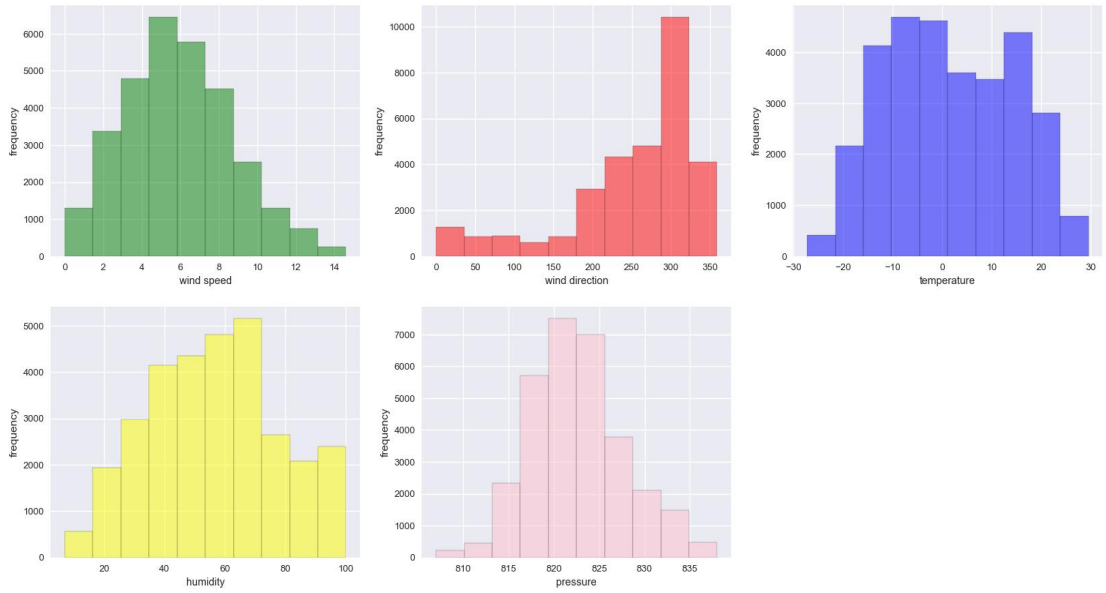

图 9 去噪后训练集天气变量直方图

对比分析图 8 和图 9 可以发现风速变量在去噪后边缘值（超出 14）减少，整体数量减少，整体数据分布更加集中，其余四个天气变量则无明显变化。这是因为 2.3.1 小节的去噪环节是以风速和功率为特征的，因此在风速的分布上会有较明显的效果。

(2) 散点图

对去噪前后的风速-功率散点图进行更加精细的绘制，如图 12 所示，左侧为去噪前的，右侧为去噪后的。

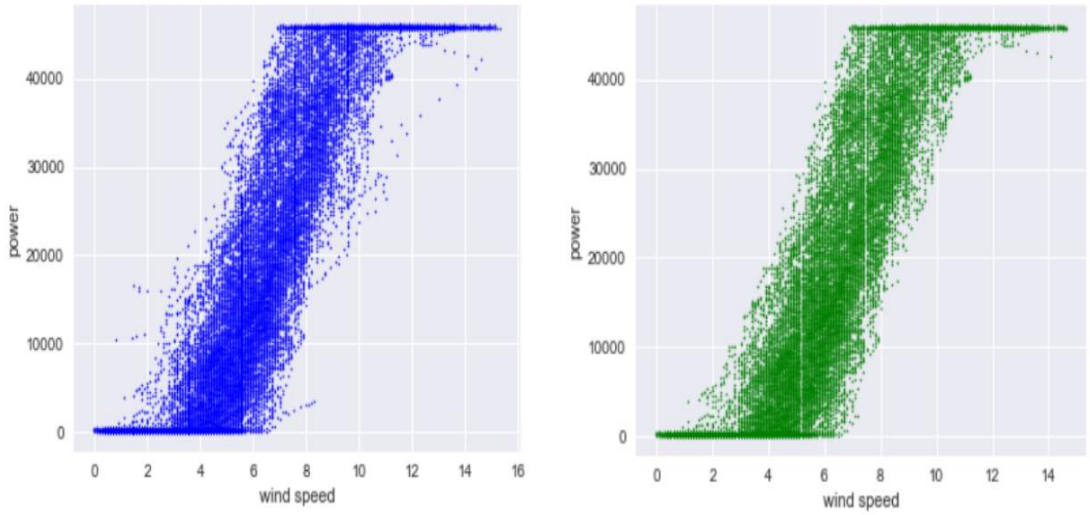

图 10 去噪前后的风速-功率散点图

(3) 箱线图

箱线图能够有效地显示数据中的异常值或离群点，而且通过箱线图可以清晰地看到数据的分布情况，绘制去噪前后的风速箱线图如下所示。

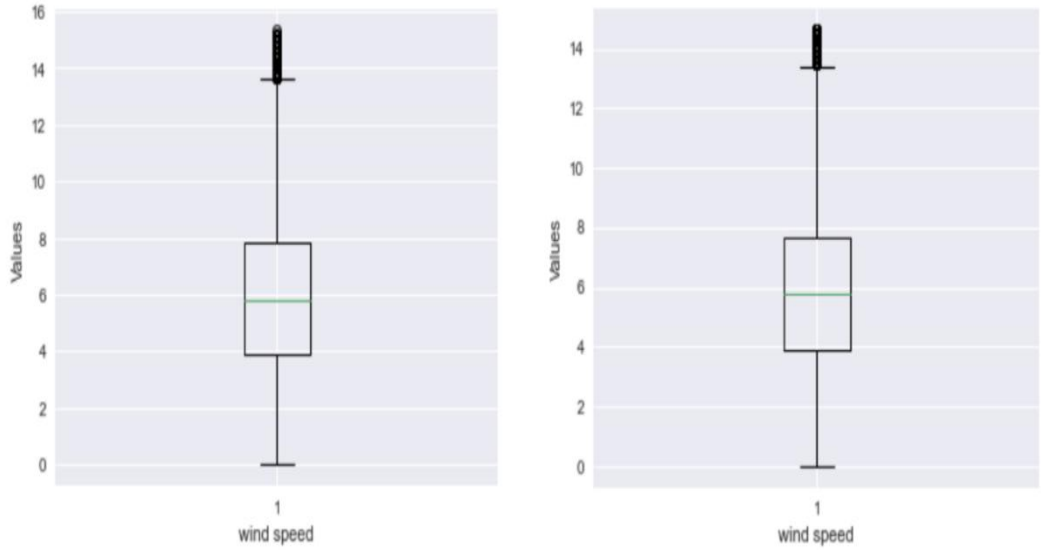

图 11 去噪前后的风速箱线图

通过上图可以看出风速指标的离群点明显减少，上限降低。

(4) 统计值

除了图 10~13 的去噪前后可视化对比，我们还在平均数、极差、标准差等统计意义上做了对比，结果如表 2。

表 2 风速变量去噪前后统计值对比

	去噪前	去噪后
均值	5.946	5.926
极差	15.300	14.600
标准差	2.807	2.783

从表 2 中可以发现去噪后的数据的极差和标准差均小于去噪前的数据，说明去噪后的数据少了极端点、离群点，数据分布更加均衡与集中。

三、相关性分析

实验中共有 5 个天气变量和 1 个功率变量，本小节采用皮尔逊相关系数来分析变量之间的相关性，根据分析结果确定预测模型的预测变量和响应变量。

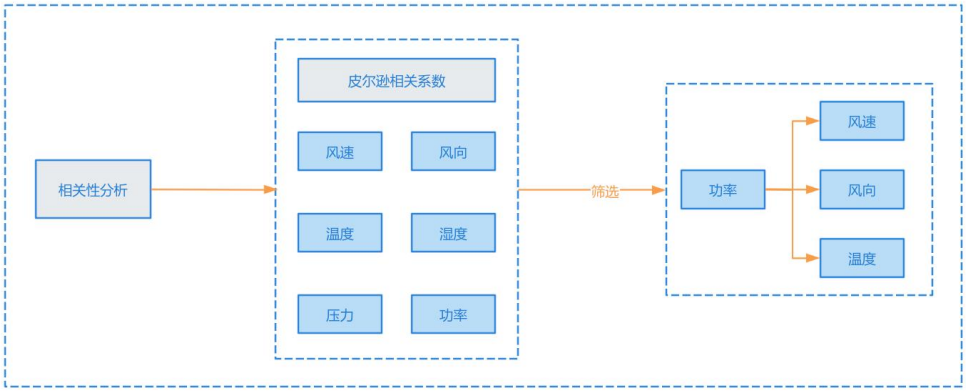

图 12 相关性分析流程图

皮尔逊相关系数（Pearson Correlation Coefficient）是用来衡量两个变量之间线性关系的强度和方向的统计指标，其值介于-1 和 1 之间。对于风电场景有以下适用性和优点：

