基于 LSTM 神经网络的空气质量多步预测

摘要

随着工业、交通、农业等的大规模发展，多种大气污染物的排放增多，导致了一定程度的空气污染。本文筛选了影响 PM2.5 浓度的因素并构建了多步预测模型，对 PM2.5 浓度、AQI 指数以及预警情况做出预测。

针对问题一，首先分别使用阈值限制法和滑动窗口法与 3σ 准则识别出数据异常点。接着计算出各种空气质量下各个指标的均值，用均值替代筛选出的异常值以及数据缺失值。在数据预处理后，使用随机森林回归模型，对每个指标袋外数据添加随机噪声干扰后计算其袋外数据误差并以此量化不同指标对 PM2.5 浓度影响程度。同时，也通过 Pearson 相关系数的绝对值量化影响程度。对两组影响程度数据进行 Z-score 标准化后加权求和计算最终得分，按得分高低筛选出影响较大的因素为 PM10 浓度，CO 浓度，气温，NO_2 浓度，SO_2 浓度和气压。

针对问题二，划分数据的前 70% 为训练集，后 30% 为测试集。利用问题一中筛选的六个影响因素作为模型的输入，PM2.5 浓度作为模型的输出来训练支持向量机回归模型和 LSTM 循环神经网络模型并分别对 PM2.5 浓度进行多步预测。使用均方根误差（RMSE）评估预测效果，比较两个模型的均方根误差，发现 LSTM 循环神经网络的均方根误差较小，其 3 步预测模型的 RMSE 为 8.96，12 步预测模型的 RMSE 为 17.34。观察到两个模型中都有预测步长越长，均方根误差越大，预测效果相对越差的情况。基于这个考虑，我们选用多步预测相结合的方法预测给定时间的 PM2.5 浓度。

针对问题三，首先应用随机森林回归模型，在气象因素中筛选影响 AQI 的因素，得到风速、降水和气温对 AQI 影响较大。仍旧划分数据的前 70% 为训练集，后 30% 为测试集，选用所有污染物指标与筛选出的三个气象指标训练支持向量机回归模型和 LSTM 神经网络模型。依据 RMSE 评估出 LSTM 神经网络模型的预测效果更好。并且在预测 AQI 时依旧存在着预测步长变长，预测效果降低的问题，因此利用两种模型并采用第二问提出的多步预测相结合预测出给定时间的 AQI 指数，结合预警等级的判断标准得到给定时间的空气质量预警等级。两种算法得到的空气质量预警等级完全一致，仅在 2023 年 5 月 2 日空气质量存在蓝色预警，其余时刻，空气质量不存在预警。

最后，使用两种预测模型应用于湖北省武汉市和恩施市的空气质量预测。将 2023 年 4 月 27 日至 4 月 29 日的污染物浓度数据分别输入问题三中训练完成的支持向量机回归模型和 LSTM 神经网络中，预测出 2023 年 4 月 30 日至 5 月 1 日的 AQI 指数。与实际值对比发现，两种预测方法的预测值和实际值误差极小，这不仅说明两种模型的**泛化能力极强、预测精度优秀**，而且证明本文模型**有较强的实际意义**。

关键字：滑动窗口法　Pearson 相关系数　随机森林回归　LSTM 神经网络　支持向量机回归

目录

一、 问题重述

1.1 问题背景

随着各行业工业生产规模不断加大，人类活动对于空气质量的影响日益显著。PM2.5 和 SO_2 等大气污染物会对人类的呼吸系统和心血管系统造成伤害，影响社会生产活动的正常开展。因此，对 PM2.5 浓度以及 AQI 指数的预测以及不同气象数据对空气质量的影响模式的研究逐渐成为人们关注的焦点。

1.2 问题提出

空气污染程度受到大量气象因素的影响。为更好地预测某些污染物的浓度以作出治理与防护，有必要对空气污染的因素进行分析。我们要解决以下问题：

问题一： 处理污染物浓度数据和气象数据，筛选出与 PM2.5 浓度变化相关性强的因素，并定量分析各因素对 PM2.5 浓度影响的程度。

问题二： 问题二是点观测数据的时域分析问题。基于污染物浓度数据和气象数据，构建 PM2.5 浓度多步预测模型，并使用均方根误差对指定步长的预测效果作出评估。将测试集和预测结果进行可视化，并预测指定时间内的 PM2.5 浓度。

问题三： 基于污染物浓度数据和气象数据，构建 AQI 浓度多步预测模型，并使用均方根误差对指定步长的预测效果作出评估。将测试集和预测结果进行可视化，并预测指定时间内的 PM2.5 浓度。

二、 问题分析

2.1 问题一分析

为筛选影响 PM2.5 浓度的因素，本文采用随机森林回归模型和计算 Pearson 相关系数两种方法来量化各个指标对 PM2.5 浓度的影响程度，并确定每个指标对于 PM2.5 浓度的最终影响程度。

2.2 问题二分析

本文使用 SVR 支持向量机回归和基于 LSTM 的深层神经网络在不同步长上预测 PM2.5 的浓度并进行均方根误差 (RMSE) 分析，从而选取预测效果最好的预测方法。

2.3 问题三分析

由于 AQI 指数由污染物浓度本身计算得到，所以考虑在气象因素中筛选出对 AQI 指数印象较大的因素；随后类似于问题二，建立两种模型进行预测和评估，并编程判断

其预警情况。

三、 模型假设

- **不考虑可能出现的极端天气情况例如沙尘暴等。**
  从现有数据可以推断出短时间内该地出现这类极端天气的概率极小。并且将这类极端天气纳入模型会影响模型的鲁棒性。
- **假设未选择的变量的变量不会对模型造成影响**
  未选择的变量经过验证都是影响程度较小的变量，可以忽略这部分变量的影响以增强模型的泛化能力。
-

四、 符号说明

符号	符号含义	符号单位
C_p	空气中 PM2.5 的浓度	$\mu g/m^3$
$P_F(X)$	随机森林中指标 X 的重要程度	/
$P_P(X)$	Pearson 相关系数中指标 X 的重要程度	/
s	重要程度数值的标准差	/
h_i	神经网络的第 i 个隐藏层	/
σ	神经网络中的 $sigmoid$ 激活函数	/

五、 数据预处理

5.1 基于阈值限制法的数据清洗

在进行具体的问题分析之前，应对数据进行预处理。首先我们应将数据限制在符合实际意义的范围内。考虑到数据合理性，对指标作出如下限制：

表 2　各项指标取值范围表

指标	取值范围	单位	指标	取值范围	单位
PM10 浓度	(0, 2080)	$\mu g/m^3$	气温	(-32, 46)	°C
PM2.5 浓度	(0, 993)	$\mu g/m^3$	气压	(800, 1200)	hPa
NO_2 浓度	(0, 350)	$\mu g/m^3$	风速	(0, 50)	m/s
SO_2 浓度	(0, 210)	$\mu g/m^3$	湿度	(0, 100)	%
CO 浓度	(0, 300)	mg/m^3			
O_3 浓度	(0, 450)	$\mu g/m^3$			

此外，注意到数据中存在一些缺失值。考虑到本题存在一个时间序列，所以在处理数据异常值和缺失值的时候不能简单地将这些数据删除。由于本题中存在“空气质量”这一指标，计算出每种空气质量对应指标的平均值，并用平均值来替换数据异常值和数据缺失值。

以气压数据为例，在替换异常值和缺损值前后的数据可视化图如下所示：

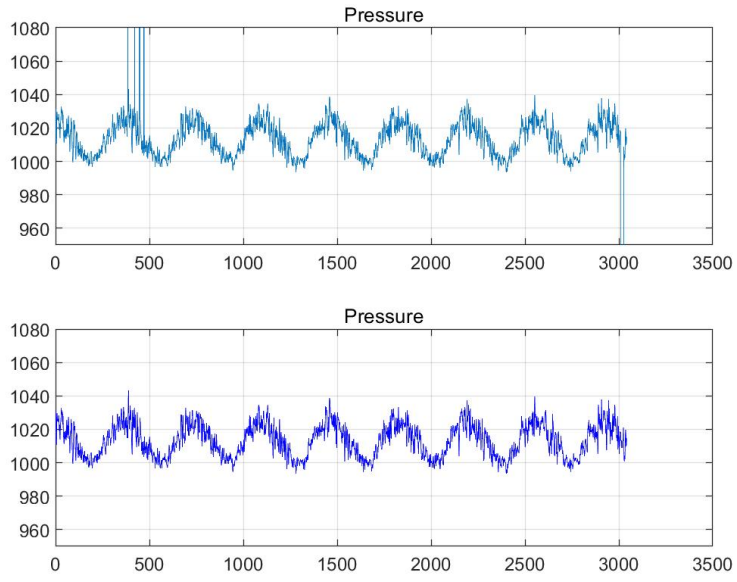


图 1　气压数据阈值限制前后对比图

可以看到在经过阈值限制处理后一些明显的异常点会被替换。

5.2 基于滑动窗口法的数据平滑

在进行了缺失值补充和初步的异常点替换后发现，得到的数据在某些点附近存在着明显的波动。这可能是当天的真实数据也有可能是测量的误差导致的。但是这些点的存在是一种误差，会影响之后训练模型的鲁棒性和泛化能力。因此，也应将这些点用对应的均值替代。

