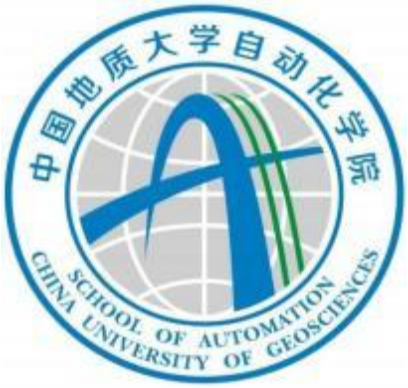

中国地质大学（武汉）自动化学院
智能制造大数据技术报告-风电数据仿真

课　　程：智能制造大数据技术实践

学　　号：20211001578

班　　级：231216

姓　　名：王愉杰

指导老师：丁敏

2024 年 6 月 25 日

目录

1 问题描述

1.1 风机运行原理

风机是一种能够将风能转化为机械能的设备，其运行原理基于伯努利原理和牛顿运动定律。

当风吹来时，风机叶片的下表面受到了高压力，而上表面则受到了低压力，这产生了一个向上的力，导致叶片开始旋转。这个旋转运动会转化成机械能，从而驱动风机的运行。

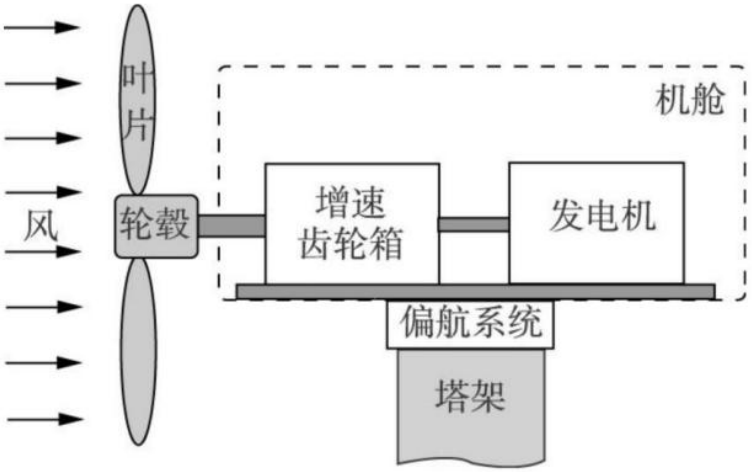

图 1 风机示意图

数学上，风机的性能可以由所谓的风功率系数来表征，表示为C_p。这个系数是描述风机输出功率和风吹来时的动能之比，因此可以用来衡量风机的效率。风功率系数由以下公式给出：

$$C_p = \frac{P}{0.5\rho A V^3}$$

其中，P是风机的输出功率，ρ是空气密度，A是风机的叶盘面积，V是风速。可以看出，当C_p越大时，风机的效率越高。

因此，要提高风机的输出功率，可以通过增大叶盘面积A、提高风速V以及改善风功率系数C_p来实现。

1.2 风电场的数据分布

风电场的数据分布特征主要涉及风速和风向两个方面。

对于风速，风电场的数据通常呈现出长期的概率分布特征，即某一特定风速出现的概率。一般而言，风速概率分布遵循韦伯分布、魏布尔分布、罗瑞斯分布等常见的概率分布模型。同时，由于不同的地理位置和时间点，风速分布的概率密度函数也会有所不同。例如，在海洋沿岸附近的风电场，风速分布可能会更加

呈现出几率低、强度高的特点，因为海洋的温度使得低温高压和高温低压相互交错，导致局部的高风速存在。

对于风向，风电场的数据分布通常为风向分布频数统计直方图。根据不同的地理位置和运行环境，风向的分布情况也存在差异，例如在平原和海上风电场，风向分布经常处于一定的周期性或者主导方向范围内，而在山区或者高山峡谷等地形复杂的区域，风向分布则更加随机和复杂。在分析风向分布时还需要注意到假象的风向分布，因为风都是在三维空间中流动的，所以地形、风机布局等因素都可能会对风向的分布产生影响。

综上所述，风电场的数据分布特征需要考虑到不同的因素，主要包括地理位置、季节气候、地形构造以及风机的布局等因素。同时，对于不同的特定风电场，需要根据观测数据进行实时监测和分析，以便及时做出调整和优化[2]。本实验根据指定风电场种给出的数据，进行分析与挖掘。

1.3 任务要求

(1) 数据清洗与预处理

✓ 完成数据重复值与缺失值处理、数据集划分与归一化。

✓ 展示风电场数据的特征分析结果，包括统计特征表、风速功率散点图、风速频率直方图、风向功率散点图、风向玫瑰图等。

✓ 展示 DBSCAN 算法去除异常值的聚类图像以及清洗后数据的可视化图像。

(2) 特征相关性分析

✓ 计算相关系数：使用皮尔逊相关系数公式计算每对特征变量之间的相关系数，该系数范围从-1 到 1，表示变量间的线性关系强度和方向。

✓ 构建相关性矩阵：将所有特征变量两两之间的相关系数整合成一个相关性矩阵，便于直观地查看和分析特征间的相关性。

✓ 可视化：通过绘制散点图矩阵或热力图等可视化工具，直观展示特征间的相关性。

✓ 特征筛选：筛选与功率相关性较高的特征作为聚类与预测模型的输入。

(3) 数据聚类

✓ 完成 K-means 聚类分析。展示肘部法与轮廓系数可视化图像，展示 K-means 聚类结果与各类别分布的可视化图像。

(4) 预测建模

✓ 采用最小二乘法、BP 神经网络与支持向量回归等方法对每一类数据进行模型训练，要求不少于两种预测方法

(5) 误差分析

✓ 分析实测功率与预测功率之间的误差特性，主要为统计关系。采用平均绝对

误差（MAE）、均方根误差（RMSE）和判定系数（R 方）来表示模型预测效果的评价指标。

2 风电场数据特征

本节对风电场的数据进行特征分析，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，流程图如图 2 所示。