(1) 线性关系的测量：皮尔逊相关系数专注于测量变量之间的线性关系，对于风电过程中常见的一些线性相关特性非常有效。

(2) 数据要求：皮尔逊相关系数对数据的要求不高，只需要成对的样本数据。此外，它假设数据服从正态分布，虽然不是严格要求，但满足该假设时结果更可靠。

(3) 处理多变量：风电过程中涉及多个变量，皮尔逊相关系数可以方便地计算这些变量之间的两两相关性，从而识别出关键影响因素。

皮尔逊相关系数的计算公式如下：

$$r=\frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2\sum(y_i-\bar{y})^2}}$$

其中，x_i 和 y_i 表示两个变量的样本值，$\bar{x}$ 和 $\bar{y}$ 表示两个变量的样本均值。相关性强弱的划分如下表：

表 3 相关性强弱划分

相关性系数	0-0.2	0.2-0.4	0.4-0.6	0.6-0.8	0.8-1.0
相关性强弱	极弱	弱	中等	强	极强

经过 Python 计算，6 个指标相互之间的相关性系数热力图如图 13 所示。

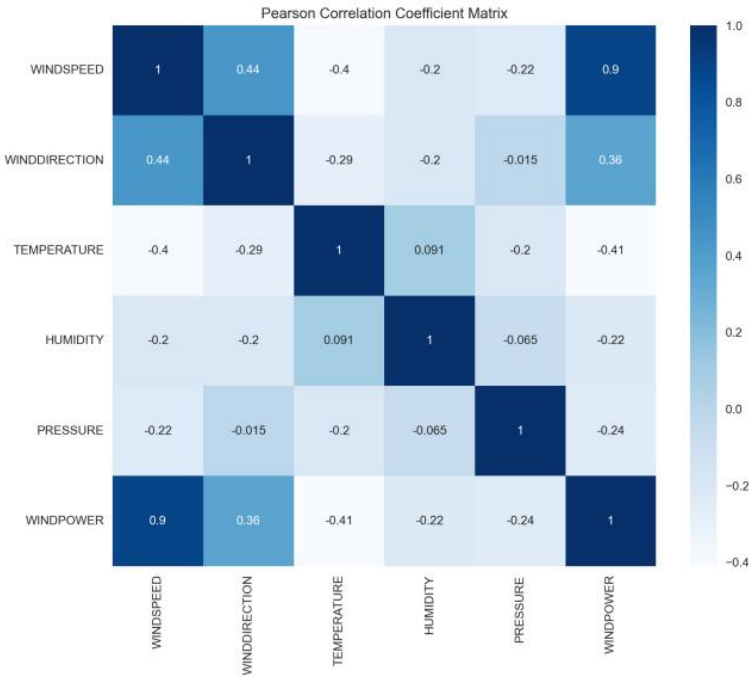

图 13 相关系数热力图

选取与风电功率相关性系数的绝对值大于 0.3 的特征，包括风速、风向、温度，得到筛选后的相关性矩阵热力图，如图 14 所示。

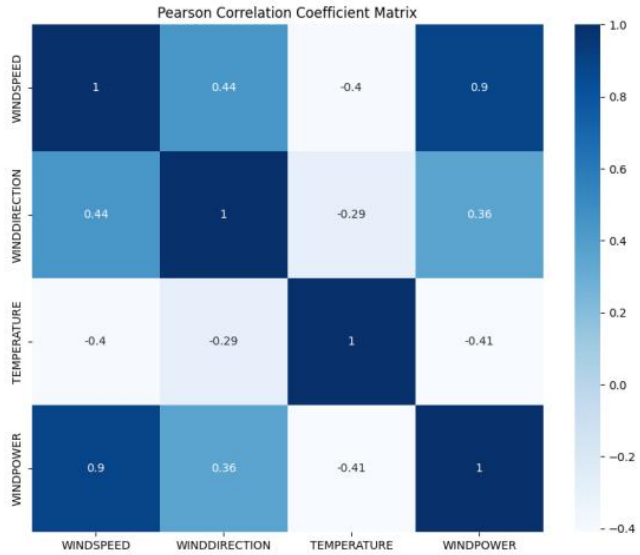

图 14 筛选后的热力图

四、工况识别

一般情况下在进行数据预处理和相关性分析后就可建立预测模型，但是很多时候这样的预测模型的性能并不好，这是因为在工业或者风电场景中存在多种工况，正常工况和异常工况下预测效果会有明显差异。

风电功率的输出受到多种因素的影响，包括风速、风向、气温和风机状态等。工况识别能够帮助我们准确地辨别和区分这些不同的影响因素，从而提高预测模型的精度。通过识别风机的运行工况，例如正常运行、维护停机或故障停机等，预测模型可以针对不同的工况采取相应的预测策略，避免因工况变化带来的预测误差。因此，工况识别在风电功率预测中起着至关重要的作用，是实现精准预测和高效管理的基础。本小节通过 K-means 聚类方法来实现工况识别。

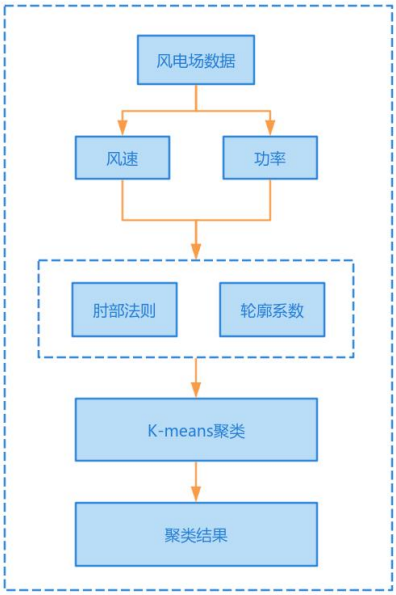

图 15 工况识别流程图

4.1 K-means 算法

K-means 是一种无监督机器学习算法，其目标是将给定的数据集分成 k 个簇，使得每个数据点都属于与其最近的簇。K-means 算法的一个重要的超参数就是簇的个数 k，也就是我们需要在算法中指定的簇的数量。由于数据集的不同和

应用的不同，确定恰当的 k 值十分关键。

以下是两种常见的为 K-means 算法选择簇数的方法：

(1) 手肘法（Elbow Method）：这是一种简单而常见的方法，它通过计算不同的 k 值下聚类模型的损失函数（通常是 sse）来找到最佳聚类数。然后，绘制 k 与该损失函数的图形，找到图形中最明显的拐点，该点对应的 k 值就是最佳的聚类数。

(2) 轮廓系数法（Silhouette Coefficient）：该方法通过计算每个样本对应的轮廓系数来评估聚类的效果，可以获得比简单的 SSE 方法更丰富的信息。轮廓系数是分配给每个观测值的度量，描述每个观测值在其所在簇中被“紧密”包围的程度。该系数越接近 1，表示在所在簇内部紧密程度高，簇分配得当，越接近-1，表示分配不当。