识别误差点时，本文选用了滑动窗口法与 3σ 准则结合的判别方法。选择数据集的第一天作为滑动窗口的起点并向前滑动。考虑到数据集的大小，选择滑动窗口大小为 30，计算出滑动窗口内 PM2.5 浓度的均值记为 μ_p，方差记为 σ_p，这段窗口中若有某一天的 PM2.5 浓度 C_p 符合以下表达式：

$$|C_p - \mu_p| \geq 3\sigma_p \tag{1}$$

则认为这天的 PM2.5 浓度存在异常，应当用均值替换。

下面展示经过数据平滑前后的 PM2.5 浓度如图 2 所示，从图中可以看出一些异常高的 PM2.5 浓度数据被很好地替换了。

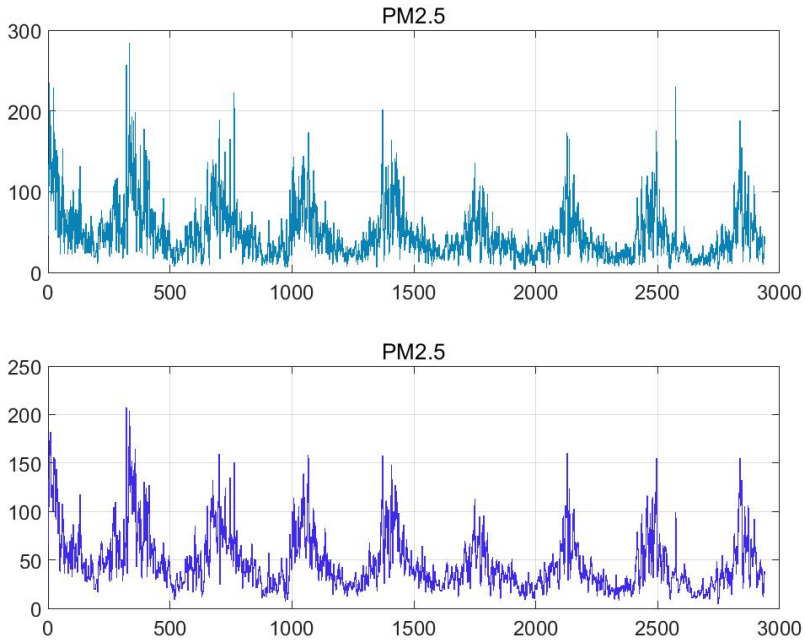


图 2　PM2.5 浓度数据平滑前后对比图

六、 问题一模型的建立及求解

6.1 基于随机森林回归模型的重要程度分析

随机森林算法是一种以决策树为估计器的引导聚集算法，通过决策树输出值的加权平均获得回归值 [1]。值得一提的是随机森林回归模型可以计算影响因素的重要程度。

因此，选择建立随机森林回归模型来筛选何种指标影响 PM2.5 的浓度变化，并量化影响的程度。

6.1.1 随机森林模型的建立

以五个气象指标和五个除 PM2.5 自身外的五个污染物浓度指标共十个指标为自变量，以 PM2.5 浓度为因变量，使用 MATLAB 建立随机森林回归预测模型。首先构建子数据集，在十个指标中，多次随机选取若干个指标作为子数据集。

然后，基于子数据集构建子决策树。在此回归问题中，本文构建了 100 棵决策树，用于进行回归分析。决策树构建算法如下：

首先定义不纯度函数——平均平方误差（MSE）：

$$H(X_m) = \sum_{i \in N_m} (y_i - \bar{y}_m)^2$$

其函数值越小，则单个节点的不纯度越低。计算各个子节点不纯度的加权和作为评价切分效果的标准：

$$G(u, v) = \frac{1}{N_S} \left(\sum_{y_i \in X_{left}} (y_i - \bar{y}_{left})^2 + \sum_{y_i \in X_{right}} (y_i - \bar{y}_{right})^2 \right)$$

其中，u 为切分变量，v 为某切分值，y_m 为当前节点样本目标变量的平均值。X_{left}, X_{right} 分别为左右子节点的训练样本集合。$G(x, v)$ 值越小，切分效果越好。使用穷举法遍历每个点，计算不纯度函数值的加权和，选取加权和最小的点也即效果最好的数值点进行切分，作为该决策树的根节点。切分之后在左右两个范围内再重复上述操作，得到若干非叶子节点，切分的次数即为树的深度，一定程度上决定了决策树的复杂度和预测效果。当切分后的预测值范围达到一定精度，切分停止，决策树构建完成。

最后将子数据集输入相应的子决策树，计算回归值。将回归值取平均数，则得到最终的回归结果。

6.1.2 影响因素重要度分析

本文通过利用平均准确率的减少量对各因素对 PM2.5 浓度的影响程度作出定量分析。步骤如下所示：

- 对于随机森林中的每一棵决策树，使用相应的 OOB(袋外数据) 数据来计算它的袋外数据误差，记为 err_{OOB_1}.
- 随机地对袋外数据 OOB 所有样本的指标 X 加入噪声干扰 (即随机的改变样本在指标 X 处的值)，再次计算其袋外数据误差，记为 err_{OOB_2}.

- 假设随机森林中有 N_{tree} 棵树，用下列公式计算指标 X 的重要度：

$$P_F(X) = \frac{\sum(err_{\mathrm{OOB}_2} - err_{\mathrm{OOB}_1})}{N_{tree}}$$

若给某个指标随机加入噪声之后，袋外的准确率大幅度降低，则说明该指标对于样本的分类结果影响很大，即该指标的重要程度较高。

6.1.3 随机森林回归模型结果展示

利用 Matlab 实现随机森林回归，我们可以得到各个指标影响 PM2.5 浓度的重要程度如下表所示：

表 3　随机森林模型所得影响 PM2.5 浓度重要程度表

指标	重要程度	指标	重要程度	指标	重要程度
PM10	4.11	CO	2.52	气温	2.77
O_3	1.07	降水	0.53	湿度	1.45
SO_2	1.23	气压	1.29		
NO_2	1.22	风速	0.84		

绘制相应的饼状图如下图所示：

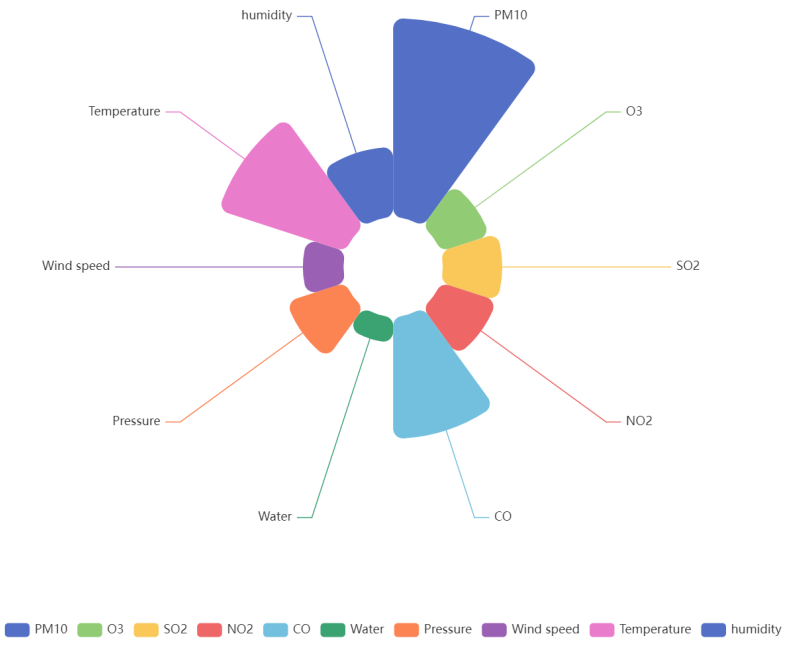


图 3　随机森林所得影响 PM2.5 浓度重要程度饼状图

图表直观表明 PM10，CO，温度，气压对 PM2.5 的浓度变化有着重要的影响。

6.2 基于 Pearson 相关系数的重要程度分析

6.2.1 Pearson 相关系数的计算

Pearson 相关系数刻画两个变量之间的相关程度，Pearson 相关系数的绝对值越大意味着两个变量之间的相关程度越大。因此本文通过计算各个指标与 PM2.5 浓度之间的 Pearson 相关系数来量化该指标影响 PM2.5 浓度变化的重要程度。记两个变量 X 和 PM2.5 浓度 C_p 之间的 Pearson 相关系数为 $corr$，该相关系数的计算公式如下所示：

$$corr(X, C_p) = \frac{Cov(X, C_p)}{\sqrt{Var(X)}\sqrt{Var(C_p)}} \tag{2}$$

6.2.2 Pearson 相关系数的结果展示

根据上述公式计算出 PM2.5 与各个指标的 Pearson 相关系数如下表所示：

表 4　各指标与 PM2.5 浓度的 Pearson 相关系数表

指标	相关系数	指标	相关系数	指标	相关系数
PM10	0.84	CO	0.74	温度	-0.52
O_3	-0.25	降水	-0.18	湿度	0.01
SO_2	0.53	气压	0.48		
NO_2	0.63	风速	-0.22		