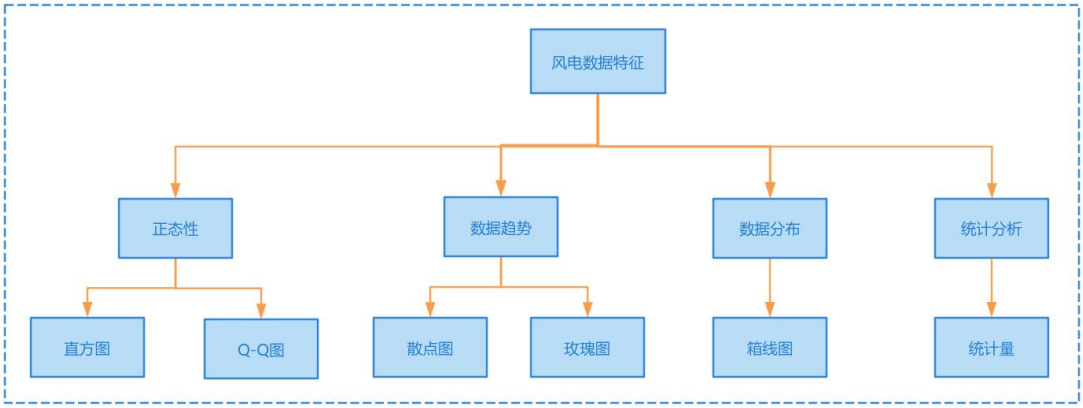

图 2 特征分析流程图

2.1 正态性检验（直方图，Q-Q 图）

本实验最终是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 3 和图 4 所示。

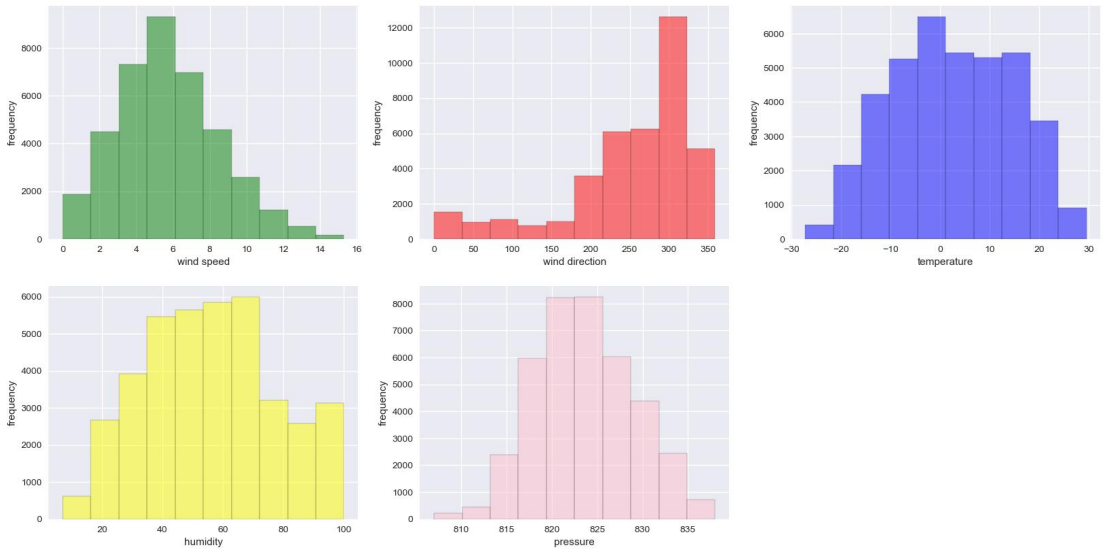

图 3 天气变量直方图

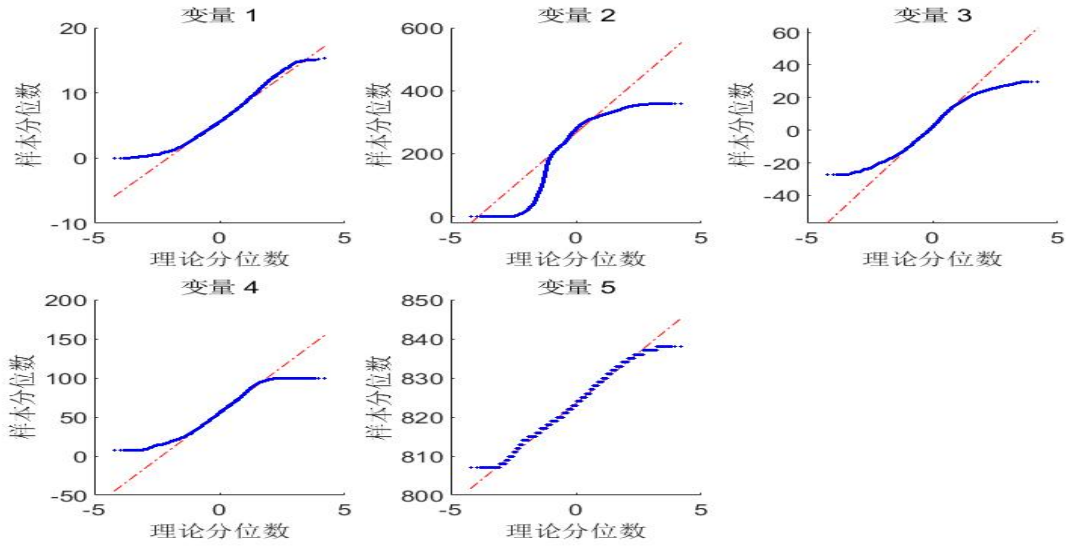

图 4 天气变量 Q-Q 图

综合观察图 3 和图 4，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。

2.2 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 5 所示。

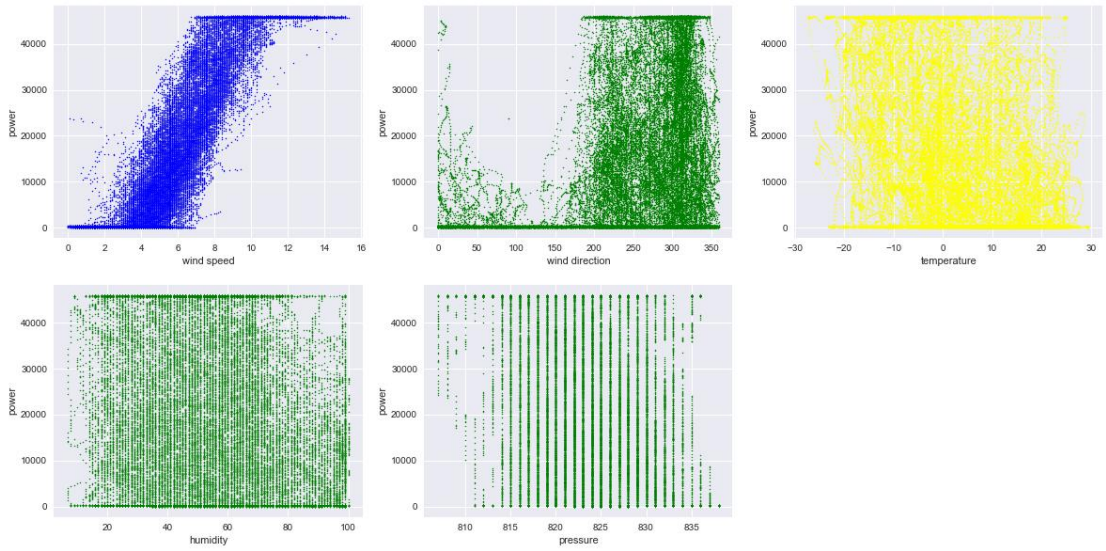

图 5 天气变量与功率的散点图

2.3 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。

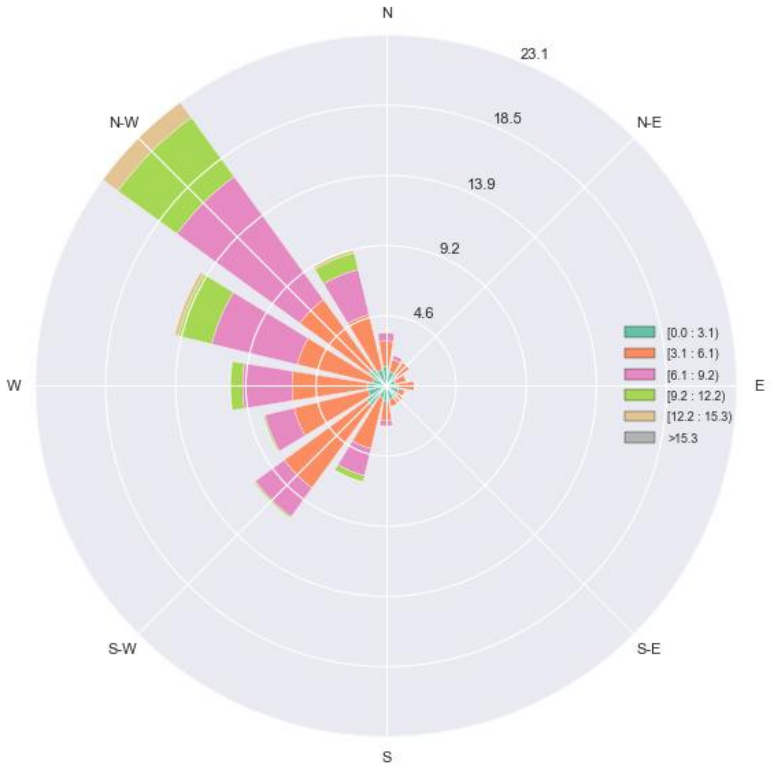

图 6 风速-风向玫瑰图

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。

2.4 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 7 所示：

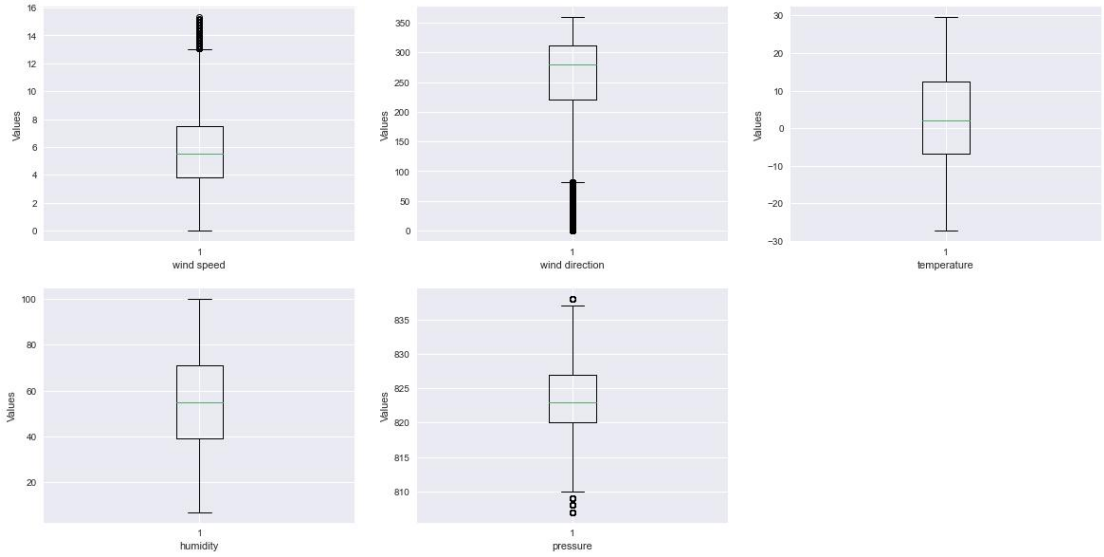

图 7 天气变量箱线图

2.5 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 六个变量的统计量值

	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

3 数据清洗与预处理

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。

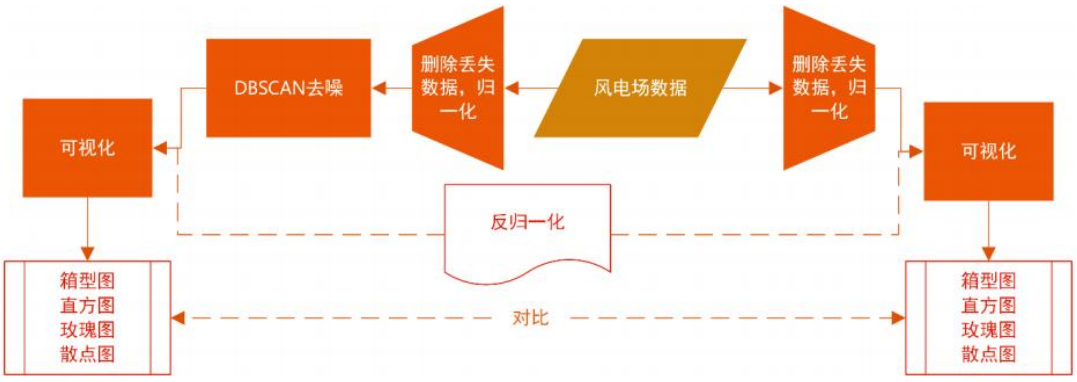

图 8 数据清洗与预处理流程图

3.1 数据清洗与归一化

3.1.1 数据清洗

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。

3.1.2 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。

3.1.3 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。

3.2 DBSCAN 去噪

3.2.1 去噪原理

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 9 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。