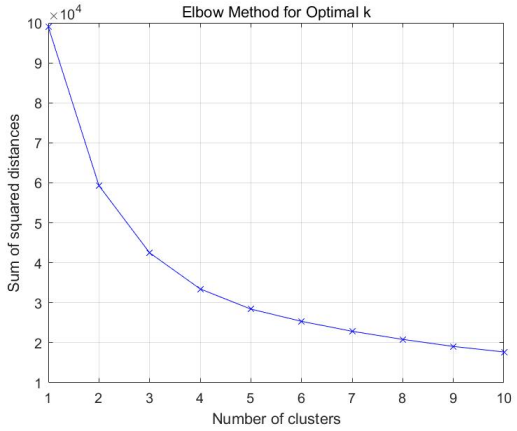

图 16 肘部法

图 16 中，曲线的“肘部”位于 k=3 或 k=4 的位置。

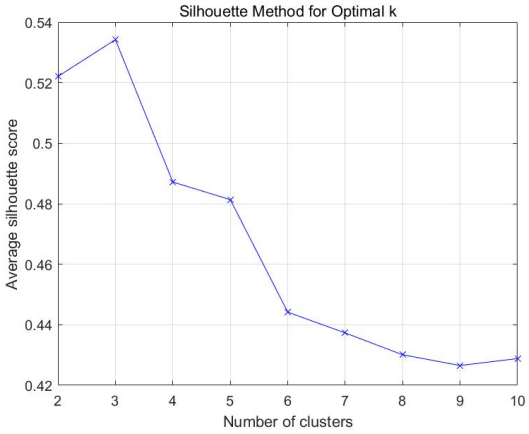

图 17 轮廓系数法

图 17 显示聚类个数为 3 时，轮廓系数最大，即 k=3 最佳。综合两种方法我们最终选择的聚类个数为 3。

4.2 聚类结果分析

图 18~20 展示了风速、风向、温度的聚类散点图，三个簇的数据分别用不同的颜色予以标注。

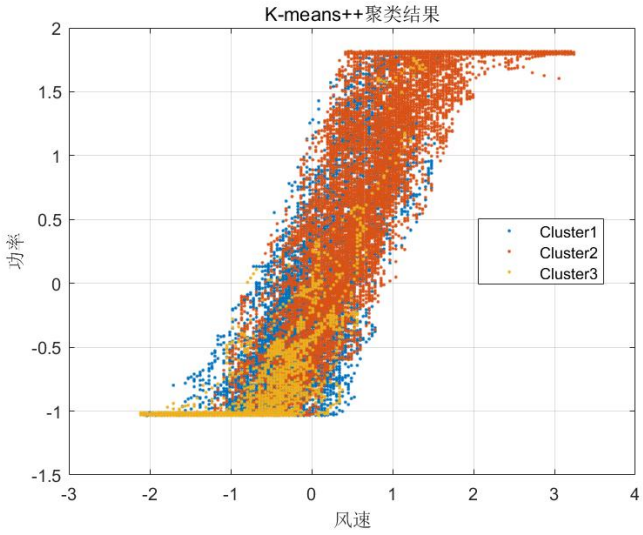

图 18 风速聚类散点图

从图 18 中可以看出黄色簇代表低风速，红色簇代表高风速，蓝色簇则既有高风速也有低风速，需要结合其它变量散点图观察其代表的类型。

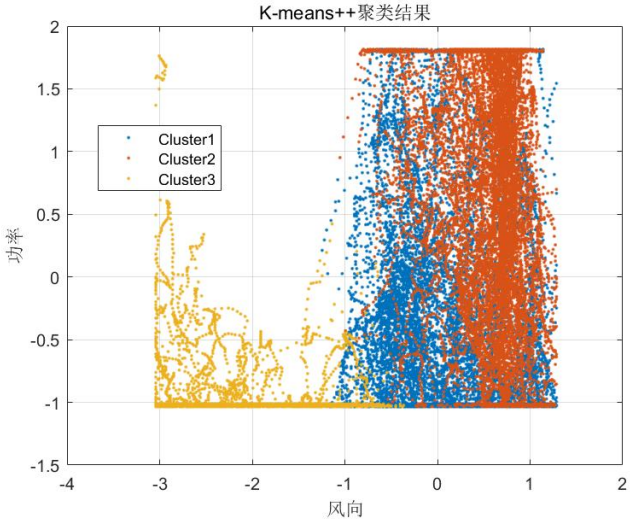

图 19 风向聚类散点图

图 19 显示三种类型的簇分别代表一个范围内的风向。

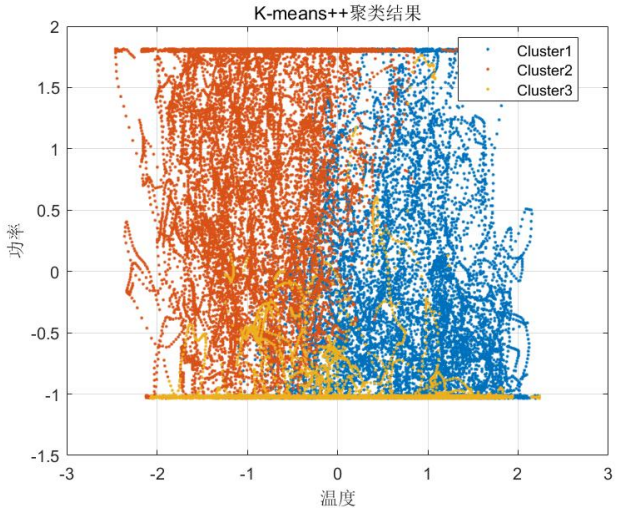

图 20 温度聚类散点图

图 20 显示红色簇代表低温，蓝色簇代表高温，黄色簇有交叉。结合 20~22 三张图我们可以大致总结出三类簇各自代表的工况，具体如下。

(1) 簇 1：中低风速，中间范围风向，高温

(2) 簇 2：高风速，右部分风向，低温

(3) 簇 3：低风速，左部分风向

图 21~23 展示了风速、风向、温度各自与功率的频率分布图，三个簇的数据分别用不同的颜色予以标注。

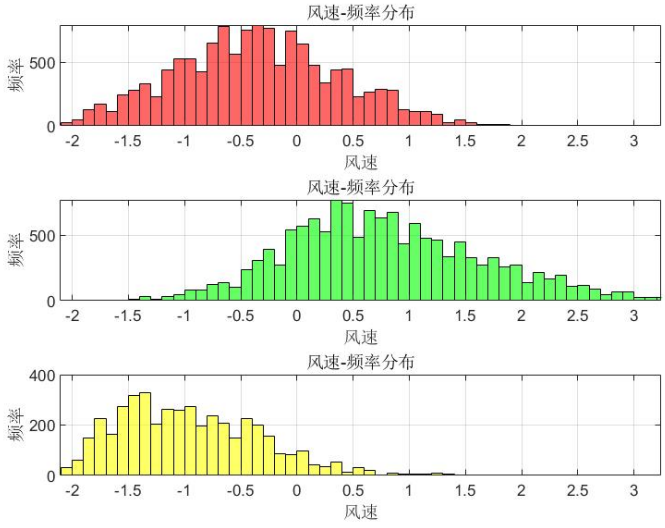

图 21 风速-频率分布图

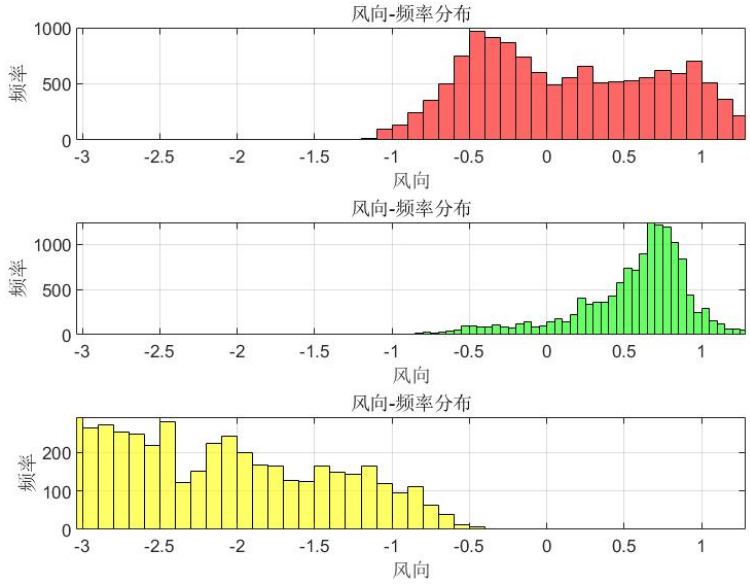

图 22 风向-频率分布图

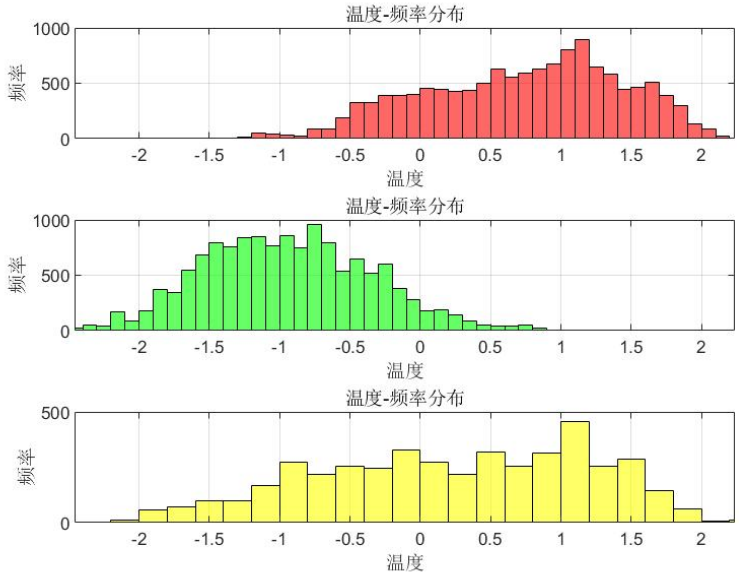

图 23 温度-频率分布图

风速、风向、温度各自与功率的频率分布图进一步验证了不问总结出的三类簇各自代表的工况。

五、风电功率预测模型

本节将在 Kmeans 聚类分析的基础上，在训练集中的三个簇内各自训练回归模型。在测试集中，我们先根据欧式距离将数据点划分到对应的簇中，然后根据训练集中各个簇建立的回归模型，预测训练集中该簇的结果，并对模型的性能进行评价，流程图如下。

