

队伍编号	MCB2403056
赛道	(A)

# 基于 RNN 及 LSTM 时间序列模型对台风的分类与预测

## 摘要

在全球气候动态变化的背景下，台风的频率和强度不断增加，随之而来的强风、暴雨和风暴潮等极端天气现象对沿海地区造成的灾害也更加严重。这些台风引发的灾害给沿海地区带来极大的经济损失，并对当地居民的生命和财产安全构成严重威胁。中国由于其特殊而复杂的地理环境，特别容易受到自然灾害的影响。台风灾害尤其突出，不仅发生频次高，且破坏力强，影响范围广泛。频繁的台风灾害加剧了沿海城市的风险与挑战，促使我们进一步探索和预测台风路径和强度，以期降低灾害带来的损失，提高沿海地区的应对能力和灾害防御水平，因此关于台风强度、路径、降水等研究具有显著意义。

**针对问题一：**需要根据台风的特征参数与各种变量的关系进行台风分类。本文首先进行数据预处理，针对特征缺失值的类型在相关气象网站中得到真实值并采用插值与均值填补将特征缺失值补全，之后过滤异常值、转换时间格式、对非数值型数据的编码处理等。为了对数据集有一个整体把握，我们对处理好的数据做了描述性统计进行分析，并给出了分析结果。接着观察数据，对台风强度、气压、风速、海洋温度等特征使用 Spearman 相关性分析探索各变量间的相关程度。针对台风分类问题，本文采用 K-均值算法对台风进行细化聚类，根据聚类结果建立三个不同尺度的分类标准。最终，凭借历史夏台风与秋台风数据给出我国 2024 年 7 月和 9 月台风类别以及途径省份，同时分析夏台风与秋台风的区别。

**针对问题二：**需要建立台风路径预测的模型。首先结合问题一中台风历史数据，将问题划分为：构建台风路径预测模型、带入台风贝碧嘉数据进行台风行进路线预测两方面求解。然后设置滑动窗口对聚类后的各类别序列进行特征提取和样本集构建，采用 RNN 循环神经网络、LSTM 长短时记忆模型分别训练预测模型，基于 GridSearch 方法进行超参数优化并利用 MSE 大小评估算法性能。对比结果表明 RNN 由于内部结构简单，对于路径波动较小的台风有较好的预测效果，对于路径波动较大的预测效果不够好；LSTM 模型更加复杂，但能够以让网络记忆更多历史信息，进而有更好的预测效果。接着依靠搜索气象网信息找到贝碧嘉台风相关数据以此进行路径预测，得到 13 至 17 日 14 时台风中心位置最后运用 DTW 算法与台风实际行进路线进行对比。

**针对问题三：**需要台风登陆后行进过程中降水量及风速的关系，及降水量与距台风中心距离的关系。本文第一步首先选取 CHM\_PRE 文件中所有的降水量场数据并加以处理，然后计算每天的平均降水量和风速，对降水量与风速进行回归分析，发现两者不存在明显的正负相关性，与我们的常识相符，接着利用随机森林回归模型对两者进行分析，得到降水量与风速的关系。第二步将问题一中处理好的台风数据与降水量场数据对应，利用大圆距离的 Haversine 公式计算台风与监测站点间距离同时匹配降水量数据，接着利用随机森林回归模型求得降水量与台风距离的关系。最后反向代入贝碧嘉台风数据预测贝碧嘉行进途中的中心风力及降水量信息。

**关键词：**Spearman 相关系数，K-均值聚类，RNN 神经网络，LSTM，随机森林回归

# 目 录

<b>一、问题重述 .....</b>	<b>1</b>
<b>1.1. 问题背景 .....</b>	<b>1</b>
<b>1.2. 问题提出 .....</b>	<b>1</b>
<b>二、问题分析 .....</b>	<b>2</b>
<b>2.1. 问题一的分析 .....</b>	<b>2</b>
<b>2.2. 问题二的分析 .....</b>	<b>2</b>
<b>2.3. 问题三的分析 .....</b>	<b>2</b>
<b>三、模型假设 .....</b>	<b>2</b>
<b>四、符号说明 .....</b>	<b>3</b>
<b>五、数据预处理 .....</b>	<b>3</b>
<b>5.1. 异常值处理 .....</b>	<b>3</b>
<b>5.2. 数据编码处理 .....</b>	<b>4</b>
<b>5.3. 时间格式处理 .....</b>	<b>4</b>
<b>5.4. 缺失值处理 .....</b>	<b>4</b>
<b>六、模型的建立与求解 .....</b>	<b>5</b>
<b>6.1. 问题一模型建立与求解 .....</b>	<b>5</b>
6.1.1. 台风强度演化的时间特征 .....	5
6.1.2. 台风特征参数与气象因素的关系 .....	7
6.1.3. 分类评价模型的建立 .....	10
6.1.4. 聚类结果 .....	11
6.1.5. 对七月夏台风和九月秋台风的强度与路径研究 .....	12
<b>6.2. 问题二：台风路径预测模型 .....</b>	<b>17</b>
6.2.1. RNN 预测模型构建与求解 .....	19
6.2.2. LSTM 预测模型构建与求解 .....	21
6.2.3. 任务预测与性能评价 .....	22
<b>6.3. 问题三：台风登陆后对降水量和风速的影响 .....</b>	<b>27</b>
6.3.1. 降水量与风速的关系模型 .....	27
6.3.2. 降水量与距台风中心距离的关系模型 .....	28
<b>七、模型的评价、与推广 .....</b>	<b>29</b>
<b>7.1. 模型优点 .....</b>	<b>29</b>
<b>7.2. 模型缺点 .....</b>	<b>30</b>
<b>7.3. 模型改进与推广 .....</b>	<b>30</b>
<b>八、参考文献 .....</b>	<b>30</b>
<b>附录 .....</b>	<b>31</b>

## 一、问题重述

### 1.1. 问题背景

台风是与人类生活和生产关系密切的降雨系统，能够带来降水。然而，台风容易造成多种灾害，例如：狂风、暴雨、风暴潮、泥石流、生态破坏、疫病流行等，且具有突发性强、破坏力大的特点，成为世界上最严重的自然灾害之一，每年我国东南沿海地区都受到台风灾害的严重影响。而台风的成因，至今仍无法准确确定，仅已知它是由热带大气内的扰动发展而来的。形成台风的众多影响因素，使得台风运动轨迹极其复杂，另外，台风的路径有时还会受到其他台风的影响，出现打转、停滞的现象。因此，根据台风的特征参数与气温、气压、季风、洋流等的关系，建立恰当的台风分类评价模型、台风路径预测模型和台风登陆后的风速与降水量预测模型，进行不同特征的台风类别划分，准确对台风进行分类并预测台风路径和台风登陆后对风速与降水量的影响。这些模型的应用不仅有助于提升对台风的理解，也为台风的预测与防范提供了有力工具，对我国防震减灾意义重大[1]。

### 1.2. 问题提出

基于前述研究背景，本题可以分为四个大部分，分别为数据处理、台风分类评价模型、台风路径预测模型、台风登陆后的风速与降水量、降水量与距台风中心距离的关系关系模型。拟要解决的问题如下：

#### 问题一：

- (1) 分析台风特征参数与气象因素的关系，建立台风的分类评价模型，明确类别划分的标准或评价算法，进行不同特征的台风类别划分及分类评价；
- (2) 根据所建立的分类评价模型，以 2024 年我国 7 月和 9 月为例，给出台风类别及途经省份的列表，并分析夏台风与秋台风的区别。

#### 问题二：

- (1) 根据气温、气压、洋流、风场等多种因素，建立台风路径预测模型；
- (2) 以 2024 年 9 月 13 日—17 日每日 14 点的第 13 号台风贝碧嘉为例，预测行进路线，运用 Dynamic Time Warping(DTW) 动态时间规整算法与台风实际行进路线进行对比，把预测结果填到表 2 中，并放于论文正文。

#### 问题三：

- (1) 建立台风在登陆后的行进过程中降水量及风速的关系，及降水量与距台风中心距离的关系；
- (2) 以 2024 年 9 月 16 日—18 日的第 13 号台风贝碧嘉为例，根据所建立的模型，预测贝碧嘉行进途中的中心风力及降水量，并进行分析。

## 二、问题分析

### 2.1. 问题一的分析

针对问题一，我们的目标是通过研究台风气象特征与气温、气压等因素关系构建台风的分类评价模型，随后根据模型对我国7月夏台风与9月秋台风进行分类、得到途径省份，最后描述夏台风与秋台风的区别。因此首先进行数据预处理，针对特征缺失值的类型在相关气象网站中得到真实值并采用插值与均值填补将特征缺失值补全，之后过滤异常值、转换时间格式、对非数值型数据的编码处理等。随后我们对处理好的数据做了描述性统计进行分析，找到。接着观察数据，对台风强度、气压、风速、海洋温度等特征使用 Spearman 相关性分析探索各变量间的相关程度。针对台风分类问题，采用 K-means 算法对台风进行细化聚类，根据聚类结果建立三个不同尺度的分类标准将台风进行分类并以此为依据对我国7月夏台风与9月秋台风的通过路径的统计分析得到对应台风途径省份和两种台风的区别。

### 2.2. 问题二的分析

针对问题二，我们的目标是根据气温、洋流、气压等因素建议台风路径预测模型，并以台风贝碧嘉为例预测行进路线，同时利用 DTW 算法与台风实际行进路线对比。首先设置时间步长对聚类后的各类别序列进行特征提取和样本集构建，采用 RNN 循环神经网络、LSTM 长短时记忆模型分别训练预测模型，随后进行超参数优化并利用 MSE 大小评估算法性能。结果表明 RNN 由于内部结构简单，对于路径波动较小的台风有较好的预测效果，对于路径波动较大的预测效果不够好；LSTM 模型更加复杂，但能够以让网络记忆更多历史信息，进而有更好的预测效果。接着搜索气象网信息取得贝碧嘉相关数据进行路径预测13至17日14时台风中心位置最后运用 DTW 算法得到路径对比结果。

### 2.3. 问题三的分析

针对问题三，我们的目标是建立台风在登陆后风速和降水量之间的关系，以及降水量与距台风中心距离之间的关系。首先选取 CHM\_PRE 文件中所有的降水量场数据并加以处理，然后计算每天的平均降水量和风速，对降水量与风速进行回归分析，发现两者不存在明显的正负相关性，与我们的常识相符，接着利用随机森林回归模型对两者进行分析，得到降水量与风速的关系。接下来将问题一中处理好的台风数据与降水量场数据对应，利用大圆距离的 Haversine 公式计算台风与监测站点间距离同时匹配降水量数据，接着利用随机森林回归模型求得降水量与台风距离的关系。最后反向代入贝碧嘉台风数据预测贝碧嘉行进途中的中心风力及降水量信息。

## 三、模型假设

1、假设数据集中所有的数据都真实有效；

- 2、假设台风的路径预测和强度预测相互独立，路径的变化不会显著影响台风强度，反之同样成立
- 3、在短时间跨度内，台风的强度、风速、气压等特征不会发生剧烈波动，即台风特征的变化是平稳且渐进的。
- 4、过去数年台风的路径和强度分布规律在当前仍然适用，因此历史数据可用于当前的模型训练和预测。

#### 四、符号说明

为了方便我们模型的建立与求解过程，我们这里对使用到的关键符合进行以下说明：

符号	说明
$\rho$	Spearman 相关系数
$Q_t$	$t$ 时刻的输出层输出
$s_t$	$t$ 时刻的隐藏状态值
$x_t$	$t$ 时刻的输入值
...	...
$D_i$	目标台风与第 $i$ 条台风的动态规整距离
$T_i(x)$	第 $i$ 棵树对输入 $x$ 的预测

#### 五、数据预处理

对于题目给出的数据一共提供了 1945 年至今 1800 多场台风的相关数据。我们需要对题目给出的数据进行必要的处理。主要包括：异常值处理、数据编码处理、时间格式处理、缺失值处理、数据描述性分析。

##### 5.1. 异常值处理

台风强度的衡量标准多采用 2 分钟持续最大风速，按照台风中心最大风力，台风强度等级划分。2006 年，我国发布了《热带气旋等级》（GB/T19201-2006），将热带气旋划分为热带低压、热带风暴、强热带风暴、台风、强台风和超强台风 6 个等级，如表 1 所示：。在中央气象台现行标准中，热带风暴、强热带风暴、台风、强台风和超强台风通常统称为“台风”。根据风力的不同强度，这些台风被分别划分为热带风暴级、强热带风暴级、台风级、强台风级和超强台风级。

台风等级	风速数值
热带低压(TD)	10.8m/s - 17.1m/s
热带风暴(TS)	17.2m/s - 24.4m/s
强热带风暴(STS)	24.5m/s - 32.6m/s
台风(TY)	32.7m/s - 41.4m/s

强台风(STY)	41.5m/s - 50.0m/s
超强台风(SuperTY)	$\geq 50.1\text{m/s}$

剔除台风经纬度不在正常值范围内的数据，以及气压小于 100，风速大于 100 的异常数据值。

## 5.2. 数据编码处理

对非数值型数据的编码处理：对于题目给出的移动方向数据，存在直接使用汉字和字母进行编码的结果，需要转化为数据才能进一步建立模型。我们创建字符到数值的映射，将东(E)、南(S)、西(W)、北(N)分别数值映射为 1、2、3、4，并进行逐个字符转换，忽略掉“偏”字；将弱热带低压或未知、热带低压(TD)、热带风暴(TS)、强热带风暴(STS)、台风(TY)、强台风(STY)、超强台风(SuperTY)分别数值映射为 0、1、2、3、4、5、6。

## 5.3. 时间格式处理

时间格式的转换：对于题目给出的数据格式，直接运行计算机无法识别，我们需要根据数据特征六小时间隔转化为 YYYYMMDDHH 格式，并提取年份和月份，方便进一步进行处理。

## 5.4. 缺失值处理

对缺失值的插值填充：对于题目存在的大量缺失值，我们可以选择插值填充，也可以选择直接删除所在行。如表 2 为我们处理后的台风数据格式。

表 2 台风数据处理后格式

台风编 号	格式化时间	台风 强度	台风 等级	风速	气压	移动 方向	移动 速度	年份	月份	调整 后的 经度	调整 后的 纬度
202102	2021042408	2	8	20	995	112	17	2021	4	131.5	23.5
202102	2021042411	2	8	20	995	112	20	2021	4	131.5	23.5
202102	2021042414	2	8	18	998	112	20	2021	4	131.5	22.5
202102	2021042417	2	8	18	998	112	23	2021	4	132.5	22.5
202101	2021021814	2	8	18	998	3	8	2021	2	132.5	7.5
202101	2021021817	2	8	18	998	3	8	2021	2	132.5	7.5

## 六、模型的建立与求解

### 6.1. 问题一模型建立与求解

#### 6.1.1. 台风强度演化的时间特征

为了探索热带气旋逐年发生频数和热带气旋逐年发生频数变化趋势，基于 Python 调用 pandas 程序包以读取经过排序处理和数据滤波处理的台风数据，按照台风年份的强度等级展示逐年不同强度的热带气旋频数如表 3 所示：

year	TS	STS	TY	STY	SSTY	all
1949	2	9	5	8	2	26
1950	4	13	8	2	5	32
1951	1	4	6	3	6	20
1952	3	7	9	5	7	31
1953	6	3	3	6	8	26
...	...	...	...	...	...	...
2017	10	5	4	4	4	27
2018	6	8	3	8	8	29
2019	8	4	5	6	6	29
2020	5	5	5	5	3	23
2021	7	4	5	1	5	22

由于我们研究的时间序列年份跨度较大并且数据较多，接下来采用描述统计分析以统计概述的方式来展示不同强度的热带气旋频数的集中趋势和离散趋势如表 4 所示：

	TS	STS	TY	STY	SSTY	all
count	73.000000	73.000000	73.000000	73.000000	73.000000	73.000000
mean	4.150685	5.986301	5.657534	4.575342	5.917808	26.287671
std	2.004656	2.756271	2.987098	2.266408	3.323959	4.774002
min	1.000000	1.000000	0.000000	0.000000	0.000000	13.000000
25%	3.000000	4.000000	4.000000	3.000000	4.000000	23.000000
50%	4.000000	6.000000	5.000000	4.000000	5.000000	26.000000
75%	5.000000	8.000000	8.000000	6.000000	8.000000	29.000000
max	10.000000	15.000000	12.000000	12.000000	15.000000	39.000000

表 4 不同强度的热带气旋频数的集中趋势和离散趋势

根据不同强度的热带气旋频数的集中趋势和离散趋势分析可知，从 1949 年到 2021 年，西北太平洋海域的热带气旋发生频率虽然有波动，但是其总体趋势呈现平稳状态。具体可以说西北太平洋热带气旋发生频数的均值和中位数为 26，均值和中位数数据较大表明台风强度和路径演化研究意义显著；离散波动的标准差为 5，离散度较小表明以历史趋势预报未来目标状态是可信的。

此外，通过对不同强度热带气旋的横向对比分析得出，不同强度等级的热带气旋年均发生频数约为 5，因此，年内热带气旋的强度分布较为均匀。对某一强度等级的热带气旋进行描述性统计分析表明，相较于热带风暴、强热带风暴、台风和强台风，超强台

风的波动幅度更大，具体而言，超强台风年发生频率最多可达 15 次，最少则为 0 次，这表明超强台风的出现具有较高偶然性。此外，根据实际经验可知，超强台风通常伴有较大破坏力，因此在对台风强度及路径变化的预测中需更多地关注超强台风的演化情况。

为了更细致地了解不同强度热带气旋频数的年际变化，本文采用折线图的形式，直观展示出不同强度热带气旋频数的逐年变化情况。如图 5：

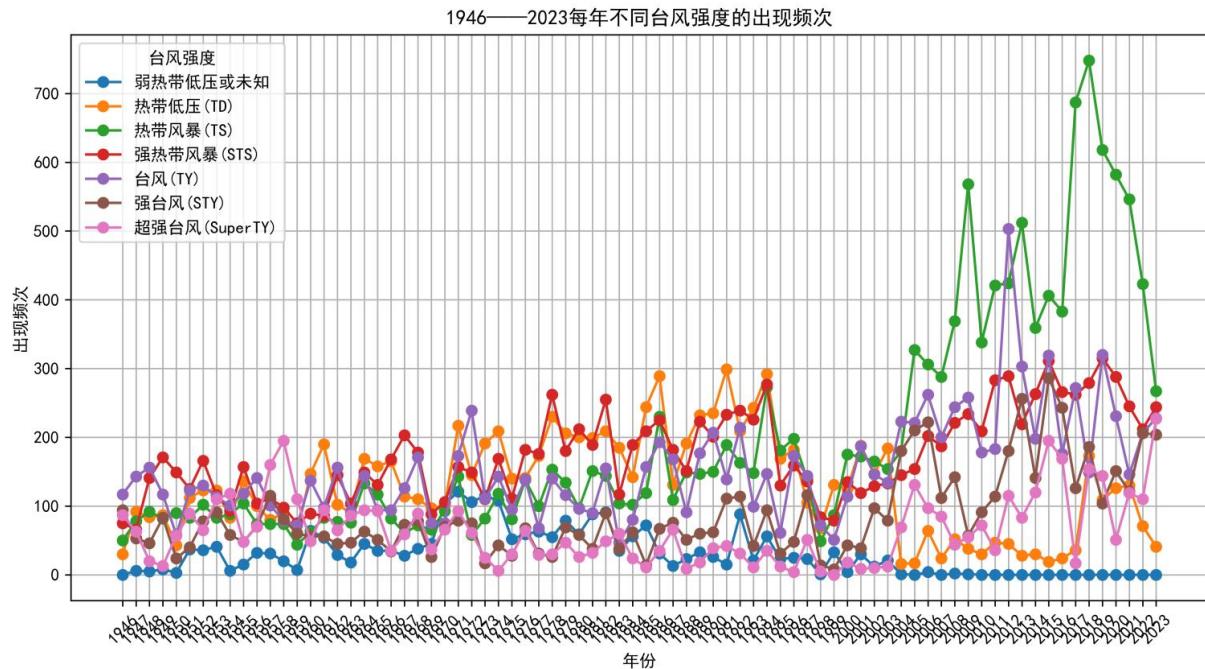


图 1 不同强度热带气旋频数的逐年变化情况

1946 – 2023 年间台风总数变化趋势

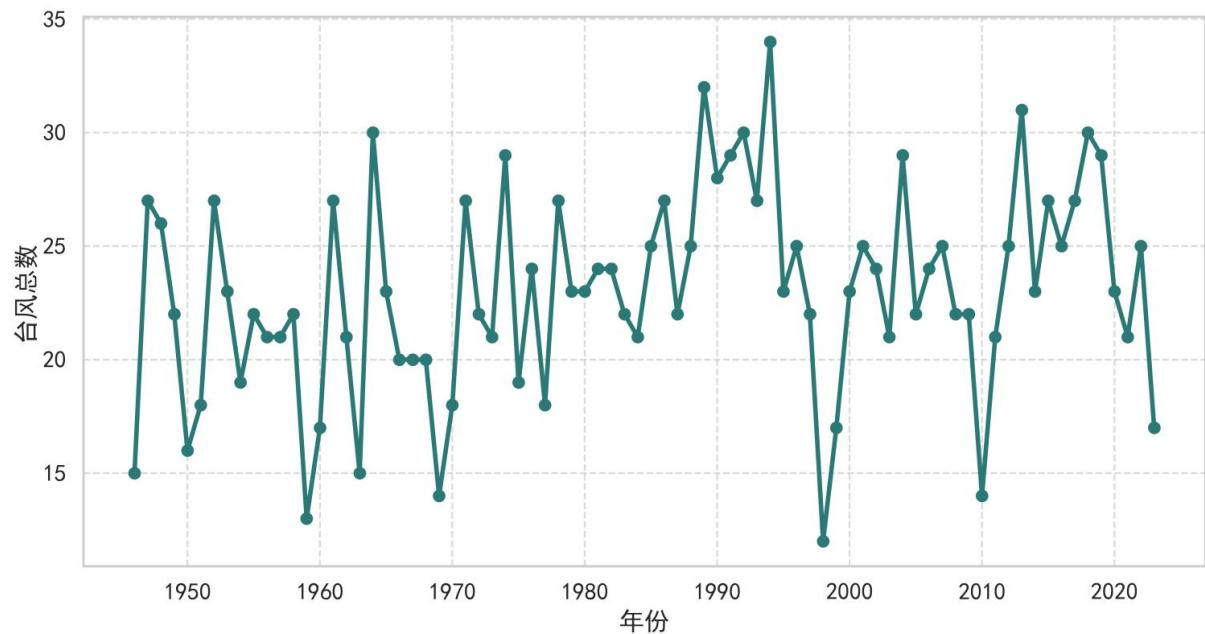


图 2 1946 – 2023 年台风总数变化趋势

不同强度热带气旋频数的逐年变化折线图直观展示了数据描述与统计分析所揭示的热带气旋年际发生频率的演变规律。

### 6.1.2. 台风特征参数与气象因素的关系

台风强度和路径演化的预报因子具有不同的单位和数据量级，因此预测结果容易受到量纲影响，导致各预报因子的权重失真，从而使预报结果偏离真实值。此外，通过对原始数据进行规范化处理，即去除量纲，可以缩小预报因子之间的绝对值差异，使最优解的搜索过程更加平滑，同时加快梯度下降算法的迭代更新速度。此外，线性去量纲操作不会改变原始数据的排序，因此对深度学习的学习效果不会造成干扰，实际上，该数据处理操作还可以提升数据的表现力。