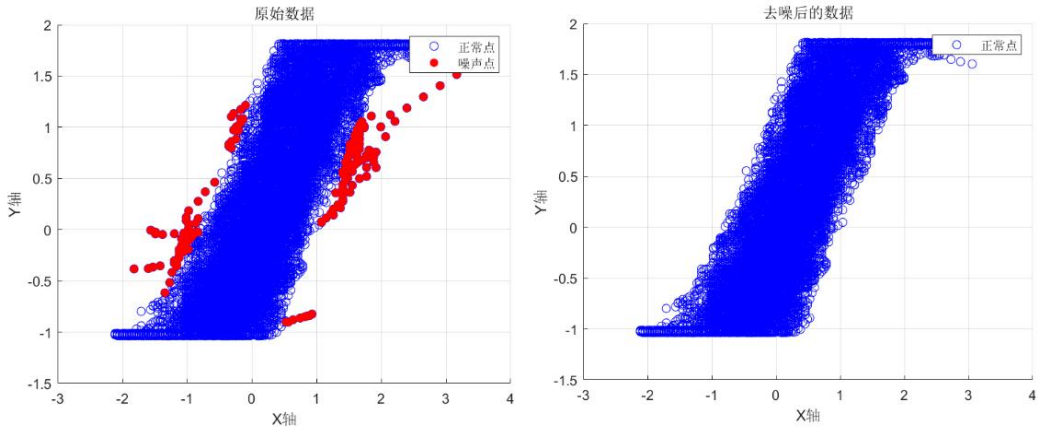

图 9 去噪前后数据对比

3.2.2 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

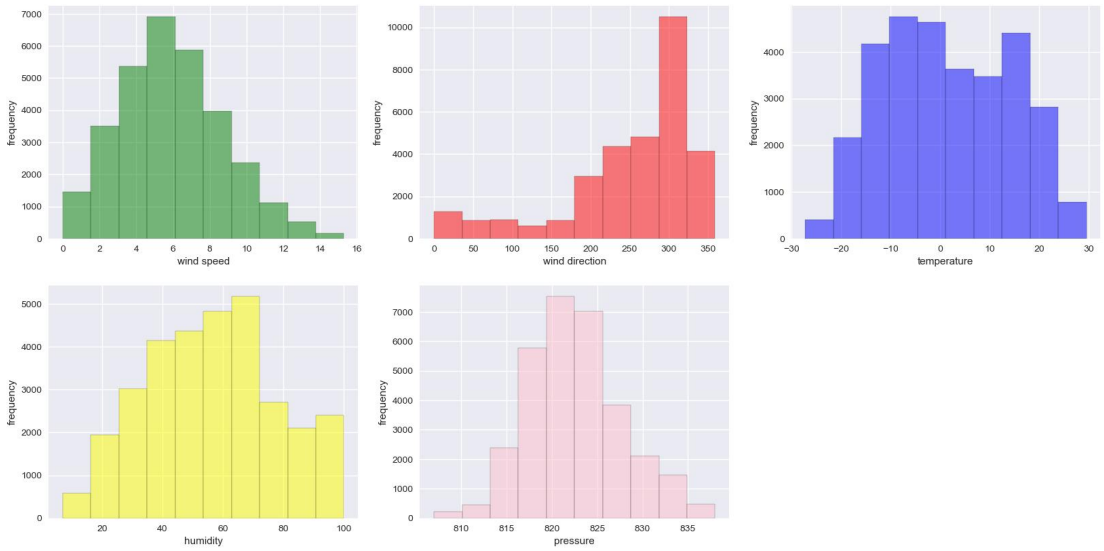

图 10 去噪前训练集天气变量直方图

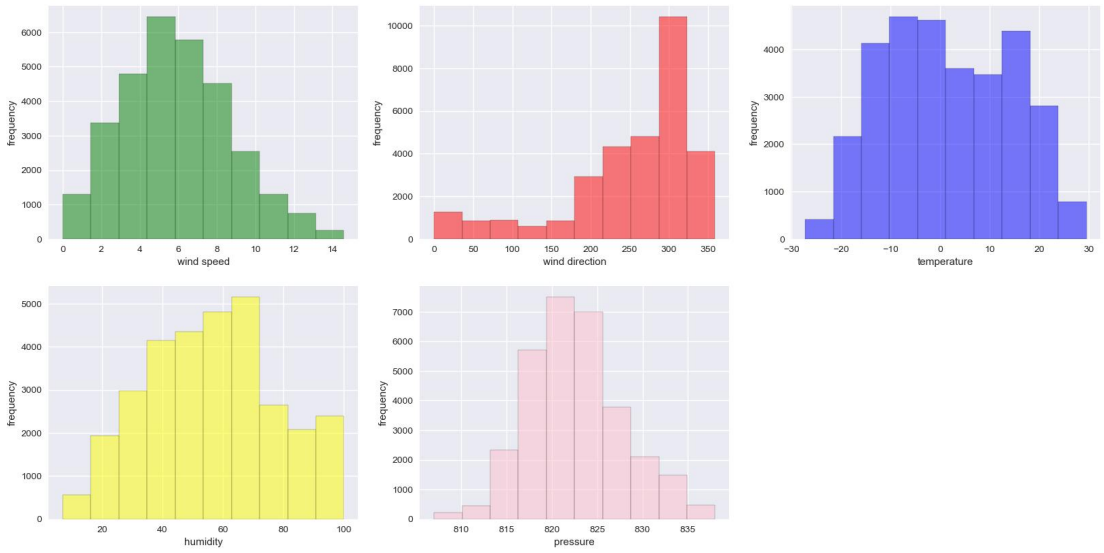

图 11 去噪后训练集天气变量直方图

对比分析图 10 和图 11 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 3.2.1 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。

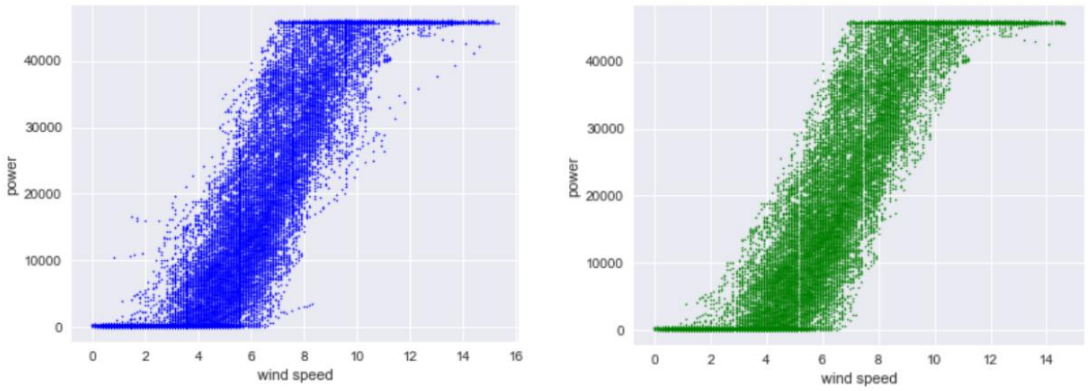

图 12 去噪前后的风速-功率散点图

(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 13 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。

