

**融合回归模型在客流量预测中的应用**

2024.5

目录

[一. 引言 2](#_Toc8429)

[1.机器学习和深度学习简介 2](#_Toc14364)

[2.回归问题简介 2](#_Toc25781)

[二. 数据处理和预处理 3](#_Toc14315)

[1. 数据读取 3](#_Toc25848)

[2.数据编码 4](#_Toc5302)

[3.数据标准化 7](#_Toc8187)

[三. 模型训练和评估 8](#_Toc12577)

[1.随机森林回归模型 8](#_Toc7549)

[2.LSTM神经网络 10](#_Toc3827)

[四．模型融合 14](#_Toc5409)

[1.加权平均融合 14](#_Toc14416)

[五． 结果可视化 17](#_Toc7500)

[1.汇总结果 17](#_Toc29618)

[2. 最终可视化 20](#_Toc19873)

# 一. 引言

## 1.机器学习和深度学习简介

机器学习是一种通过数据训练模型以进行预测或分类的技术。常见的机器学习模型包括回归模型、决策树、支持向量机等。深度学习是机器学习的一个分支，通过神经网络模拟人脑的工作方式来处理数据，特别适用于处理复杂的模式识别问题。

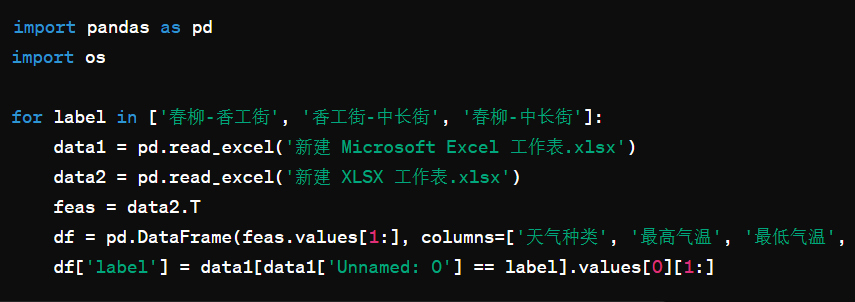
## 2.回归问题简介

回归问题是一种监督学习任务，目标是根据输入特征预测连续的输出值。例如，根据天气数据和其他特征预测某一天的客流量。常见的回归模型包括线性回归、决策树、随机森林、支持向量回归等。在本代码中，我们将使用随机森林和LSTM（长短期记忆）神经网络来解决回归问题。

# 数据处理和预处理

## 数据读取

首先，我们需要从Excel文件中读取数据。我们使用pandas库来处理数据。pandas是Python中一个非常强大的数据分析库，可以方便地读取和处理各种格式的数据，包括CSV、Excel、SQL数据库等。



**解释：**

1. **import pandas as pd：**导入pandas库，并将其简化为pd，以便在后续代码中更方便地使用。

pandas库是一个功能强大且易于使用的数据分析库，提供了多种数据结构和数据分析工具。通过将其简化为pd，可以更方便地在代码中调用。

1. **import os：**导入os库，用于与操作系统进行交互，但在这个代码中没有实际使用。

os库提供了一种方便的方式来与操作系统进行交互，可以执行诸如文件和目录操作等任务。在这个代码中，虽然导入了os库，但实际上并没有使用它。

1. **for label in ['春柳-香工街', '香工街-中长街', '春柳-中长街']：**遍历每个路线标签进行操作。这是一个for循环，用于遍历三个不同的路线标签。每次循环都会处理一个标签的数据。
2. **data1 = pd.read\_excel('新建 Microsoft Excel 工作表.xlsx')：**读取第一个Excel文件中的数据。使用pandas的read\_excel函数从Excel文件中读取数据。这里的文件名是'新建 Microsoft Excel 工作表.xlsx'。
3. **pd.read\_excel：**pandas库中的一个函数，用于读取Excel文件，并将其转换为DataFrame对象，方便后续的数据处理。
4. **data2 = pd.read\_excel('新建 XLSX 工作表.xlsx')：**读取第二个Excel文件中的数据。同样使用read\_excel函数从另一个Excel文件中读取数据。文件名是'新建 XLSX 工作表.xlsx'。