为了直观展示数据分布及变量之间的关联关系，首先对原始数据进行了无量纲化的预处理；随后，绘制了变量的散点图矩阵，如图 3 所示：

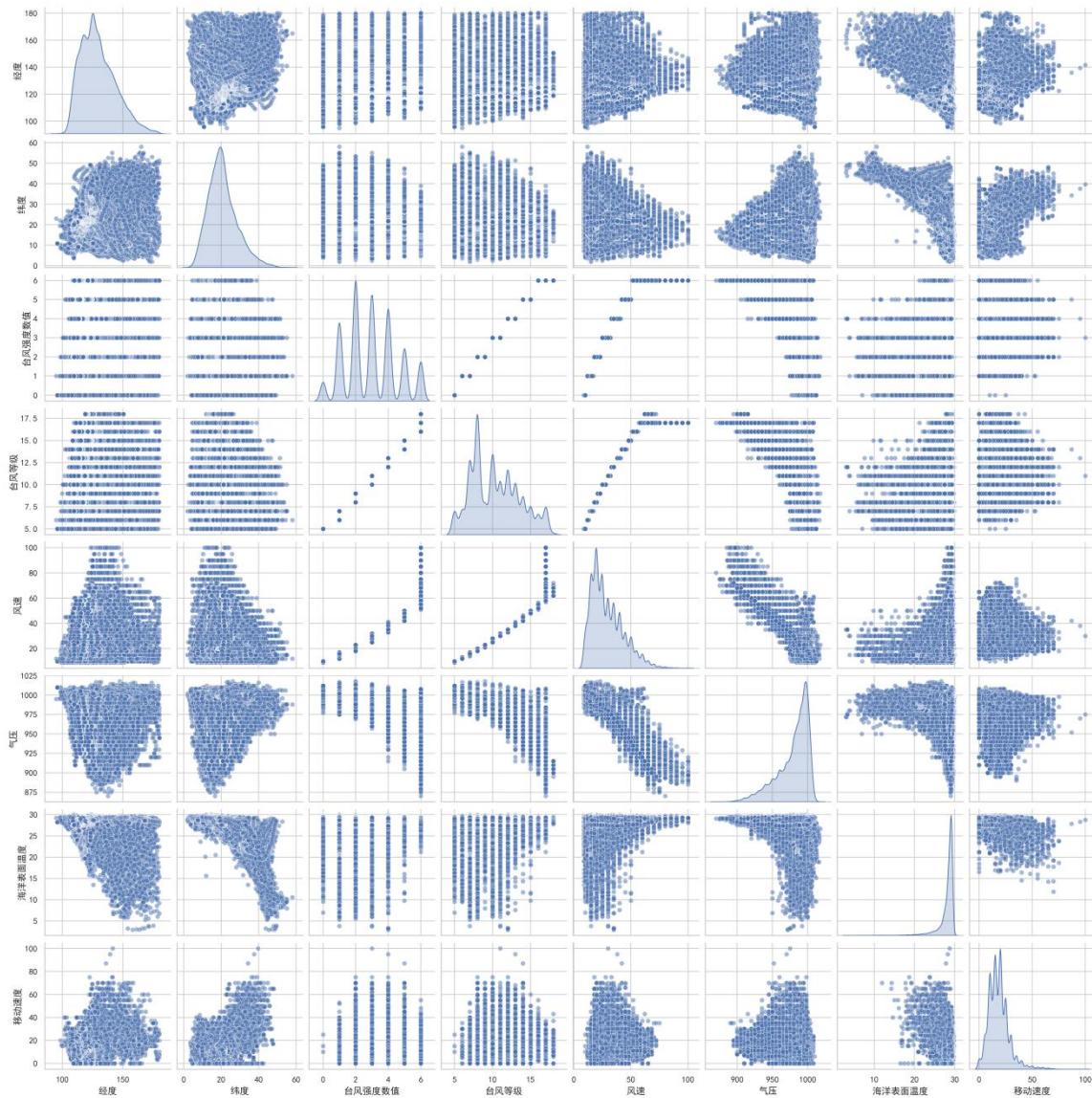


图 3 变量的散点图矩阵

变量的散点图直观展示了台风中心最低气压）、台风强度数值和台风轨迹中心风速之间存在关联关系，而其他变量之间的关联关系则不明显。

根据题目提供数据，台风特征信息主要为台风强度、台风等级、风速、气压、移动方向和移动速度。首先，对台风特征参数与气温、气压进行相关性分析，了解各个指标之间的关系。考虑到台风特征数据(如台风强度、风速、气压等)不符合正态分布，我们

这里使用 Spearman 相关系数来分析变量之间的关系，Spearman 相关系数是一种非参数的相关性度量方法，用于衡量两个变量之间的单调关系，适合处理非正态分布的数据。  
**Spearman 相关系数：**

用于度量两个变量之间的单调相关性，定义为两个变量的秩值之间的皮尔逊相关系数。假设我们有两个变量 X 和 Y，Spearman 相关系数  $\rho$  计算如下：

- (1) 数据排序：对两个变量的数据进行排序，得到其秩次  $R(X)$  和  $R(Y)$ ，分别表示 X 和 Y 的秩序统计量。
- (2) 计算相关系数：

$$\rho = 1 - \frac{6 \sum_{i=1}^n (R(X_i) - R(Y_i))^2}{n(n^2 - 1)}$$

其中  $R(X)$  和  $R(Y)$  分别是变量 X 和 Y 的秩值， $n$  为数据对的数量。

相关性系数的范围为  $[-1, 1]$ ，其中：

- $\rho > 0$  表示正相关，即一个变量增加，另一个变量也倾向于增加；
- $\rho \leq 0$  表示负相关，即一个变量增加，另一个变量也倾向于减少；
- $|\rho|$  的绝对值越大，表示相关性越强。

**计算相关性矩阵：**

在本研究中，我们计算台风特征变量间的 Spearman 相关系数矩阵。矩阵中的每个元素代表两个特征间的相关性。通过计算特征集合的 Spearman 相关性矩阵，可以显示各变量之间的 Spearman 相关系数。计算各预报因子间和 Spearman 相关系数以进行变量相关性的分析，计算结果如表 5 所示：

Spear man	经度	纬度	台风强 度	台风等 级	风速	气压	海表温 度	移动方 向	移动速 度
经度	1.000 000	0.05 1643	0.0508 49	0.0488 38	0.0548 99	-0.001 031	-0.071 504	0.1145 39	0.1955 66
纬度	0.051 643	1.00 0000	0.1048 20	0.1077 53	0.1073 74	-0.196 521	-0.507 114	0.0802 21	0.2279 13
台风 强度	0.050 849	0.10 4820	1.0000 00	0.9895 38	0.9861 22	-0.921 265	-0.064 170	0.0615 71	0.0256 25
台风 等级	0.048 838	0.10 7753	0.9895 38	1.0000 00	0.9965 38	-0.933 053	-0.064 170	0.0635 24	0.0325 04
风速	0.054 899	0.10 7374	0.9861 22	0.9965 38	1.0000 00	-0.934 019	-0.066 035	0.0625 78	0.0350 01
气压	-0.00 1031	-0.1 9652	-0.921 1	-0.933 053	-0.934 019	1.0000 00	0.0957 86	-0.057 140	-0.033 057
海表 温度	-0.07 1504	-0.0 5071	-0.064 170	-0.064 300	-0.066 035	0.0957 86	1.0000 00	0.0481 23	-0.146 454
移动 方向	0.114 539	0.08 0221	0.0615 71	0.0635 24	0.0625 78	-0.057 140	0.0481 23	1.0000 00	0.0336 05

移动速度	0.195 566	0.22 7913	0.02 5625	0.0325 04	0.03 5001	-0.033 057	-0.146 454	0.0336 05	1.0000 00
------	--------------	--------------	--------------	--------------	--------------	---------------	---------------	--------------	--------------

表 5 Spearman 相关系数

数据表明，台风中心最低气压对台风强度演化预测具有显著的预报效果。因此，基于变量间的关联性分析，最终选取台风强度）、台风轨迹近中心风速）和台风中心气压作为台风强度演化预测模型的输入向量。在台风路径演化方面，台风强度标识、台风轨迹近中心风速和台风中心气压与台风轨迹纬度坐标及经度坐标的 Spearman 相关系数较小，表明几乎不存在线性或非线性关联。这表明上述变量对台风路径演化的预测效果不显著。因此，最终根据变量间的关联性分析，选取台风轨迹纬度和经度作为台风路径演化预测模型的输入向量。

接着，为直观展示变量间的关联性绘制的台风强度和路径 演化相关变量的热力图如图 4 所示

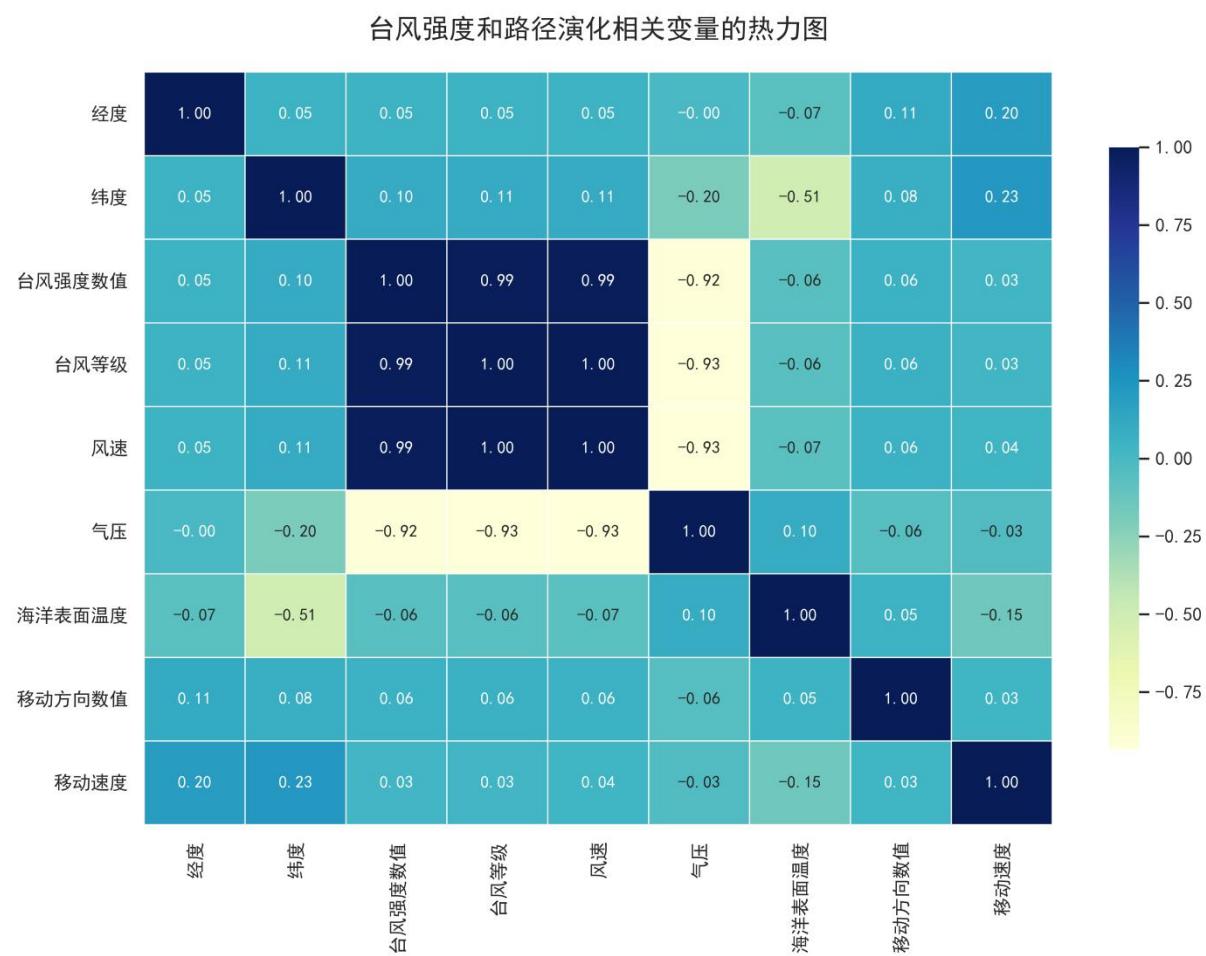


图 4 台风强度和路径演化相关变量的热力图

台风等级与风速的 Spearman 相关系数为 1,这意味着风速越高，台风等级也越高，这种结果符合我们对台风的物理特性理解。风速是影响台风等级的重要因素，台风的破坏力主要由其风速决定，因此两者呈现出如此强的正相关性。此外，气压与风速之间呈现出高度负相关的关系，相关系数为-0.93，这也符合气象学中的常识，即气压越低，风速通常越大。类似的，气压与台风等级也存在高度的负相关关系，相关系数为-0.93，说

明低气压通常意味着较高的台风等级，这一结果也与物理学中台风的形成原理相吻合。

在分析移动速度时，发现其与其他变量的关系同样较弱。移动速度与气压的 Spearman 相关系数为 -0.03，说明虽然气压可能在一定程度上影响台风的移动速度，但这种影响并不显著。此外，移动速度与台风等级的相关性为 0.03、与风速的相关性为 0.04，这些数值都表明移动速度的变化并不能由风速或台风等级单独解释，它可能更多地受到整体大气环流系统的影响，例如副热带高压的强弱和洋流的变化等。

为了更直观地展示各个变量之间的相关性，我们绘制了 Spearman 相关性矩阵的热图。在热图中，颜色的深浅代表相关性的强弱。颜色越接近深红色，代表变量之间的正相关性越强，例如台风等级与风速之间的关系；颜色越接近深蓝色，则代表负相关性越强，例如气压与风速之间的关系；颜色接近浅色或白色则代表相关性较弱，如移动方向与其他变量之间的关系。

综合来看，Spearman 相关性分析结果揭示了台风特征参数与气象因素之间的重要关系。风速、台风等级和气压之间存在显著的相关性，这些变量是台风强度和破坏力的主要决定因素。而移动方向和移动速度与其他变量的相关性较弱，显示这些因素受更复杂的外部条件影响。通过对各变量之间关系的分析，为后续的台风分类和路径预测提供了重要的特征选择依据，帮助我们更好地理解台风的形成和演变机制。

### 6.1.3. 分类评价模型的建立

基于上述分析的结果，采用 K-均值算法对台风进行聚类，因为此时的 k 值不被指定，因此采用肘部法来估计聚类数量。

在聚类的过程中，随着聚类数的增大，样本划分会变得更加精细，每个类别的聚合程度更高，那么误差平方和（SSE）会逐渐变小，误差平方和即该类重心与其内部成员位置距离的平方和。SSE 是手肘法的核心指标，其公式为：

$$SSE = \sum_{i=1}^k \sum_{p \in C_i} |p - m_i|^2$$

其中， $c_i$  是第  $i$  个簇， $p$  是  $c_i$  中的样本点， $m_i$  是  $c_i$  的质心（ $c_i$  中所有样本均值），代表了聚类效果的好坏。当  $k$  小于真实聚类数时，由于  $k$  的增大会增加每个簇的聚合程度，故 SSE 的下降幅度会很大；而当  $k$  到达真实聚类数时，再增加  $k$  所得到的聚合程度回报会迅速变小，所以 SSE 的下降幅度会骤减，然后随着  $k$  值的继续增大而趋于平缓。也就是说 SSE 和  $k$  的关系图是一个手肘的形状，而这个肘部对应的  $k$  值就是数据的真实聚类数。算法执行流程如表 5 所示：

---

#### 算法 1：K-means 聚类算法

---

输入：类别数量  $k$ ，最大迭代次数  $t_{max}$ ，质心变化阈值  $\delta$

选择  $k$  个随机样本作为初始质心

while  $i < t_{max}$  or 质心变化量  $> \delta$

    分配样本到最近的质心，并计算每个聚类的平均值

    更新质心，并计算当前质心与更新后质心的差距

$i = i + 1$

end while

输出：聚类结果

---

表 5 K-means 聚类算法流程

利用肘部法最优  $k$  值的寻优过程如图 3 所示。

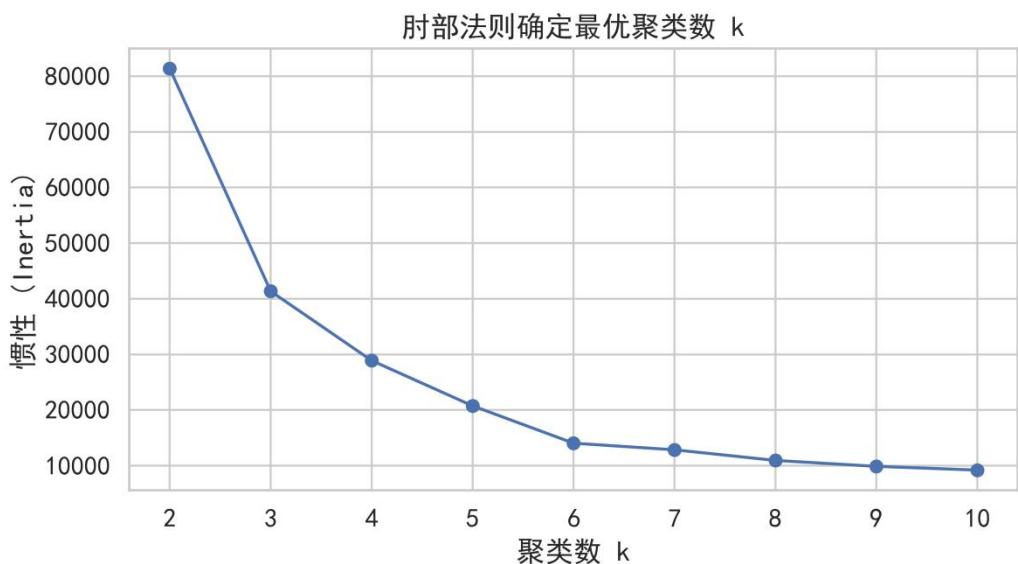


图 3 肘部法则确定最佳  $k$  值

#### 6.1.4. 聚类结果

基于 K-means 算法，对台风特征如台风等级、台风强度、风速、气压等特征进行聚合，将聚类的结果进行可视化如图 6。

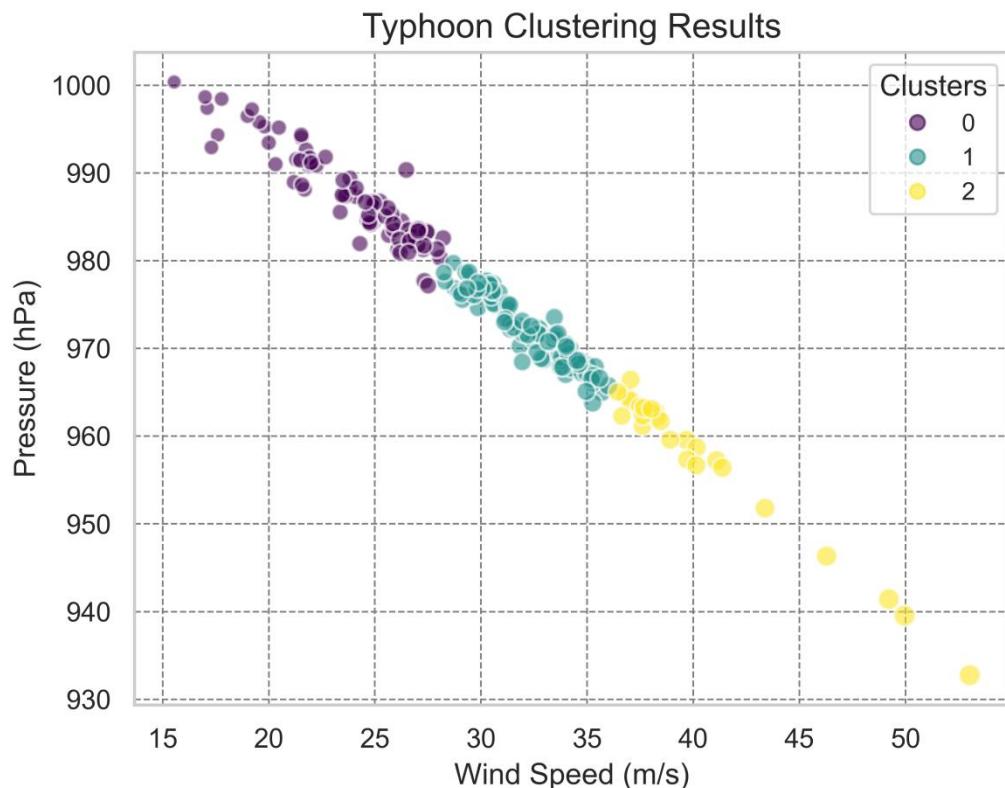


图 6 K-means 聚类后的可视化分类图

从图中可以看出，本文构建的聚类方案可以较好地挖掘出序列中类分离的显著特征。各簇边界明显且簇类中心点相互远离，使同一类别需求上的特征最为相似。对此我们将台风分为三类：第一类台风的气压与风速约在区间[979.6, 1005.0]和[16.1, 27.9]之间我们称其为弱能量台风，第二类台风的气压与风速约在区间[963.4, 979.6]和[27.9, 36.5]