4 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。

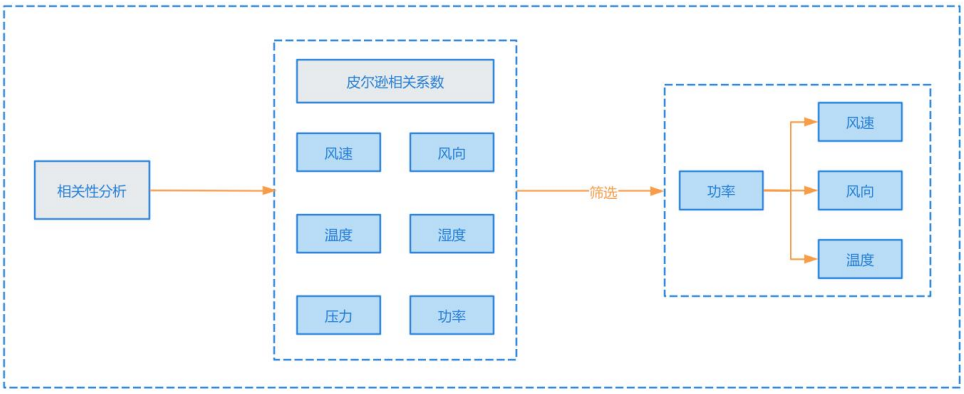

图 14 相关性分析流程图

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 15 所示。

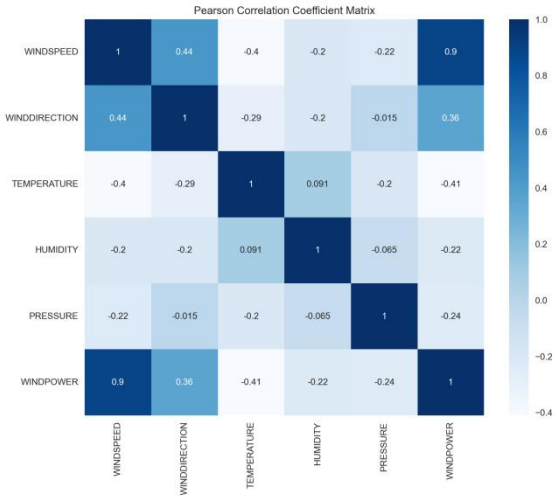

图 15 相关系数热力图

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 16 所示。

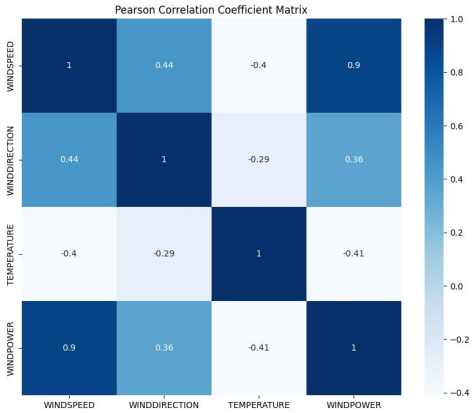

图 16 筛选后的热力图

5 工况识别-聚类分析

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。

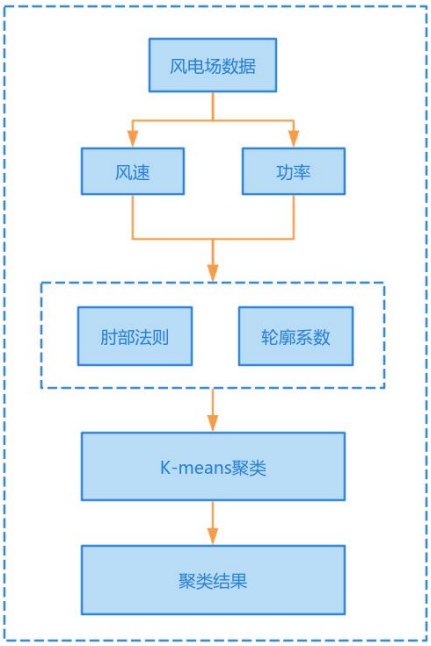

图 17 聚类分析流程图

5.1 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。

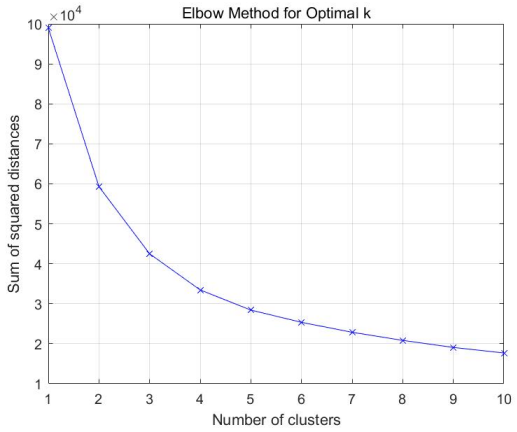

图 18 肘部法

图 18 中，曲线的“肘部”位于 k=3 或 k=4 的位置。

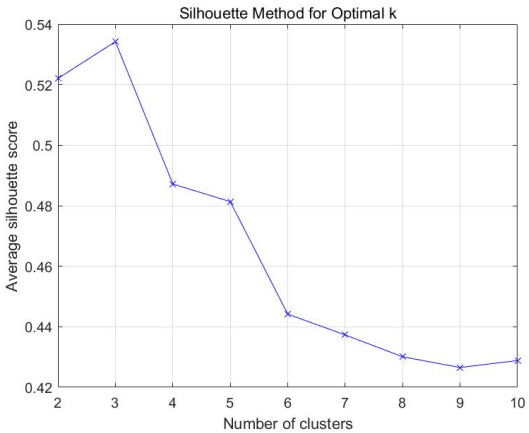

图 19 轮廓系数法

图 19 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。

5.2 聚类结果分析

图 20~22 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。

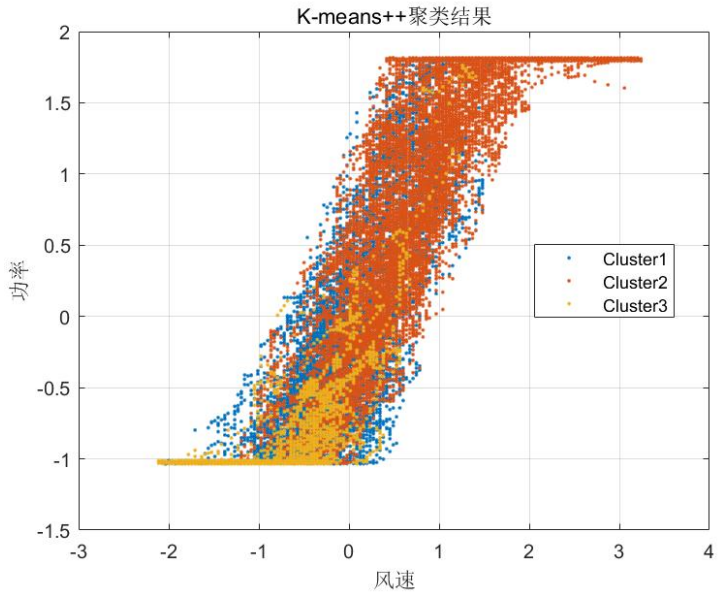

图 20 风速聚类散点图

图 20 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。

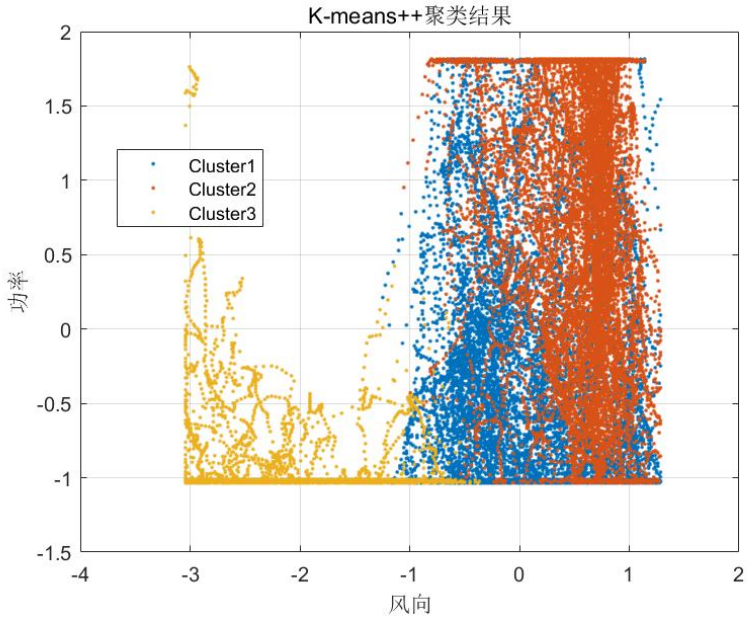

图 21 风向聚类散点图

图 21 显示三种类型的簇分别代表一个范围内的风向。

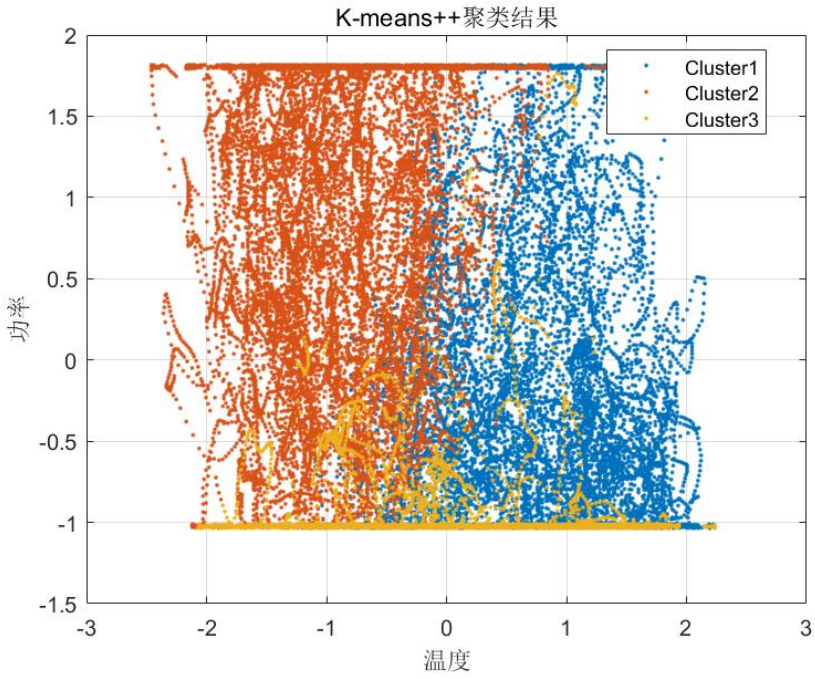

图 22 温度聚类散点图

图 22 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 23~25 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。

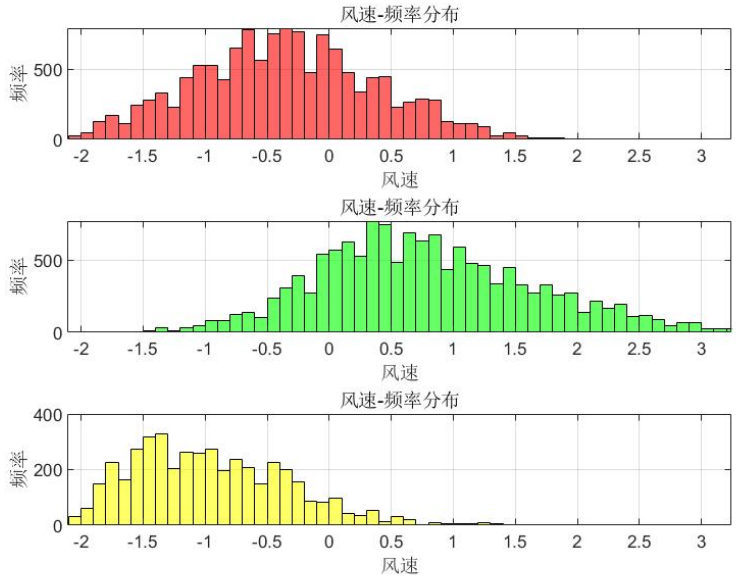

图 23 风速-频率分布图

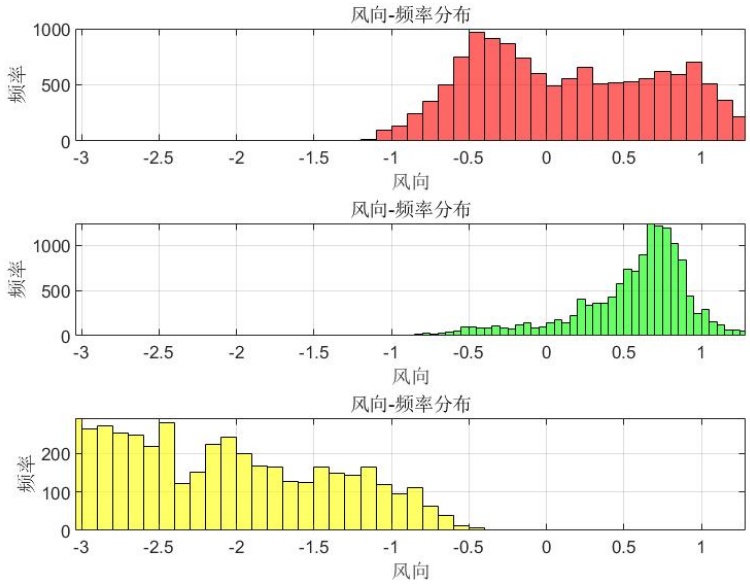

图 24 风向-频率分布图

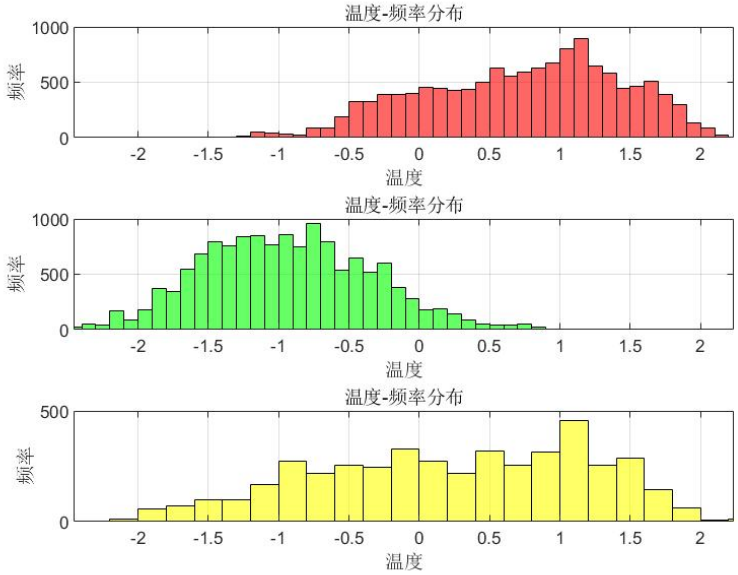

图 25 温度-频率分布图

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。

6 回归分析

本节的回归分析中，我们将在 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。

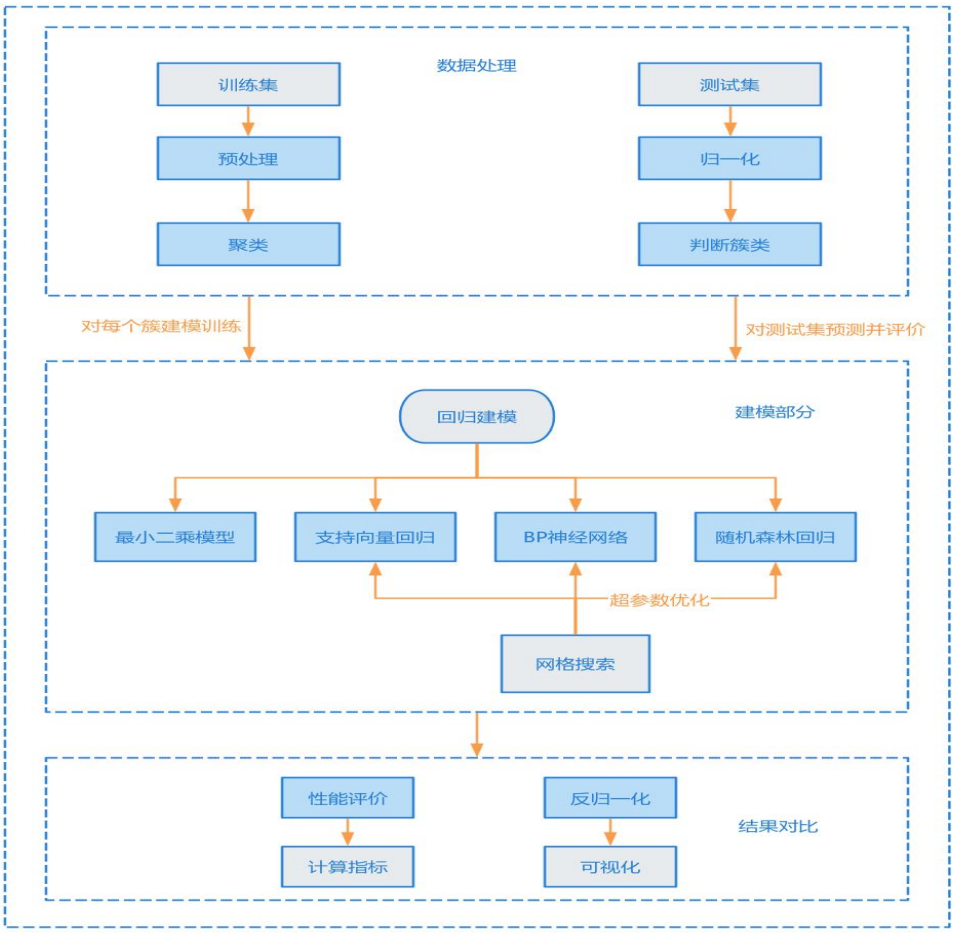

图 26 回归分析流程图

6.1 BP 神经网络

BP 神经网络（Back Propagation Neural Network）是一种常用的人工神经网络模型，是一种有监督学习算法，又称多层感知机(MLP)用于处理多类别分类或回归问题。BP 神经网络由多个神经元组成的多层前馈网络，通过前向传播和误差反向传播两个过程来计算和调整网络的权重和阈值。