相应地，绘制出相关系数的热力图如下图所示：

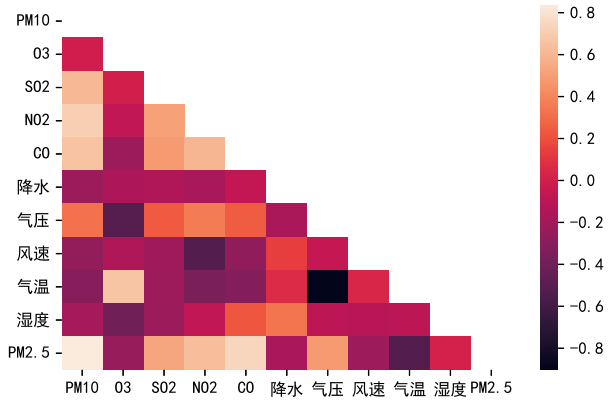


图 4　相关系数热力图

取相关系数的绝对值作为不同因素对 PM2.5 浓度影响的重要程度值，即

$$P_P(X) = |corr(X, C_p)| \tag{3}$$

可以得到影响 PM2.5 浓度变化的因素有 PM10，CO，NO_2，SO_2，气压等。

6.3 基于 Z-score 标准化的重要程度综合分析

经过随机森林回归和 Pearson 相关系数分析，得到了两组有关十个指标的重要程度数据值。利用 Z-score 标准化方法对两组重要性程度数据向量处理后进行加权求和，从而筛选出对 PM2.5 影响较大的因素并定量分析。

6.3.1 Z-score 标准化

对于随机森林计算得出的重要程度数值 $P_F(X)$, 作如下处理：

$$P'_F(X) = \frac{P_F(X) - \bar{P}_F(X)}{\sqrt{Var(P_F(X))}} \tag{4}$$

其中 $\bar{x}_F$ 表示随机森林计算所得重要程度数值的均值，σ_F 表示随机森林计算所得重要程度数值的标准差。

类似地，对 Pearson 相关系数计算所得重要程度数值 $P_P(X)$，作如下处理：

$$P'_P(X) = \frac{P_P(X) - \bar{P}_P(X)}{\sqrt{Var(P_P(X))}} \tag{5}$$

6.3.2 加权求和

由于 Z-score 标准化处理将数据的标准差化为 1，并消除了量纲的影响 [4]，又考虑到应充分体现两种方法的作用，所以将两个经处理后的重要程度向量的权重都赋为 0.5 后求和。

$$P_X = 0.5P'_F(X) + 0.5P'_P(X) \tag{6}$$

得到的结果如下表所示：

表 5　影响 PM2.5 浓度的程度加权求和结果表

指标	影响程度	指标	影响程度	指标	影响程度
PM10	1.85	CO	0.94	温度	0.64
O_3	-0.64	降水	-1.03	湿度	-0.92
SO_2	-0.05	气压	-0.12		
NO_2	0.14	风速	-0.80		

和相关系数不同的是，加权结果代数值越小，代表影响程度越低，从表中我们按 P_X 值从高到低的顺序依次筛选出六个指标：PM10 浓度，CO 浓度，气温，NO_2 浓度，SO_2 浓度和气压作为影响 PM2.5 浓度的重要指标。

七、 问题二模型的建立及求解

考虑到题目中提到的划分训练集和验证集，本文采用机器学习的方法来预测 PM2.5 的浓度，选用支持向量机回归和 LSTM 长短期记忆网络这两种方法。

7.1 基于支持向量机回归模型预测 PM2.5 浓度

7.1.1 支持向量机回归模型的建立

支持向量机回归模型的目标是找到权重 w 与偏置 b 来构建输入 x_i 与输出 y_i 之间的关系：$y_i = wx_i + b$。优化目标为：

$$\min_{w,b} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} l_\epsilon(y_i(wx_i+b)-1) \tag{7}$$

其中 l_ϵ 代表一个损失函数，其计算公式如下所示：

$$l_\epsilon(z) = \begin{cases} 0, & if\ |z| < \epsilon \\ |z| - \epsilon, & otherwise \end{cases} \tag{8}$$

基于此，将 PM10 浓度，CO 浓度，气温，NO_2 浓度，SO_2 浓度和气压作为模型中的 x 而 PM2.5 作为模型中的 y 来建立支持向量机回归模型。

7.1.2 支持向量机回归模型的求解

我们运用 Matlab 中 $fitkernel$ 中的函数训练支持向量机多步预测模型，得到的模型在验证集上的表现效果如下图所示：

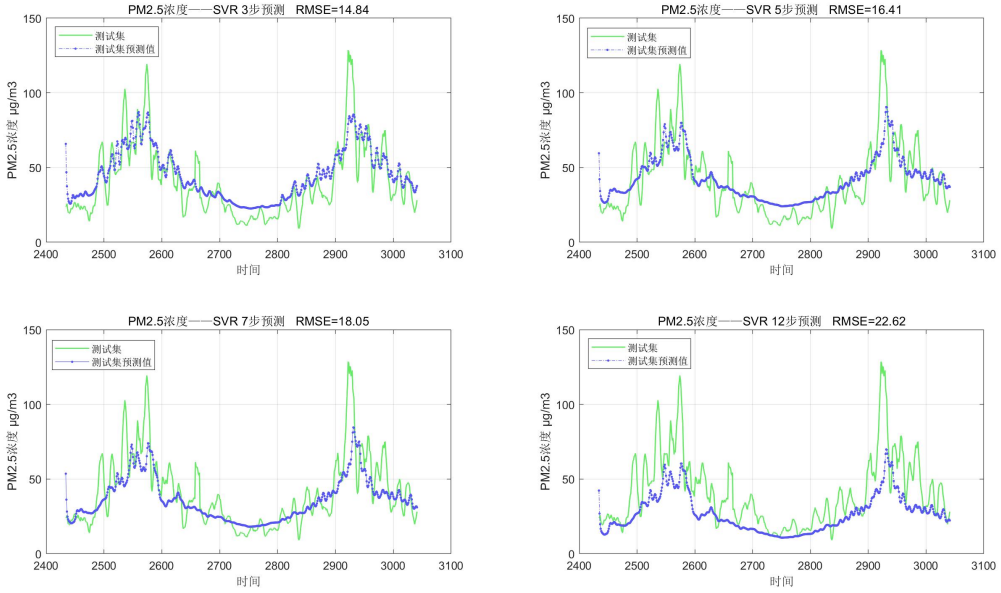


图 5　支持向量机回归多步预测 PM2.5 效果图

得到 3 步，5 步，7 步和 12 步预测模型的 RMSE 如下表所示：

表 7　支持向量机回归多步预测 PM2.5 均方根误差表

预测步长	3 步预测	5 步预测	7 步预测	12 步预测
RMSE	14.84	16.41	18.05	22.62

可以看到随着预测步长的增加，模型的 RMSE 也随之增加。

7.2 基于 LSTM 神经网络预测 PM2.5 浓度

7.2.1 LSTM 神经网络模型的建立

考虑到 PM2.5 浓度的变化涉及一个时间序列，所以我们考虑使用 LSTM 长短期记忆网络，因为长短期记忆网络能够考虑到较长的历史信息 [2]。

LSTM 是一种特殊的循环神经网络，循环神经网络的结构示意图如下所示：

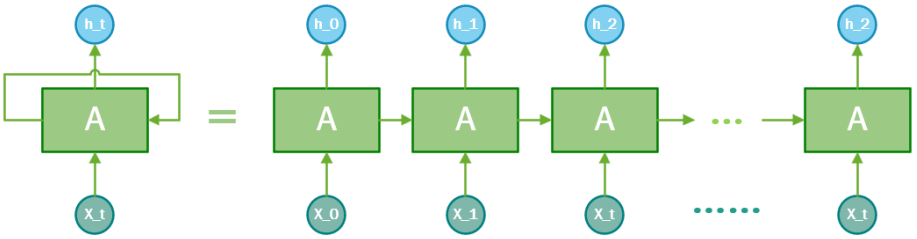


图 6　循环神经网络的结构示意图

结构示意图中的 A 代表的就是一个神经网络，因此循环神经网络本质上是神经网络的一种递归。而 LSTM 长短期记忆网络特殊在它的模块 A 的构造，其模块 A 的构造如下图所示：