之间我们称其为中等能量台风，第三类的气压与风速约在区间[933.7, 963.4]和[36.5, 53.7]之间我们称其为强能量台风。显示如下表 6：

台风类别	气压 (hPa)	风速 (m/s)
弱能量台风	[979.6, 1005.0]	[16.1, 27.9]
中等能量台风	[963.4, 979.6]	[27.9, 36.5]
强能量台风	[933.7, 963.4]	[36.5, 53.7]

表 6 台风分类结果

### 6.1.5. 对七月夏台风和九月秋台风的强度与路径研究

#### 6.1.5.1. 七、九月台风强度分析

首先我们从历史台风数据中提取七月和九月的台风，并对七月和九月的台风数据进行描述性统计分析如表 7、表 8。

	风速	气压	台风强度数值
Count	9788.000000	9788.000000	9788.000000
Mean	27.642011	981.372701	2.785145
Std	13.350054	20.704415	1.548677
Min	9.000000	896.000000	0.000000
25%	18.000000	972.750000	2.000000
50%	25.000000	988.000000	3.000000
75%	35.000000	996.00000	4.000000
Max	90.000000	1010.000000	6.000000

表 7 七月台风数据统计性描述

	风速	气压	台风强度数值
Count	13122.000000	13122.000000	13122.000000
Mean	31.237769	975.447737	3.175812
Std	15.068287	24.429484	1.655706
Min	9.000000	896.000000	0.000000
25%	20.000000	960.00000	2.000000
50%	28.000000	983.000000	3.000000
75%	40.000000	995.00000	4.000000
Max	90.000000	1010.000000	6.000000

表 8 九月台风数据统计性描述

根据上表七月台风的统计性描述，我们可以知道风速的平均值为 27.64 m/s，略低于九月台风的平均风速，表明七月的台风相对较温和。风速标准差为 13.35，显示风速的分布有一定的波动，风速最低为 9 m/s，最高为 90 m/s，表明七月的台风中包含强烈的台风，25% 的风速在 18 m/s 以下，显示出较弱的台风，可能主要是热带低压或热带风暴，中位数（50%）为 25 m/s，说明多数台风的风速在这个范围上下，75% 的风速在 35 m/s 以下，表明大部分台风的风速未达到极强级别。气压平均值为 981.37 hPa，比九月台风略高，说明七月的台风平均强度略弱，气压标准差为 20.70，显示气压分布较稳定，变化性稍低于九月，最低气压为 896 hPa，较九月台风的低气压高出许多，表明七月的台风极端强度稍弱。最高气压为 1010 hPa，接近常规气压，反映出其中包含一些较弱的台

风或热带低压，台风强度的平均值为 2.79，接近“热带风暴(TS)”和“强热带风暴(STS)”的强度等级，强度数值标准差为 1.55，反映出台风强度变化性较大，强度数值最低为 0（对应热带低压），最高为 6（对应超强台风），显示七月台风中有少量极端强台风。七月夏台风的数据表明，台风风速和气压范围涵盖了从弱到强的不同等级，但整体强度和风速均略低于九月台风。平均气压略高，风速相对低于九月，说明七月台风的强度普遍较弱。然而，极端数据表明也存在部分强烈台风，尤其是接近最大强度的少数台风。

对于九月秋台风风速的平均值为 31.24 m/s，属于台风的中等强度，风速的标准差为 15.07，表明数据分布较为分散，有不同强度的台风出现，风速最低为 9 m/s，最高为 100 m/s，表明有从弱到强的台风，涵盖热带低压到超强台风的强度范围。25% 的风速在 20 m/s 以下，显示出较弱的台风或热带低压中位数（50%）为 28 m/s，说明多数台风的风速在这个范围左右，75% 的风速在 40 m/s 以下，代表了大多数台风的风速不会超过 40 m/s，即达到强台风级别的风速。气压平均为 975.45 hPa，符合台风较低气压的特征气压标准差为 24.43 hPa，反映出气压数据相对稳定，但仍然存在较大的差异，最低气压为 876 hPa，表明部分台风可能非常强烈；而最高气压为 1011.2 hPa，接近正常海平面气压值，可能是一些弱台风或热带低压台风强度的平均值为 3.18，接近“强热带风暴”的强度等级，强度数值标准差为 1.66，反映出台风强度的变化较大，强度数值的最低为 0（对应热带低压），最高为 6（对应超强台风），表明九月份台风强度分布广泛，从热带低压到超强台风均有出现。从以上描述来看，九月份台风的风速和气压呈现出较大波动，其中以中等强度到强台风为主，少数达到超强台风级别。气压的变化范围较大，气压越低，风速和强度一般越高，符合台风发展强度的气压特征。总体来看，九月份台风的强度变化大，风速和气压指标在不同强度级别的台风中有显著差异。

下面分别是七月与九月台风风速以及气压的箱型如图 7、图 8 所示，我们从中也可以看到和上述结论类似的箱型图表现：

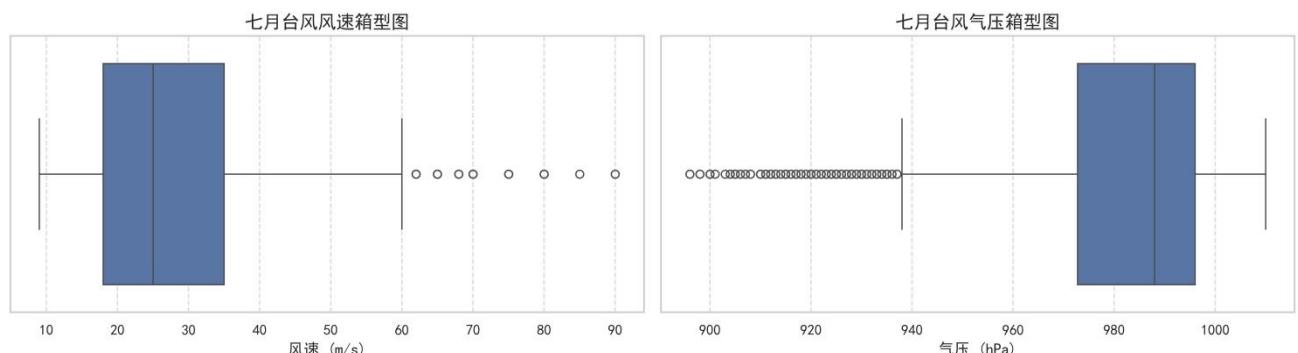


图 7 七月台风风速和气压箱型图

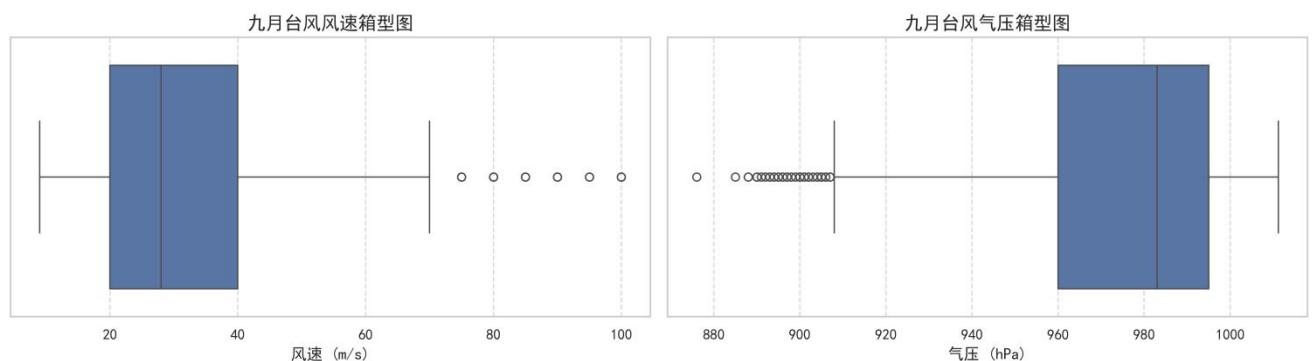


图 8 九月台风风速和气压箱型图

### 6.1.5.2. 七、九月台风路径分析

为了对我国七月夏台风和九月秋台风进行分类和途径省份的研究，我们首先利用Panda库统计出自2000年至2023年齐乐风台风的数量为86，随后给出七月台风路径及途径省份图如图9与图10：

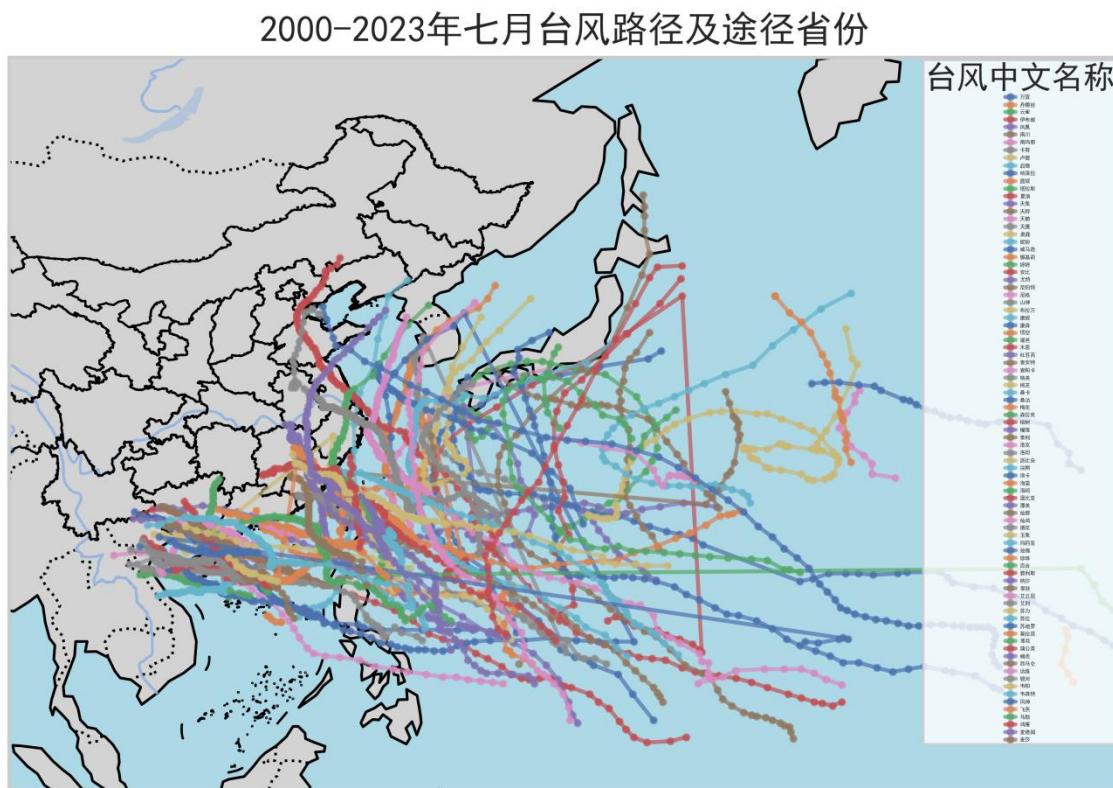


图 9 2000 至 2023 年七月台风路径及途径省份（见上图）

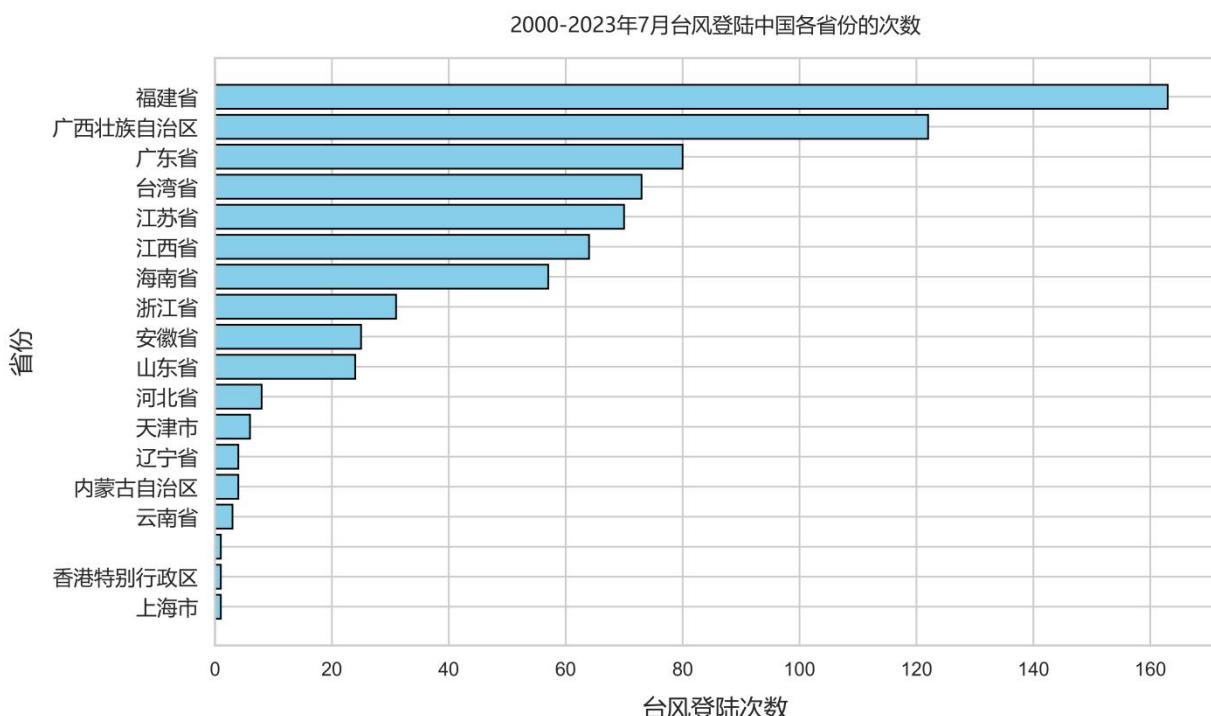


图 10 2000 年至 2023 年七月台风途径省份统计

根据 2000 年至 2023 年七月份台风途径省份数量的排序，可以分析台风在各省份的影响频率和分布规律。首先是高频台风影响省份，福建省（163 次）：作为受台风影响频率最高的省份，福建在东南沿海位置较为突出，处于台风登陆或途经的主要路径上。由于福建位于西北太平洋台风路径的前沿，台风多从东南沿海登陆，使得福建成为七月份台风影响的重灾区；广西次于福建，成为台风经常影响的地区之一。福建沿海台风登陆后，经常沿西南方向进入广西，导致广西成为台风路径上的高频省份，尤其是台风减弱后深入内陆；广东也是台风重要的登陆和经过区域之一，特别是在珠三角地区，由于地处南海和太平洋台风的交汇区，广东常年受到台风的侵袭，但数量上稍低于福建和广西。

随后是中等频次台风影响省份，台风常常沿东南方向路径经过台湾。虽然多数台风不会在台湾直接登陆，但其外围风雨影响显著。台湾的山地地形还会加剧台风带来的降水量，导致较大的风雨灾害；江苏省作为东部沿海的省份，江苏台风影响频次较高，可能与台风登陆后的路径有关。江苏虽然不在典型的台风登陆点，但受到登陆后向北转移台风的间接影响；江西位于内陆地区，但七月份台风从东南沿海进入内陆时会经过江西，特别是福建台风登陆后向西移动时，常途经或影响江西；海南由于地理位置直接面向南海，常受热带气旋的影响，特别是南海生成的台风，容易对海南造成直接影响；浙江虽然位于东南沿海，但在七月份受台风影响的频率相对较低，主要因为七月台风路径偏向偏南，登陆点多集中在福建及广东。

最后是低频次台风影响省份安徽省与山东省这两个省份在台风影响频率中属于相对低频的省份。多数情况下，台风在内陆逐渐减弱，影响力逐渐降低，故在较偏北的安徽和山东受到较少直接影响；河北和天津在台风路径上的频次更低，主要由于其地理位置较偏北，台风进入北方时通常已显著减弱；辽宁省与内蒙古自治区受到台风的间接影响通常为减弱后的风暴或强降水。路径较偏北的台风才能影响到辽宁和内蒙古；云南受台风影响较少，但偶有来自南海的热带气旋影响西南地区，导致降水和风力影响。虽然香港和上海位于沿海地区，但由于七月份台风路径多偏向西南方向，香港和上海在该时段受直接影响的概率较小。

我们可以总结：七月份台风路径多集中在东南沿海，尤其是福建、广东、广西和台湾等地。台风登陆后，通常向西北方向推进，经过江西、江苏等地。随着台风深入内陆，强度逐渐减弱，影响范围扩展至中东部省份。尤其是在台风减弱为热带风暴时，仍会带来强降水和大风，但影响范围较福建、广东等沿海省份显著减弱。同时数据显示东南沿海省份（如福建、广西、广东）的台风频次要远高于内陆或北方省份，这与台风在南海或西北太平洋生成的路径分布一致。总的来说七月份的台风影响频次呈现由东南沿海向内陆逐渐递减的趋势。