BP 神经网络模型的基本流程包括输入层、隐含层和输出层。网络的训练过程包括前向传播和反向传播两个过程。在前向传播过程中，网络根据输入数据和当前的权重和阈值计算输出值。在反向传播过程中，网络根据真实标签和前向传播的输出值计算损失函数，并使用误差反向传播算法调整网络的权重和阈值。

设计 BP 神经网络需要考虑 4 个问题

(1) 网络层数：具有至少一个 S 型隐含层加上一个线性输入层的网络，能够逼近任何有理函数。增加层数可以进一步的降低误差，提高精度，但同时也使网络复杂化。

(2) 隐含层神经元：网络训练精度的提高，可以通过采用一个隐含层，而增加其

神经元数的方法来获得。

(3) 初始权值选取：BP 神经网络是非线性优化算法，初始值设置不当，可能陷入局部极小。

(4) 学习速率：决定每一次循环训练中所产生的权值变化量。为了减少寻找学习速率的训练次数以及训练时间，比较合适的方法是采用变化的自适应学习速率，使网络的训练在不同的阶段设置不同大小的学习速率。

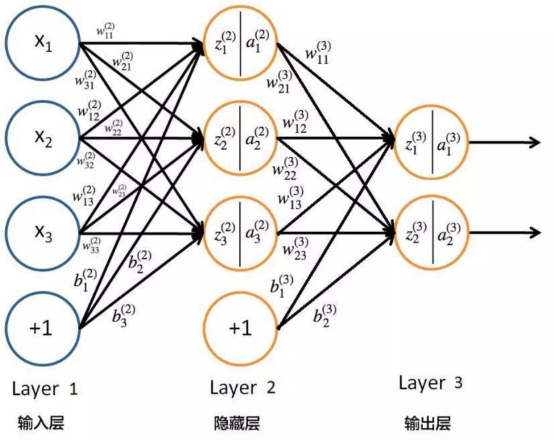

图 27 BP 神经网络示意图

BP 模型的优点在于它可以模拟非线性函数，具有较好的灵活性和适应性。另外，BP 神经网络还可以通过加入正则项和调整学习率等方式来控制过拟合问题。然而，BP 神经网络容易陷入局部最优解、训练时间较长等问题，同时模型的可解释性也较差，不太适合直观理解和解释。

训练的模型对测试集的预测结果与实际值的对比图如下。

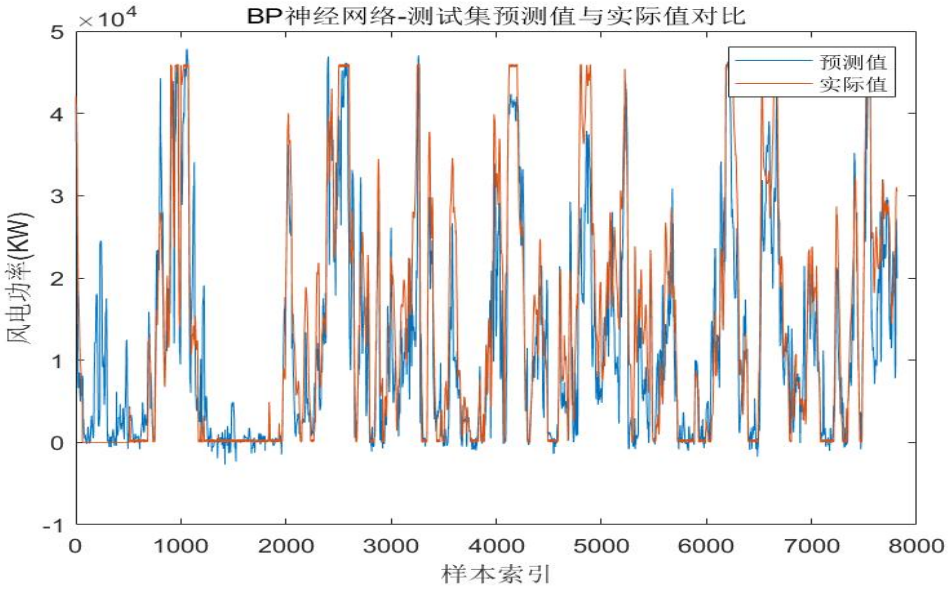

图 28 BP 神经网络预测结果

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，测试集的各评价指标值如下表所示。

表 4 BP 神经网络各评价指标值

评价指标	测试集
R 方	0.8084
RMSE	0.4061
MAE	0.2786

从表 4 和图 28 中可以看出 BP 神经网络的预测效果整体还不错，R 方值达到 0.8，误差也较小，但是在平稳期的预测效果不如振荡期。

6.2 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：

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

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

(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。

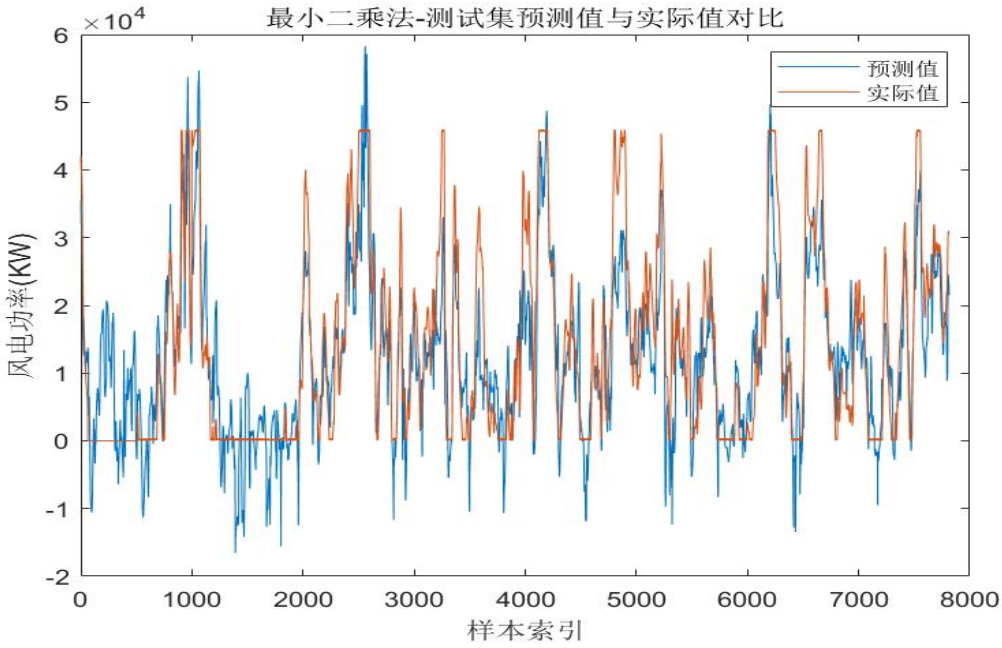

图 29 最小二乘模型预测结果与实际值对比图

训练集和测试集的各评价指标值如下表所示。

表 5 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 5 和图 29 中可以看出最小二乘方法的预测效果整体还不错，但是在平稳期的预测效果不如振荡期。

6.3 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。

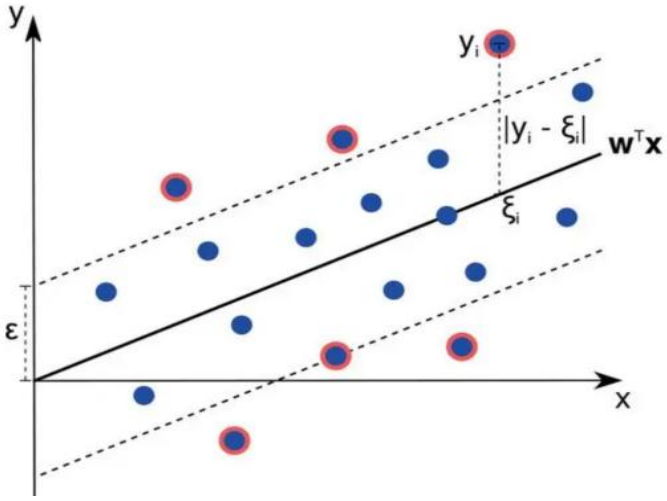

图 30 SVR 示意图

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 3.2.1 小节分析选择的 5 个预测变量建立成分变量预测模型，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。

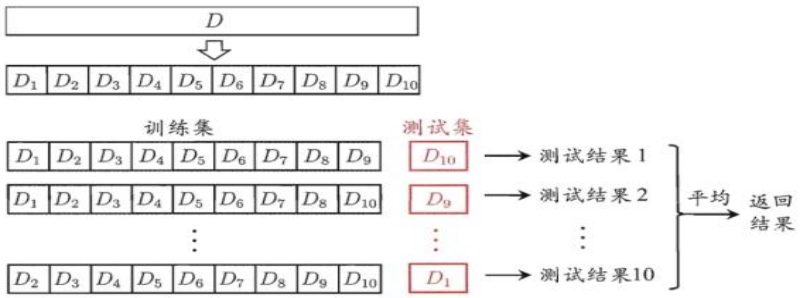

图 31 十折交叉验证示意图

(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 6 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 7 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

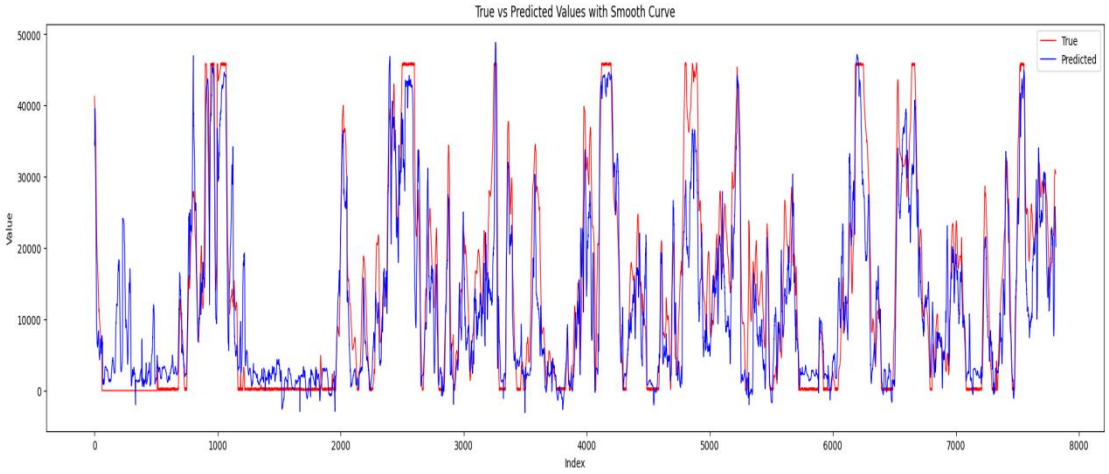

图 32 SVR 预测结果与真实值对比图

从表 7 和图 32 中可以看出 SVR 的预测效果相比前两种更加好，但是依旧存在平稳期的预测效果欠佳的问题。

6.4 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。

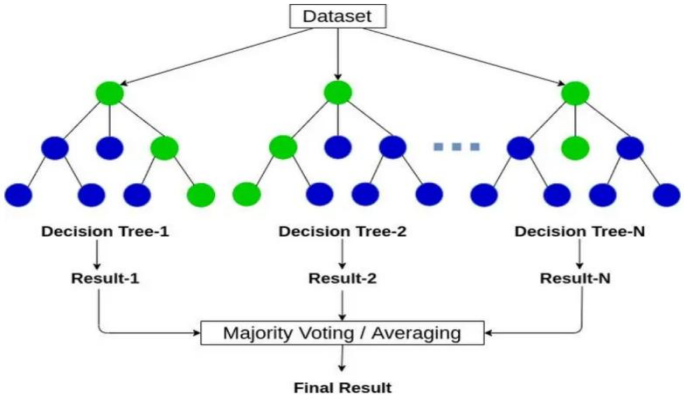

图 33 随机森林示意图

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 8 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 9 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。