读取的数据存储在data2变量中，后续将对其进行处理。

1. **feas = data2.T：**将第二个Excel文件的数据进行转置：
   * **data2.T：**将data2中的数据进行转置操作，即行列互换。这是为了使数据结构符合后续处理的需要。
2. **df = pd.DataFrame(feas.values[1:], columns=['天气种类', '最高气温', '最低气温', '是否为周末', '是否为节假日'])：**将转置后的数据转换为DataFrame，并设置列名。
3. **feas.values[1:]：**提取转置后数据的所有行（除去第一行，因为第一行通常是列名）。
4. **pd.DataFrame：**将提取的数据转换为一个DataFrame对象，并指定列名为‘天气种类’，‘最高气温’，‘最低气温’，‘是否为周末’，‘是否为节假日’。这样做的目的是为了方便后续的数据操作和处理。
5. **df['label']=data1[data1['Unnamed:0']== label].values[0][1:]：**从第一个Excel文件中提取对应路线标签的流量数据作为标签列：
   * **data1[data1['Unnamed: 0'] == label]：**筛选出第一个Excel文件中与当前标签匹配的行。
   * **.values[0][1:]：**提取匹配行中的所有列（除去第一列），这些数据将作为标签列添加到df中。
   * **df['label']：**将提取的数据赋值给df中的‘label’列，‘label’列用于存储每条路线的流量数据。

## 2.数据编码

在机器学习中，模型通常只能处理数值型数据，而不能直接处理类别型数据。因此，我们需要将类别数据转换为数值数据。这一步骤称为数据编码。

例子：

我们有一个列表示天气种类，可能包含如“晴天”、“雨天”、“阴天”等文本数据。为了让模型能够处理这些数据，我们需要将它们转换为整数编码。



**解释：**

1. **def trans(x)：**定义一个函数用于将‘是’和‘否’转换为1和0。这是一个自定义的函数，用于将表示二进制分类的数据从字符串形式转换为数值形式。

def关键字用于定义函数，trans是函数名，x是函数的参数。

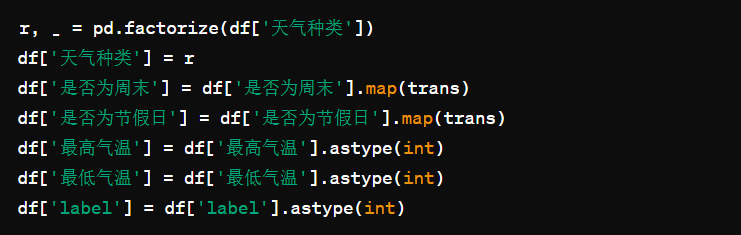
1. **if x == '是': return 1：**如果输入为‘是’，则返回1。

这是一个条件判断语句，用于检查输入参数x是否等于‘是’。如果条件为真，函数返回值为1。

1. **else: return 0：**否则返回0。

如果上述条件不成立，函数将返回0。这确保了函数能够处理两种可能的输入：‘是’和‘否’。

这个函数用于将‘是’转换为1，‘否’转换为0，以便后续将二分类数据（如是否为周末、是否为节假日）转换为数值数据。



**解释：**

1. **r, \_ = pd.factorize(df['天气种类'])：**对‘天气种类’列进行因子化处理，将其转换为数字编码。因子化是指将类别数据转换为整数，以便于机器学习模型处理。
   * **pd.factorize：**pandas库中的一个函数，用于将分类数据转换为数字编码。
   * **df['天气种类']：**要进行编码的列名。
   * **r：**编码后的结果。
   * **\_：**因子化的唯一值（在这里我们不需要）。
2. **df['天气种类'] = r：**将因子化后的结果赋值给‘天气种类’列。