同样我们也对九月台风路径进行统计如图 11、12（见下）；

2000-2023年九月台风路径及途径省份

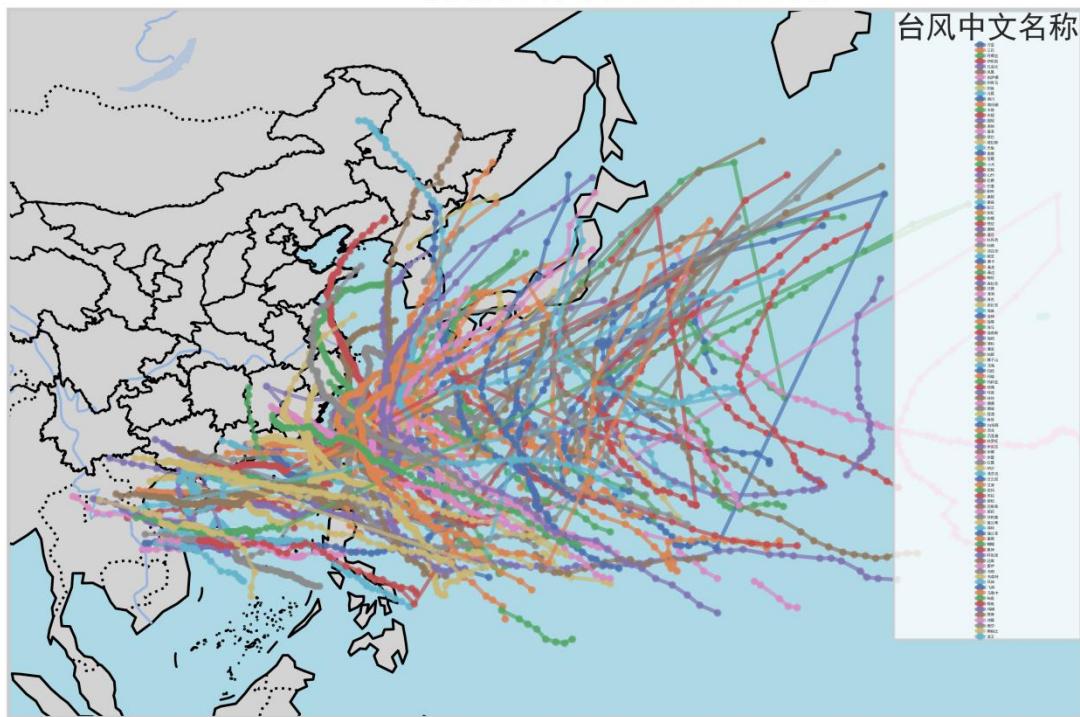


图 11 2000 至 2023 年九月台风路径及途径省份（见上）

2000-2023年9月台风登陆中国各省份的次数

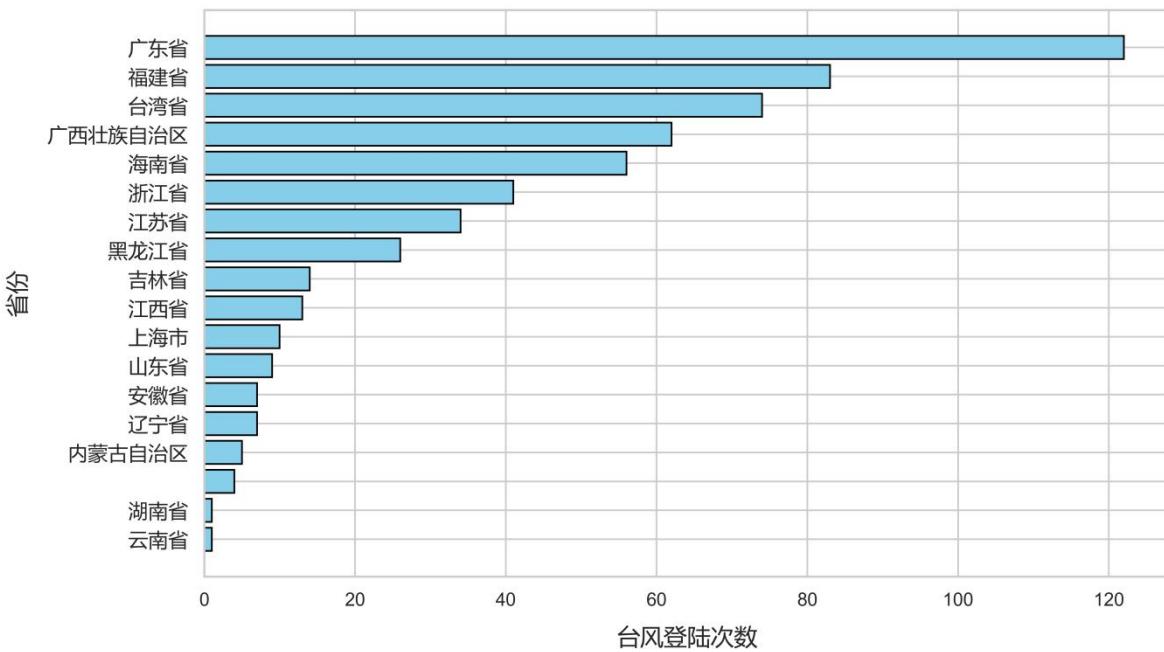


图 12 2000 年至 2023 年九月台风登陆中国个省份次数

与七月数据类似分析得九月份的台风主要影响的是南方沿海省份，尤其是广东、福建和台湾等地。与七月份的情况类似，九月份台风频次显示出明显的空间分布特征，主要集中在东南沿海地区。其中九月份的台风路径多集中在南海和东南沿海，特别是广东和福建等省份受影响明显，显示出典型的台风影响模式。台风在九月的路径通常由东南沿海向北或西北移动，造成了内陆省份的影响减少，但仍有一些省份如广西、海南受到直接影响。数据显示，九月份东南沿海省份的台风频次明显高于北方或内陆省份，反映

出台风形成与路径的区域性特点。总而言之，九月份台风在影响频次上表现出明显的地理分布，沿海地区面临的台风风险高于内陆，尤其是广东和福建等省份。

因此我们根据 6.1.4 的聚类结果，对 2024 年我国 7 月和 9 月台风，给出台风类别及途经省份的列表如下表 9：

月份	台风类别	途径省份
7 月夏台风	弱能量台风	广西壮族自治区，上海市，河北省，辽宁省，福建省，广东省，天津市，内蒙古自治区
	中等能量台风	广西壮族自治区，河北省，广东省，福建省，云南省，江苏省，海南省，浙江省
	强能量台风	广西壮族自治区，广东省，福建省，云南省，海南省，江西省，台湾省
9 月秋台风	弱能量台风	广西壮族自治区，上海市，广东省，福建省，云南省，黑龙江省，海南省，浙江省
	中等能量台风	广西壮族自治区，吉林省，上海市，辽宁省，广东省，福建省，内蒙古自治区
	强能量台风	广西壮族自治区，广东省，福建省，湖南省，江西省，台湾省

表 9 2024 年我国 7 月和 9 月台风类别及途经省份（见上表）

由此我们可以得到夏台风和秋台风的区别：夏季台风的生成数量明显高于秋季台风；然而，从强度上看，秋季的超强台风数量明显超过夏季的超强台风。由此可见，秋季台风的发生频率更高且实力更强。从不同类别台风的发生比例来看，夏季的热带风暴级和强热带风暴级台风较多，而 12 级以上的台风则在秋季更为常见，特别是强台风及以上的发生比例明显高于夏季。

秋季的台风强度通常比夏季的台风强，这主要与海洋表面的高温有关。从春分到夏至，太阳直射点从赤道移动到北回归线，导致北半球低纬度海域被加热；而从夏至后进入秋季时，太阳直射点又回移至赤道，使热带海域再次被加热。这种反复的加热过程创造了温暖的海温条件，有利于台风的形成和能量增强，从而导致强台风或超强台风的出现频率增加。此外，与夏季台风相比，秋季台风的登陆地点更偏南，主要集中在海南、广东和台湾一带。由于台风在西太平洋副热带高压的南侧生成和活动，其移动路径受到该高压的影响和制约。进入秋季后，西太平洋高压逐渐南撤东移，导致台风的路径和登陆地点向南偏移。

## 6.2. 问题二：台风路径预测模型

多元时间序列是我们日常生活中常见的现象，包括股票市场中不同股票的价格序列，以及不同城市的温度和降雨量等。顾名思义，多元时间序列的首要特征是按照时间顺序采样的具有时序性的序列数据。这类数据广泛应用于金融市场、气候预测等领域。通过对系统中多个时间依赖变量的研究，可以分析其内在的规律性和不规律性，以预测未来的趋势。

本文研究的台风路径同样是以多元时间序列为研究对象，涉及与台风路径相关的经纬度信息、曲率等轨迹趋势，以及与台风空间结构相关的一定范围内的环境场预测因子。

本文我们采用 RNN（循环神经网络）与 LSTM 预测算法对各类台风数据时间序列构建预测模型。重要实验参数设置如下表 10：

参数	参数值
滑动窗口大小	6
batchsize	32
Dropout_rate	0.3
激活函数	ReLU、Sigmoid
早停参数	10
Epoch	100
LSTM 学习率	0.001
优化器	ADAM
损失函数	MSE

表 10 重要实验参数设置

在本研究中，滑动窗口的大小决定了使用多少条历史台风 (TC) 记录来预测未来台风中心位置。滑动窗口的选择至关重要：窗口过大会导致历史数据长度超过预测所需的数据量，带来冗余信息；而窗口过小则可能导致模型获取的信息不足，无法充分捕捉台风路径演化的趋势。为了找到合适的窗口大小，我们对不同的历史长度进行了实验，以分析不同滑动窗口设置对模型预测效果的影响，从而选出最佳用于模型训练的窗口大小为 6。

在本文中，batch\_size（批大小）指在每次迭代中用于训练的样本数。通过采用 mini-batch 的方式，可以在模型训练时有效提高训练速度和优化效率。mini-batch 方法的优势在于其能够在每次迭代中利用部分样本进行参数更新，相较于全量数据训练更为高效，且比单样本训练更稳定。通过使用 RandomizedSearchCV 进行超参数优化，本文选取的 batch\_size 大小为 32。

在本文中，Epoch（训练轮次）指的是所有数据在神经网络上完成一次完整的前向和反向传播的过程。通常，模型的训练需要多轮迭代，以便通过不断更新参数充分学习数据中的潜在信息，从而获得最优模型。训练轮次不能过少，以避免模型欠拟合；但训练过多则可能导致模型过拟合，即过度拟合训练数据，难以泛化到新数据。本文初始选择的 Epoch 数为 100 轮。

早停 (Early Stopping) 参数的设置旨在解决 Epoch 数量的手动设置问题。我们知道，Epoch 轮次过少可能导致模型欠拟合，过多则容易过拟合。早停法的核心目标是防止过拟合，因此可以看作是一种正则化方法[6]。其原理是，在每个 Epoch 结束后，用测试集评估模型性能；如果性能优于之前最佳模型，就将当前模型保存为最优模型。本文采用了 max\_patience 参数来控制早停。具体来说，当某一轮的损失未比前一轮更低时，继续观察后续的 patience 个 Epoch；如果在此期间损失没有进一步下降，则停止训练。本文中 patience 设为 10，即如果 10 个 Epoch 内没有性能提升，则训练将提前停止。

激活函数的作用在于帮助模型学习和理解复杂的非线性信息。在本文中，第一个全连接层后采用了 ReLU 激活函数，以更高效地提取非线性特征。ReLU 激活函数的一个显著特点是当输入小于 0 时输出为 0，即在该区间内完全不激活，这种特性能够加快模型的收敛速度，同时也减少计算复杂度，从而提高训练效率。因此，本文选用了 ReLU 函数作为激活函数，以便更好地提取台风路径预测中的重要特征信息。

本文的学习率优化选用了 Adam 算法。选择 Adam 的主要原因在于传统的随机梯度下降方法在训练过程中可能遇到局部最优或鞍点，从而影响模型的收敛效果。虽然目前 SGD 已经引入了一阶动量以帮助跳过这些特殊情况，但它无法根据每个参数的重要性来实现差异化的更新步长。相比之下，Adam 算法结合了 SGD 的一阶动量和 RMSProp 的二阶动量，使其在学习过程中具有更好的自适应性，对不同参数进行适当的调整。基于

这些优点，本文选用 Adam 优化算法，初始学习率设定为 0.001，以便更高效地优化模型。

### 6.2.1. RNN 预测模型构建与求解

循环神经网络（Recurrent Neural Network，RNN）是一种特殊的神经网络模型，具有与传统神经网络不同的特性。传统神经网络通常由输入层、隐藏层和输出层构成，且同一层的神经元之间不进行相互通信，当前时刻的输出与上一时刻的隐藏状态没有直接关系。下图（图 13）展示了传统神经网络的结构：

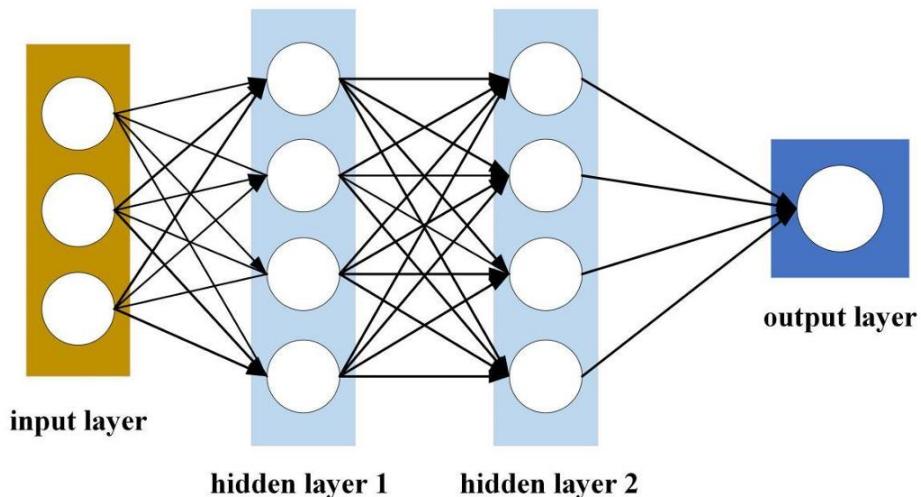


图 13 RNN 神经网络模型

台风路径预测数据属于多元时间序列数据，具有自相关性。相邻台风路径记录点的气象信息和路径信息等，对预测未来台风位置坐标具有不同程度的相关性。RNN 由于其独特的结构，能够使所有隐藏层的神经元之间进行相互通信，使得上一次的结果以数据形式存储在隐藏层中。当下一时刻传递消息时，前一时刻的隐藏结果也会影响当前输出。通过持续这一过程，最终输出将包含前面所有的信息。这种信息传递方式体现了时间序列数据的时序特性。通用的 RNN 模型如图 14 所示，其中  $Q_t$  表示在  $t$  时刻的输出层输出； $s_t$  表示在  $t$  时刻的隐藏状态值； $x_t$  表示在  $t$  时刻的输入值； $b$  表示偏置项； $V$  表示从隐藏层到输出层的权重矩阵； $W$  表示从隐藏层到隐藏层的权重矩阵； $U$  表示从输入层到隐藏层的权重矩阵。

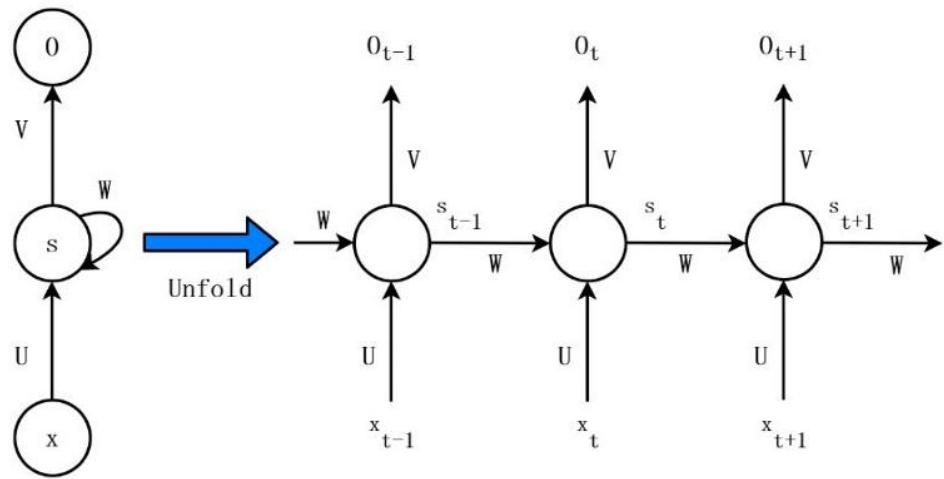


图 14 RNN 网络结构

公式表示如下：

$$\begin{aligned}
 O_t &= V \cdot s_t + b_o \\
 s_t &= f(W \cdot s_{t-1} + U \cdot x_t + b_h) \\
 O_t &= V \cdot f(W \cdot f(U \cdot x_{t-1} + \dots) + U \cdot x_t) + b_o
 \end{aligned}$$

从上图的 RNN 结构可以看出，RNN 具有两个显著特点。一方面，由于其串联结构，体现了“前因后果”的关系，后续结果的生成依赖于前面的信息，因此网络的输出与历史信息密切相关。另一方面，所有特征共享同一套参数，这不仅确保了信息传递的公平性，还大大减少了训练参数的数量。然而，普通 RNN 由于其串联结构，在处理长序列问题时容易遇到瓶颈，因此衍生出了一系列 RNN 的变形优化。

利用 RNN 模型对历史台风的经纬度预测与实际经纬度对比效果如下图(图 15)：

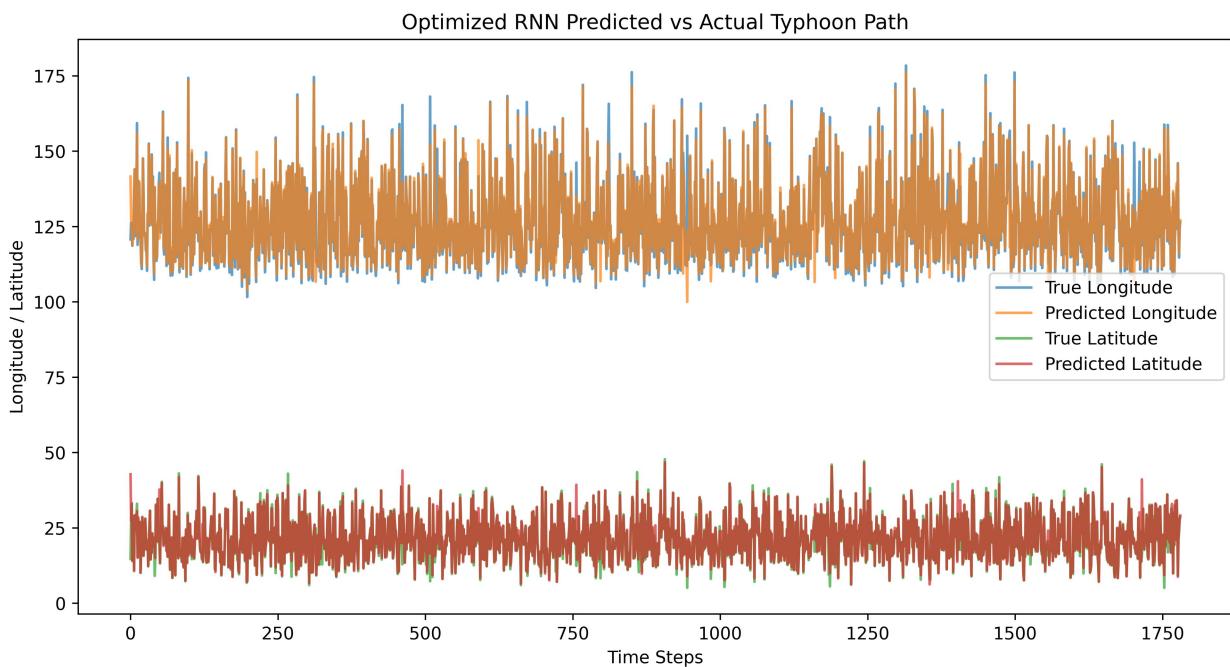


图 15 优化后 RNN 预测与实际台风经纬度对比

接下来利用 RNN 模型对台风贝碧嘉的行进路径进行预测，结果见附件“RNN 贝碧嘉预测.xlsx”。

## 6.2.2. LSTM 预测模型构建与求解

在预测任务中，LSTM（长短期记忆网络，Loog Short Term Memory）是一种非常常用的预测方法，尤其适用于时间序列数据。与传统的前馈神经网络不同，LSTM 是一种递归神经网络（RNN）的特殊变体，能够在较长时间内保留并利用历史信息。它通过门控机制对长期和短期信息进行筛选和保留，能有效避免梯度消失或爆炸问题，使得模型在较长时间依赖的任务上具有更好的表现。本文台风数据具备明显的时间序列特征，数据记录点之间存在时间依赖性，因此使用 LSTM 模型来预测台风路径可以具有较高的合理性和效果预期。LSTM 能够充分利用历史轨迹和环境变量，提取其中的时序模式，从而提供对未来台风位置的有效预测。

整个 LSTM 网络模型学习过程的前向传递包含了遗忘门  $f_t$ 、输入门  $i_t$  和输出门  $o_t$ 。其中遗忘门  $f_t$  用来控制  $t - 1$  时刻的记忆单元状态  $C_{t-1}$  是否被保留，表达式为：

$$f_t = \sigma(W_f \cdot [h_{t-1} \ x_t] + b_f)$$

式中， $x_t$  为  $t$  时刻的输入向量， $h_{t-1}$  为  $t - 1$  时刻的隐藏状态向量， $\sigma$  为 Sigmoid 激活函数， $W_f$  和  $b_f$  分别为遗忘门的权重矩阵与偏置向量。 $i_t$  用于控制  $t$  时刻的输入向量  $x$  是否被更新到  $t$  时刻的记忆单元更新值  $\tilde{C}_t$ ，分别表示为：

$$i_t = \sigma(W_i \cdot [h_{t-1} \ x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1} \ x_t] + b_c)$$

式中， $W_i$  和  $b_i$  为输入门的权重矩阵与偏置向量， $W_c$  和  $b_c$  为记忆单元更新值的权重矩阵与偏置向量。

$t$  时刻的记忆单元状态  $C_t$  通过遗忘门  $f_t$  和输入门  $i_t$  控制  $t - 1$  时刻的记忆单元状态以及  $C_{t-1}$  更新值  $\tilde{C}_t$  得到：

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

$o_t$  基于  $C_t$  更新  $t$  时刻的隐藏状态向量  $h_t$ ，表达式分别为：

$$o_t = \sigma(W_o \cdot [h_{t-1} \ x_t] + b_o)$$
$$h_t = o_t \cdot \tanh(C_t)$$

式中， $W_o$  和  $b_o$  为输出门的权重矩阵与偏置向量。

LSTM 通过不同的“门”来控制抛弃或者增加信息，从而实现遗忘或记忆的功能，模拟神经网络。“门”是一种使信息选择性通过的结构，由一个 Sigmoid 函数和一个点乘操作组成。Sigmoid 函数为生物学中常用的 S 型函数，其值域在  $[0,1]$  区间，当“门”输出为 0 时代表此信息完全丢弃，为 1 时代表完全通过。一个 LSTM 单元有三个这样的门，分别是遗忘门、输入门、输出门。简单来说，LSTM 中的三个门就是通过三个  $[0,1]$  之间的函数来有选择的过滤旧的信息，从而加入新的信息。

本文通过 Python 编程来实现 LSTM 对数据进行预测，将台风各类特征数据以及气温等作为输入，经纬度信息即台风路径作为输出。从而利用 LSTM 神经网络对台风数据进行训练，最终将训练得到的模型对台风的经纬度进行预测。