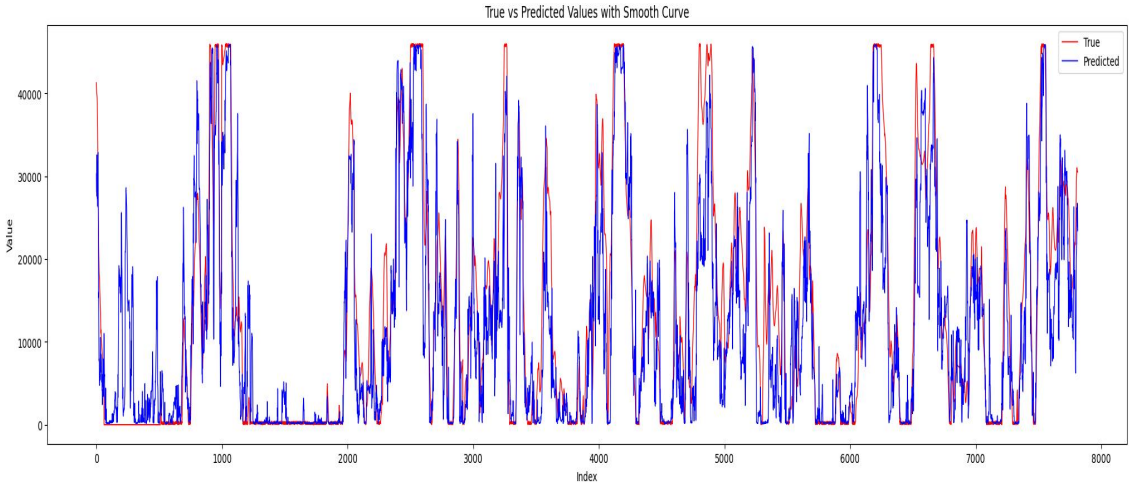

图 34 RF 实际值与预测值对比图

从表 9 和图 24 中可以看出 RF 的预测效果也很不错，与 SVR 不相上下。

6.5 模型比较与评价

在上文中，分别使用了最小二乘、BP 神经网络、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 10 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
BP 神经网络	0.8084	0.4061	0.2786
支持向量	0.8124	0.4030	0.2926
随机森林	0.7621	0.4296	0.2909

分析表 10，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 BP 神经网络模型和 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 BP、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。

7 总结与感悟

7.1 总结

本次的大数据风电功率预测建模实验共分为数据预处理和可视化、相关性分析、聚类分析和回归分析四大部分，从数据预处理到特征提取再到回归和分类模型的建立，层层递进，环环相扣。实验内容与大数据课程的课内理论知识关系密切，大大加深了我对理论部分的理解。

在数据预处理和可视化仿真部分，与第一阶段的 TEP 仿真实验类似，主要包括数据可视化、统计分析、数据去噪、数据变化等内容。与第一阶段有所不同，本次我使用到了不同的方法，包括可视化中的风向玫瑰图、去噪中的 DBSCAN 算法，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。

通过自己动手实验，我真切地感受到不同可视化方法的异同点，每一种图形都有自己的着重点和功能；将数据可视化后，可以直观地看到原始数据的分布情况和噪声的影响，接着利用数据去噪方法去除噪声，提高数据质量，便于后续预测模型的建立。

在聚类分析部分，我自然而然地联想到了大数据课程中的工况识别内容，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。

在回归分析部分，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，然后使用最小二乘、BP 神经网络、SVM 和 RF 分别建立预测模型，并对模型的性能进行评估。 最后通过对比分析得出 BP 神经网络和 SVR 的表现都不错，R 方指标达到 0.8，RMSE 在 0.4 左右。

综上所述，实验内容包括了大数据课程几乎所有重点知识，对数据挖掘知识的理解和应用有很大的帮助。

7.2 感悟

丁老师指导的大数据实践的风电功率预测实验让我受益匪浅，我在上一阶段的基础上学到了更多的大数据方法。

在内容上，实验刚开始的时候老师就在线上详细介绍了实验的内容和要点，两位助教分别演示了 Matlab 和 Python 上的实验流程，并在指导书中清晰写明，非常直观的让我知道自己做的是哪一块的哪一部分。之前做别的实验时，总是感觉与讲课内容关系不是很密切，但是，本实验内容与理论部分关系非常密切，从数据预处理到特征提取再到模型建立，这都是在之前的理论课中讲述过的内容，并不会感到陌生，从认真听到动手做，这门实践课真正的做到了理论与实验的结合，也让我切身感受到了动手敲代码的重要性，学习算法又怎么能缺少动手实践呢。

实验中我印象深刻的有两部分，第一部分是数据预处理，之前我在网络上看到一句话说“大数据分析的大部分时间花在数据预处理上”，当时我还诧异为什么看似最简单的数据处理却需要这么多时间，直到自己动手操作才真正体会到数据预处理的重要性，数据类型多样，噪声类型不一，选择什么去噪方法和归一化方法是非常重要的。如果数据没有处理好，会严重影响后续预测模型的准确率。第二部分是预测模型的建立，回归分析实验中我刚开始用了 Matlab 的工具箱，虽然方便但是模型的性能并不好，工具箱的有些模型也不能用网格搜索寻找最优超参数，于是我重新在 jupyter notebook 中写了 SVR 和 RF 的回归预测模型，使用了课内学过的交叉验证和网格搜索，不过期间我也深刻体会到电脑配置在大数据分析中的重要性，我的笔记本电脑带不动大范围和网格搜索，只能一小段的去搜，因此最后只能得出相对较优的超参数。

在指导上，丁老师和助教们做实验时都会亲自到实验室答疑，认真严谨的工作态度也感染了我，我不再摸鱼划水，而是专注实验，争取高效完成；此外，面对我们的各种疑问助教们都会耐心解答，并会抛出一些思考性的问题，培养我们的思考能力。大数据理论课程和实践课程让我对数据挖掘产生了很大的兴趣，看到自己的模型一步一步使预测值接近真实值是非常有趣且有成就感的！

伴随着实践课的结束，我对所学知识也有了更深入的理解。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，带给我启迪！

附录

任务 1：数据预处理实验程序（Matlab）

```
%初始化
clc
clear
%% 处理重复值和缺失值
load data.mat
% 提取第一列
first_column = data(:, 1);

% 找到唯一的值及其索引，'first' 表示保留第一次出现的行
[~, unique_indices] = unique(first_column, 'first');

% 排序索引以保持原始顺序
unique_indices = sort(unique_indices);

% 得到处理后的数据
data_process = data(unique_indices, :);

% 查找缺失值
missing_values = isnan(data_process);

% 检查是否存在缺失值
if any(missing_values, 'all')
% 获取缺失值的行和列索引
[row, col] = find(missing_values);
% 显示缺失值的行和列索引
disp('缺失值的行和列索引：');
disp(table(row, col));
else
% 输出无缺失值的消息
disp('无缺失值');
end
```

```
%% QQ
% 绘制 22 个过程变量的 QQ 图
figure;
for i = 1:5
subplot(2, 3, i); % 创建 5 行 9 列的子图布局
h = qqplot(data_process(:, (i+1))); % 绘制每一列的 QQ 图
% 调整 QQ 图数据点的大小
set(h, 'MarkerSize', 1); % h(1) 是数据点对象
title(['变量 ', num2str(i)]);
xlabel('理论分位数');
ylabel('样本分位数');
End

%% 数据归一化
% 初始化归一化后的矩阵
z_score_normalized_data = zeros(size(data_process));

% 对每一列分别进行 Z 分数归一化
for col = 1:size(data_process, 2)
col_data = data_process(:, col);
mean_val = mean(col_data);
std_val = std(col_data);
z_score_normalized_data(:, col) = (col_data - mean_val) / std_val;
end
%% 将数据集划分为训练集和测试集
% 计算数据集的大小
num_samples = size(z_score_normalized_data, 1);

% 计算训练集的大小（80%）
num_train = round(0.8 * num_samples);

% 划分训练集和测试集
train_data = z_score_normalized_data(1:num_train, :);
test_data = z_score_normalized_data(num_train+1:end, :);
```

```
%% % 选择特征进行 DBSCAN 聚类去噪（风速和功率）
feature_1 = train_data(:, 2);
feature_2 = train_data(:, 7);
features = [feature_1,feature_2];

% 设置 DBSCAN 的参数
epsilon = 0.2; % 邻域半径
minPts = 150; % 最小点数

% 使用 DBSCAN 进行聚类
clusterIdx = dbscan(features, epsilon, minPts);

% 获取正常点和噪声点
normalPoints = features(clusterIdx ~= -1, :);
noisePoints = features(clusterIdx == -1, :);

% 去除噪声点后的完整数据
cleanedData = train_data(clusterIdx ~= -1, :);

% 绘制去噪前后的数据
figure();
% 去噪前的散点图（包括正常点和噪声点）
scatter(features(:, 1), features(:, 2), 'b'); % 所有点
hold on;
scatter(noisePoints(:, 1), noisePoints(:, 2), 'r', 'filled'); % 噪声点
title('原始数据');
xlabel('X 轴');
ylabel('Y 轴');
legend('正常点', '噪声点');
grid on;
hold off;

% 去噪后的散点图（仅正常点）
figure();
```

```
grid on;
scatter(normalPoints(:, 1), normalPoints(:, 2), 'b'); % 正常点
grid on;
title('去噪后的数据');
xlabel('X 轴');
ylabel('Y 轴');
legend('正常点');
%% 反归一化（用于绘图，与去噪前进行对比，jupyter notebook）
% 原始数据的均值和标准差
meanVals = [mean(data_process(:, 1)), mean(data_process(:, 2)),
mean(data_process(:, 3)), mean(data_process(:, 4)), mean(data_process(:, 5)),
mean(data_process(:, 6)),mean(data_process(:, 7))];
stdVals = [std(data_process(:, 1)), std(data_process(:, 2)),
std(data_process(:, 3)), std(data_process(:, 4)), std(data_process(:, 5)),
std(data_process(:, 6)),std(data_process(:, 7))];

% 反归一化
originalData = cleanedData .* stdVals + meanVals;
```