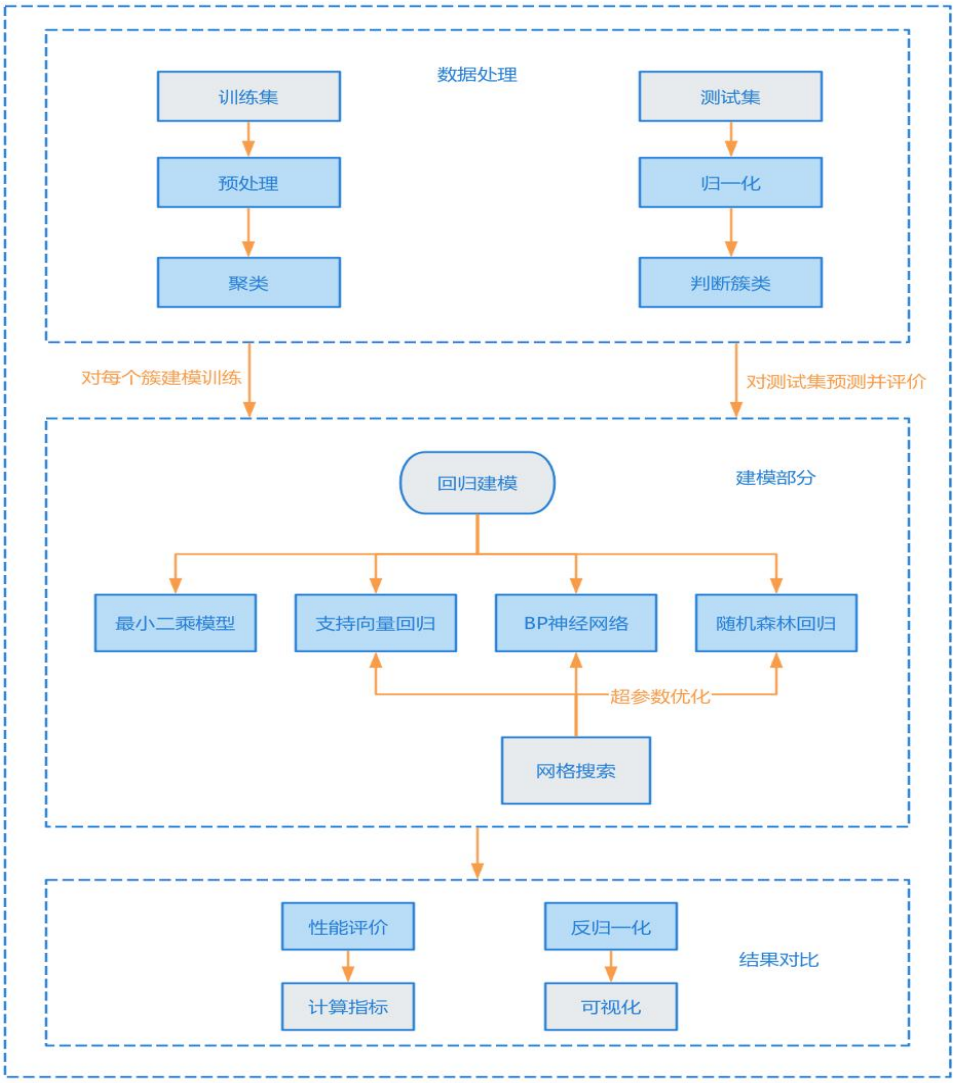

图 24 风电功率预测模型建立的流程图

5.1 支持向量回归

支持向量回归（Support Vector Regression）是一种基于支持向量机的回归方法，使用核函数来处理非线性回归问题。它通过映射数据到高维空间，在高维空间中寻找最佳拟合曲线。下面是对 SVR 的一些关键概念和实现步骤的介绍。

(1) 核函数

核函数是支持向量机中常用的核函数，定义如下：

$$K\left(x_i, x_j\right)=\exp(-\gamma\left\|x_i-x_j\right\|^2)$$

SVR 的目标函数由两个部分组成：损失函数和正则化项。目标是找到最佳的 w 和 b 使得回归误差最小化，同时保持模型的复杂度尽量小。

$$\min \frac{1}{2}\|w\|^2+C\sum\left(\xi_i+\xi_i^*\right)$$

其中，w 是模型的权重向量。b 是偏置项。ξ_i 和 ξ_i^* 是松弛变量，表示允许的误差。C 是惩罚参数，控制模型复杂度和误差的权衡。

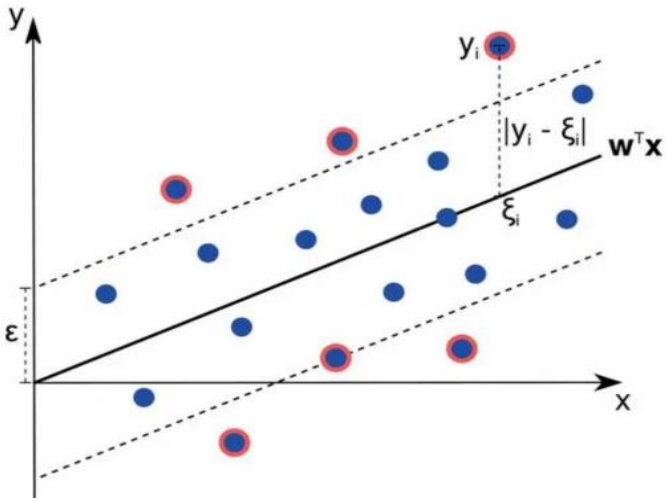

图 25 SVR 示意图

在化学反应过程中，预测成分变量是一个重要的任务。SVR 在这一场景中的应用契合度高，主要体现在以下几个方面：

(1) **非线性关系处理能力**：化学反应过程中的成分变量往往具有复杂的非线性关系。SVR 特别适合这种场景，因为它可以通过核方法（如 RBF 核）将数据映射到高维特征空间，在该空间中处理原本非线性的关系，从而实现更准确的预测。

(2) **鲁棒性**：SVR 对异常值具有一定的鲁棒性，不敏感损失函数允许在一定范围内忽略误差，不会因为少数异常值而对模型造成重大影响。这在实际化学反应中非常重要，因为实验数据中常常存在一些异常值。

(3) **高维数据处理**：化学反应过程中可能涉及多个变量，导致数据维度较高。SVR 在处理高维数据方面表现优异，因为它的优化目标和约束条件只依赖于支持向量的数量，而不是特征的数量，这使得 SVR 在高维空间中仍能高效地进行计算。

(4) **灵活性**：SVR 可以通过选择不同的核函数适应不同的数据分布和关系模型，具有很高的灵活性。

训练中，本文选择成分变量 7 作为响应变量，以相关性分析选择的 3 个预测变量建立风电功率预测模型，在原始数据集中选取 80%作为训练集。训练中使用的超参数优化方法是网格搜索结合十折交叉验证。