我们首先对已有数据来进行训练集，验证集及其测试集的划分，规定训练集为 80%，验证集与测试集均为 10%。随后使用 Python 中的 Keras 以及 Sklearn 库建立 LSTM 算法利用训练集进行神经网络的训练得到最终的 LSTM 预测模型，结果如下图 16：

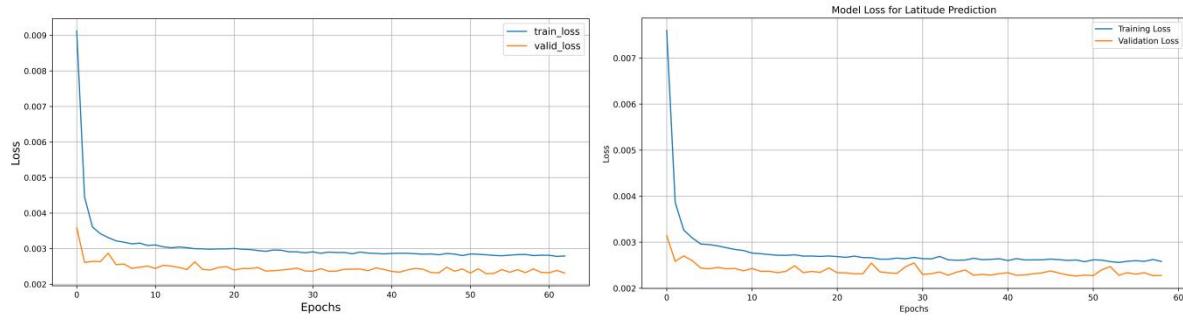


图 16 LSTM 经纬度预测损失曲线

在经度训练过程中，模型的损失值从初始的 0.0162 逐渐减少至 0.0027，并逐步趋于稳定。在训练过程中，最大 loss 下降幅度出现在第一轮训练，下降了 0.0114。此次训练总共进行了 63 轮，在模型的 loss 值不再显著下降后，自动触发早停机制，停止训练。在纬度训练过程中，模型的损失值从初始的 0.0123 逐渐减少至 0.0025，并逐渐趋于稳定。最大 loss 下降幅度出现在第一轮训练，下降了 0.0083。本次训练共进行了 59 轮，随后因 loss 值趋于稳定，早停机制被触发，训练自动停止。

同时我们带入 2023 年台风 Sanba 的数据进行路径预测结果如下图：

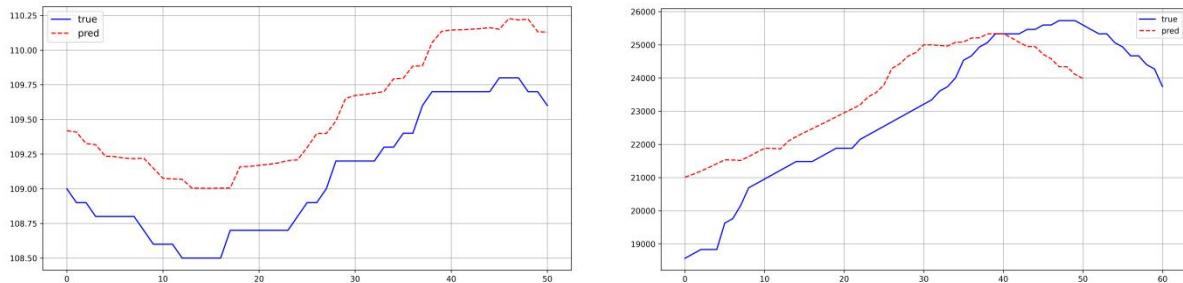


图 17 LSTM 对 2023 年台风 Sanba 经纬度预测

由经纬度的预测结果来看，模型对于中高经度的区域预测效果好于低经度区域，同时也可以看出模型对高精度区域的精度相对较高，对于低纬度地区的变化更不敏感。同样我们也利用 LSTM 预测模型对台风贝碧嘉进行路径预测，结果见附表“LSTM 贝碧嘉预测结果.xlsx”。

### 6.2.3. 任务预测与性能评价

本文采用运用 Dynamic Time Warping(DTW)动态时间规整算法与台风实际行进路线进行对比来实现预测路径与实际路径间的相似性分析，同时对 RNN 及 LSTM 模型的性能采用均方误差函数（MeanSquareError，MSE）和平均绝对误差（MeanAbsoluteError，MAE）表示预测经纬度点和 真实经纬度点之间的差异。

MSE 的计算公式为：

$$MSE = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

MSE 值越小，表示预测值越接近真实值

MAE 的计算公式为：

$$MAE(X, h) = \frac{1}{m} \sum_{i=1}^m |h(x_i) - y_i|$$

为了更直观的展示出 RNN 与 LSTM 预测模型的路线预测结果，以下是不同模型与台风贝碧嘉实际路径线路对比图(图 18、图 19):

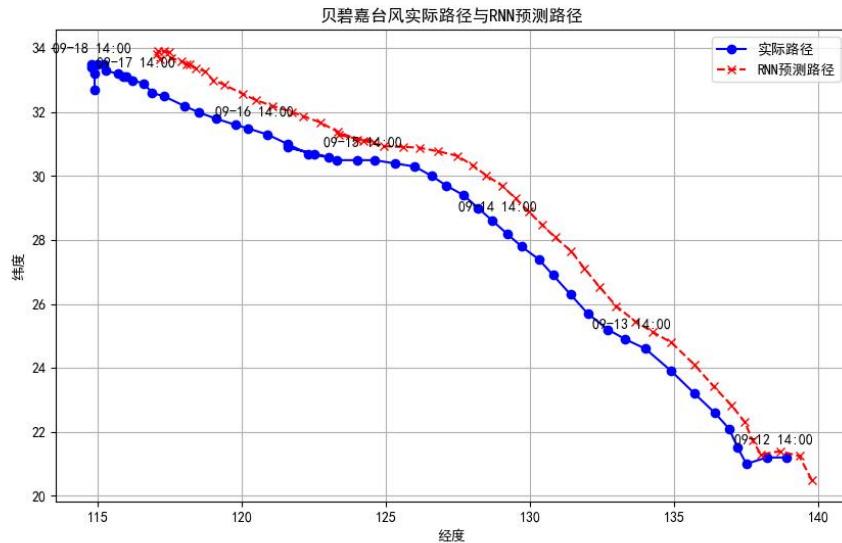


图 18 贝碧嘉台风 RNN 路径预测

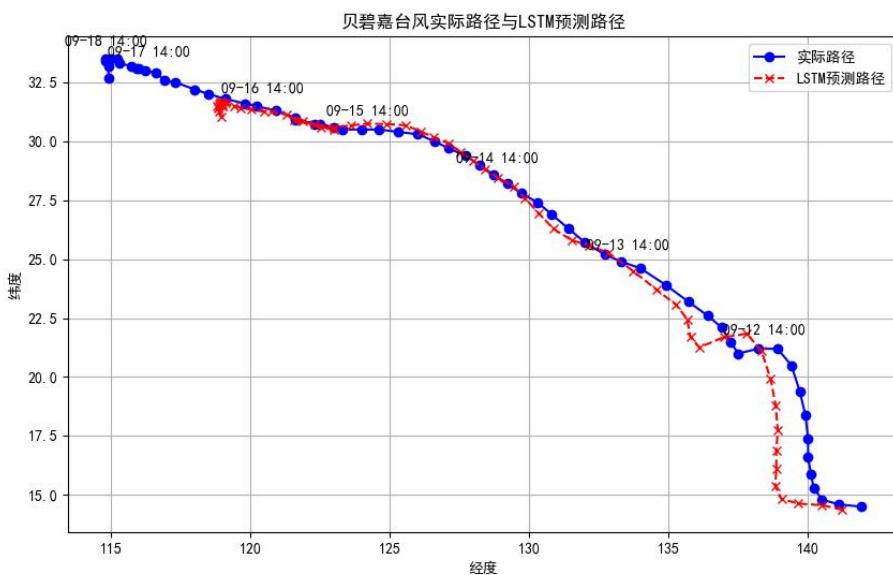


图 19 贝碧嘉台风 LSTM 路径预测

从图中我们可以看到 RNN 神经网络在高经度低纬度区域预测效果要明显优于低经度高纬度区域和中经度中纬度地区，并且在越低的经纬度地区 RNN 模型的预测偏差要逐渐变高。而 LSTM 预测模型在中经度中纬度区域的预测效果非常好，在高经度低纬度区域效果要略差于中经度中纬度区域，而在低经度高纬度区域则出现明显的偏差。但 LSTM 模型的总体预测情况要好于 RNN 神经网络，这与我们的模型构建初衷和预期相符合。

下面我们利用 DTW 动态时间规整算法对预测的台风路径与实际台风路径做相似性分析。

在时间序列分析中，动态时间规整算法（Dynamic Time Warping, DTW）是一种用于测量两个时间序列之间相似性的有效工具。DTW 算法能够对序列在时间上的变速进行对齐，使得即便时间序列在速度上有所不同，也能进行有效的比较。此算法在多个领

域都有广泛的应用，其中语音识别是最常见的应用之一，能够处理说话者的语速差异。

在本文中，以台风路径为研究对象，DTW 算法可用于检测两条台风路径在不同时间段的相似性。由于台风在移动过程中可能存在速度变化，DTW 能够帮助分析这些变化，揭示两条台风路径之间的相似程度，从而为台风路径的预测和研究提供有价值的信息。通过应用 DTW，可以更好地理解台风的动态行为及其对环境的响应。

定义 2 条时间序列 A:  $a_1, a_2 \dots, a_i \dots, a_n$ ; B:  $b_1, b_2 \dots, b_j \dots, b_m$ ，长度分别为 n 和 m。利用动态规整算法计算它们之间的规整路径见图 20。

为了提高搜索效率，本文作以下约束：(1) 单调性：路径不能往回走。(2) 连续性：路径一次向前移动一步。(3) 有界性：路径从左下方开始，到右上方结束。(4) 扭曲度：路径不能偏离对角线太远。(5) 斜率限制：路径不能太陡或太平缓。

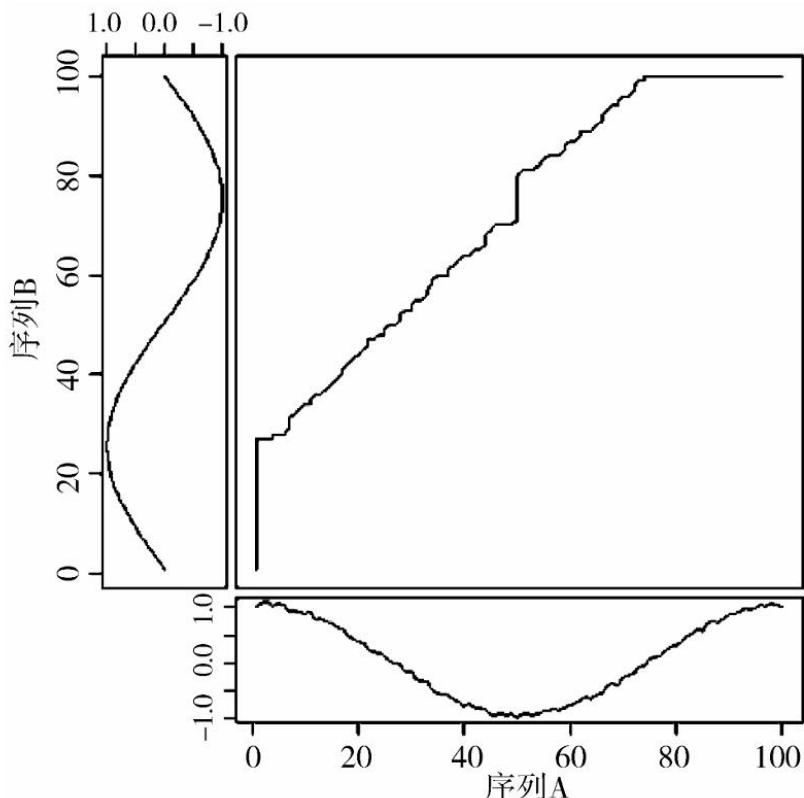


图 20 序列 A 和 B 间的规整路径

动态规整算法主要通过损失矩阵寻找最优路径，最后得到 2 条序列的规整路径，主要包含以下步骤：

1) 设 2 条时间序列的长度分别为  $n$  和  $m$ ；

2) 网格的规模为  $n \times m$ ；

3) 计算损失矩阵  $C$ ,  $C_{ij} = |A_i - B_j|$ ；

4) 计算规整矩阵  $D$ , 令  $D(1,1) = 0$ ,

$$D(i+1, j+1) = C_{ij} + \min \left\{ \begin{array}{l} D(i, j+1) \\ D(i+1, j+1) \\ D(i+1, j) \end{array} \right\}$$

5) 得到动态规整的规整矩阵  $D$ , 则动态规整距离为  $D(n, m)$ 。

接下来是计算相似度：计算目标台风与第  $i$  条台风的相似度，其取值范围为  $[0,1]$ ，相似度越接近 1 表明 2 条序列越相似，定义相似度如下：

$$S_i = e^{-D_i}$$

式中： $D_i$  为目标台风与第  $i$  条台风的动态规整距离。

利用构建的 DTW 动态规整模型对台风贝碧嘉的 RNN 神经网络的预测路径与实际路径做对比，我们发现经度的 DTW 距离为 6.1496，纬度的 DTW 距离为 1.8440。经纬度对比效果图如下图 21：

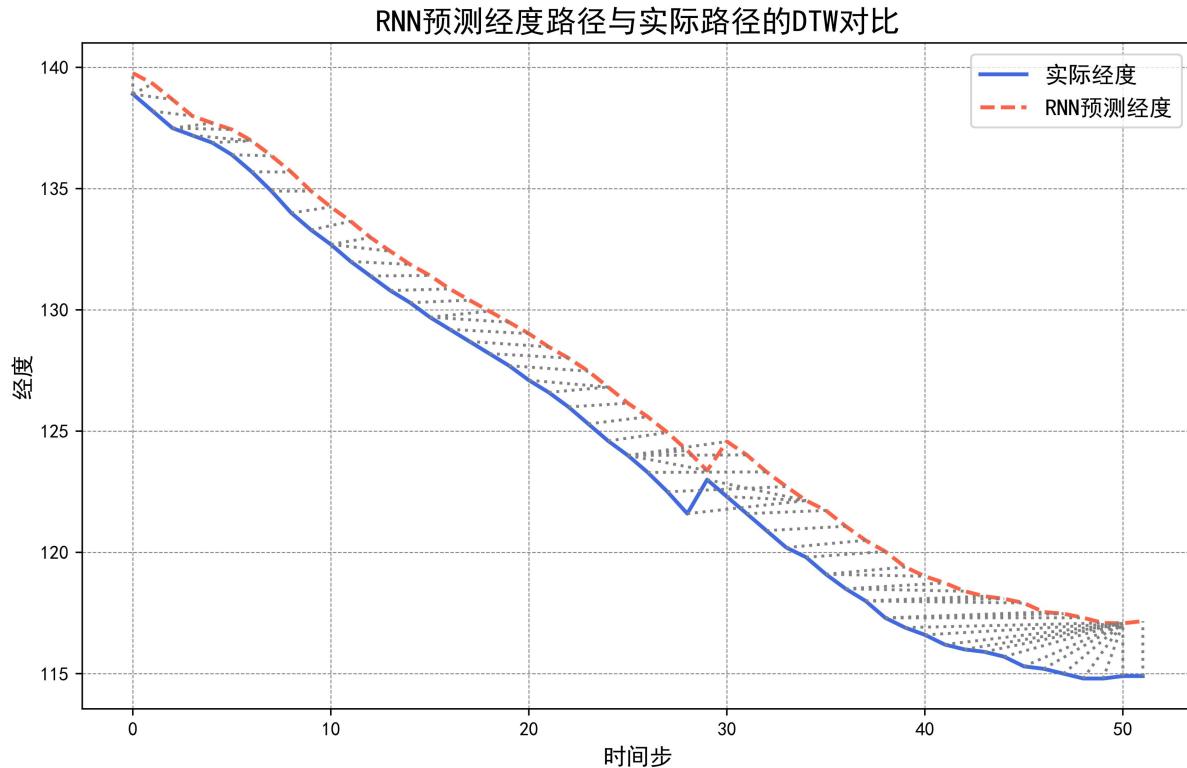


图 21RNN 预测经度路径与实际路径 DTW 对比

RNN预测纬度路径与实际路径的DTW对比

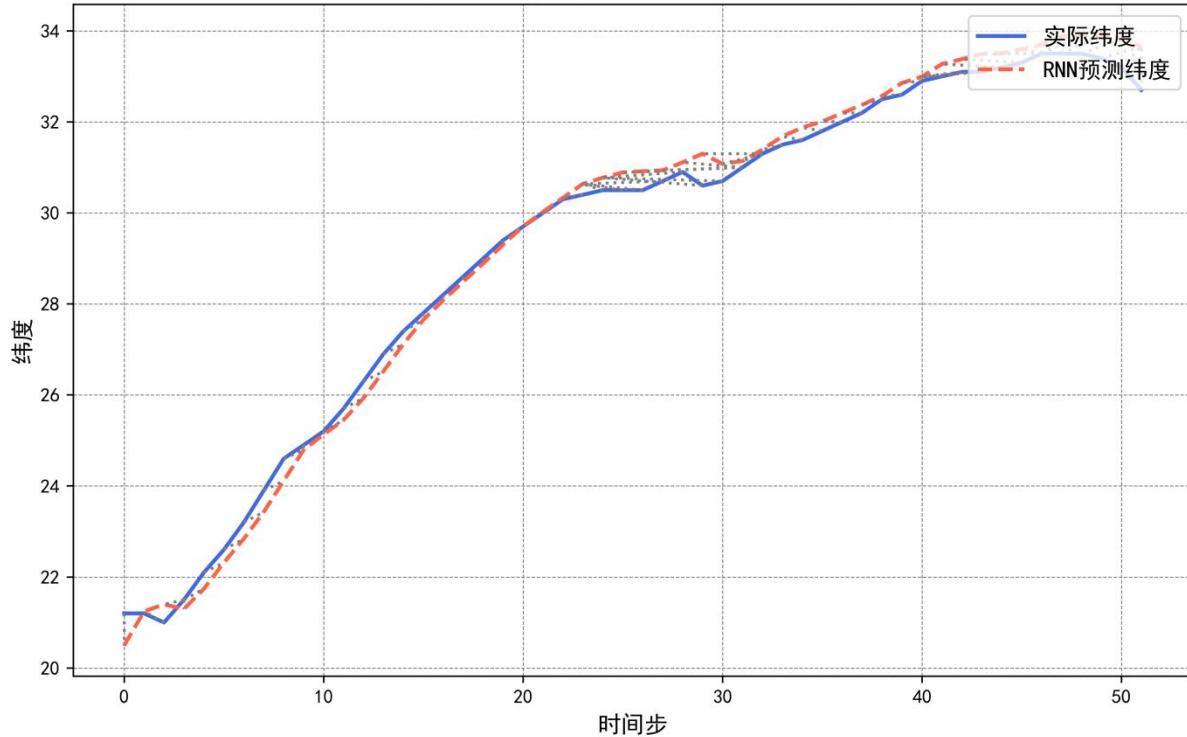


图 22RNN 预测纬度路径与实际路径 DTW 对比

同样我们对 LSTM 预测模型做类似的 DTW 相似度计算，我们得到 LSTM 模型下的预测路径和实际路径的经度的 DTW 距离为 12.3173，纬度的 DTW 距离为 6.3018，效果图如图 23：

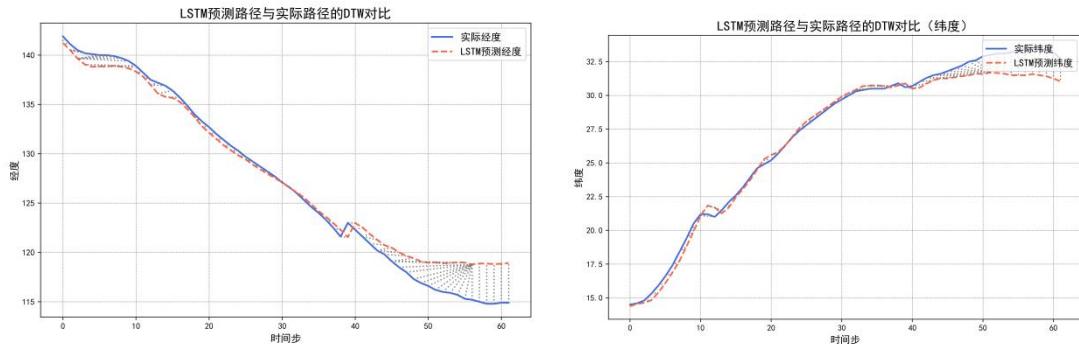


图 23 LSTM 预测经纬度路径与实际路径 DTW 对比

由此我们发现在经度的 DTW 距离上 RNN 神经网络小于 LSTM 模型，这可能是因为 LSTM 在低经度区域的误差过大，较大程度影响了 DTW 距离的计算，而在纬度的 DTW 距离上 RNN 神经网络为 1.8440 仍小于 LSTM 的纬度 DTW 距离 6.3018，这是因为在高纬度区域 RNN 展示出的预测效果更好，明显优于 LSTM 模型，因此我们通过 DTW 动态规整算法，有充分的理由认为在贝碧嘉路径预测上 RNN 模型的预测效果更好。