任务 1：可视化仿真实验程序（Python）

```
### 导入第三方库
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from math import pi
import windrose
from windrose import WindroseAxes, WindAxes, plot_windrose
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
import matplotlib.cm as cm
from sklearn import metrics
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans
import seaborn as sns
# %config InlineBackend.figure_format = 'svg'
```

```
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import warnings
warnings.filterwarnings("ignore")
from scipy.spatial.distance import cdist
from sklearn.model_selection import KFold
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

##可视化原数据集数据
data = pd.read_excel(io=r"D:\桌面\预处理后的数据集.xls")
df = pd.DataFrame(data) #表格型数据结构
print(np.any(df.isna())) #是否存在缺失值
df = df.dropna() ##删掉含缺失值的样本点
display(df)
plt.style.use('seaborn')
#风速频率直方图
plt.figure(figsize=(20,10),dpi = 100)
plt.subplot(231)
plt.xlabel("wind speed")
plt.ylabel("frequency")
plt.hist(df['WINDSPEED'], color='green', alpha=0.5, edgecolor='black')
#风向频率直方图
plt.subplot(232)
plt.xlabel("wind direction")
plt.ylabel("frequency")
plt.hist(df['WINDDIRECTION'], color='red', alpha=0.5, edgecolor='black')
#温度频率直方图
plt.subplot(233)
plt.xlabel("temperature")
```

```
plt.ylabel("frequency")
plt.hist(df['TEMPERATURE'], color='blue', alpha=0.5, edgecolor='black')
#湿度频率直方图
plt.subplot(234)
plt.xlabel("humidity")
plt.ylabel("frequency")
plt.hist(df['HUMIDITY'], color='yellow', alpha=0.5, edgecolor='black')
#plt.subplots_adjust(wspace=1.5, hspace=1.5)
#气压频率直方图
plt.subplot(235)
plt.xlabel("pressure")
plt.ylabel("frequency")
plt.hist(df['PRESSURE'], color='pink', alpha=0.5, edgecolor='black')
#plt.subplots_adjust(wspace=1.5, hspace=1.5)
plt.show()
#玫瑰图
ax = WindroseAxes.from_ax()
#ax.figure.set_dpi(300)
ax.bar(df['WINDDIRECTION'], df['WINDSPEED'], normed=True, opening=0.8,
cmap=cm.Set2)
ax.set_legend(loc='right')
plt.show()
#统计特征
print("1.功率的统计特征：")
#均值
mean_power = np.mean(df['WINDPOWER'])
print ("均值  =%10.3f" %mean_power)
#极差
ptp_power = df['WINDPOWER'].max() - df['WINDPOWER'].min()
print ("极差  =%10.3f" %ptp_power)
#标准差
std_power = np.std(df['WINDPOWER'])
print ("标准差  =%10.3f" %std_power)
print('')
```

```
print("2.风速的统计特征：")
#均值
mean_fs = np.mean(df['WINDSPEED'])
print ("均值  =%10.3f" %mean_fs)
#极差
ptp_fs = df['WINDSPEED'].max() - df['WINDSPEED'].min()
print ("极差  =%10.3f" %ptp_fs)
#标准差
std_fs = np.std(df['WINDSPEED'])
print ("标准差  =%10.3f" %std_fs)
print('')

print("3.风向的统计特征：")
#均值
mean_fx = np.mean(df['WINDDIRECTION'])
print ("均值  =%10.3f" %mean_fx)
#极差
ptp_fx = df['WINDDIRECTION'].max() - df['WINDDIRECTION'].min()
print ("极差  =%10.3f" %ptp_fx)
#标准差
std_fx = np.std(df['WINDDIRECTION'])
print ("标准差  =%10.3f" %std_fx)
print('')

print("4.温度的统计特征：")
#均值
mean_tem = np.mean(df['TEMPERATURE'])
print ("均值  =%10.3f" %mean_tem)
#极差
ptp_tem = df['TEMPERATURE'].max() - df['TEMPERATURE'].min()
print ("极差  =%10.3f" %ptp_tem)
#标准差
std_tem = np.std(df['TEMPERATURE'])
```

```
print ("标准差 =%10.3f" %std_tem)
print('')

print("5.湿度的统计特征：")
#均值
mean_hum = np.mean(df['HUMIDITY'])
print ("均值 =%10.3f" %mean_hum)
#极差
ptp_hum = df['HUMIDITY'].max() - df['HUMIDITY'].min()
print ("极差 =%10.3f" %ptp_hum)
#标准差
std_hum = np.std(df['HUMIDITY'])
print ("标准差 =%10.3f" %std_hum)
print('')

print("6.气压的统计特征：")
#均值
mean_pre = np.mean(df['PRESSURE'])
print ("均值 =%10.3f" %mean_pre)
#极差
ptp_pre = df['PRESSURE'].max() - df['PRESSURE'].min()
print ("极差 =%10.3f" %ptp_pre)
#标准差
std_pre = np.std(df['PRESSURE'])
print ("标准差 =%10.3f" %std_pre)
print('')
# 绘制风速-功率散点图
plt.figure(figsize=(20,10))
plt.subplot(231)
plt.xlabel("wind speed")
plt.ylabel("power")
plt.scatter(df['WINDSPEED'], df['WINDPOWER'], color='blue' , marker='.',s=5)
#绘制风向-功率散点图
plt.subplot(232)
```

```
plt.xlabel("wind direction")
plt.ylabel("power")
plt.scatter(df['WINDDIRECTION'], df['WINDPOWER'], color='green' ,
marker='.',s=5)
plt.subplots_adjust(wspace=0.6, hspace=0.6)
#绘制温度-功率散点图
plt.subplot(233)
plt.xlabel("temperature")
plt.ylabel("power")
plt.scatter(df['TEMPERATURE'], df['WINDPOWER'], color='yellow' ,
marker='.',s=5)
plt.subplots_adjust(wspace=0.6, hspace=0.6)
#绘制湿度-功率散点图
plt.subplot(234)
plt.xlabel("humidity")
plt.ylabel("power")
plt.scatter(df['HUMIDITY'], df['WINDPOWER'], color='green' , marker='.',s=5)
plt.subplots_adjust(wspace=0.6, hspace=0.6)
#绘制气压-功率散点图
plt.subplot(235)
plt.xlabel("pressure")
plt.ylabel("power")
plt.scatter(df['PRESSURE'], df['WINDPOWER'], color='green' , marker='.',s=5)
plt.subplots_adjust(wspace=0.2, hspace=0.2)
plt.show()
#箱线图
plt.figure(figsize=(20, 10))

plt.subplot(231)
plt.boxplot(df['WINDSPEED'])
plt.xlabel('wind speed')
plt.ylabel('Values')

plt.subplot(232)
```

```
plt.boxplot(df['WINDDIRECTION'])
plt.xlabel('wind direction')
plt.ylabel('Values')

plt.subplot(233)
plt.boxplot(df['TEMPERATURE'])
plt.xlabel('temperature')
plt.ylabel('Values')

plt.subplot(234)
plt.boxplot(df['HUMIDITY'])
plt.xlabel('humidity')
plt.ylabel('Values')

plt.subplot(235)
plt.boxplot(df['PRESSURE'])
plt.xlabel('pressure')
plt.ylabel('Values')



#plt.subplots_adjust(wspace=1, hspace=3)
plt.show()
```

任务 2：相关性分析（Python）

```
# 计算皮尔逊相关系数矩阵
correlation_matrix = np.corrcoef(data_array, rowvar=False)

# 设置变量名
variables = ['WINDSPEED', 'WINDDIRECTION', 'TEMPERATURE', 'HUMIDITY',
'PRESSURE', 'WINDPOWER']

# 画出热力图
plt.figure(figsize=(10, 8),dpi=200)
sns.heatmap(correlation_matrix, annot=True, cmap='Blues', xticklabels=variables,
yticklabels=variables)
```

```
plt.title('Pearson Correlation Coefficient Matrix')
plt.show()
# 计算皮尔逊相关系数矩阵
correlation_matrix_1 = np.corrcoef(data_array[:,[0,1,2,5]], rowvar=False)

# 设置变量名
variables = ['WINDSPEED', 'WINDDIRECTION','TEMPERATURE',
'WINDPOWER']

# 画出热力图
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix_1, annot=True, cmap='Blues', xticklabels=variables,
yticklabels=variables)
plt.title('Pearson Correlation Coefficient Matrix')
plt.show()
```

任务 3：聚类分析（Matlab）

```
%% 聚类分析（K-means）
% 生成示例数据
data_clu = cleanedData(:,[2,3,4]);
power = cleanedData(:,7);
wind_speed = cleanedData(:,2);
wind_direction = cleanedData(:,3);
temperature = cleanedData(:,4);
% 使用肘部法则确定最佳聚类个数
sum_of_squared_distances = [];
for k = 1:10
[idx, ~, sumd] = kmeans(data_clu, k, 'Distance', 'sqeuclidean', 'Replicates',
5, 'Start', 'plus');
sum_of_squared_distances(k) = sum(sumd);
end

% 绘制肘部法则图
figure;
plot(1:10, sum_of_squared_distances, 'bx-');
```

```
xlabel('Number of clusters');
ylabel('Sum of squared distances');
title('Elbow Method for Optimal k');
grid on;

% 使用轮廓系数法确定最佳聚类个数
silhouette_avg = [];
for k = 2:10
idx = kmeans(data_clu, k, 'Distance', 'sqeuclidean', 'Replicates', 5, 'Start',
'plus');
s = silhouette(data_clu, idx);
silhouette_avg(k) = mean(s);
end

% 绘制轮廓系数法图
figure;
plot(2:10, silhouette_avg(2:end), 'bx-');
xlabel('Number of clusters');
ylabel('Average silhouette score');
title('Silhouette Method for Optimal k');
grid on;

% 根据轮廓系数法选择最佳聚类个数
[~, optimal_k] = max(silhouette_avg);

% 使用最佳聚类个数进行 K-means++聚类
[idx, C] = kmeans(data_clu, optimal_k, 'Distance', 'sqeuclidean', 'Replicates',
5, 'Start', 'plus');

% 绘制聚类后的数据
figure;
gscatter(data_clu(:,1),power, idx);
hold on;
xlabel('风速');
```

```
ylabel('功率');
title('K-means++聚类结果')
legend('Cluster1','Cluster2','Cluster3');
grid on;
hold off;

figure;
gscatter(data_clu(:,2),power, idx);
hold on;
xlabel('风向');
ylabel('功率');
title('K-means++聚类结果')
legend('Cluster1','Cluster2','Cluster3');
grid on;
hold off;

figure;
gscatter(data_clu(:,3),power, idx);
hold on;
xlabel('温度');
ylabel('功率');
title('K-means++聚类结果')
legend('Cluster1','Cluster2','Cluster3');
grid on;
hold off;

%绘制不同类别的子图
cc = ['r','g','y','c'];
%风速频率图
figure();
for i = 1:optimal_k
cluster_wind_speed = wind_speed(idx==i);
subplot(optimal_k,1,i);
%绘制直方图
```

```
histogram(cluster_wind_speed,'FaceColor',cc(i));
title('风速-频率分布');
xlabel('风速');
ylabel('频率');
xlim([min(wind_speed) max(wind_speed)]);
grid on;
end
%风向频率图
figure();
for i = 1:optimal_k
cluster_wind_direction = wind_direction(idx==i);
subplot(optimal_k,1,i);
%绘制直方图
histogram(cluster_wind_direction,'FaceColor',cc(i));
title('风向-频率分布');
xlabel('风向');
ylabel('频率');
xlim([min(wind_direction) max(wind_direction)]);
grid on;
end

%温度频率图
figure();
for i = 1:optimal_k
cluster_temperature = temperature(idx==i);
subplot(optimal_k,1,i);
%绘制直方图
histogram(cluster_temperature,'FaceColor',cc(i));
title('温度-频率分布');
xlabel('温度');
ylabel('频率');
xlim([min(temperature) max(temperature)]);
grid on;
end
```