(1) 十折交叉验证

交叉验证是一种评估模型性能的技术，通过将数据集划分为多个子集，分别用于训练和验证，以获得模型在不同数据集上的性能表现。十折交叉验证的步骤如下：

① 将数据集划分为 10 个互斥的子集。

② 每次使用其中 9 个子集进行训练，剩下的 1 个子集进行验证。重复这一过程 10 次，每个子集都被用作一次验证集。

③ 计算每次验证的 R 方值，最后取 10 次验证的平均 R 方值作为该超参数组合的性能指标。

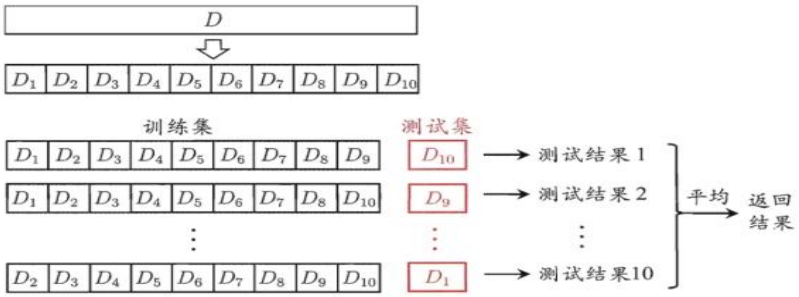

图 26 十折交叉验证示意图

(2) 网格搜索

网格搜索是一种系统地遍历预定义的超参数集合的方法，以选择出使模型

表现最优的超参数组合。对于每个超参数组合，都会训练模型并评估其性能，最终选择最佳的超参数组合。网格搜索的步骤如下：

① 定义超参数范围，包括 C 的取值范围、ε 的取值范围和核尺度（Kernel Scale）的取值范围。

② 遍历所有可能的超参数组合。

③ 对每个超参数组合，进行十折交叉验证，计算模型的平均 R 方指标。

网格搜索得出的超参数为：

表 4 SVR 超参数

超参数	取值
C	50
gamma	0.1
kernel	rbf
epsilon	0.2

测试集的各评价指标值如下表所示。

表 5 测试集的各评价指标值

评价指标	测试集
R 方	0.8054
RMSE	0.4030
MAE	0.2926

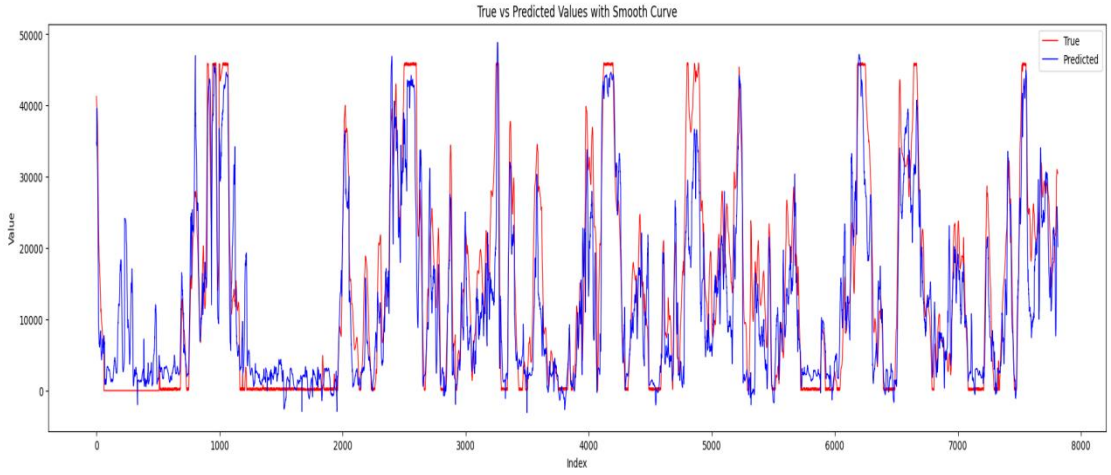

图 27 SVR 预测结果与真实值对比图

从表 5 和图 27 中可以看出 SVR 的预测效果不过，但是存在平稳期的预测效

果不如振荡期的问题。

5.2 最小二乘法

最小二乘预测模型（Least Squares Prediction Model）是一种常用的统计方法，主要用于拟合数据并进行预测。其基本思想是通过最小化预测值与实际值之间的误差平方和来确定模型参数。其目的是通过一条直线（回归线）来描述一个自变量 X 和一个因变量 Y 之间的线性关系。回归线的方程通常表示为：

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

其中，Y 和 X 分别为因变量和自变量，β_0 表示截距，β_1 表示斜率，ε 表示误差项。

建立一元线性回归模型的步骤如下：

(1) 划分训练集和测试集

将预处理后的数据以 7:3 划分为训练集和测试集，训练集用于拟合模型，测试机用于模型评估。

(2) 拟合模型

拟合一元线性回归模型主要是估计回归系数 β_0 和 β_1，常用的方法是最小二乘法，目的是找到使得所有观测值的残差平方和最小的回归系数。

公式如下：

$$\beta_1 = \frac{\sum (X_i - \overline{X})(Y_i - \overline{Y})}{\sum (X_i - \overline{X})^2}$$

$$\beta_0 = \overline{Y} - \beta_1 \overline{X}$$

(3) 模型预测

模型拟合好后，使用该模型对测试集进行预测，然后在模型评估中分析预测效果。

(4) 模型评估

本次实验中选择 R 方、均方根误差（RMSE）以及平均绝对误差（MAE）对模型的拟合效果和预测效果进行分析。

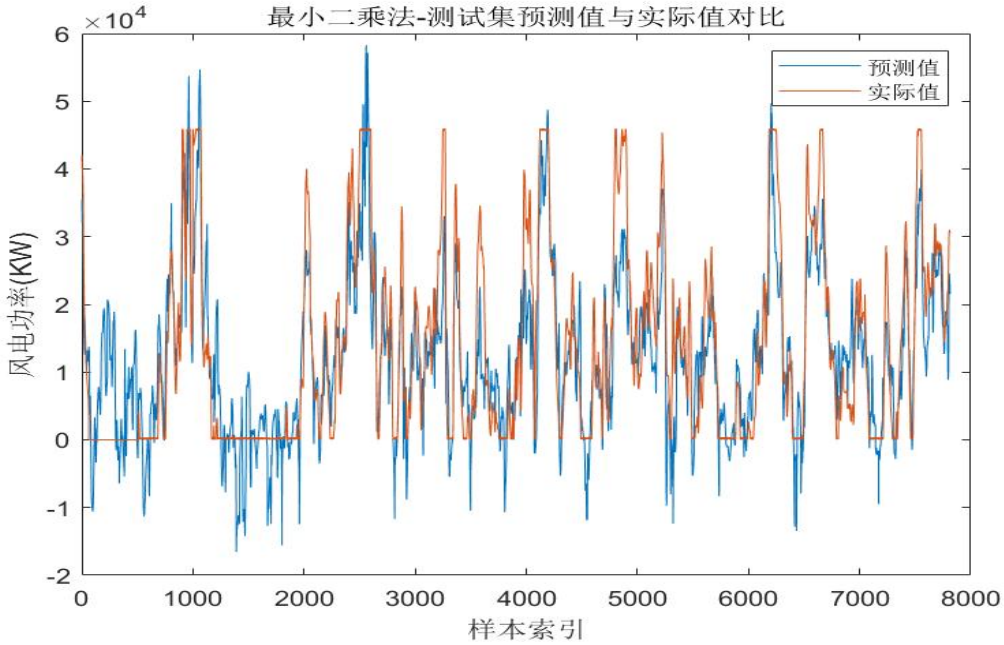

图 28 最小二乘模型预测结果与实际值对比图

训练集和测试集的各评价指标值如下表所示。

表 6 最小二乘模型指标值

评价指标	测试集
R 方	0.7457
RMSE	0.4554
MAE	0.3467

从表 6 和图 28 中可以看出最小二乘方法的预测效果整体还不错，但是在 RMSE 和 MAE 的指标上不如 SVR。

5.3 随机森林回归

随机森林回归（Random Forest Regression）是一种集成学习方法，通过构建多个决策树并结合其结果来进行预测，从而提高模型的准确性和稳定性。随机森林回归在处理非线性、复杂和高维数据时表现出色，广泛应用于各种回归问题。

随机森林回归的基本构件是决策树。决策树是一个递归的分割过程，将数据集分成更小的子集，同时在每个子集上建立预测模型。虽然单个决策树容易过拟合，但其集成方法可以缓解这一问题。随机森林通过构建多个决策树并将

其结果结合起来进行预测的。每棵树的预测结果通过平均来决定最终的结果。

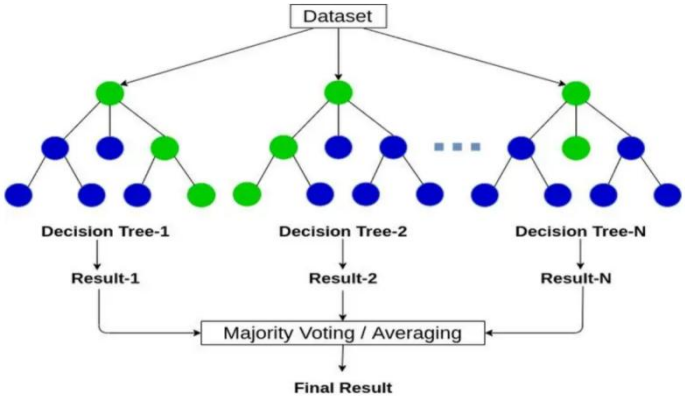

图 29 随机森林示意图

风电过程涉及多个变量，这些变量之间可能存在复杂的非线性相互作用。随机森林算法在化工过程中成分变量的预测中具有多项优势和高度契合性，主要体现在以下几个方面：

(1) **处理非线性关系**：风电过程中的成分变量可能具有复杂的非线性关系。随机森林通过集成多个决策树，可以很好地捕捉这些复杂关系，提供准确的预测结果

(2) **高泛化能力**：由于随机森林使用了多棵决策树的集成方法，并且每棵树都在不同的随机子集上进行训练，这减少了单个模型过拟合的风险，提高了整体模型的泛化能力。

(3) **鲁棒性和抗噪性**：风电数据通常会包含噪声和异常值，随机森林对这些噪声数据有较高的鲁棒性。即使存在一些异常值，也不会显著影响整体模型的预测性能。

模型训练和 SVR 一样，选择功率作为响应变量，以风速、风向和温度作为预测变量建立风电功率预测模型，在原始数据集中选取 80%作为训练集，20%作为测试集。使用的超参数优化方法是网格搜索结合十折交叉验证。网格搜索结合十折交叉验证在上一小节中已介绍，此处不再赘述。