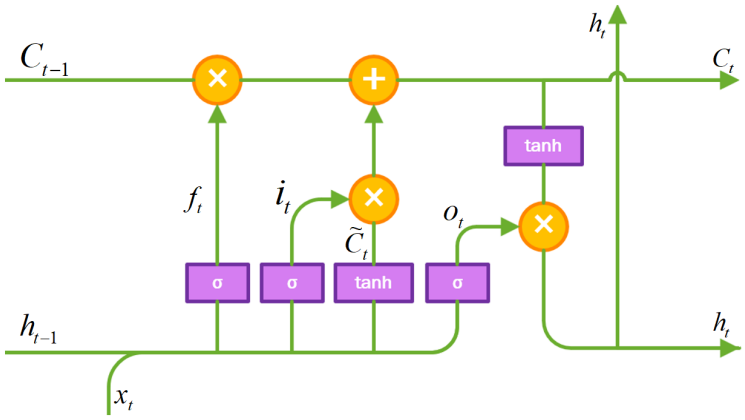


图 7　模块 A 的结构示意图

在模块 A 中的输入为初始数据 x_t 和迭代得到的 C_{t-1} 与 h_{t-1}。我们将 $sigmoid$ 函数记为 $\sigma(x)$，其表达式为：

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{9}$$

此外，$\tanh(x)$ 函数作为激活函数 [3]，该函数的表达式为：

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{10}$$

图中的 f_t 函数的表达式为：

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{11}$$

i_t 函数的表达式为：

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{12}$$

$\tilde{C}_t$ 的表达式为：

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{13}$$

然后我们计算出 C_t，其表达式为：

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{14}$$

接着我们计算 o_t，其表达式为：

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{15}$$

最后计算出 h_t 其表达式为：

$$h_t = o_t * \tanh(C_t) \tag{16}$$

这样一个模块 A 的构建就完成了，对其进行循环我们就能搭建出 LSTM 长短期记忆神经网络。

7.2.2 LSTM 神经网络模型的求解

本文中选择了数据的前 70% 作为训练集，后 30% 作为验证集，迭代了 1000 次后的 3 步预测模型在训练集和验证集上的效果如下图所示：

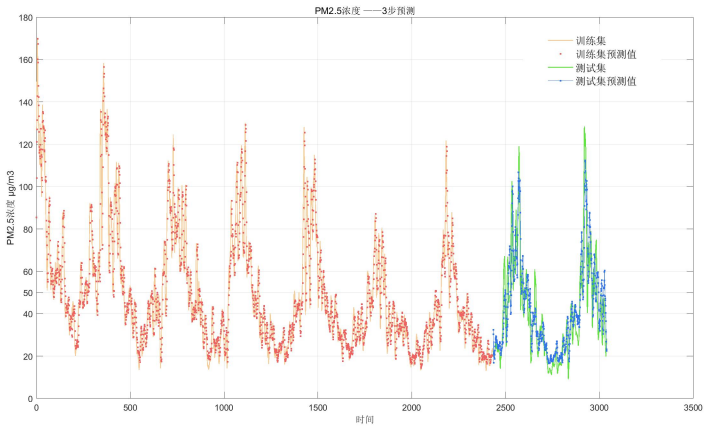


图 8　LSTM 神经网络 3 步预测模型效果图

从图中我们可以看出基于 LSTM 神经网络的三步预测模型在训练集和验证集上的效果都很好。此外，为了更好的比较步长对于分步预测模型的 RMSE 的影响，绘制出 3 步，5 步，7 步和 12 步预测模型在验证集上的效果图如下所示：

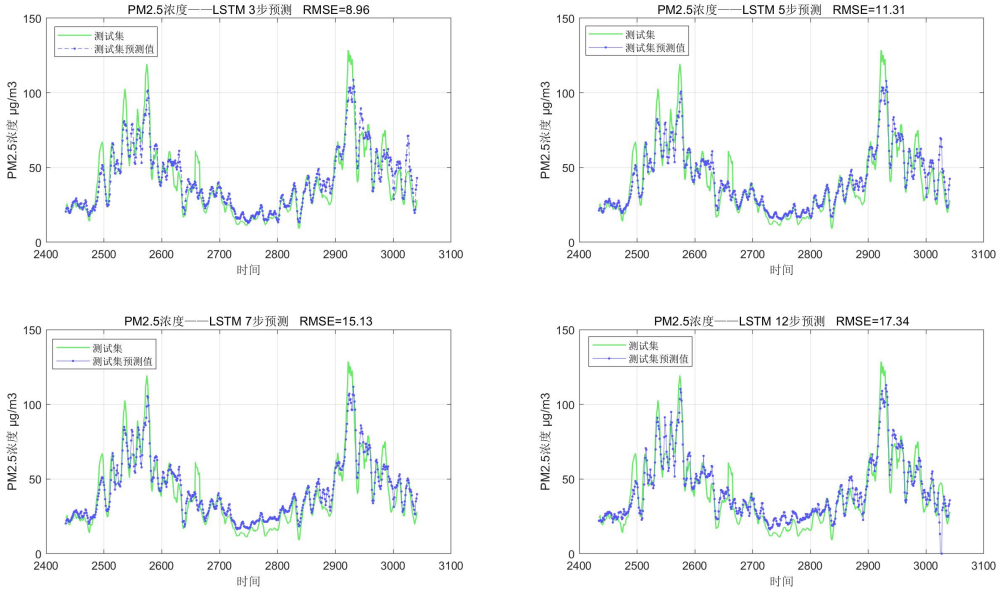


图 9　LSTM 神经网络多步预测 PM2.5 效果图

此外，3 步，5 步，7 步和 12 步预测模型的 RMSE 如下表所示：

表 8　LSTM 神经网络多步预测 PM2.5 均方根误差表

预测步长	3 步预测	5 步预测	7 步预测	12 步预测
RMSE	8.96	11.31	15.13	17.34

同样地，预测步长增加，RMSE 会相应增加。

7.3 两种方法预测给定时间 PM2.5 浓度

利用搭建好的网络，考虑到步长越长，模型的 RMSE 越大，我们用 3 步预测模型预测 4 月 30 日至 5 月 1 日的 PM2.5 浓度，5 月 3 日至 5 月 4 日的用 5 步预测模型的预测结果补全，依次类推，计算得到两种方法预测给定时间的 PM2.5 浓度如下表所示：

表 9　给定时间两种方法 PM2.5 浓度预测表

时间	PM2.5(SVR)	**PM2.5(LSTM)**	时间	PM2.5(SVR)	**PM2.5(LSTM)**
2023/4/30	49.53	**55.46**	2023/5/6	18.37	**23.53**
2023/5/1	35.98	**43.33**	2023/5/7	19.23	**21.97**
2023/5/2	30.71	**37.85**	2023/5/8	20.76	**22.85**
2023/5/3	27.19	**33.92**	2023/5/9	25.21	**29.98**
2023/5/4	25.33	**31.02**	2023/5/10	29.47	**34.47**
2023/5/5	21.40	**27.05**	2023/5/11	35.53	**38.71**

绘制出两种方法预测的给定时间 PM2.5 浓度图如下所示：

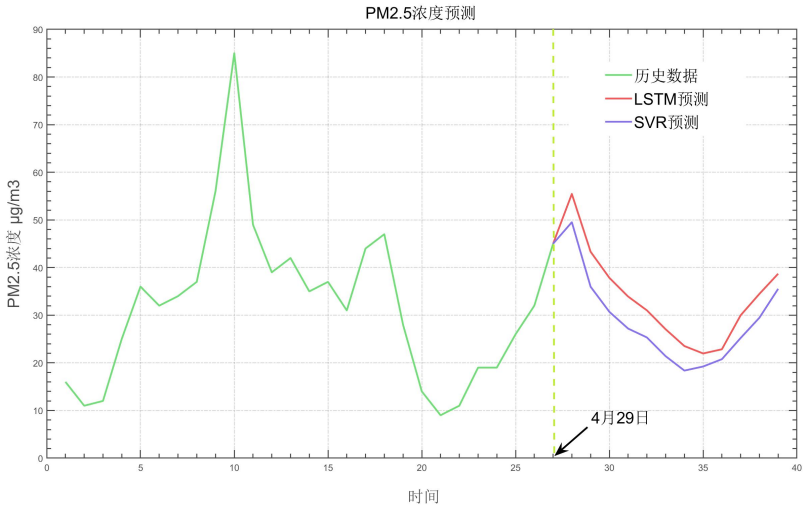


图 10　PM2.5 浓度预测图

从图中可以看到预测值基本符合历史变化趋势。考虑到两种模型的均方根误差，我们认为 LSTM 神经网络模型预测出的结果更为可靠，并将 LSTM 模型的预测值作为预测结果。