这一步将因子化后的数据替换原有的‘天气种类’列。现在，‘天气种类’列中的每个值都是一个整数，表示不同的天气种类。

1. **df['是否为周末'] = df['是否为周末'].map(trans)：**将‘是否为周末’列中的‘是’和‘否’转换为1和0。map函数将trans函数应用于‘是否为周末’列中的每个值。转换后的结果是一个包含1和0的列，表示是否为周末。
   * **df['是否为周末'].map(trans)：**使用自定义的trans函数将‘是否为周末’列中的值进行转换。
2. **df['是否为节假日'] = df['是否为节假日'].map(trans)：**将‘是否为节假日’列中的‘是’和‘否’转换为1和0。

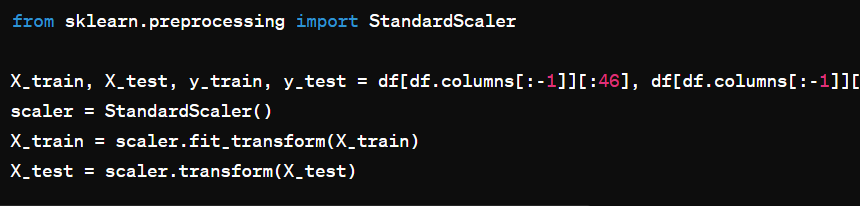
同样使用trans函数将‘是否为节假日’列中的值进行转换。这一步确保‘是否为节假日’列中的每个值都是数值类型。

1. **df['最高气温'] = df['最高气温'].astype(int)：**将‘最高气温’列的数据类型转换为整数。
   * **astype(int)：**pandas函数，用于将数据类型转换为整数。
2. **df['最低气温'] = df['最低气温'].astype(int)：**将‘最低气温’列的数据类型转换为整数。同样使用astype(int)函数将‘最低气温’列中的值转换为整数类型。
3. **df['label'] = df['label'].astype(int)：**将‘label’列的数据类型转换为整数。

label列存储了路线的流量数据，使用astype(int)函数将其转换为整数类型，确保所有数据都是数值类型，以便后续的模型训练和评估。

## 3.数据标准化

为了使数据在模型训练过程中表现更好，我们通常需要对特征进行标准化处理，使其具有相同的尺度和分布。标准化的目的是将不同特征的值转换到相同的范围，以避免模型训练过程中某些特征的值过大或过小而导致模型性能下降。



**解释：**

1. **from sklearn.preprocessing import StandardScaler：**从sklearn库中导入StandardScaler类，用于数据标准化。
   * StandardScaler是sklearn库中的一个类，用于对特征数据进行标准化处理。
   * 标准化处理将特征数据转换为均值为0，标准差为1的分布，以减少不同特征之间的量纲差异。
2. **X\_train, X\_test, y\_train, y\_test：**将数据分为训练集和测试集，前46条数据作为训练集，后面的数据作为测试集。
   * **df[df.columns[:-1]][:46]：**提取前46行的特征数据，作为训练集的特征数据。
   * **df[df.columns[:-1]][46:]：**提取从第46行开始的特征数据，作为测试集的特征数据。
   * **df['label'][:46]**：提取前46行的标签数据，作为训练集的标签数据。
   * **df['label'][46:]：**提取从第46行开始的标签数据，作为测试集的标签数据。

这里我们将数据分为训练集和测试集，以便后续的模型训练和评估。

1. **scaler = StandardScaler()：**创建一个StandardScaler对象，用于数据标准化处理。
   * **StandardScaler()：**初始化StandardScaler对象。