训练结束后，网格搜索得出的超参数为：

表 7 RF 超参数

超参数	取值
max_depth	100
min_samples_split	3
n_estimators	500

训练集和测试集的各评价指标值如下表所示。

表 8　RF 各评价指标值

评价指标	测试集
R 方	0.7621
RMSE	0.4296
MAE	0.2909

训练的 RF 模型对测试集的预测结果与实际值的对比图如下。

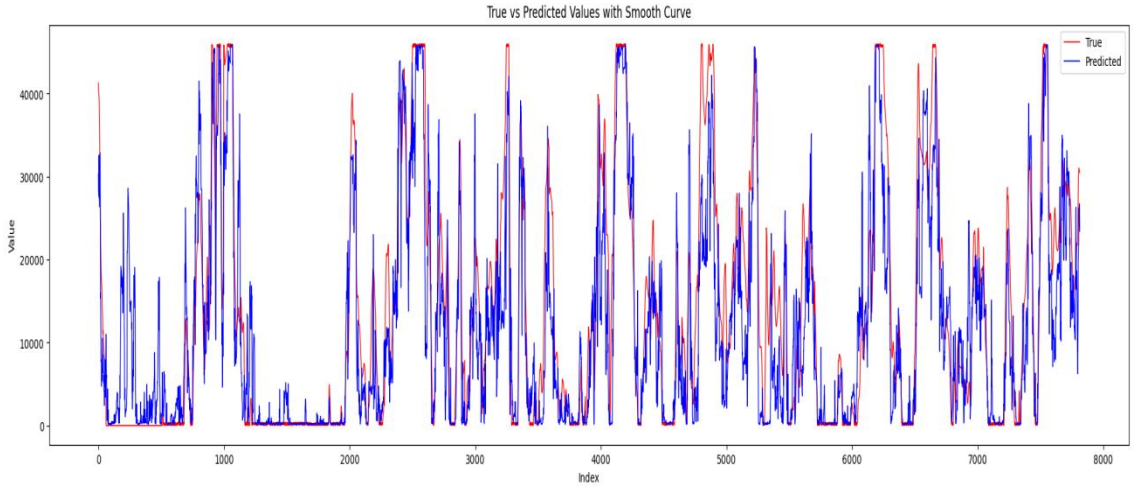

图 30 RF 实际值与预测值对比图

从表 8 和图 30 中可以看出 RF 的预测效果也很不错，整体效果与 SVR 不相上下，但 SVR 在 RMSE 上的表现略胜一筹。

5.4 预测模型比较与评价

在上文中，分别使用了最小二乘、支持向量回归和随机森林回归算法建立了风电功率预测模型。并使用 R 方、RMSE 和 MAE 指标对模型进行评价。不同模型在测试集上的表现如下表所示：

表 9 不同模型在测试集上的表现

	R 方	RMSE	MAE
最小二乘	0.7457	0.4554	0.3467
支持向量	0.8124	0.4030	0.2926
随机森林	0.7621	0.4296	0.2909

分析表 9，可以得出如下结论：

(1) 最小二乘预测效果整体不如机器学习算法的预测效果，可能与风电过程的特

性有关，变量间可能是非线性关系，简单模型不能准确描述天气变量与风电功率之间的关系。

(2) 支持向量回归模型属于非线性模型，预测效果比前两种有明显提升，尤其是 R 方指标，拟合度更好。

(3) 整体上 SVR 模型的表现最佳，在 R 方、MSE 和 RMSE 三个指标上均体现出一定的优势，拟合效果和预测效果不错。

(4) 由于笔记本电脑的性能问题，实验中网格搜索的范围很有限，因此 SVR 和 RF 模型还可以在高性能计算机上进一步优化，得到效果更好的预测模型。

六、风电功率优化

在风电功率预测的基础上，本章将介绍如何利用模拟退火算法对风电功率进行优化，考虑变量间的约束条件，寻找在指定风速下最优的风向和温度组合，从而最大化风电功率输出。模拟退火算法是一种基于物理退火过程的全局优化算法，具有跳出局部最优解的能力，非常适用于解决复杂的优化问题。

6.1 模拟退火算法原理

模拟退火算法模拟的是固体物质在高温下逐渐冷却的过程。在物理退火过程中，物质在高温时，分子具有较高的能量，可以在不同状态之间自由移动；随着温度的降低，分子逐渐失去能量，最终稳定在一个低能量状态。模拟退火算法通过引入“温度”参数，允许在搜索过程中偶尔接受一些劣解，以避免陷入局部最优解。

模拟退火算法的基本步骤如下：

(1) 初始解与初始温度：选择一个初始解，并设定初始温度。

(2) 邻域解生成：在当前解的邻域中生成新的解。

(3) 目标函数计算：计算新解的目标函数值。

(4) 解的接受准则：根据目标函数值差异及当前温度，决定是否接受新解。通常采用 Metropolis 准则：

$$P(\Delta E)=\begin{cases}1 & \Delta E \le 0\\ \exp\left(-\dfrac{\Delta E}{T}\right) & \Delta E > 0\end{cases}$$

其中，ΔE 为新解与当前解的目标函数值差异，T 为当前温度。

(5) 温度更新：根据降温策略降低温度。

(6) 终止条件：当温度降至设定值或迭代次数达到预定值时，终止算法，输出当前解。

6.2 建立风电功率优化模型

6.2.1 目标函数

本文优化模型的目标函数为最大化风电功率输出，由于模拟退火算法寻优过程是取最小值，所以在功率前加负号“-”。

$$\min -P(v,\theta,T)$$

其中，P 为风电功率，v 为风速，θ 风向，T 为温度。

6.2.2 约束处理

在风电功率预测中，风速、风向和温度的直接不等式约束可以用数学表达式表示。这些约束反映了风电机组在不同环境条件下的运行限制，确保预测模型的合理性和实际应用的可行性。经过调研，风电场中有以下几条约束。

(1) 风速约束

风速的约束主要包括启动风速、额定风速和停机风速。

$$v_{cut-in} \le v \le v_{cut-out}$$

其中，v_{cut-in} 是风电机组的启动风速（通常为 1-4 m/s），$v_{cut-out}$ 是风电机组的停机风速（通常为 20-25 m/s）。

(2) 风向约束

风向的约束反映了风电机组需要调整到最佳方向的范围

$$\theta_{\min} \le \theta \le \theta_{\max}$$

其中，$\theta_{\min}$ 和 $\theta_{\max}$ 是风电机组可以调整的最小和最大风向角度范围（通常为 0°到 360°）。

(3) 温度约束

温度的约束确保风电机组在安全的工作温度范围内运行。

$$T_{\min} \le T \le T_{\max}$$

其中，$T_{\min}$ 是风电机组的最低工作温度（通常为-30℃），$T_{\max}$ 是风电机组的最高工作温度（通常为 30℃）。

(4) 组合约束

高风速和极端温度的安全性：高风速条件下，设备的安全性和材料性能可能会受到极端温度的影响。例如，当风速接近停机风速时，如果温度也接近设备的工作极限，可能需要增加额外的保护措施。其中 k_1 是一个调整系数，用于反映温度对高风速条件下安全性的影响。

$$v \le v_{cut-out} - k_1 \cdot (T - T_{\min})$$

风向变化和温度影响：在极端温度条件下，风向调整系统（偏航控制系统）的效率可能下降。因此，在高温或低温条件下，风向调整的响应速度可能需要考虑额外的约束。θ_{opt} 其中是最优风向，k_2 是一个调整系数，用于反映温度对风向调整效率的影响。

$$\left|\theta - \theta_{opt}\right| \le k_2 \cdot (T_{\max} - T)$$

综上，风电场中约束约束条件可整理如下：

$$\begin{cases} v_{cut-in} \le v \le v_{cut-out} \\ \theta_{\min} \le \theta \le \theta_{\max} \\ T_{\min} \le T \le T_{\max} \\ v \le v_{cut-out} - k_1 \cdot (T - T_{\min}) \\ \left|\theta - \theta_{opt}\right| \le k_2 \cdot (T_{\max} - T) \end{cases}$$

对于风速、风向和温度的最大最小约束，可以直接在优化算法中定义，而对于最后两个组合不等式约束，本文采用罚函数的方法，将约束融入到原始目标函数，形成一个组合的目标函数，如下所示。

Python 约束处理代码

```
# 定义参数约束
param_bounds = {
    'direction': (-3, 1.3),   # 风向
```

```
    'temperature': (-2, 2.3) #  温度
}
#  目标函数
def objective_function(params,penalty_factor=1000):
    X_new = np.array([[-0.25, params['direction'], params['temperature']]])
    y_pred = models[0].predict(X_new)
    penalty1 = max(0, constraint1(params))   #  线性适应性罚项
    penalty2 = max(0, constraint2(params))   #  线性适应性罚项
    return -y_pred[0]+ penalty_factor * penalty1+penalty_factor * penalty2     #  取负数，因为我们要最大化功率
#  定义不等式约束函数  g(x) <= 0，例如  x[0] + x[1] - 1 <= 0
def constraint1(params):
    return -0.25 + 0.1 * (params['temperature']+30) - 25
def constraint2(params):
    cc = np.abs(params['direction'])- 0.05 * (30 - params['temperature'])
    return cc
```