八、 问题三模型的建立及求解

8.1 基于随机森林回归的 AQI 影响因素抉择

我们对 AQI 历史数据插值来完成缺损值填补。考虑到 AQI 本身是由污染物浓度计算的，所以我们这里只考虑气象因素对 AQI 的影响程度，和问题一中相同，本文使用

五个气象因素进行随机森林回归，得到的影响因素重要程度如下表所示：

表 10　随机森林回归所得影响 AQI 重要程度表

风速	降水	气温	气压	湿度
3.19	1.79	1.40	0.64	2.82

可以看到在气象因素中，风速、降水和气温对 AQI 的影响程度较大，因此选用这三个指标与六个污染物指标和历史的 AQI 指标作为网络的输入来预测 AQI 的值。

8.2 基于支持向量机回归模型的 AQI 预测

和第二问类似，仍然将前 70% 的数据划分为训练集，后 30% 的数据划分为验证集。用上述的九个指标与 AQI 的历史数据训练支持向量机回归模型，得到的 3 步，5 步，7 步和 12 步预测效果如下图所示，下图的数据分别为验证集上的真实数据和预测数据：

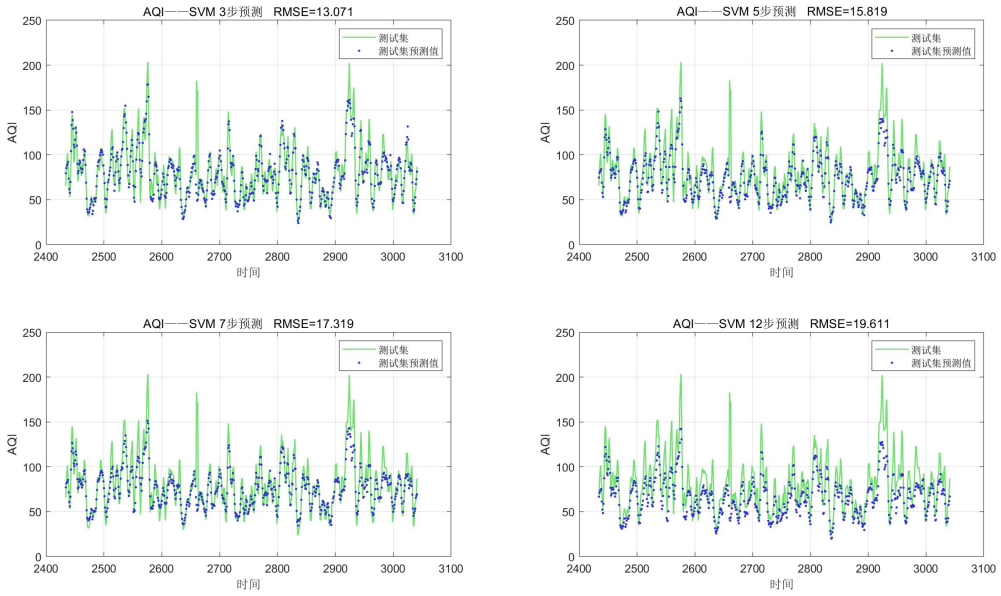


图 11　支持向量机回归多步预测 AQI 效果图

支持向量机回归多步预测 AQI 的均方根误差如下表所示：

表 11　支持向量机回归多步预测 AQI 均方根误差表

预测步长	3 步预测	5 步预测	7 步预测	12 步预测
RMSE	13.07	15.82	17.32	19.61

8.3 基于 LSTM 长短期神经网络的 AQI 预测

和第二问类似，仍然将前 70% 的数据划分为训练集，后 30% 的数据划分为验证集。用上述的九个指标与 AQI 的历史数据训练 LSTM 长短期神经网络，得到的 3 步，5 步，7 步和 12 步预测效果如下图所示，下图的数据分别为验证集上的真实数据和预测数据：

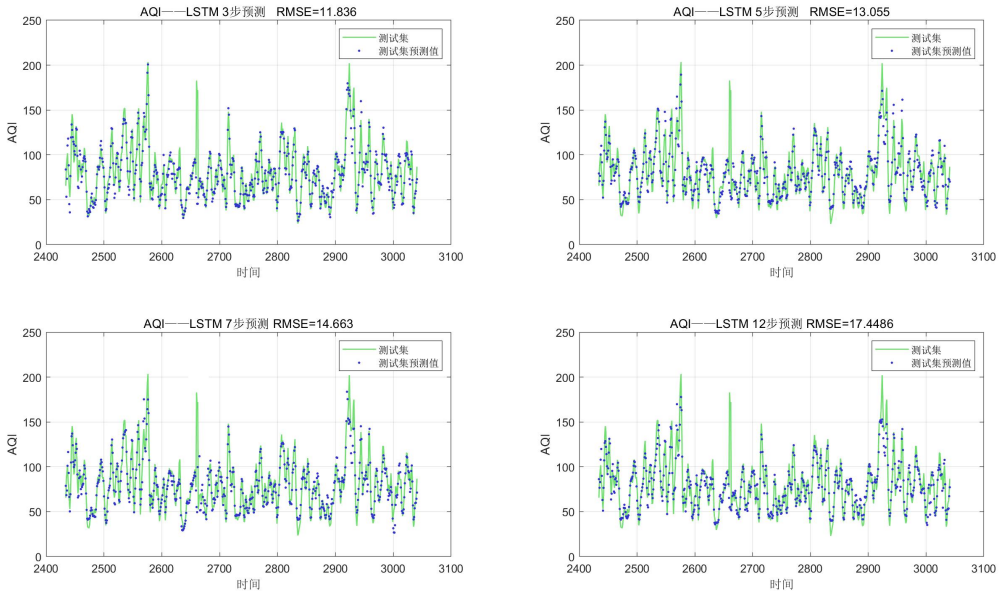


图 12　LSTM 神经网络多步预测 AQI 效果图

LSTM 神经网络多步预测 AQI 的均方根误差如下表所示：

表 12　LSTM 神经网络多步预测 AQI 均方根误差表

预测步长	3 步预测	5 步预测	7 步预测	12 步预测
RMSE	11.84	13.06	14.66	17.45

对比两种模型的均方根误差值发现 LSTM 模型的预测效果更为精确，所以选取 LSTM 的预测值作为预测结果。

8.4 两种方法预测结果展示与比较

首先训练支持向量机回归模型预测 AQI，并根据 AQI 得到预警程度结果如下表所示：

表 13　支持向量机回归预测 AQI 及预警等级颜色结果表

日期	AQI(SVR)	预警等级颜色	日期	AQI(SVR)	预警等级颜色
2023/4/30	61.40	无	2023/5/6	32.20	无
2023/5/1	86.74	无	2023/5/7	45.64	无
2023/5/2	127.61	蓝色预警	2023/5/8	68.71	无
2023/5/3	103.92	无	2023/5/9	54.08	无
2023/5/4	54.23	无	2023/5/10	60.25	无
2023/5/5	31.38	无	2023/5/11	83.81	无

接着训练 LSTM 长短期记忆神经网络预测 AQI，并根据 AQI 得到预警程度结果如下表所示：

表 14　LSTM 神经网络预测 AQI 及预警等级颜色结果表

日期	**AQI(LSTM)**	预警等级颜色	日期	**AQI(LSTM)**	预警等级颜色
2023/4/30	**44.43**	无	2023/5/6	**33.71**	无
2023/5/1	**73.99**	无	2023/5/7	**44.39**	无
2023/5/2	**128.75**	蓝色预警	2023/5/8	**65.97**	无
2023/5/3	**110.30**	无	2023/5/9	**67.25**	无
2023/5/4	**58.98**	无	2023/5/10	**55.24**	无
2023/5/5	**38.98**	无	2023/5/11	**78.12**	无