StandardScaler对象将用于对训练集和测试集的特征数据进行标准化处理。

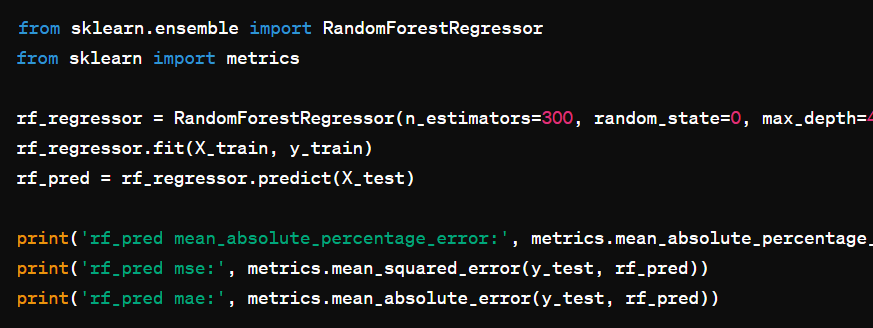
1. **X\_train = scaler.fit\_transform(X\_train)：**对训练数据进行标准化转换，使其具有均值为0和标准差为1的分布。
   * **scaler.fit\_transform(X\_train)：**首先拟合StandardScaler对象到训练数据，然后对训练数据进行标准化转换。
   * 通过标准化转换，训练数据中的特征将具有均值为0和标准差为1的分布。
   * 这一步的目的是使训练数据中的特征值在相同的量纲上，有助于提升模型的训练效果。
2. **X\_test = scaler.transform(X\_test)：**使用与训练数据相同的标准化器对测试数据进行转换。
   * **scaler.transform(X\_test)：**使用之前拟合到训练数据的StandardScaler对象对测试数据进行标准化转换。

这一步确保测试数据中的特征值与训练数据中的特征值在相同的尺度上，以便模型能够正确地进行预测。

# 三. 模型训练和评估

## 1.随机森林回归模型

随机森林是一种集成学习方法，通过构建多个决策树来进行训练和预测。每棵树都是在数据的不同子集上训练的，最后通过综合这些树的预测结果来提高整体模型的准确性和稳定性。随机森林在处理高维数据和防止过拟合方面表现出色。



**解释：**

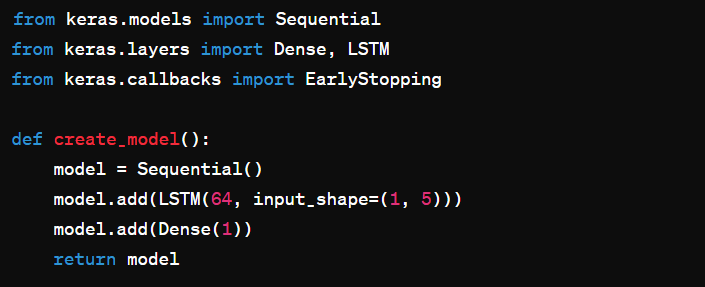
1. **from sklearn.ensemble import RandomForestRegressor：**从sklearn库中导入RandomForestRegressor类，用于构建随机森林回归模型。
   * RandomForestRegressor是sklearn库中的一个类，用于构建随机森林回归模型。
   * 随机森林通过训练多个决策树并结合它们的预测结果来提高模型的准确性和稳定性。
2. **from sklearn import metrics：**从sklearn库中导入metrics模块，用于评估模型性能。
   * metrics模块提供了多种评估模型性能的工具，包括均方误差（MSE）、平均绝对误差（MAE）等。
   * 这些评估指标可以帮助我们了解模型的预测性能。
3. **rf\_regressor=RandomForestRegressor(n\_estimators=300,random\_state=0, max\_depth=4)：**创建一个随机森林回归模型，设置决策树数量为300，随机种子为0，最大深度为4。
   * **RandomForestRegressor(n\_estimators=300,random\_state=0, max\_depth=4)：**初始化一个RandomForestRegressor对象，设置参数：

* **n\_estimators=300：**决策树的数量为300。
* **random\_state=0：**随机种子为0，确保结果可重复。
* **max\_depth=4：**决策树的最大深度为4，限制树的深度以防止过拟合。