时间	台风中心位置(经度/纬度)
13 日 14: 00	(24.565797,135.163392)
14 日 14: 00	(28.352878,130.558080)
15 日 14: 00	(30.852047,126.376480)
16 日 14: 00	(31.586166,122.917786)

进而根据 RNN 预测的经纬度信息，本文得出 2024 年 9 月 13 日-17 日每日 14 点的第 13 号台风贝碧嘉的中心位置如表 11 所示。

### 6.3. 问题三：台风登陆后对降水量和风速的影响

已知，台风登陆后，风速和雨量将逐渐减弱，且某地区受台风影响 而产生的风速和降水量，随着其与台风中心距离的增加而减小。本文分别建立降水量与风速的关系模型和降水量与距台风中心距离的关系模型，最后反代贝碧嘉台风数据预测贝碧嘉行进途中的中心风力及降水量。

#### 6.3.1. 降水量与风速的关系模型

本文首先提取 1949-2018\_DailyPrecipitation 文件中的台风时间、站点、经纬度、降水量等数据和 1949-2018\_Wind 中的台风编号，站点，风速等信息。利用 Pandas 库中的 merge 函数根据时间列合并两文件数据，接着依靠每日平均降水量和风速绘制散点图(图 24)。

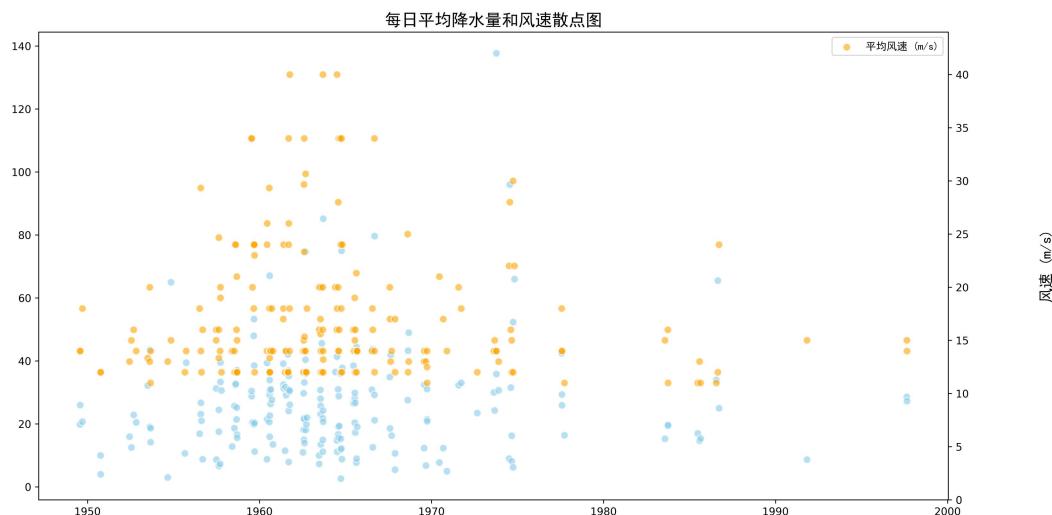


图 24 每日平均降水量和风速散点图

从图中得知降水量与风速间并不存在明显的线性关系，这与我们的常识认知相符合，即降水量与风速一般不会同时增大。接下来我们建立随机森林回归模型进一步探究两者的非线性相关性。

随机森林是基于集成学习的思想，通过组合多个基学习器（即决策树）来改善预测性能。集成学习的关键在于多个模型的结合可以降低单个模型的偏差和方差，每棵树通过递归地对数据进行分割，以实现数据的分类或回归。在回归任务中，树的输出是根据平均值进行预测。

同时引入样本随机性和特征随机性两个层次的随机性，以提高模型的多样性和鲁棒性。对于回归问题，随机森林的最终预测是所有决策树预测结果的平均值。具体来说，如果有 N 棵树，预测结果为：

$$\hat{y} = \frac{1}{N} \sum_{i=1}^N T_i(x)$$

其中  $T_i(x)$  是第  $i$  棵树对输入  $x$  的预测。

模型结果如下：

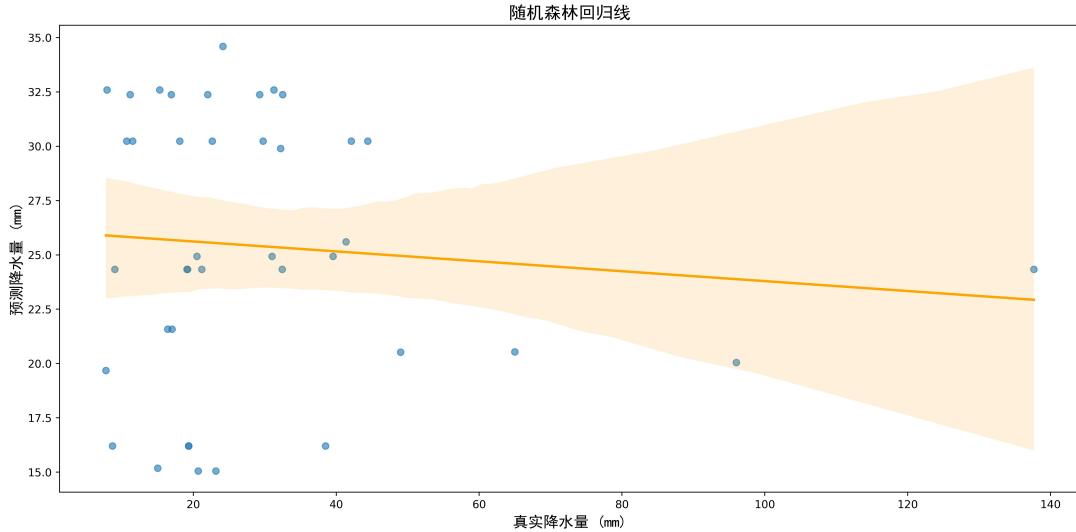
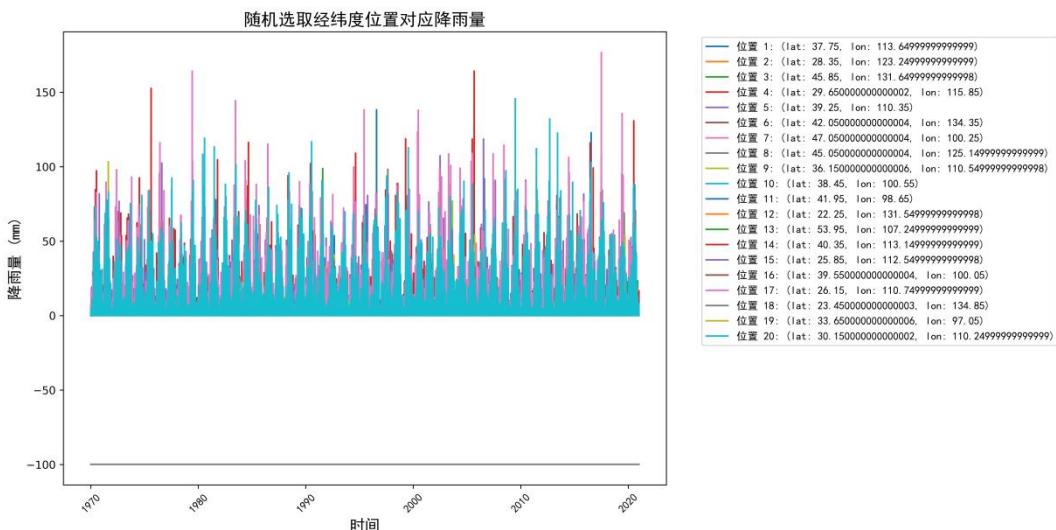


图 25 降水量随机森林回归

可知降水量与风速间的关系呈较弱的正相关关系，解释是合理的。

### 6.3.2. 降水量与距台风中心距离的关系模型

本文首先读取 CHM\_PRE\_0.1dg\_19612022 文件中 1970 到 2020 年的台风降水量场数据同时筛选出 [98,180], [2,58] 的有效经纬度，接下来我们随机抽取二十个样本点的经纬度，并统计随机选取的经纬度位置的对应降水量，结果如图 26：



我们发现存在有降水量数据值接近-100 的情况，视为无效降水量数据，由于文件数据量过大，限制于计算机硬件条件，本文难以将无效数据剔除。于是

从图中得到在不同的经纬度位置一般具有不同的降水量情况，在某些地区可能会出现降水量明显高于其他地区的情况，而现实天气的降水量往往与台风影响以外还与其他天气系统的相互作用如全球气温升高，洋流的不确定性等因素有关，因此我们认为结果

合理的。接下来我们同样对降水量以及距台风中心距离做随机森林回归以研究他们的非线性关系如下图 27:

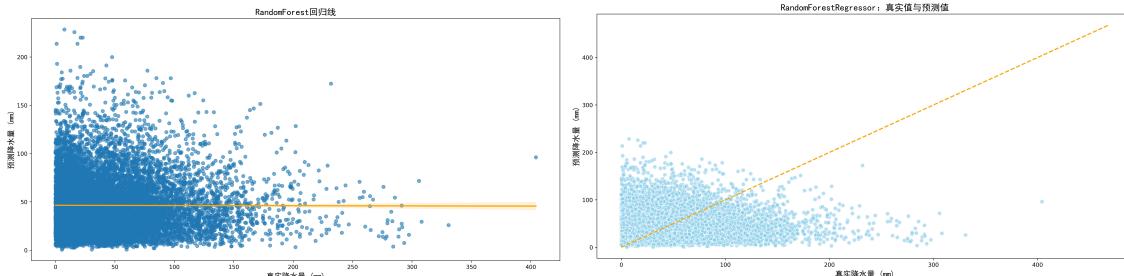


图 27 RandomForestRegressor 真实值与预测值散点图和回归线

对于贝碧嘉台风数据，本文先取贝碧嘉路径的中心点[129.7083,25.1958]，再抽取里中心点不同距离的采样点，利用大圆距离使用 Haversine 公式计算采样点离中心点的距离随后根据随机森林模型得到对应的降水量进而得到台风中心风速大小结果如下表 12 所示：

表 2 贝碧嘉行进途中的中心风力及降水量

	经度	纬度	距离中心点距离/km	降水量/mm	风速 m/s
中心点坐标	129.7083	25.1958			
采样点 1	146.8000	10.8000	2409.24	24.22	16.76
采样点 2	143.8000	13.6000	1958.44	48.21	23.41
采样点 3	140.0000	16.6000	1432.86	68.57	28.47
采样点 4	137.2000	21.5000	867.98	70.46	30.54
采样点 5	132.0000	25.7000	236.82	65.13	27.38
采样点 6	127.7000	29.4000	507.83	70.37	33.79
采样点 7	122.5000	30.7000	935.45	69.42	44.76
采样点 8	119.1000	31.8000	1269.25	44.73	20.64
采样点 9	115.9000	33.1000	1600.97	22.64	16.52
采样点 10	114.9000	32.7000	1662.52	15.79	13.55

## 七、模型的评价、与推广

### 7.1. 模型优点

1. 本文广泛利用可靠数据集，同时采取多种模型，对数据集使用充分，对数据集的处理比较完整。
2. 在对 RNN 和 LSTM 模型调参过程中使用了 GridSearch 和 RandomizedSearchCV 进行超参数优化，提高了模型的泛化能力，验证模型的稳健性，防止过拟现象发生。
3. 模型具有可拓展性，可以引入更多性能优良的时序预测模型或考虑更多解释变量，来支撑更精确的台风预测任务。

## 7.2. 模型缺点

- 1.LSTM 模型运行耗时较长，在并行处理上存在劣势。
- 2.受限于时间，时序预测模型参数可以进一步寻优。

## 7.3. 模型改进与推广

1、可引入更多的预测模型（如 GRU、Seq2Seq 模型等），再使用以卷积神经网络为核心的模型处理卫星云图资料，进行赋权处理提高整体的预测能力，最后还可以用到集成预报的方法。

2、本文建立的模型可以广泛运用到台风强度和台风路径等气象预测中使用。

## 八、参考文献

- [1] 王瀚. 基于深度学习的台风路径预测多模型算法研究[D].成都:电子科技大学硕士论文, 2020.
- [2] 徐光宁. 基于深度学习的台风路径与强度预测方法研究[D]. 哈尔滨:哈尔滨工业大学硕士论文, 2020
- [3] 徐高扬, 刘姚. LSTM 网络在台风路径预测中的应用 [J]. 计算机与现代化, 2019, No.285(05): 64-68+73
- [4] 裴艳萍. 基于深度学习的台风强度和路径演化机制研究[D]. 中南财经政法大学, 2023. DOI:10.27660/d.cnki.gzcuzu.2023.002612.
- [5] 刘峻, 高珊. 基于 XGBoost 和 LSTM 的台风强度预测模型分析 [J]. 无线互联科技, 2022, 19(06):46-48.
- [6] 高鹏. 基于深度学习对西北太平洋台风路径的预测研究[D]. 中国民用航空飞行学院, 2023. DOI:10.27722/d.cnki.gzgmh.2023.000281.
- [7] 高珊 . 基于深度学习的台风强度预测研究 [D]. 广西大学, 2021. DOI:10.27034/d.cnki.ggxu.2021.000721.
- [8] Yu, Z., Y. Wang, and H. Xu, 2015: Observed Rainfall Asymmetry in Tropical Cyclones Making Landfall over China. J. Appl. Meteor. Climatol., 54, 117 - 136, <https://doi.org/10.1175/JAMC-D-13-0359.1.4>
- [9] Ying, M., W. Zhang, H. Yu, X. Lu, J. Feng, Y. Fan, Y. Zhu, and D. Chen, 2014: An Overview of the China Meteorological Administration Tropical Cyclone Database. J. Atmos. Oceanic Technol., 31, 287-301, <https://doi.org/10.1175/JTECH-D-12-00119.1>.

## 附录

### No. 1 Filename: Question\_1.ipynb

```
1. import pandas as pd
2. import numpy as np
3. file_path = '/Users/z/Desktop/Mathor/数据.xlsx'
4. df = pd.read_excel(file_path)
5. # 过滤不符合条件的经纬度数据
6. df = df[(df['经度'] >= 95) & (df['经度'] <= 180) & (df['纬度'] >= -2) & (df['纬度'] <= 58)]
7. df = df[(df['气压'] > 100) & (df['风速'] <= 100)]
8. import pandas as pd
9.
10. def convert_direction(direction):
11.     # 如果移动方向为空, 直接返回空
12.     if pd.isnull(direction):
13.         return direction
14.
15.     # 创建字符到数值的映射
16.     char_mapping = {
17.         '东': '1',
18.         '南': '2',
19.         '西': '3',
20.         '北': '4',
21.         '偏': '0',
22.         'E': '1',
23.         'S': '2',
24.         'W': '3',
25.         'N': '4',
26.     }
27.
28.     # 初始化转换结果
29.     converted = ''
30.
31.     # 逐个字符转换, 忽略“偏”字
32.     for char in direction:
33.         if char in char_mapping:
34.             converted += char_mapping[char]
35.
36.     # 返回转换后的结果
37.     return converted
38.
39. # 应用转换函数
40. df['移动方向数值'] = df['移动方向'].apply(convert_direction)
41.
42. # 如果需要将新列放到原列后面
43. current_direction_index = df.columns.get_loc('移动方向')
```

```

44.     df.insert(current_direction_index + 1, '移动方向数值', df.pop('移动方向数值'))
45.
46.     # 假设你要转换的列名为‘台风起始时间’，‘台风结束时间’，‘当前台风时间’
47.     df['台风起始时间'] = pd.to_datetime(df['台风起始时间'])
48.     df['台风结束时间'] = pd.to_datetime(df['台风结束时间'])
49.     df['当前台风时间'] = pd.to_datetime(df['当前台风时间'])
50.     # 提取年份
51.     df['年份'] = df['当前台风时间'].dt.year
52.     df['月份'] = df['当前台风时间'].dt.month
53.     # 定义一个函数将经纬度转换为以.5 结尾的值
54.     def convert_to_half(value):
55.         if value % 1 < 0.25: # 小于 x.25
56.             return np.floor(value) + 0.5
57.         elif value % 1 < 0.75: # 在 x.25 和 x.75 之间
58.             return np.floor(value) + 0.5
59.         else: # 大于等于 x.75
60.             return np.ceil(value) - 0.5
61.
62.     # 应用函数转换经纬度
63.     df['调整后的经度'] = df['经度'].apply(convert_to_half)
64.     df['调整后的纬度'] = df['纬度'].apply(convert_to_half)
65.
66.     # 根据年份分类
67.     grouped = df.groupby('年份')[['调整后的经度', '调整后的纬度']].apply(lambda x: x.reset_index(drop=True))
68.     # 替换台风强度中的“超强台风（Super TY）”为“超强台风（SuperTY）”
69.     df['台风强度'] = df['台风强度'].replace('超强台风(Super TY)', '超强台风(SuperTY)')
70.
71.     # 删掉年份为 1945 的行
72.     df = df[df['年份'] != 1945]
73.     # 计算气压值不为 0 的平均值
74.     pressure_mean = df.loc[df['气压'] != 0, '气压'].mean()
75.     # 使用 loc 来进行替换，以避免警告
76.     df.loc[df['气压'] == 0, '气压'] = pressure_mean
77.     import pandas as pd
78.
79.     # 假设 df 是你的 DataFrame，且风速列名为‘风速’
80.     # 定义风速的区间和对应的台风强度类别
81.     bins = [-1, 10.8, 17.1, 24.4, 32.6, 41.4, 50.9, float('inf')]
82.     labels = ['弱热带低压或未知', '热带低压(TD)', '热带风暴(TS)', '强热带风暴(STS)', '台风(TY)', '强台风(STY)', '超强台风(SuperTY)']
83.
84.     # 仅针对“台风强度”为 NaN 且“风速”不为空或不为零的行进行分类
85.     mask = df['台风强度'].isnull() & (df['风速'].notnull() & (df['风速'] > 0))
86.
87.     # 使用 cut() 方法根据风速分类
88.     df.loc[mask, '台风强度'] = pd.cut(df.loc[mask, '风速'], bins=bins, labels=labels)

```

```

89.
90.    # 对海洋表面温度的空缺值进行插值填补
91.    df['海洋表面温度'] = df['海洋表面温度'].interpolate(method='linear')
92.
93.    # 如果想使用其他插值方法, 比如时间插值(需要时间索引)可以使用:
94.    #df['海洋表面温度'] = df['海洋表面温度'].interpolate(method='time')
95.    df['气压'] = df['气压'].interpolate(method='linear')# 对气压的空缺值进行线性插值
96.    # 通过布尔条件剔除台风强度和风速均为零或空值的行
97.    df = df[~((df['台风强度'].isnull() | (df['台风强度'] == 0)) &
98.              (df['风速'].isnull() | (df['风速'] == 0)))]
99.    # 创建台风强度的映射
100.   typhoon_intensity_mapping = {
101.       '弱热带低压或未知': 0,
102.       '热带低压(TD)': 1,
103.       '热带风暴(TS)': 2,
104.       '强热带风暴(STS)': 3,
105.       '台风(TY)': 4,
106.       '强台风(STY)': 5,
107.       '超强台风(SuperTY)': 6,
108.   }
109.
110.   # 将台风强度列转换为数值并保留原列
111.   df['台风强度数值'] = df['台风强度'].map(typhoon_intensity_mapping)
112.   current_intensity_index = df.columns.get_loc('台风强度')
113.   df.insert(current_intensity_index + 1, '台风强度数值', df.pop('台风强度数值'))
114.   # 创建台风强度的映射
115.   typhoon_intensity_mapping = {
116.       '弱热带低压或未知': 0,
117.       '热带低压(TD)': 1,
118.       '热带风暴(TS)': 2,
119.       '强热带风暴(STS)': 3,
120.       '台风(TY)': 4,
121.       '强台风(STY)': 5,
122.       '超强台风(SuperTY)': 6,
123.   }
124.
125.   # 将台风强度列转换为数值并保留原列
126.   df['台风强度数值'] = df['台风强度'].map(typhoon_intensity_mapping)
127.   current_intensity_index = df.columns.get_loc('台风强度')
128.   df.insert(current_intensity_index + 1, '台风强度数值', df.pop('台风强度数值'))
129.   # 将时间格式化为所需的字符串格式
130.   df['格式化时间'] = df['当前台风时间'].dt.strftime('%Y%m%d%H')
131.   # 获取 '当前台风时间' 的列索引
132.   current_time_index = df.columns.get_loc('当前台风时间')
133.
134.   # 将 '格式化时间' 列插入到 '当前台风时间' 的后面

```

```

135. df.insert(current_time_index + 1, '格式化时间', df.pop('格式化时间'))
136. # 计算按年份统计不同台风强度的频数
137. frequency_by_year = df.groupby(['年份', '台风强度数值']).size().unstack(fill_value=0)
138.
139. # 计算描述统计
140. stats_summary = frequency_by_year.describe()
141. # 显示描述统计
142. print("按年份统计的热带气旋频数的描述统计分析: ")
143. stats_summary
144. import os
145. import pandas as pd
146. # 定义文件保存路径
147. output_file_path = '/Users/z/Desktop/Mathor/处理后的数据.xlsx'
148. # 检查文件是否存在, 如果存在则删除
149. if os.path.exists(output_file_path):
150.     os.remove(output_file_path)
151. # 将 DataFrame 保存为 Excel 文件
152. df.to_excel(output_file_path, index=False)
153. print(f"数据已成功保存到 {output_file_path}")
154. # 计算按年份统计不同台风强度的频数
155. frequency_by_year = df.groupby(['年份', '台风强度数值']).size().unstack(fill_value=0)
156.
157. # 创建折线图
158. plt.figure(figsize=(12, 6))
159.
160. for column in frequency_by_year.columns:
161.     plt.plot(frequency_by_year.index, frequency_by_year[column], marker='o', label=f'强度 {column}')
162.
163. # 添加图例和标签
164. plt.title('1946—2023 年间不同台风强度的出现频次')
165. plt.xlabel('年份')
166. plt.ylabel('出现频次')
167. plt.xticks(frequency_by_year.index, rotation=45) # 旋转年份标签
168. plt.legend(title='台风强度', labels=['弱热带低压或未知', '热带低压(TD)', '热带风暴(TS)', '强热带风暴(STS)', '台风(TY)', '强台风(STY)', '超强台风(SuperTY)'])
169. plt.grid()
170.
171. # 保存图像
172. plt.savefig('1946—2023 年间不同台风强度的出现频次', dpi=600, bbox_inches='tight') # 设置分辨率和边距
173. # 显示图形
174. plt.tight_layout()
175. plt.show()
176. import pandas as pd
177. import matplotlib.pyplot as plt
178. import matplotlib.font_manager as fm
179.