可以看到在预警程度上，两种方法得到的结果是一致的，只有 2023 年 5 月 2 日是蓝色预警，其余时间都没有预警。预警天气的统计表如下所示：

预警等级	蓝色	黄色	橙色	红色	无	合计
天数（天）	1	0	0	0	11	12

表 15　给定时间内预警等级统计

此外，绘制出两种方法预测的给定时间 AQI 指数图如下所示

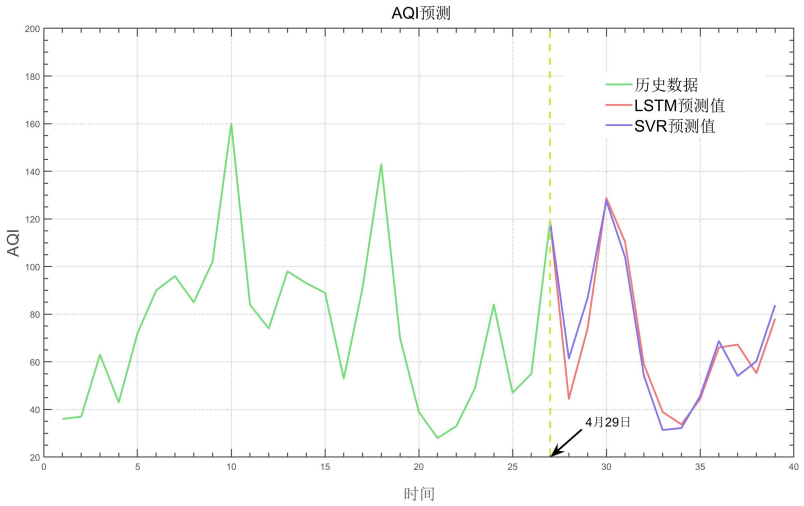
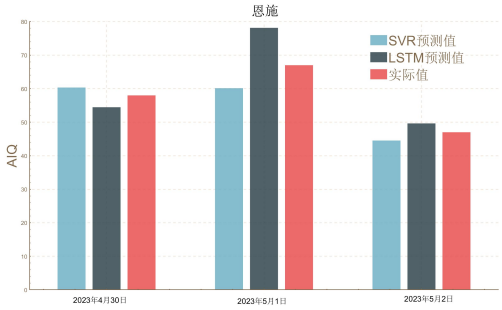


图 13　AQI 指数预测图

九、 预测模型在实际情况中的检验

考虑到我们模型训练的样本量足够充分，本文尝试将模型应用于实际情况以检验其效果。我们选取了恩施、武汉两地从 2023 年 4 月 27 日到 4 月 29 日的污染物与天气数据来预测该地在 2023 年 4 月 30 日到 5 月 2 日的 AQI，并与真实值比较。其中气压，降水与湿度数据难以获取，用历史平均值代替。绘制出的条形图如下所示：



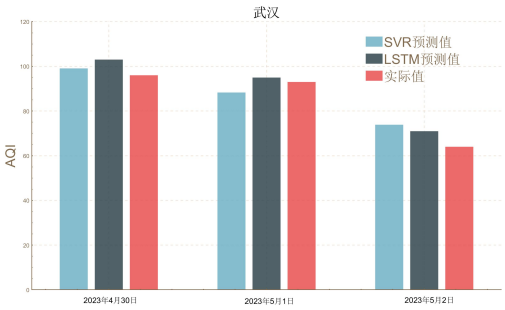


图 14　恩施 AQI 预测值与真实值对比图　**图 15　武汉 AQI 预测值与真实值对比图**

从条形图中可以看出预测值与真实值相差不大，为了更精确地比较预测效果，我们列出恩施与武汉两地在 2023 年 4 月 30 日到 5 月 2 日 AQI 指标的两种模型预测值与真实值如下表所示：

表 16　恩施与武汉两地预测值与真实值对比表

恩施				武汉			
日期	LSTM	SVR	实际值	日期	LSTM	SVR	实际值
2023/4/30	60.35	54.47	58.00	2023/4/30	99.06	103.01	96.00
2023/5/1	60.15	78.15	67.00	2023/5/1	88.28	94.97	93.00
2023/5/2	44.53	49.64	47.00	2023/5/2	73.86	70.97	64.00

条形图和对比表直观反映了预测的效果。可以发现两种模型的预测效果都比较优秀，尤其是 LSTM 神经网络的预测值与两城市当地的实际 AQI 指数之间的误差几乎达到了 10% 以内，证明模型可以较好地在实际的空气质量预测工作中应用

十、 模型的评价

10.1 模型的优点

- 使用滑动窗口方法对数据进行了平滑处理，从而对部分离群值作出数据替换，减小偶然误差对模型训练的影响，增强了模型的泛化能力。
- 针对筛选影响因素的问题，我们通过两种方法做出评价并标准化后加权求和，充分利用两种方法求得的数据，使筛选决策更科学。
- 选取了 LSTM 长短期记忆神经网络，能够将较长时间内的历史信息纳入考虑，增强了模型的预测效果。

10.2 模型的缺点

- 模型未考虑节假日因素对空气污染情况的影响，导致预测结果可能与实际数据有误差。
- 对于极端天气，模型并不具备很好的预测能力。
- 模型对空气质量较差的地区适用能力有所欠缺，对这部分地区可能需要对预测模型进行重新训练。

参考文献

[1] 李欣海. 随机森林模型在分类与回归分析中的应用 [J]. 应用昆虫学报, 2013, 50(04):1190-119

[2] 杨丽, 吴雨茜, 王俊丽, 刘义理. 循环神经网络研究综述 [J]. 计算机应用, 2018, 38(S2):1-6+26.

[3] 蒋昂波, 王维维.ReLU 激活函数优化研究 [J]. 传感器与微系统, 2018, 37(02):50-52.DOI:10.13873/J.1000-9787(2018)02-0050-03.

[4] Ghasemi A, Zahediasl S. Normality tests for statistical analysis: a guide for non-statisticians[J]. International journal of endocrinology and metabolism, 2012, 10(2): 486.

[5] 丁世飞, 齐丙娟, 谭红艳. 支持向量机理论与算法研究综述 [J]. 电子科技大学学报, 2011, 40(01):2-10.