任务 4：回归分析（Matlab，Python）

```
%% 预测
%提取特征
power = cleanedData(:,7);
wind_speed = cleanedData(:,2);
wind_direction = cleanedData(:,3);
temperature = cleanedData(:,4);

%将特征矩阵合并
features =[wind_speed, wind_direction, temperature];

%初始化存储模型的单元数组
models = cell(optimal_k, 1);

%训练每个聚类的 BP 神经网络
for k=1:optimal_k
% 提取属于当前聚类的数据
clusterIndices = (idx==k);
clusterFeatures = features(clusterIndices,:);
clusterPower = power(clusterIndices);
%创建 BP 神经网络
net = feedforwardnet(10);
net.trainParam.showWindow = false;%关闭训练窗口
%训练 BP 神经网络
net =train(net,clusterFeatures',clusterPower');
%存储模型
models{k}=net;

% 创建最小二乘回归模型
lsrModel = fitlm(clusterFeatures, clusterPower); % 使用最小二乘法拟合模型
% 存储最小二乘回归模型
models2{k} = lsrModel;
% 创建支持向量机回归模型
svmModel = fitrsvm(clusterFeatures, clusterPower); % 使用支持向量机回归拟合模
```

```
型
% 存储支持向量机回归模型
models3{k} = svmModel;
end

%使用训练好的 k-means 模型的中心来判断测试数据的类别
test_features = test_data(:,[2,3,4]);
normalize_test_power = test_data(:,7);
test_power = normalize_test_power.*std(data_process(:,
7))+mean(data_process(:, 7));
dists = pdist2(test_features,C);%计算测试数据点到各聚类中心的距离
[~,test_idex] = min(dists,[],2);%找出最近的聚类中心

%使用相应类别的 BP 神经网络进行预测

for k = 1:optimal_k
%提取属于当前类别的数据
clusterIndices = (test_idex == k );
clusterFeatures = test_features(clusterIndices,:);

%使用模型进行预测
if ~isempty(clusterFeatures)
test_predictedPower1(clusterIndices) = models{k}(clusterFeatures')';
test_predictedPower2(clusterIndices) = predict(models2{k}, clusterFeatures);
test_predictedPower3(clusterIndices) = predict(models3{k}, clusterFeatures);
end
end
test_predictedPower1 = test_predictedPower1';
%计算归一化预测误差
normalize_test_predictionError1 = normalize_test_power -
test_predictedPower1;
%计算归一化数据的平均绝对误差
normalize_test_mae1 = mean(abs(normalize_test_predictionError1));
%计算归一化数据的均方根误差
```

```
normalize_test_rmse1 = sqrt(mean(normalize_test_predictionError1 .^2));
%计算归一化数据的决定系数
R = corrcoef(normalize_test_power, test_predictedPower1);
% 相关系数
r = R(1, 2);
% 计算 R²
normalize_test_r2_1 = r^2;
%计算反归一化预测值
test_predictedPower1 = test_predictedPower1.*std(data_process(:,
7))+mean(data_process(:, 7));

%打印归一化数据的模型评估结果
disp("神经网络预测归一化数据的平均绝对误差:");
disp(normalize_test_mae1);
disp("神经网络预测归一化数据的均方根误差:");
disp(normalize_test_rmse1);
disp("神经网络预测归一化数据的 R 方:");
disp(normalize_test_r2_1);

figure;
plot(test_predictedPower1);
hold on;
plot(test_power);
title('BP 神经网络-测试集预测值与实际值对比')
legend('预测值','实际值')
xlabel('样本索引')
ylabel('风电功率(KW)')
%% 最小二乘回归
test_predictedPower2 = test_predictedPower2';
%计算归一化预测误差
normalize_test_predictionError2 = normalize_test_power -
test_predictedPower2;
%计算归一化数据的平均绝对误差
normalize_test_mae2 = mean(abs(normalize_test_predictionError2));
```

```
%计算归一化数据的均方根误差
normalize_test_rmse2 = sqrt(mean(normalize_test_predictionError2 .^2));
%计算归一化数据的决定系数
R_2 = corrcoef(normalize_test_power, test_predictedPower2);
% 相关系数
r_2 = R_2(1, 2);
% 计算 R²
normalize_test_r2_2 = (r_2)^2;
%计算反归一化预测值
test_predictedPower2 = test_predictedPower2.*std(data_process(:,
7))+mean(data_process(:, 7));

%打印归一化数据的模型评估结果
disp("最小二乘法预测归一化数据的平均绝对误差:");
disp(normalize_test_mae2);
disp("最小二乘法预测归一化数据的均方根误差:");
disp(normalize_test_rmse2);
disp("最小二乘法预测归一化数据的 R 方:");
disp(normalize_test_r2_2);

figure;
plot(test_predictedPower2);
hold on;
plot(test_power);
title('最小二乘法-测试集预测值与实际值对比')
legend('预测值','实际值')
xlabel('样本索引')
ylabel('风电功率(KW)')
#风电功率预测模型----SVR
#导入需要的库
import numpy as np
import pandas as pd
from sklearn.svm import SVR
from sklearn.cluster import KMeans
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt

##读取去噪后的数据(归一化)
file_path = r"D:\桌面\风电实验\训练集.xls" #  替换为你的 Excel 文件路径
data = pd.read_excel(file_path)
data_array = data.to_numpy()
data_array.shape

#  划分数据集为 82 开
X = data_array[:, 0:3]    #  特征
y = data_array[:, -1]     #  目标变量
X_train = X[:31059,:]
X_test = X[31059:,:]
y_train = y[:31059]
y_test = y[31059:]

#  聚类
optimal_k = 3    #  假设已经确定最佳聚类个数为 3
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', n_init=10,
random_state=1)
kmeans.fit(X_train)
clusters_train = kmeans.predict(X_train)
clusters_test = kmeans.predict(X_test)

#  为每个簇分别建立 SVR 模型
models = {}
for cluster in range(optimal_k):
    #  筛选出当前簇的数据
    X_train_cluster = X_train[clusters_train == cluster]
    y_train_cluster = y_train[clusters_train == cluster]

    #  训练 SVR 模型
```

```
        svr = SVR(kernel='rbf', C=50, gamma=0.1,epsilon=0.2)
        svr.fit(X_train_cluster, y_train_cluster)
        models[cluster] = svr

# 对测试集进行预测
y_pred = np.zeros_like(y_test)
for i in range(len(X_test)):
        cluster = clusters_test[i]
        svr = models[cluster]
        y_pred[i] = svr.predict([X_test[i]])

# 计算评价指标
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

# 打印结果
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'R² : {r2}')
print(f'Mean Absolute Error (MAE): {mae}')

#反归一化
mean = 16706.7718213679
std_dev = 16123.6643570036

# 测试集实际功率反标准化
original__true_power = y_test * std_dev + mean
# 测试集预测功率反标准化
original__pred_power = y_pred * std_dev + mean

# 设置画图的宽度和高度，单位为英寸
figure_width = 25
```

```
figure_height = 6

# 创建一个 figure
fig = plt.figure(figsize=(figure_width, figure_height))

# 绘制实际值
plt.plot(range(len(original__true_power)), original__true_power, color='red',
label='True',linewidth=1)

# 绘制预测值
plt.plot(range(len(original__pred_power)), original__pred_power, color='blue',
label='Predicted',linewidth=1)

# 添加图例
plt.legend()

# 显示图形
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('True vs Predicted Values with Smooth Curve')
plt.show()

#风电功率预测模型----随机森林回归
#导入需要的库
import numpy as np
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

##读取去噪后的数据(归一化)
file_path = r"D:\桌面\风电实验\训练集.xls" #  替换为你的 Excel 文件路径
```

```
data = pd.read_excel(file_path)
data_array = data.to_numpy()
data_array.shape

# 划分数据集为 82 开
X = data_array[:, 0:3]   # 特征
y = data_array[:, -1]    # 目标变量
X_train = X[:31059,:]
X_test = X[31059:,:]
y_train = y[:31059]
y_test = y[31059:]

# 使用 K-means++在训练集上进行聚类
optimal_k = 3   # 假设已经确定最佳聚类个数为 3
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', n_init=10,
random_state=1)
kmeans.fit(X_train)
clusters_train = kmeans.predict(X_train)
clusters_test = kmeans.predict(X_test)

# 为每个聚类簇建立随机森林回归模型
models = {}
for k in range(optimal_k):
    cluster_indices = np.where(clusters_train == k)
    X_cluster = X_train[cluster_indices]
    y_cluster = y_train[cluster_indices]
    rf = RandomForestRegressor(n_estimators=500, max_depth=100,
min_samples_split=3, random_state=1)
    rf.fit(X_cluster, y_cluster)
    models[k] = rf

# 对测试集进行预测
predictions = np.zeros(len(X_test))
for i in range(len(X_test)):
```

```
        sample = X_test[i].reshape(1, -1)
        cluster = clusters_test[i]
        model = models[cluster]
        predictions[i] = model.predict(sample)

# 计算评价指标
mse = mean_squared_error(y_test, predictions)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, predictions)
mae = mean_absolute_error(y_test, predictions)

# 打印结果
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'R² : {r2}')
print(f'Mean Absolute Error (MAE): {mae}')

#反归一化
mean = 16706.7718213679
std_dev = 16123.6643570036

# 测试集实际功率反标准化
original__true_power = y_test * std_dev + mean
# 测试集预测功率反标准化
original__pred_power = predictions * std_dev + mean

# 设置画图的宽度和高度，单位为英寸
figure_width = 25
figure_height = 6

# 创建一个 figure
fig = plt.figure(figsize=(figure_width, figure_height))

# 绘制实际值
```

```
plt.plot(range(len(original__true_power)), original__true_power, color='red',
label='True',linewidth=1)

# 绘制预测值
plt.plot(range(len(original__pred_power)), original__pred_power, color='blue',
label='Predicted',linewidth=1)

# 添加图例
plt.legend()

# 显示图形
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('True vs Predicted Values with Smooth Curve')
plt.show()
```