本文中约束条件各物理量的含义和取值如下表所示。

表 10 约束条件各物理量的含义和取值

物理量	含义	取值
v_{cut-in}	启动风速	1
$v_{cut-out}$	停机风速	20
$\theta_{\min}$	最小风向角	0
θ_{out}	最大风向角	360
$T_{\min}$	最低工作温度	-30
T_{out}	最高工作温度	30
k_1	调整系数	0.1

k_2	调整系数	0.05
θ_{opt}	最优风向	0

6.2.3 优化过程

模拟退火算法优化风电功率的具体步骤如下：

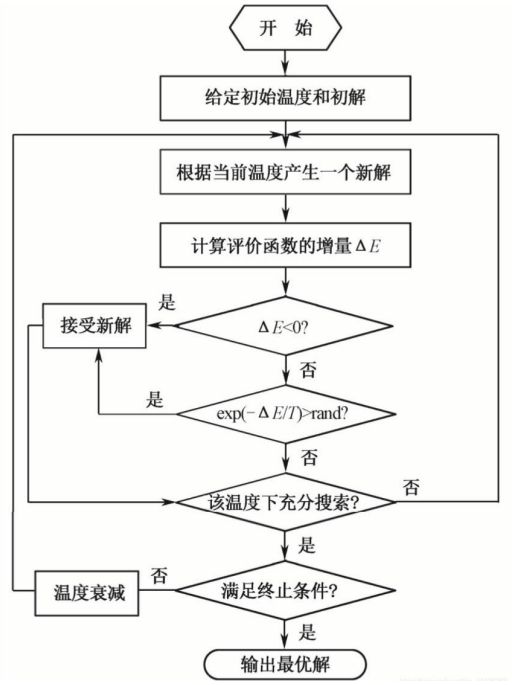

图 31 模拟退火算法流程图

(1) 初始解与温度设定：选择一个初始风向和温度组合，并设定最大迭代次数和初始温度 T_0。

(2) 目标函数计算：根据支持向量回归模型，计算初始组合下的风电功率。

(3) 邻域解生成：在当前风向和温度的邻域内随机生成新的风向和温度组合。

(4) 新解评估：使用支持向量回归模型，计算新组合下的风电功率，并根据目标函数差值和当前温度，决定是否接受新解。

(5) 温度更新：采用指数退火策略更新温度：

$$T_{k+1} = \alpha T_k$$

其中，α 为降温系数，通常取 0.8-0.99。

(6) 迭代终止：当温度降至设定值或迭代次数达到预定值时，终止算法，输出最优的风向和温度组合。

伪代码如下

Algorithm: Simulated Annealing Algorithm

Input: 初始风向 θ0, 初始温度 T0, 初始风速 W, 初始降温系数 α, 最小温度 Tmin, 最大迭代次数 max_iter

Output: 最优风向 θ_opt, 最优温度 T_opt, 最大风电功率 P_opt

1. 初始化

θ_current ← θ0

T_current ← T0

P_current ← CalculatePower(W, θ_current, T0)

θ_opt ← θ_current

T_opt ← T0

P_opt ← P_current

2. While T_current > Tmin and iteration < max_iter do

3. 在 θ_current 和 T_current 的邻域内生成新的解 (θ_new, T_new)

θ_new ← GenerateNeighbor(θ_current)

T_new ← GenerateNeighbor(T_current)

4. 计算新解的风电功率

P_new ← CalculatePower(W, θ_new, T_new)

5. 计算功率差

ΔP ← P_new - P_current

6. If ΔP > 0 then

接受新解

θ_current ← θ_new

T_current ← T_new

P_current ← P_new

If P_current > P_opt then

更新最优解

θ_opt ← θ_current

T_opt ← T_current

P_opt ← P_current

```
        Else
            根据 Metropolis 准则决定是否接受新解
            If exp(-ΔP / T_current) > Random(0, 1) then
                接受新解
                θ_current ← θ_new
                T_current ← T_new
                P_current ← P_new
    7. 更新温度
        T_current ← α * T_current
    8. 迭代计数器 +1
        iteration ← iteration + 1
9. 输出最优解 θ_opt, T_opt 和最大风电功率 P_opt
Function CalculatePower(W, θ, T):
    // 根据支持向量回归模型计算风电功率
    return SVR_Prediction(W, θ, T)
Function GenerateNeighbor(value):
    // 在当前值的邻域内生成新的值
    neighbor ← value + Random(-δ, δ)
    // 保证生成的值在合法范围内
    If neighbor < lower_bound then
        neighbor ← lower_bound
    If neighbor > upper_bound then
        neighbor ← upper_bound
    return neighbor
```

6.2.4 实验结果

本文建立的风电功率预测模型的预测变量有风速、风向和温度三个变量。在风力发电厂，直接调控环境中的风速、风向和温度是不现实的。但随着科技的发展，目前可以通过多种手段来间接调控和管理风电机组内部及其关键部件的温度，例如加热系统和冷却系统。此外现代风电机组通常配备有偏航控制系统，可以自动调整风轮对准风向，以最大化风能的捕获。因此本文的优化目标是在指定风速下，优化风向和温度，以达到最高的风电功率。

在第四章的工况识别中，我们一共归纳出三种工况，分别代表低风速、中风速、高风速。

表 11 三种工况的变量范围

工况	风速范围(m/s)	风向范围(°)	温度范围(°C)
1	4.4~7.2	170~360	-25~30
2	7.2~15	210~360	-27~8.5
3	1~4.4	3~210	-20~26

接下来分别在三种工况下，结合 6.2.2 的整体约束和表 11 中的工况约束，运行模拟退火功率优化模型，并将优化前后的功率对比分析。

表 12 优化前后的功率对比

风速(m/s)	优化前功率(kW)	优化后功率(kW)	最优风向(°)	最优温度(°C)
5	31796.91	37236.44	218.79	-21.67
14	45886.49	77984.74	309.52	4.07
2	9780.52	14242.09	201.94	-10.73

从表 12 可以看出，在指定风速条件下，模拟退火算法能够有效地在限定范围内找到最优的风向和温度组合，使风电功率输出达到最大化，达到了预期目标。

七、总结与感悟

7.1 总结

本次的工业过程智能优化技术课程报告共分为预测模型建立与优化模型建立大部分，环环相扣。报告内容与课内理论知识关系密切，大大加深了我对理论部分的理解。

第一部分是预测模型的建立，这与大数据课程设计的内容相仿，主要包括数据可视化、统计分析、数据去噪、数据变化等内容。在常见方法的基础上，我还使用到了风向玫瑰图、DBSCAN 算法去噪等新内容，扩大了知识面。此外，本次可视化绘图中我使用了 Python，体会到了其绘图功能的强大和多样性，为

之后绘图提供了更多的选择。在工况分析中，我深刻意识到识别不同工况并分别建立预测模型是非常巧妙和有效的方法。在回归模型建立中，主要是建立风电功率预测模型，和之前实验的步骤类似，先对数据进行预处理，然后使用 SVR、最小二乘法和 RF 分别建立预测模型，并对模型的性能进行评估。

第二部分是功率优化模型的建立，这也是这门课的重中之重，前面预测模型的建立和工况识别其实就是为优化打基础。在得到预测效果优良的模型后，本文采用了模拟退火算法分别在三种工况相对应的风速下，寻找最优风向和温度，使得风电功率达到最大。

7.2 感悟

甘老师的工业过程智能优化技术课程让我受益匪浅，与以往的课程不同，这门课有很多课堂互动和现场小测验，并不是一味地填鸭式教学和读 ppt，而是真正注重这门课的重点知识，以及培养我们的思考能力和主观能动性。

我印象深刻是上课期间甘老师总会在讲解前问问大家有什么 idea，让我们充分思考后再开始相关知识或步骤的讲解。刚开始可能会觉得有点困难，但细细想来这就是在培养学生的思考能力。在前几个学期我总是一味地听课，忽视自我思考与辩证，以至于每学完和考完一门课就把那些知识抛之脑后了，很多时候不知道自己学了什么。如果能够做到在学习过程中充分加入自己的思考和创新，我想自己的主观能动性会大大提高，有助于之后研究生阶段的发展。在之后，我将努力践行甘老师教给我的理念，注重思考为什么要这样做，以及自己会怎么做，深入挖掘知识点。

行文至此，工业过程智能优化技术课程也迎来结束。未来在控制科学与工程的道路上，课程蕴含的知识和老师认真严谨的工作态度永远会引领我前行！祝愿老师身体健康，工作顺利。

参考文献

[1] Gan C, Cao W H, Liu K Z, et al. A new hybrid bat algorithm and its application to the ROP optimization in drilling processes[J]. IEEE Transactions on Industrial Informatics, 2019, 16(12): 7338-7348.