A 附录一：支撑材料列表

1. 论文源文件
2. 代码
    - 问题一
    - 问题二
    - 问题三
3. 数据
    - 问题一
    - 问题二
    - 问题三

B 附录二：可视化补充

2.1 LSTM 神经网络可视化补充

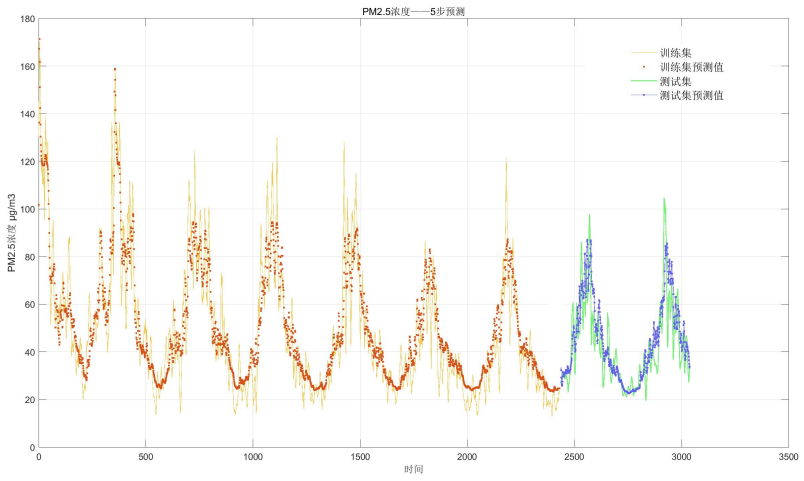


图 16　LSTM 神经网络 5 步预测模型效果图

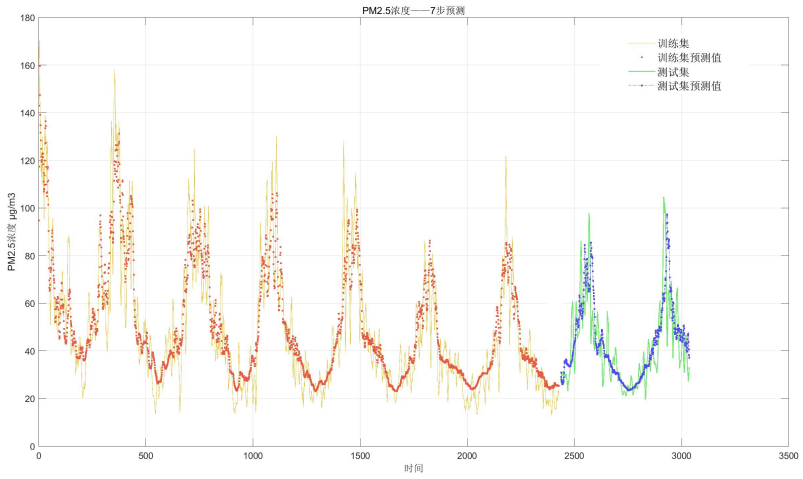


图 17　LSTM 神经网络 7 步预测模型效果图

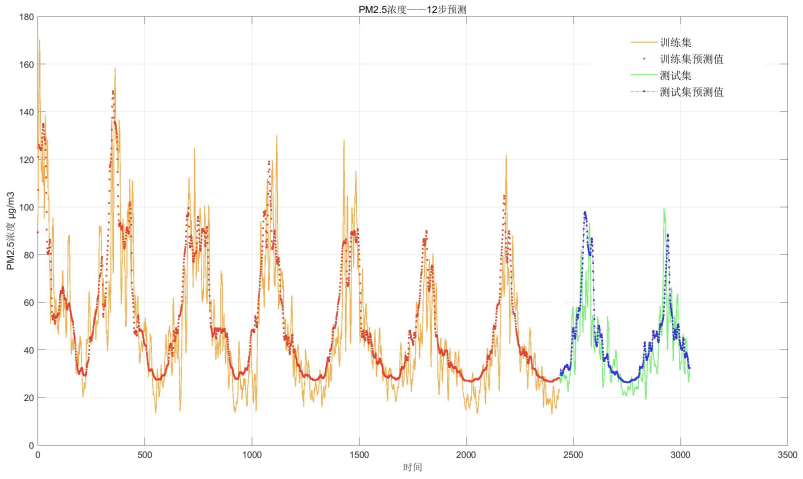


图 18　LSTM 神经网络 12 步预测模型效果图

C 附录二：可运行源代码

3.1 数据预处理代码

3.1.1 阈值限制数据处理代码

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

# 读取数据
df = pd.read_csv('data.csv')
```

```
# 计算各指标的均值
means1 = df.groupby('质量等级')['PM10'].median()
means2 = df.groupby('质量等级')['O3'].median()
means3 = df.groupby('质量等级')['SO2'].median()
means4 = df.groupby('质量等级')['PM2.5'].median()
means5 = df.groupby('质量等级')['CO'].median()
means6 = df.groupby('质量等级')['Water'].median()
means7 = df.groupby('质量等级')['Pressure'].median()
means8 = df.groupby('质量等级')['Wind speed'].median()
means9 = df.groupby('质量等级')['Temperature'].median()
means10 = df.groupby('质量等级')['humidity'].median()
means11 = df.groupby('质量等级')['NO2'].median()

df['PM10'].fillna(-50, inplace=True)
df['O3'].fillna(-50, inplace=True)
df['SO2'].fillna(-50, inplace=True)
df['PM2.5'].fillna(-50, inplace=True)
df['CO'].fillna(-50, inplace=True)
df['Water'].fillna(-50, inplace=True)
df['Pressure'].fillna(-50, inplace=True)
df['Wind speed'].fillna(-50, inplace=True)
df['Temperature'].fillna(-50, inplace=True)
df['humidity'].fillna(-50, inplace=True)
df['NO2'].fillna(-50, inplace=True)

## 替代缺失值和不合法值
for i, row in df.iterrows():
    if row['PM10'] <= 0 or row['PM10'] > 1000 or pd.isna(row['PM10']):
        df.at[i, 'PM10'] = means1[row['质量等级']]

for i, row in df.iterrows():
    if row['O3'] <= 0 or row['O3'] > 1000 or pd.isna(row['O3']):
        df.at[i, 'O3'] = means2[row['质量等级']]

for i, row in df.iterrows():
    if row['SO2'] <= 0 or row['SO2'] > 1000 or pd.isna(row['SO2']):
        df.at[i, 'SO2'] = means3[row['质量等级']]

for i, row in df.iterrows():
    if row['PM2.5'] <= 0 or row['PM2.5'] > 1000 or pd.isna(row['PM2.5']):
        df.at[i, 'PM2.5'] = means4[row['质量等级']]

for i, row in df.iterrows():
    if row['CO'] <= 0 or row['CO'] > 1000 or pd.isna(row['CO']):
        df.at[i, 'CO'] = means5[row['质量等级']]

for i, row in df.iterrows():
```

```
    if row['Water'] < 0 or row['Water'] > 1870 or pd.isna(row['Water']):
        df.at[i, 'Water'] = means6[row['质量等级']]


for i, row in df.iterrows():
    if row['Pressure'] < 850 or row['Pressure'] >=1200 or pd.isna(row['Pressure']):
        df.at[i, 'Pressure'] = means7[row['质量等级']]

for i, row in df.iterrows():
    if row['Wind speed'] < 0 or row['Wind speed'] >100 or pd.isna(row['Wind speed']):
        df.at[i, 'Wind speed'] = means8[row['质量等级']]

for i, row in df.iterrows():
    if row['Temperature'] < -20 or row['Temperature'] >45 or pd.isna(row['Temperature']):
        df.at[i, 'Temperature'] = means9[row['质量等级']]


for i, row in df.iterrows():
    if row['humidity'] < 0 or row['humidity'] >100 or pd.isna(row['humidity']):
        df.at[i, 'humidity'] = means10[row['质量等级']]

for i, row in df.iterrows():
    if row['NO2'] < 0 or row['NO2'] >100 or pd.isna(row['NO2']):
        df.at[i, 'NO2'] = means11[row['质量等级']]

df.to_csv('data_clc_1.csv', encoding='gbk', index=False)
```

3.1.2 滑动窗口法数据降噪代码

```
smoothedData = SmoothData(data, 30);

function [y, winsz] = SmoothData(A, varargin)
narginchk(1, inf);

    if ~isnumeric(A) && ~islogical(A) && ~isa(A, 'tabular')
        error(message("badArray"));
    end
    if isinteger(A) && ~isreal(A)
        error(message("complexIntegers"));
    end

    sparseInput = issparse(A);
    if sparseInput
        A = full(A);
    end
```

```
[dim, method, winsz, nanflag, t, sgdeg, dvars] = parseInputs(A, varargin{:});

if isempty(A) || (dim > ndims(A)) || (size(A, dim) < 2)
    if isa(A, 'tabular')
        y = A;
        for j = dvars
            y.(j) = convertToFloat(y.(j));
        end
    else
        y = convertToFloat(A);
        if sparseInput
            y = sparse(y);
        end
    end
    return;
end

if isa(A, 'tabular')
    y = A;
    if isempty(dvars)
        return;
    end
    singleCheck = varfun(@(x) isa(x, 'single'), A, ...
        'InputVariables', dvars, 'OutputFormat', 'uniform');
    sparseCheck = varfun(@issparse, A, 'InputVariables', dvars, ...
        'OutputFormat', 'uniform');
    if (all(singleCheck) || ~any(singleCheck)) && ~any(sparseCheck)
        if ~any(singleCheck)
            for j = dvars
                y.(j) = double(y.(j));
            end
        end
        y{:, dvars} = smoothNumericArray(y{:, dvars}, method, dim, ...
            winsz, nanflag, t, sgdeg);
    else
        for j = dvars
            if issparse(y.(j))
                y.(j) = sparse(smoothNumericArray(full(y.(j)), method, ...
                    dim, winsz, nanflag, t, sgdeg));
            else
                y.(j) = smoothNumericArray(y.(j), method, dim, winsz, ...
                    nanflag, t, sgdeg);
            end
        end
    end
else
```

```
        y = smoothNumericArray(A, method, dim, winsz, nanflag, t,  sgdeg);
    end

    if sparseInput
        y = sparse(y);
    end
end
```

3.2 随机森林回归代码

```
num_train=ceil(size(data, 1));
choose=randperm(length(data));
train_data=data(choose(1:num_train), 1:end);label_train=train_data(:, end);
test_data=data(choose(num_train+1:end), 1:end);
n1=size(data, 2);
y_train_data=train_data(:, n1);
x_train_data=train_data(:, 1:n1-1);
[x_train_regular, x_train_maxmin]=mapminmax(x_train_data');
[y_train_regular, y_train_maxmin]=mapminmax(y_train_data');
x_train_regular=x_train_regular';
y_train_regular=y_train_regular';
ntree=100;
RF_Model=TreeBagger(ntree, x_train_regular, y_train_regular,
'Method', "regression", 'OOBPredictorImportance', "on");
imp=RF_Model.OOBPermutedPredictorDeltaError;
figure;
bar(imp);
title('Curvature Test');
ylabel('Predictor importance estimates');
xlabel('Predictors');
```

3.3 Pearson 相关系数计算和可视化代码

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

# read dataset
df = pd.read_csv('C:/Users/tsuki/Desktop/C/result.csv', encoding='utf-8')
# get correlations
df_corr = df.corr() # 13X13
print(df_corr)
```

```
np.ones_like(df_corr, dtype=np.bool)

mask = np.triu(np.ones_like(df_corr, dtype=np.bool))

sb.heatmap(df_corr, mask=mask)
plt.savefig('corr.eps')
plt.show()
```