```

```
180. # 使用 Seaborn 设置风格
181. import seaborn as sns
182. sns.set(style="whitegrid", font_scale=1.2)
183.
184. # 手动加载中文字体（这里以 SimHei 为例）
185. font_path = r'C:\Windows\Fonts\simhei.ttf' # 请确保路径正确
186. font_prop = fm.FontProperties(fname=font_path)
187. plt.rcParams['font.family'] = font_prop.get_name() # 设置全局字体
188.
189. # 创建图表
190. plt.figure(figsize=(10, 6))
191. plt.plot(annual_typhoon_counts.index, annual_typhoon_counts.values, marker='o', color="#2B7A78",
192.           markersize=6, linewidth=2.5, label='台风总数')
193.
194. # 设置标题和轴标签
195. plt.title('1946-2023 年间台风总数变化趋势', fontsize=16, weight='bold', pad=20, fontproperties=font_prop)
196. plt.xlabel('年份', fontsize=14, weight='bold', fontproperties=font_prop)
197. plt.ylabel('台风总数', fontsize=14, weight='bold', fontproperties=font_prop)
198.
199. # 显示网格
200. plt.grid(visible=True, linestyle='--', alpha=0.7)
201. plt.tight_layout(pad=2.0)
202.
203. # 保存图像
204. output_image_path = '1946-2023 年台风总数变化趋势.png'
205. plt.savefig(output_image_path, dpi=600, bbox_inches='tight', format='png') # 提高 DPI
206. plt.show()
207.
208. import pandas as pd
209. import numpy as np
210. from sklearn.cluster import KMeans
211. from sklearn.preprocessing import StandardScaler
212. from sklearn.metrics import silhouette_score
213. import matplotlib.pyplot as plt
214.
215. # 加载并选择特征
216. # 假设 df 包含台风强度、台风等级、风速、气压的列
217. X = df[['台风强度数值', '台风等级', '风速', '气压']]
218.
219. # 标准化数据
220. scaler = StandardScaler()
221. X_scaled = scaler.fit_transform(X)
222.
223. # 肘部法则
224. inertia = []
225. k_values = range(2, 11) # 设置 k 的范围为 2 到 10
```

```

226.
227.     for k in k_values:
228.         kmeans = KMeans(n_clusters=k, random_state=0)
229.         kmeans.fit(X_scaled)
230.         inertia.append(kmeans.inertia_)
231.
232.     # 绘制肘部法则图
233.     plt.figure(figsize=(8, 4))
234.     plt.plot(k_values, inertia, marker='o', linestyle='--', color='b')
235.     plt.title('肘部法则确定最优聚类数 k')
236.     plt.xlabel('聚类数 k')
237.     plt.ylabel('惯性 (Inertia)')
238.     plt.grid(True)
239.     # 保存图像
240.     plt.savefig('肘部法则确定最优聚类数 k', dpi=600, bbox_inches='tight') # 设置分辨率和边距
241.     plt.show()
242.
243. import os
244. os.environ["OMP_NUM_THREADS"] = "1" # 设置环境变量
245.
246. import pandas as pd
247. import numpy as np
248. import matplotlib.pyplot as plt
249. import seaborn as sns
250. from sklearn.preprocessing import StandardScaler
251. from sklearn.cluster import KMeans
252. from matplotlib.font_manager import FontProperties
253.
254. # 假设 df 是你的原始数据
255. # 1. 按台风名称分组，对数值列进行聚合
256. df_aggregated = df.groupby('台风中文名称').agg({
257.     '风速': 'mean',
258.     '气压': 'mean',
259.     '台风强度数值': 'mean',
260.     '台风等级': 'mean',
261. }).reset_index()
262.
263. # 2. 数据标准化
264. scaler = StandardScaler()
265. X_scaled = scaler.fit_transform(df_aggregated[['风速', '气压', '台风强度数值', '台风等级']])
266.
267. # 3. K-means 聚类
268. kmeans = KMeans(n_clusters=3, random_state=42, init='k-means++')
269. df_aggregated['Cluster'] = kmeans.fit_predict(X_scaled)
270.
271. # 4. 可视化聚类结果

```

```

272. sns.set(style="whitegrid")
273.
274. scatter = plt.scatter(df_aggregated['风速'], df_aggregated['气压'],
275.                         c=df_aggregated['Cluster'], cmap='viridis', alpha=0.6,
276.                         edgecolors='w', s=df_aggregated['台风等级'] * 5)
277.
278. # 添加图例
279. handles, labels = scatter.legend_elements()
280. legend = plt.legend(handles, labels, title="Clusters", loc='upper right', fontsize=10)
281.
282. # 添加标题和标签
283. plt.title("Typhoon Clustering Results", fontsize=14)
284. plt.xlabel("Wind Speed (m/s)", fontsize=12)
285. plt.ylabel("Pressure (hPa)", fontsize=12)
286.
287. # 增加主要网格线
288. plt.grid(color='gray', linestyle='--', linewidth=0.7)
289. plt.savefig('Typhoon Clustering Results.png', dpi=600, bbox_inches='tight') # 设置分辨率和边距
290. # 显示图形
291. plt.tight_layout()
292. plt.show()
293. import pandas as pd
294. import matplotlib.pyplot as plt
295. import seaborn as sns
296. # 假设 df 是包含所有台风数据的 DataFrame，并且有一个日期列可以用来筛选月份
297. # 先确保日期列是 datetime 类型
298. # df['当前台风时间'] = pd.to_datetime(df['当前台风时间'])
299.
300. # 设置字体和负号
301. plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来显示中文
302. plt.rcParams['axes.unicode_minus'] = False # 用来正常显示负号
303.
304.
305. # 过滤出七月和九月的数据
306. july_typhoons = df[df['当前台风时间'].dt.month == 7]
307. september_typhoons = df[df['当前台风时间'].dt.month == 9]
308.
309. # 对七月和九月的台风数据进行描述性统计分析
310. july_stats = july_typhoons[['风速', '气压', '台风强度数值']].describe()
311. september_stats = september_typhoons[['风速', '气压', '台风强度数值']].describe()
312. print("七月台风统计:")
313. july_stats
314. # 创建一个 1x2 的子图
315. fig, axes = plt.subplots(1, 2, figsize=(14, 4))
316.
317. # 绘制七月风速的箱型图

```

```
318. sns.boxplot(x=july_typhoons['风速'], ax=axes[0])
319. axes[0].set_title('七月台风风速箱型图', fontsize=14)
320. axes[0].set_xlabel('风速 (m/s)', fontsize=12)
321. axes[0].grid(axis='x', linestyle='--', alpha=0.7)
322.
323. # 绘制七月气压的箱型图
324. sns.boxplot(x=july_typhoons['气压'], ax=axes[1])
325. axes[1].set_title('七月台风气压箱型图', fontsize=14)
326. axes[1].set_xlabel('气压 (hPa)', fontsize=12)
327. axes[1].grid(axis='x', linestyle='--', alpha=0.7)
328.
329. # 调整布局
330. plt.tight_layout()
331.
332. # 保存图像
333. plt.savefig('七月风速与气压箱型图.png', dpi=600, bbox_inches='tight') # 设置分辨率和边距
334. plt.show() # 显示图形
335. # 创建一个 1x2 的子图
336. fig, axes = plt.subplots(1, 2, figsize=(14, 4))
337.
338. # 绘制九月风速的箱型图
339. sns.boxplot(x=september_typhoons['风速'], ax=axes[0])
340. axes[0].set_title('九月台风风速箱型图', fontsize=14)
341. axes[0].set_xlabel('风速 (m/s)', fontsize=12)
342. axes[0].grid(axis='x', linestyle='--', alpha=0.7)
343.
344. # 绘制九月气压的箱型图
345. sns.boxplot(x=september_typhoons['气压'], ax=axes[1])
346. axes[1].set_title('九月台风气压箱型图', fontsize=14)
347. axes[1].set_xlabel('气压 (hPa)', fontsize=12)
348. axes[1].grid(axis='x', linestyle='--', alpha=0.7)
349.
350. # 调整布局
351. plt.tight_layout()
352.
353. # 保存图像
354. plt.savefig('九月风速与气压箱型图.png', dpi=600, bbox_inches='tight') # 设置分辨率和边距
355. plt.show() # 显示图形
356. import matplotlib.pyplot as plt
357. import cartopy.crs as ccrs
358. import cartopy.feature as cfeature
359. import cartopy.io.shapereader as shapereader
360. # 过滤出七月份的台风数据
361. # 假设时间列名为 '当前台风时间', 且为 datetime 类型
362. df['当前台风时间'] = pd.to_datetime(df['当前台风时间'])
```

```

363. july_typhoons = df[(df['当前台风时间'].dt.month == 7) & (df['当前台风时间'].dt.year >= 2000) & (df['当前台风时间'].dt.year <= 2023)]
364.
365. # 检查七月份历史台风数量
366. num_typhoons = july_typhoons['台风中文名称'].nunique()
367. print(f"七月份历史台风数量: {num_typhoons}")
368. # 绘制台风路径
369. plt.figure(figsize=(9, 5), dpi=300) # 设置图像大小和分辨率
370. ax = plt.axes(projection=ccrs.PlateCarree())
371. ax.set_extent([95, 180, 2, 58], crs=ccrs.PlateCarree()) # 设置经纬度范围
372. # 添加底图
373. ax.add_feature(cfeature.LAND, facecolor='lightgray') # 添加陆地
374. ax.add_feature(cfeature.OCEAN, facecolor='lightblue') # 添加海洋
375. ax.add_feature(cfeature.BORDERS, linestyle=':', linewidth=1) # 添加国家边界
376. ax.add_feature(cfeature.COASTLINE) # 添加海岸线
377. ax.add_feature(cfeature.LAKES, alpha=0.5) # 添加湖泊
378. ax.add_feature(cfeature.RIVERS) # 添加河流
379. # 添加中国省份边界
380. shp_info = shpreader.Reader("C:\\\\Users\\\\z\\\\Downloads\\\\中华人民共和国\\\\中华人民共和国.shp") # 替换为你的省界 shapefile 路径
381. ax.add_geometries(shp_info.geometries(), crs=ccrs.PlateCarree(), edgecolor='black', facecolor='none', linewidth=0.8)
382. # 绘制台风路径
383. for name, group in july_typhoons.groupby('台风中文名称'):
384.     ax.plot(group['经度'], group['纬度'], marker='o', markersize=2, linestyle='-', label=name, alpha=0.7)
385. # 添加图例
386. plt.title('2000-2023 年七月台风路径及途径省份', fontsize=14)
387. plt.xlabel('经度', fontsize=14)
388. plt.ylabel('纬度', fontsize=14)
389. plt.legend(title='台风中文名称', fontsize=2, loc='upper right', bbox_to_anchor=(1, 1))
390.
391. # 保存图像
392. plt.savefig('七月台风路径及省份图.png', dpi=600, bbox_inches='tight') # 设置分辨率和边距
393. plt.show() # 显示图形
394. import matplotlib.pyplot as plt
395. import cartopy.crs as ccrs
396. import cartopy.feature as cfeature
397. import cartopy.io.shapereader as shpreader
398. import pandas as pd
399.
400. # 过滤出九月份的台风数据
401. # 假设时间列名为 '当前台风时间'，且为 datetime 类型
402. df['当前台风时间'] = pd.to_datetime(df['当前台风时间'])
403. september_typhoons = df[(df['当前台风时间'].dt.month == 9) & (df['当前台风时间'].dt.year >= 2000) & (df['当前台风时间'].dt.year <= 2023)]
404.
405. # 绘制台风路径
406. plt.figure(figsize=(9, 5), dpi=300) # 设置图像大小和分辨率

```

```

407. ax = plt.axes(projection=ccrs.PlateCarree())
408. ax.set_extent([95, 180, 2, 58], crs=ccrs.PlateCarree()) # 设置经纬度范围
409.
410. # 添加底图
411. ax.add_feature(cfeature.LAND, facecolor='lightgray') # 添加陆地
412. ax.add_feature(cfeature.OCEAN, facecolor='lightblue') # 添加海洋
413. ax.add_feature(cfeature.BORDERS, linestyle=':', linewidth=1) # 添加国家边界
414. ax.add_feature(cfeature.COASTLINE) # 添加海岸线
415. ax.add_feature(cfeature.LAKES, alpha=0.5) # 添加湖泊
416. ax.add_feature(cfeature.RIVERS) # 添加河流
417.
418. # 添加中国省份边界
419. shp_info = shpreader.Reader("C:\\\\Users\\\\z\\\\Downloads\\\\中华人民共和国\\\\中华人民共和国.shp") # 替换为你的省界 shapefile 路径
420. ax.add_geometries(shp_info.geometries(), crs=ccrs.PlateCarree(), edgecolor='black', facecolor='none', linewidth=0.8)
421.
422. # 绘制九月份台风路径
423. for name, group in september_typhoons.groupby('台风中文名称'):
424.     ax.plot(group['经度'], group['纬度'], marker='o', markersize=2, linestyle='-', label=name, alpha=0.7) # 修改 markersize 为 2
425.
426. # 添加图例
427. plt.title('2000-2023 年九月台风路径及途径省份', fontsize=14)
428. plt.xlabel('经度', fontsize=14)
429. plt.ylabel('纬度', fontsize=14)
430. plt.legend(title='台风中文名称', fontsize=1.5, loc='upper right', bbox_to_anchor=(1, 1))
431.
432. # 保存图像
433. plt.savefig('九月台风路径及途径省份省份.png', dpi=600, bbox_inches='tight') # 设置分辨率和边距
434. plt.show() # 显示图形
435.
436. # 检查九月份历史台风数量
437. num_typhoons_september = september_typhoons['台风中文名称'].nunique()
438. print(f"九月份历史台风数量: {num_typhoons_september}")
439. import geopandas as gpd
440. import pandas as pd
441. from shapely.geometry import Point
442. from dbfread import DBF
443. # 读取 .dbf 文件
444. dbf_file_path = "C:\\\\Users\\\\z\\\\Downloads\\\\中华人民共和国\\\\中华人民共和国.dbf"
445. dbf_data = DBF(dbf_file_path, encoding='utf-8') # 尝试使用 'gbk' 或 'gb2312'
446. # 将 dbf 数据转换为 DataFrame
447. province_df = pd.DataFrame(iter(dbf_data))
448. # 读取台风数据 DataFrame (确保已存在)
449. # 假设 df 包含经度和纬度, 并且已经过滤了七月份的数据
450. df['当前台风时间'] = pd.to_datetime(df['当前台风时间'], errors='coerce') # 转换为日期时间格式
451. july_typhoons = df[(df['当前台风时间'].dt.month == 7) &

```

```

452.             (df['当前台风时间'].dt.year >= 2000) &
453.             (df['当前台风时间'].dt.year <= 2023)]
454.     # 创建 GeoDataFrame
455.     geometry = [Point(xy) for xy in zip(july_typhoons['经度'], july_typhoons['纬度'])]
456.     typhoon_gdf = gpd.GeoDataFrame(july_typhoons, geometry=geometry, crs="EPSG:4326")
457.     # 读取省份 shapefile
458.     shp_file_path = "C:\\\\Users\\\\z\\\\Downloads\\\\中华人民共和国\\\\中华人民共和国.shp"
459.     provinces_gdf = gpd.read_file(shp_file_path)
460.     # 检查并修复无效的几何图形
461.     provinces_gdf['geometry'] = provinces_gdf.geometry.buffer(0)
462.     # 将 DataFrame 中的名称列与 provinces_gdf 进行合并
463.     provinces_gdf = provinces_gdf.merge(province_df[['adcode', 'name']], on='adcode', how='left')
464.
465.     # 统计台风登陆的省份
466.     province_counts = {}
467.     for _, typhoon in typhoon_gdf.iterrows():
468.         typhoon_geometry = typhoon.geometry
469.         for _, province in provinces_gdf.iterrows():
470.             if typhoon_geometry.within(province.geometry):
471.                 province_name = province['name_y'] # 替换为实际的省份名称列
472.                 province_counts[province_name] = province_counts.get(province_name, 0) + 1
473.
474.     # 将结果按降序排列
475.     sorted_province_counts = sorted(province_counts.items(), key=lambda x: x[1], reverse=True)
476.
477.     # 提取省份和登陆次数
478.     provinces, counts = zip(*sorted_province_counts)
479.     provinces, counts
480.     import seaborn as sns
481.     import matplotlib.pyplot as plt
482.     from matplotlib import rcParams
483.     from matplotlib import font_manager
484.     # 指定字体路径（根据实际字体文件位置替换路径）
485.     font_path = "C:/Windows/Fonts/msyh.ttc" # 微软雅黑字体路径
486.     font_prop = font_manager.FontProperties(fname=font_path)
487.
488.     # 设置字体属性
489.     rcParams['font.sans-serif'] = [font_prop.get_name()] # 使用指定路径的字体
490.     rcParams['axes.unicode_minus'] = False # 解决坐标轴负号显示问题
491.     # 使用 Seaborn 设置风格
492.     sns.set_style("whitegrid")
493.     # 使用 Seaborn 设置风格
494.     sns.set_style("whitegrid")
495.     fig, ax = plt.subplots(figsize=(10, 6), dpi=300)
496.     # 绘制条形图
497.     bars = ax.bars(provinces, counts, color='skyblue', edgecolor='black')

```

```

498.     ax.invert_yaxis()
499.     # 添加标签和标题
500.     ax.set_xlabel("台风登陆次数", fontsize=14, labelpad=10, fontproperties=font_prop)
501.     ax.set_ylabel("省份", fontsize=14, labelpad=10, fontproperties=font_prop)
502.     ax.set_title("2000-2023 年 7 月台风登陆中国各省份的次数", fontsize=16, pad=15, fontproperties=font_prop)
503.     # 强制设置 y 轴刻度标签
504.     ax.set_yticks(range(len(provinces)))
505.     ax.set_yticklabels(provinces, fontproperties=font_prop)
506.     plt.tight_layout()
507.     plt.savefig("2000-2023 年 7 月台风登陆中国各省份的次数.png", format="png", dpi=600, bbox_inches='tight')
508.     plt.show()
509.     # 修改后的代码筛选九月的台风数据
510.     # 读取 .dbf 文件
511.     dbf_file_path = "C:\\\\Users\\\\z\\\\Downloads\\\\中华人民共和国\\\\中华人民共和国.dbf"
512.     dbf_data = DBF(dbf_file_path, encoding='utf-8') # 尝试使用 'gbk' 或 'gb2312'
513.
514.     # 将 dbf 数据转换为 DataFrame
515.     province_df = pd.DataFrame(iter(dbf_data))
516.
517.     # 读取台风数据 DataFrame (确保已存在)
518.     # 假设 df 包含经度和纬度，并且已经过滤了九月份的数据
519.     df['当前台风时间'] = pd.to_datetime(df['当前台风时间'], errors='coerce') # 转换为日期时间格式
520.     september_typhoons = df[(df['当前台风时间'].dt.month == 9) &
521.                                 (df['当前台风时间'].dt.year >= 2000) &
522.                                 (df['当前台风时间'].dt.year <= 2023)]
523.
524.     # 创建 GeoDataFrame
525.     geometry = [Point(xy) for xy in zip(september_typhoons['经度'], september_typhoons['纬度'])]
526.     typhoon_gdf = gpd.GeoDataFrame(september_typhoons, geometry=geometry, crs="EPSG:4326")
527.
528.     # 读取省份 shapefile
529.     shp_file_path = "C:\\\\Users\\\\z\\\\Downloads\\\\中华人民共和国\\\\中华人民共和国.shp"
530.     provinces_gdf = gpd.read_file(shp_file_path)
531.
532.     # 检查并修复无效的几何图形
533.     provinces_gdf['geometry'] = provinces_gdf.geometry.buffer(0)
534.
535.     # 将 DataFrame 中的名称列与 provinces_gdf 进行合并
536.     provinces_gdf = provinces_gdf.merge(province_df[['adcode', 'name']], on='adcode', how='left')
537.
538.     # 统计台风登陆的省份
539.     province_counts = {}
540.     for _, typhoon in typhoon_gdf.iterrows():
541.         typhoon_geometry = typhoon.geometry
542.         for _, province in provinces_gdf.iterrows():
543.             if typhoon_geometry.within(province.geometry):

```

```

544.         province_name = province['name_y'] # 替换为实际的省份名称列
545.         province_counts[province_name] = province_counts.get(province_name, 0) + 1
546.
547.     # 将结果按降序排列
548.     sorted_province_counts = sorted(province_counts.items(), key=lambda x: x[1], reverse=True)
549.
550.     # 提取省份和登陆次数
551.     provinces, counts = zip(*sorted_province_counts)
552.     provinces, counts
553.     # 使用 Seaborn 设置风格
554.     sns.set_style("whitegrid")
555.     fig, ax = plt.subplots(figsize=(10, 6), dpi=300)
556.     # 绘制条形图
557.     bars = ax.barr(provinces, counts, color='skyblue', edgecolor='black')
558.     ax.invert_yaxis()
559.     # 添加标签和标题
560.     ax.set_xlabel("台风登陆次数", fontsize=14, labelpad=10, fontproperties=font_prop)
561.     ax.set_ylabel("省份", fontsize=14, labelpad=10, fontproperties=font_prop)
562.     ax.set_title("2000-2023 年 9 月台风登陆中国各省份的次数", fontsize=16, pad=15, fontproperties=font_prop)
563.     # 强制设置 y 轴刻度标签
564.     ax.set_yticks(range(len(provinces)))
565.     ax.set_yticklabels(provinces, fontproperties=font_prop)
566.     plt.tight_layout()
567.     plt.savefig("2000-2023 年 9 月台风登陆中国各省份的次数.png", format="png", dpi=600, bbox_inches='tight')
568.     plt.show()
569.     import os
570.     import pandas as pd
571.     import numpy as np
572.     import matplotlib.pyplot as plt
573.     import seaborn as sns
574.     from sklearn.preprocessing import StandardScaler
575.     from sklearn.cluster import KMeans
576.     from shapely.geometry import Point
577.     import geopandas as gpd
578.     from dbfread import DBF
579.
580.     # 设定环境变量
581.     os.environ["OMP_NUM_THREADS"] = "1"
582.
583.     # 读取省份 shapefile
584.     shp_file_path = "C:\\\\Users\\\\z\\\\Downloads\\\\中华人民共和国\\\\中华人民共和国.shp"
585.     provinces_gdf = gpd.read_file(shp_file_path)
586.     # 将 DataFrame 中的名称列与 provinces_gdf 进行合并
587.     provinces_gdf = provinces_gdf.merge(province_df[['adcode', 'name']], on='adcode', how='left')
588.     provinces_gdf['geometry'] = provinces_gdf.geometry.buffer(0) # 检查并修复无效的几何图形
589.