[2] Gan C, Cao W H, Wu M, et al. Prediction of drilling rate of penetration (ROP) using hybrid support vector regression: A case study on the Shennongjia area, Central China[J]. Journal of Petroleum Science and Engineering, 2019, 181: 106200.

[3] 余青何, 摆玉龙, 范满红. 融合支持向量机算法的数据驱动型数据同化方法研究[J]. 遥感技术与应用, 2024, 39(02): 381-392.

[4] 何山, 郝雄博, 赵宇明, 等. 基于遗传算法优化支持向量回归的电池 SOH 预测[J]. 汽车技术, 2024(05): 31-36. DOI: 10. 19620/j. cnki. 1000-3703. 20230606.

[5] 吴振龙, 莫艺鹏, 王荣花, 等. 基于 LSTM 和粒子群算法的多机组风电功率预测[J/OL]. 郑州大学学报(工学版), 1-8[2024-07-08]. https://doi.org/ 10. 13705/ j.issn. 1671-6833. 2024. 06. 005.

[6] 门茂琛, 赵睿, 张金帅, 等. 基于改进模拟退火-粒子群的配电网分布式光伏承载力评估[J]. 浙江大学学报(工学版), 2024, 58(06): 1255-1265.

附录-上机实验钻速优化结果与代码

在上机实验中，我在钻速预测模型与钻进约束条件的基础上，综合运用智能优化方法优化钻压、钻速等操作参数，实现钻速优化。钻速预测模型采用的是支持向量回归算法，优化算法为模拟退火算法。对于不等式约束条件采用线性适应性罚项方法，将约束条件加在目标函数中。

指定井深为 1125m，在 Jupyter notebook 中运行程序，结果如下所示。

```
Iteration 1988: Best Fitness = 4.8862150730524
Iteration 1989: Best Fitness = 4.8862150730524
Iteration 1990: Best Fitness = 4.8862150730524
Iteration 1991: Best Fitness = 4.8862150730524
Iteration 1992: Best Fitness = 4.8862150730524
Iteration 1993: Best Fitness = 4.8862150730524
Iteration 1994: Best Fitness = 4.8862150730524
Iteration 1995: Best Fitness = 4.8862150730524
Iteration 1996: Best Fitness = 4.8862150730524
Iteration 1997: Best Fitness = 4.8862150730524
Iteration 1998: Best Fitness = 4.8862150730524
Iteration 1999: Best Fitness = 4.8862150730524
Iteration 2000: Best Fitness = 4.8862150730524
Optimized Parameters: {'WOB': 17.543087178276927, 'RPM': 94.38917242289511, 'TORQUE': 2759.02765908232, 'STANDPIPE_PRESSURE': 2.0199571351
905496}

Out[2]: {'WOB': 17.543087178276927,
         'RPM': 94.38917242289511,
         'TORQUE': 2759.02765908232,
         'STANDPIPE_PRESSURE': 2.0199571351905496}

In [3]: X_new = np.array([[1125, best_solution['WOB'], best_solution['RPM'], best_solution['TORQUE'], best_solution['STANDPIPE_PRESSURE']]])  # 新数据
        y_pred = best_svr.predict(X_new)
        print("Predicted Drilling Speed:", y_pred)

        Predicted Drilling Speed: [4.88621507]
```

从上图可以看出，优化后的钻速值为 4.8862，操作参数如下：

参数	数值
WOB	17.543087178276927
RPM	94.38917242289511
TORQUE	2759.02765908232
STANDPIPE_PRESSURE	2.0199571351905496

Python 程序如下（Jupyter notebook 运行）：

```
import numpy as np
import pandas as pd
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

```
import random

# 读取数据
file_path = r"D:\桌面\优化数据集.xls"  # 替换为你的 Excel 文件路径
data = pd.read_excel(file_path)
data_array = data.to_numpy()

X = data_array[:, 0:5]  # 特征变量：井深、钻压、转速、扭矩、立压
y = data_array[:, 5]    # 响应变量：钻速

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.16, random_state=42)

# 定义参数网格
param_grid = {
    'kernel': [ 'rbf'],
    'C': [80],
    'epsilon': [0.0445],
    'gamma': [0.000314]
}

# 使用网格搜索和交叉验证来优化 SVR 模型
svr = SVR()
grid_search = GridSearchCV(svr, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

# 获取最优参数和模型
best_svr = grid_search.best_estimator_
best_params = grid_search.best_params_
print("Best Parameters:", best_params)
```

```
# 预测和计算误差
y_pred = best_svr.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

print("MSE:", mse)
print("RMSE:", rmse)

# 绘制预测值和真实值的对比图
plt.figure(figsize=(10, 6))
plt.plot(y_test, label='True Values')
plt.plot(y_pred, label='Predicted Values')
plt.xlabel('Sample Index')
plt.ylabel('Drilling Speed')
plt.title('True vs Predicted Drilling Speed')
plt.legend()
plt.show()

# 定义参数约束
param_bounds = {
    'WOB': (1, 23.14),  # 钻压
    'RPM': (88, 102),  # 转速
    'TORQUE': (2000, 4000),  # 扭矩
    'STANDPIPE_PRESSURE': (2, 5)  # 立压
}
'''
# 目标函数
def objective_function(params):
    X_new = np.array([[0, params['WOB'], params['RPM'], params['TORQUE'],
params['STANDPIPE_PRESSURE']]])  # 假设井深为 0
    y_pred = svr.predict(X_new)
```

```
    return -y_pred[0]  # 取负数，因为我们要最大化钻速
'''
# 目标函数
def objective_function(params,penalty_factor=1000):
    X_new = np.array([[1125, params['WOB'], params['RPM'], params['TORQUE'], params['STANDPIPE_PRESSURE']]])  # 假设井深
    y_pred = best_svr.predict(X_new)
    penalty1 = max(0, constraint1(params))  # 线性适应性罚项
    penalty2 = max(0, constraint2(params))  # 线性适应性罚项
    return -y_pred[0] + penalty_factor * penalty1+penalty_factor * penalty2  # 取负数，因为我们要最大化钻速

# 定义不等式约束函数 g(x) <= 0，例如 x[0] + x[1] - 1 <= 0
def constraint1(params):
    return params['WOB'] * params['RPM'] - 1657.45
def constraint2(params):
    cc = 200-((2*(5.59-0.0161*params['WOB']))/(7.8805e-4*(0.5*params['RPM']+2.18e-5*params['RPM']**3)))
    return cc
'''
# 适应性罚函数方法将约束转化为目标函数的一部分
def penalized_objective(x, penalty_factor=100):
    penalty1 = max(0, constraint1(x))  # 线性适应性罚项
    penalty2 = max(0, constraint2(x))  # 线性适应性罚项
    return objective_function(params) + penalty_factor * penalty1+penalty_factor * penalty2
'''


# 生成初始解
def generate_initial_solution():
```

```
    return {
        'WOB': 19,
        'RPM': 97,
        'TORQUE': 2749,
        'STANDPIPE_PRESSURE': 4
    }

# 生成邻域解
def generate_neighbor(solution):
    neighbor = solution.copy()
    param_to_modify = random.choice(list(param_bounds.keys()))
    neighbor[param_to_modify] = random.uniform(*param_bounds[param_to_modify])
    return neighbor

# 模拟退火算法
def simulated_annealing(max_iter=2000, initial_temp=100, cooling_rate=0.9):
    current_solution = generate_initial_solution()
    current_fitness = objective_function(current_solution)
    best_solution = current_solution
    best_fitness = current_fitness
    temperature = initial_temp

    for i in range(max_iter):
        neighbor = generate_neighbor(current_solution)
        neighbor_fitness = objective_function(neighbor)

        if neighbor_fitness < current_fitness or random.uniform(0, 1) < np.exp((current_fitness - neighbor_fitness) / temperature):
            current_solution = neighbor
            current_fitness = neighbor_fitness
```

```
                if current_fitness < best_fitness:
                    best_solution = current_solution
                    best_fitness = current_fitness

            temperature *= cooling_rate

            print(f'Iteration {i+1}: Best Fitness = {-best_fitness}')

        return best_solution

# 运行模拟退火算法
best_solution = simulated_annealing()
print("Optimized Parameters:", best_solution)
best_solution

X_new = np.array([[1125, best_solution['WOB'], best_solution['RPM'], best_solution['TORQUE'], best_solution['STANDPIPE_PRESSURE']]])  # 新数据示例：假设井深为 0，其他参数为示例值
y_pred = best_svr.predict(X_new)
print("Predicted Drilling Speed:", y_pred)
```