3.4 浅层神经网络代码

```
num_train=ceil(0.8*size(data, 1));
num_test=size(data, 1)-num_train;

train_data=data(1:num_train, 1:end);
% label_train=dataTrain(:, end);%训练标签
test_data=data(num_train+1:end, 1:end);%测试集

label_train=train_data(:, end);%训练标签
n1=size(data, 2); %指标数量
y_train_data=train_data(4:end, n1);
y_train_data=y_train_data';
x_train_data=train_data(1:end-3, 1:n1-1);
x_train_data=x_train_data';
y_test_data=test_data(4:end, n1);
y_test_data=y_test_data';
x_test_data=test_data(1:end-3, 1:n1-1);
x_test_data=x_test_data';
label_train=train_data(:, end);%训练标签
n1=size(data, 2); %指标数量
y_train_data=train_data(:, n1);
y_train_data=y_train_data';
x_train_data=train_data(:, 1:n1-1);
x_train_data=x_train_data';
y_test_data=test_data(:, n1);
y_test_data=y_test_data';
x_test_data=test_data(:, 1:n1-1);
x_test_data=x_test_data';

[x_train_regular, x_train_maxmin]=mapminmax(x_train_data);
x_test_regular= mapminmax('apply', x_test_data, x_train_maxmin);

[y_train_regular, y_train_maxmin]=mapminmax(y_train_data);
y_test_regular= mapminmax('apply', y_test_data, y_train_maxmin);
```

```
%创建SVM回归模型(自动寻优) 可以查看matlab中的fitrkernel
[SVMModel, FitInfo, HyperparameterOptimizationResults]
=fitrkernel(x_train_regular', y_train_regular',
'OptimizeHyperparameters', "auto",
'HyperparameterOptimizationOptions',
struct('AcquisitionFunctionName', 'expected-improvement-plus'));

%
%进行预测
x_test_regular=mapminmax('apply', x_test_data, x_train_maxmin);
predictSVMre=predict(SVMModel, x_test_regular');
predictSVM=mapminmax('reverse', predictSVMre, y_train_maxmin);
%计算平均绝对值误差
errors_nn=sum(abs(predictSVM-y_test_data)./(y_test_data))/length(y_test_data);
%绘图

color=[111, 168, 86;128, 199, 252;112, 138, 248;184, 84, 246]/255;
plot(y_test_data, "LineWidth", 1)
hold on
plot(predictSVM', '.')
hold on
legend('True', 'predict');
```

3.5 LSTM 长短期记忆神经网络与可视化代码

```
%data为一个列向量
%% 取data的前70%为训练集，后30%为测试集
num_train=ceil(0.8*size(data, 1));
num_test=size(data, 1)-num_train;

train_data=data(1:num_train, 1:end);
% label_train=dataTrain(:, end);%训练标签
test_data=data(num_train+1:end, 1:end);%测试集

label_train=train_data(:, end);%训练标签
n1=size(data, 2); %指标数量
y_train_data=train_data(:, n1);
y_train_data=y_train_data';
x_train_data=train_data(:, 1:n1-1);
x_train_data=x_train_data';
y_test_data=test_data(12:end, n1);
y_test_data=y_test_data';
x_test_data=test_data(1:end, 1:n1-1);
x_test_data=x_test_data';
label_train=train_data(:, end);%训练标签
```

```
n1=size(data, 2); %指标数量
y_train_data=train_data(:, n1);
y_train_data=y_train_data';
x_train_data=train_data(:, 1:n1-1);
x_train_data=x_train_data';
y_test_data=test_data(:, n1);
y_test_data=y_test_data';
x_test_data=test_data(:, 1:n1-1);
x_test_data=x_test_data';

%数据预处理
[x_train_regular, x_train_maxmin]=mapminmax(x_train_data);
x_test_regular= mapminmax('apply', x_test_data, x_train_maxmin);

[y_train_regular, y_train_maxmin]=mapminmax(y_train_data);
y_test_regular= mapminmax('apply', y_test_data, y_train_maxmin);

%layer = fullyConnectedLayer(numHiddenUnits, 'Activation', 'sigmoid');



numFeatures=size(x_train_regular, 1);%输入层的指标维度
numResponses=size(y_train_regular, 1);%输出层的指标维度
numHiddenUnits = 100; %创建LSTM回归网络，指定LSTM层的隐含单元个数200。可调

layers = [ ...
    sequenceInputLayer(numFeatures) %输入层，这里为n维
    lstmLayer(numHiddenUnits, 'StateActivationFunction', "tanh") %学习层设置(cell层)
    dropoutLayer(0.4)
%     lstmLayer(numHiddenUnits-30, 'GateActivationFunction', 'sigmoid') % lstm层，
     如果是构建多层的LSTM模型，可以修改
%     dropoutLayer(0.4)
%     lstmLayer(numHiddenUnits-60, 'StateActivationFunction', 'tanh') % lstm层，
     如果是构建多层的LSTM模型，可以修改
%     dropoutLayer(0.6)
    fullyConnectedLayer(numResponses) %为全连接层，是输出层，这里维数为一维。
    regressionLayer]; %其计算回归问题的半均方误差模块 。即说明这不是在进行分类问题


%指定训练选项，求解器设置为rmsprop，500轮训练
%梯度阈值设置为 1。指定初始学习率 0.01，每 50 轮训练后通过乘以因子 0.2 来降低学习率。
options = trainingOptions('rmsprop', ...
    'MaxEpochs', 100, ...
    'GradientThreshold', 1, ...%梯度下降阈值
    'InitialLearnRate', 0.01, ...
    %全局学习率。默认率是0.01
    'LearnRateSchedule', 'piecewise', ...
```

```
    %在训练期间降低学习率的选项，默认学习率不变。'piecewise' 表示在学习过程中降低学习率
    'LearnRateDropPeriod', 40, ...
    % 降低学习率的轮次数量，每次通过指定数量的epoch时，将全局学习率与学习率降低因子相乘
    'LearnRateDropFactor', 0.15, ...    %学习下降因子
    'Verbose', 0,  ... %如果将其设置为true，则有关训练进度的信息将被打印到命令窗口中。默认值为true
    'Plots', 'training-progress');
    %构建曲线图 将'training-progress'替换为none

%% LSTM网络训练
net = trainNetwork(x_train_regular, y_train_regular, layers, options);

numTimeStepsTrain = numel(x_train_regular(1, :));
% for i = 1:numTimeStepsTrain
%     [net, predictLSTMre_train(i)] = predictAndUpdateState(net, x_train_regular(:, i),
    'ExecutionEnvironment', 'cpu');
%     %[net, predictLSTMre_train(i)] = predictAndUpdateState(net, x_train_regular(:, i),
    'ExecutionEnvironment', 'cpu');
% end
predictLSTMre_train=predict(net, x_train_regular);
predictLSTMre_train=double(predictLSTMre_train');
predictLSTM_train=mapminmax('reverse', predictLSTMre_train, y_train_maxmin);

%%BiLSTM测试数据
numTimeStepsTest = numel(x_test_regular(1, :));
% for i = 1:numTimeStepsTest
%     [net, predictLSTMre_test(i)] = predictAndUpdateState(net, x_test_regular(:, i),
    'ExecutionEnvironment', 'cpu');
% end
predictLSTMre_test=predict(net, x_test_regular);
predictLSTMre_test=double(predictLSTMre_test');
predictLSTM_test = mapminmax('reverse', predictLSTMre_test, y_train_maxmin);
    %用训练的最后一步来进行预测第一个预测值，给定一个初始值。这是用预测值更新网络状态特有的


%预测的测试数据评价
y=y_test_data*0.673+13;
ae=abs(predictLSTM_test - y');
RMSE=mean(ae.^2).^(0.5);
MSE = mean(ae.^2);
MAE = mean(ae);
disp('预测结果评价指标: ')
disp(['RMSE = ', num2str(RMSE)])
disp(['MSE = ', num2str(MSE)])
disp(['MAE = ', num2str(MAE)])

%画出训练集、测试集的情况
plot(y_train_data(1:end))
```

```
hold on
idx=1:num_train;
plot(idx, predictLSTM_train, '.-')
hold on

idx=num_train+1:num_train+num_test;
plot(idx, y', 'g.-')
hold on
idx=num_train+1:num_train+num_test;
plot(idx, predictLSTM_test, 'b')
```

3.6 判断预警程度代码

```
   % 输入 AQI 预测结果，以及每天的小时数
aqi_predictions = input('按顺序输入每天的 AQI 预测结果（用空格或逗号分隔）：', 's');
hours_per_day = input('每天的小时数：');

% 将输入字符串转换为数字数组
aqi_predictions = str2num(strrep(aqi_predictions, ', ', ' '));

% 计算 AQI 预测结果持续时间（天数）
duration_days = ceil(length(aqi_predictions) / hours_per_day);

% 根据 AQI 预测结果和持续时间计算预警级别
aqi_max = max(reshape(aqi_predictions, hours_per_day, duration_days));
if any(aqi_max > 150) && duration_days >= 2
    warning_level = '蓝色预警';
elseif any(aqi_max > 200) && duration_days >= 2
    warning_level = '黄色预警';
elseif any(aqi_max > 200) && duration_days >= 3 || ...
       any(aqi_max > 150) && duration_days >= 3
    warning_level = '橙色预警';
elseif all(aqi_max > 200) && duration_days >= 3 && ...
       all(aqi_max > 300) && duration_days >= 1
    warning_level = '红色预警';
else
    warning_level = '无预警';
end

% 输出预警级别
fprintf('预警级别：%s\n', warning_level);
```