```

```

590. # 读取 .dbf 文件
591. dbf_file_path = "C:\\\\Users\\\\z\\\\Downloads\\\\中华人民共和国\\\\中华人民共和国.dbf"
592. dbf_data = DBF(dbf_file_path, encoding='utf-8')
593. province_df = pd.DataFrame(iter(dbf_data))
594.
595. # 读取台风数据 DataFrame (假设 df 已存在并包含九月数据)
596. df['当前台风时间'] = pd.to_datetime(df['当前台风时间'], errors='coerce')
597. september_typhoons = df[(df['当前台风时间'].dt.month == 9) &
598.                           (df['当前台风时间'].dt.year.between(2000, 2023))]
599.
600. # 1. 按台风名称分组, 对数值列进行聚合
601. df_aggregated = df.groupby('台风中文名称').agg({
602.     '风速': 'mean',
603.     '气压': 'mean',
604.     '台风强度数值': 'mean',
605.     '台风等级': 'mean',
606. }).reset_index()
607.
608. # 2. 数据标准化
609. scaler = StandardScaler()
610. X_scaled = scaler.fit_transform(df_aggregated[['风速', '气压', '台风强度数值', '台风等级']])
611.
612. # 3. K-means 聚类
613. kmeans = KMeans(n_clusters=3, random_state=42, init='k-means++')
614. df_aggregated['Cluster'] = kmeans.fit_predict(X_scaled)
615.
616. # 4. 将聚类结果合并到九月份台风数据中
617. september_typhoons = september_typhoons.merge(df_aggregated[['台风中文名称', 'Cluster']], on='台风中文名称', how='left')
618.
619. # 5. 统计每个聚类的途径省份
620. september_typhoons['途经省份'] = None
621.
622. # 创建 GeoDataFrame
623. geometry = [Point(xy) for xy in zip(september_typhoons['经度'], september_typhoons['纬度'])]
624. typhoon_gdf = gpd.GeoDataFrame(september_typhoons, geometry=geometry, crs="EPSG:4326")
625.
626.
627. # 遍历每个台风
628. for name, group in typhoon_gdf.groupby('Cluster'):
629.     provinces_list = []
630.
631.     for _, point in group.iterrows():
632.         point_geom = point.geometry
633.
634.         for _, province in provinces_gdf.iterrows():
635.             if province.geometry.contains(point_geom):

```

```

636.         province_name = province['name_y'] # 确保使用省份名称列
637.         provinces_list.append(province_name)
638.
639.     # 将结果添加到 DataFrame 中
640.     unique_provinces = list(set(provinces_list))
641.     # 过滤掉 None 值
642.     unique_provinces = [p for p in unique_provinces if p is not None]
643.
644.     typhoon_gdf.loc[typhoon_gdf['Cluster'] == name, '途经省份'] = ", ".join(unique_provinces)
645.
646. # 输出每个聚类的途经省份情况
647. print('九月夏台风不同类别途径省份情况')
648. cluster_provinces = typhoon_gdf.groupby('Cluster')['途经省份'].unique().reset_index()
649. print(cluster_provinces)
650. import os
651. import pandas as pd
652. import numpy as np
653. import matplotlib.pyplot as plt
654. import seaborn as sns
655. from sklearn.preprocessing import StandardScaler
656. from sklearn.cluster import KMeans
657. from shapely.geometry import Point
658. import geopandas as gpd
659. from dbfread import DBF
660. # 设定环境变量
661. os.environ["OMP_NUM_THREADS"] = "1"
662. # 读取省份 shapefile
663. shp_file_path = "C:\\\\Users\\\\z\\\\Downloads\\\\中华人民共和国\\\\中华人民共和国.shp"
664. provinces_gdf = gpd.read_file(shp_file_path)
665. # 将 DataFrame 中的名称列与 provinces_gdf 进行合并
666. provinces_gdf = provinces_gdf.merge(province_df[['adcode', 'name']], on='adcode', how='left')
667. provinces_gdf['geometry'] = provinces_gdf.geometry.buffer(0) # 检查并修复无效的几何图形
668. # 读取 .dbf 文件
669. dbf_file_path = "C:\\\\Users\\\\z\\\\Downloads\\\\中华人民共和国\\\\中华人民共和国.dbf"
670. dbf_data = DBF(dbf_file_path, encoding='utf-8')
671. province_df = pd.DataFrame(iter(dbf_data))
672. # 读取台风数据 DataFrame (假设 df 已存在并包含九月数据)
673. df['当前台风时间'] = pd.to_datetime(df['当前台风时间'], errors='coerce')
674. september_typhoons = df[(df['当前台风时间'].dt.month == 7) &
675.                             (df['当前台风时间'].dt.year.between(2000, 2023))]
676. # 1. 按台风名称分组, 对数值列进行聚合
677. df_aggregated = df.groupby('台风中文名称').agg({
678.     '风速': 'mean',
679.     '气压': 'mean',
680.     '台风强度数值': 'mean',
681.     '台风等级': 'mean',

```

```

682.     }).reset_index()
683.     # 2. 数据标准化
684.     scaler = StandardScaler()
685.     X_scaled = scaler.fit_transform(df_aggregated[['风速', '气压', '台风强度数值', '台风等级']])
686.     # 3. K-means 聚类
687.     kmeans = KMeans(n_clusters=3, random_state=42, init='k-means++')
688.     df_aggregated['Cluster'] = kmeans.fit_predict(X_scaled)
689.     # 4. 将聚类结果合并到九月份台风数据中
690.     september_typhoons = september_typhoons.merge(df_aggregated[['台风中文名称', 'Cluster']], on='台风中文名称', how='left')
691.     # 5. 统计每个聚类的途径省份
692.     september_typhoons['途经省份'] = None
693.     # 创建 GeoDataFrame
694.     geometry = [Point(xy) for xy in zip(september_typhoons['经度'], september_typhoons['纬度'])]
695.     typhoon_gdf = gpd.GeoDataFrame(september_typhoons, geometry=geometry, crs="EPSG:4326")
696.     # 遍历每个台风
697.     for name, group in typhoon_gdf.groupby('Cluster'):
698.         provinces_list = []
699.
700.         for _, point in group.iterrows():
701.             point_geom = point.geometry
702.
703.             for _, province in provinces_gdf.iterrows():
704.                 if province.geometry.contains(point_geom):
705.                     province_name = province['name_y'] # 确保使用省份名称列
706.                     provinces_list.append(province_name)
707.             # 将结果添加到 DataFrame 中
708.             unique_provinces = list(set(provinces_list))
709.             # 过滤掉 None 值
710.             unique_provinces = [p for p in unique_provinces if p is not None]
711.             typhoon_gdf.loc[typhoon_gdf['Cluster'] == name, '途经省份'] = ", ".join(unique_provinces)
712.     # 输出每个聚类的途经省份情况
713.     cluster_provinces = typhoon_gdf.groupby('Cluster')['途经省份'].unique().reset_index()
714.     print('七月夏台风不同类别途径省份情况')
715.     print(cluster_provinces)

```

## No. 2 Filename: RNN.ipynb

```

1.     # 读取贝碧嘉台风数据
2.     df_BEBINCA = pd.read_excel('贝碧嘉数据.xlsx')
3.
4.     # 转换时间列为日期时间格式
5.     df_BEBINCA['时间'] = pd.to_datetime(df_BEBINCA['时间'])
6.
7.     # 转换时间为步长，并将其加入到特征中
8.     df_BEBINCA['时间步长'] = (df_BEBINCA['时间'] - df_BEBINCA['时间'].min()).dt.total_seconds() / 21600 # 每 6 小时一次
9.
10.    # 使用经度、纬度和时间步长作为输入
11.    data_BEBINCA = df_BEBINCA[['经度', '纬度', '时间步长']].values

```

```

12.     data_BEBINCA = scaler.transform(data_BEBINCA) # 使用之前的缩放器进行缩放
13.
14.     # 创建输入数据集
15.     X_BEBINCA, _ = create_dataset(data_BEBINCA, time_step=20) # 不需要返回 y
16.
17.     # 使用训练好的模型进行预测
18.     predictions_BEBINCA = best_model.predict(X_BEBINCA)
19.
20.     # 反归一化预测结果
21.     predictions_unscaled_BEBINCA = scaler.inverse_transform(np.concatenate((predictions_BEBINCA, np.zeros((predictions_BEBINCA.shape[0], 1))), axis=1))[:, :2]
22.
23.     # 将预测结果保存到 DataFrame
24.     df_BEBINCA['预测经度'] = np.nan
25.     df_BEBINCA['预测纬度'] = np.nan
26.
27.     # 将预测结果放入 DataFrame 中
28.     start_index = 20 # 从第 20 个位置开始填充预测结果
29.     df_BEBINCA.iloc[start_index:start_index + len(predictions_unscaled_BEBINCA), df_BEBINCA.columns.get_loc('预测经度')] = predictions_unscaled_BEBINCA[:, 0]
30.     df_BEBINCA.iloc[start_index:start_index + len(predictions_unscaled_BEBINCA), df_BEBINCA.columns.get_loc('预测纬度')] = predictions_unscaled_BEBINCA[:, 1]
31.
32.     # 保存结果到 Excel 文件
33.     output_file = 'RNN 贝碧嘉预测.xlsx'
34.     df_BEBINCA.to_excel(output_file, index=False)
35.
36.     print(f"预测结果已保存到 {output_file}")
37.     import pandas as pd
38.     import numpy as np
39.     import matplotlib.pyplot as plt
40.     from scipy.interpolate import interp1d
41.     import datetime
42.
43.     # 设置中文字体
44.     plt.rcParams['font.sans-serif'] = ['SimHei'] # 使用黑体显示中文
45.     plt.rcParams['axes.unicode_minus'] = False # 使负号显示正常
46.
47.     # 1. 读取 RNN 预测结果 Excel 文件
48.     rnn_file_path = "C:\\\\Users\\\\z\\\\Desktop\\\\Mathor\\\\RNN 贝碧嘉预测.xlsx"
49.     rnn_data = pd.read_excel(rnn_file_path)
50.
51.     # 假设 Excel 文件包含以下列：实际经度、实际纬度、RNN 预测经度、RNN 预测纬度、时间
52.     # 根据你的文件实际列名来调整
53.     actual_lon_rnn = rnn_data['经度']
54.     actual_lat_rnn = rnn_data['纬度']

```

```

55.     rnn_predicted_lon = rnn_data['预测经度']
56.     rnn_predicted_lat = rnn_data['预测纬度']
57.     time_rnn = pd.to_datetime(rnn_data['时间'])
58.
59.     # 2. 插值处理
60.     # 提取 2024 年 9 月 12 日到 2024 年 9 月 18 日的日期和对应的时间
61.     target_dates = [datetime.datetime(2024, 9, day, 14, 0) for day in range(12, 19)]
62.     target_times_numeric = [(target_date - datetime.datetime(1970, 1, 1)).total_seconds() for target_date in target_dates]
63.
64.     # 转换时间为浮点数形式（时间戳）
65.     time_numeric_rnn = time_rnn.astype(np.int64) / 1e9 # 转换为秒
66.
67.     # 使用线性插值
68.     interp_actual_lon_rnn = interp1d(time_numeric_rnn, actual_lon_rnn, fill_value="extrapolate")
69.     interp_actual_lat_rnn = interp1d(time_numeric_rnn, actual_lat_rnn, fill_value="extrapolate")
70.     interp_rnn_predicted_lon = interp1d(time_numeric_rnn, rnn_predicted_lon, fill_value="extrapolate")
71.     interp_rnn_predicted_lat = interp1d(time_numeric_rnn, rnn_predicted_lat, fill_value="extrapolate")
72.
73.     # 获取插值结果
74.     interp_actual_lon_values_rnn = interp_actual_lon_rnn(target_times_numeric)
75.     interp_actual_lat_values_rnn = interp_actual_lat_rnn(target_times_numeric)
76.     interp_rnn_predicted_lon_values = interp_rnn_predicted_lon(target_times_numeric)
77.     interp_rnn_predicted_lat_values = interp_rnn_predicted_lat(target_times_numeric)
78.
79.     # 3. 绘图
80.     plt.figure(figsize=(10, 6))
81.
82.     # 找到 RNN 预测经度和纬度都存在值的起始索引
83.     valid_start_index_rnn = None
84.     for i in range(len(rnn_predicted_lon)):
85.         if not pd.isnull(rnn_predicted_lon[i]) and not pd.isnull(rnn_predicted_lat[i]):
86.             valid_start_index_rnn = i
87.             break
88.
89.     # 如果找到了有效的起始索引
90.     if valid_start_index_rnn is not None:
91.         # 只绘制从有效起始索引开始的数据
92.         plt.plot(actual_lon_rnn[valid_start_index_rnn:], actual_lat_rnn[valid_start_index_rnn:], label=' 实际路径',
93.                  color='blue', marker='o')
94.
95.         # 添加时间标记
96.         for i, txt in enumerate(target_dates):
97.             plt.annotate(txt.strftime('%m-%d %H:%M'), (interp_actual_lon_values_rnn[i], interp_actual_lat_values_rnn[i]), textcoords="offset points", xytext=(0,10), ha='center')

```

```

98.
99.     plt.title('贝碧嘉台风实际路径与 RNN 预测路径')
100.    plt.xlabel('经度')
101.    plt.ylabel('纬度')
102.    plt.legend()
103.    plt.grid()
104.    plt.savefig('贝碧嘉台风 RNN 路径预测.png')
105.    plt.show()
106.
107.    # 输出插值结果
108.    interpolation_results_rnn = pd.DataFrame({
109.        '日期': target_dates,
110.        '插值经度（实际）': interp_actual_lon_values_rnn,
111.        '插值纬度（实际）': interp_actual_lat_values_rnn,
112.        '插值经度（RNN）': interp_rnn_predicted_lon_values,
113.        '插值纬度（RNN）': interp_rnn_predicted_lat_values
114.    })
115.
116.    interpolation_results_rnn.to_excel('贝碧嘉台风 RNN 插值结果.xlsx', index=False)
117.    import pandas as pd
118.    import numpy as np
119.    import matplotlib.pyplot as plt
120.    from dtw import dtw
121.
122.    # 1. 读取 RNN 预测结果 Excel 文件
123.    rnn_file_path = "C:\\\\Users\\\\z\\\\Desktop\\\\Mathor\\\\RNN 贝碧嘉预测.xlsx"
124.    rnn_data = pd.read_excel(rnn_file_path)
125.
126.    # 假设 Excel 文件包含以下列：实际经度、实际纬度、RNN 预测经度、RNN 预测纬度、时间
127.    actual_lon_rnn = rnn_data['经度']
128.    actual_lat_rnn = rnn_data['纬度']
129.    rnn_predicted_lon = rnn_data['预测经度']
130.    rnn_predicted_lat = rnn_data['预测纬度']
131.
132.    # 2. 选择需要对比的有效路径（去除空值）
133.    valid_indices_lon = ~pd.isnull(actual_lon_rnn) & ~pd.isnull(rnn_predicted_lon)
134.    valid_indices_lat = ~pd.isnull(actual_lat_rnn) & ~pd.isnull(rnn_predicted_lat)
135.
136.    actual_lon_valid = actual_lon_rnn[valid_indices_lon].to_numpy()
137.    predicted_lon_valid = rnn_predicted_lon[valid_indices_lon].to_numpy()
138.
139.    actual_lat_valid = actual_lat_rnn[valid_indices_lat].to_numpy()
140.    predicted_lat_valid = rnn_predicted_lat[valid_indices_lat].to_numpy()
141.
142.    # 3. 计算 DTW 距离
143.    dtw_distance_lon = dtw.distance(actual_lon_valid, predicted_lon_valid)

```

```

144.     dtw_distance_lat = dtw.distance(actual_lat_valid, predicted_lat_valid)
145.
146.     # 输出 DTW 距离
147.     print(f"经度的 DTW 距离: {dtw_distance_lon:.4f}")
148.     print(f"纬度的 DTW 距离: {dtw_distance_lat:.4f}")
149.
150.     # 4. 可视化 DTW 路径
151.     alignment_lon = dtw.warping_path(actual_lon_valid, predicted_lon_valid)
152.     alignment_lat = dtw.warping_path(actual_lat_valid, predicted_lat_valid)
153.
154.     # 设置中文字体
155.     plt.rcParams['font.sans-serif'] = ['SimHei']
156.     plt.rcParams['axes.unicode_minus'] = False
157.
158.     # 设置字体和大小
159.     plt.rcParams['font.size'] = 12
160.     plt.rcParams['axes.titlesize'] = 14
161.     plt.rcParams['legend.fontsize'] = 12
162.     plt.rcParams['xtick.labelsize'] = 10
163.     plt.rcParams['ytick.labelsize'] = 10
164.
165.     # 绘制经度对比图
166.     plt.figure(figsize=(10, 6))
167.     plt.plot(actual_lon_valid, label='实际经度', color='royalblue', linewidth=2)
168.     plt.plot(predicted_lon_valid, label='RNN 预测经度', color='tomato', linestyle='--', linewidth=2)
169.
170.     # 绘制经度 DTW 对齐路径
171.     for i, j in alignment_lon:
172.         plt.plot([i, j], [actual_lon_valid[i], predicted_lon_valid[j]], color='gray', linestyle='dotted')
173.
174.     plt.title('RNN 预测经度路径与实际路径的 DTW 对比', fontsize=16)
175.     plt.xlabel('时间步', fontsize=12)
176.     plt.ylabel('经度', fontsize=12)
177.     plt.legend(loc='upper right', fontsize=12)
178.     plt.grid(color='gray', linestyle='--', linewidth=0.5)
179.
180.     # 调整边距
181.     plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
182.
183.     # 保存经度图像
184.     plt.savefig('RNN 预测经度路径与实际路径 DTW 对比.png', dpi=500, bbox_inches='tight')
185.     plt.show()
186.
187.     # 绘制纬度对比图
188.     plt.figure(figsize=(10, 6))
189.     plt.plot(actual_lat_valid, label='实际纬度', color='royalblue', linewidth=2)

```

```
190. plt.plot(predicted_lat_valid, label='RNN 预测纬度', color='tomato', linestyle='--', linewidth=2)
191.
192. # 绘制纬度 DTW 对齐路径
193. for (i, j) in alignment_lat:
194.     plt.plot([i, j], [actual_lat_valid[i], predicted_lat_valid[j]], color='gray', linestyle='dotted')
195.
196. plt.title('RNN 预测纬度路径与实际路径的 DTW 对比', fontsize=16)
197. plt.xlabel('时间步', fontsize=12)
198. plt.ylabel('纬度', fontsize=12)
199. plt.legend(loc='upper right', fontsize=12)
200. plt.grid(color='gray', linestyle='--', linewidth=0.5)
201.
202. # 调整边距
203. plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
204.
205. # 保存纬度图像
206. plt.savefig('RNN 预测纬度路径与实际路径 DTW 对比.png', dpi=500, bbox_inches='tight')
207. plt.show()
```