1. **rf\_regressor.fit(X\_train, y\_train)：**用训练数据训练随机森林模型。
   * **rf\_regressor.fit(X\_train, y\_train)：**将训练数据（特征和标签）传递给随机森林模型，进行模型训练。
   * 模型将学习训练数据中的特征和标签之间的关系。
2. **rf\_pred = rf\_regressor.predict(X\_test)：**使用训练好的模型对测试数据进行预测。
   * **rf\_regressor.predict(X\_test)：**使用训练好的随机森林模型对测试数据进行预测，得到预测结果rf\_pred。
   * 预测结果是模型对测试数据中的流量进行预测的值。
3. **print('rf\_pred mean\_absolute\_percentage\_error:', metrics.mean\_absolute\_percentage\_error(y\_test, rf\_pred))：**计算并打印预测结果的均绝对百分比误差（MAPE）。
   * **metrics.mean\_absolute\_percentage\_error(y\_test, rf\_pred)：**计算测试数据真实值y\_test与预测值rf\_pred之间的均绝对百分比误差。
   * MAPE是评估回归模型性能的一种常用指标，表示预测值与真实值之间的平均百分比误差。
4. **print('rf\_pred mse:', metrics.mean\_squared\_error(y\_test, rf\_pred))：**计算并打印预测结果的均方误差（MSE）。
   * **metrics.mean\_squared\_error(y\_test, rf\_pred)：**计算测试数据真实值y\_test与预测值rf\_pred之间的均方误差。
   * MSE是评估回归模型性能的一种常用指标，表示预测值与真实值之间的平均平方误差。
5. **print('rf\_pred mae:', metrics.mean\_absolute\_error(y\_test, rf\_pred))：**计算并打印预测结果的平均绝对误差（MAE）。
   * **metrics.mean\_absolute\_error(y\_test, rf\_pred)：**计算测试数据真实值y\_test与预测值rf\_pred之间的平均绝对误差。
   * MAE是评估回归模型性能的一种常用指标，表示预测值与真实值之间的平均绝对误差。

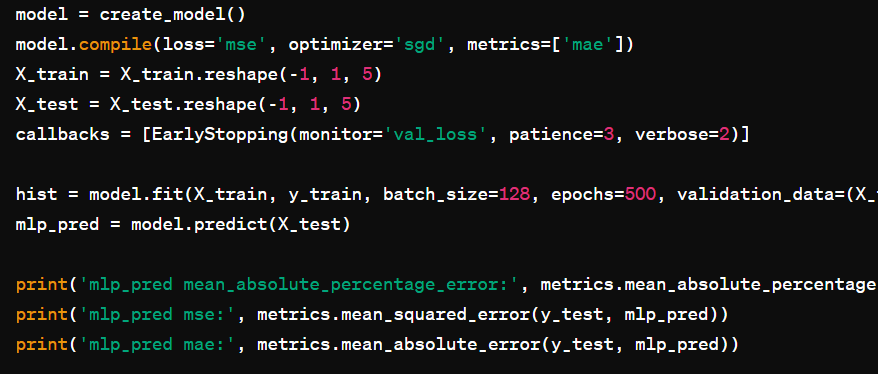
## 2.LSTM神经网络

**LSTM（长短期记忆）**是一种特殊的递归神经网络（RNN），特别适用于处理和预测时间序列数据。LSTM通过引入记忆单元，可以捕捉数据中的长期依赖关系。



**解释：**

1. **from keras.models import Sequential：**从keras库中导入Sequential类，用于构建神经网络的顺序模型。
   * Sequential是Keras中的一个类，用于创建顺序模型。顺序模型是多个网络层线性堆叠的模型。
2. **from keras.layers import Dense, LSTM：**从keras库中导入Dense和LSTM层，用于构建神经网络层。
   * **Dense层：**全连接层，LSTM层：长短期记忆层，用于处理时间序列数据。
3. **from keras.callbacks import EarlyStopping：**从keras库中导入EarlyStopping回调函数，用于在训练过程中监控验证损失并提前停止训练以防止过拟合。
   * EarlyStopping是Keras中的一个回调函数，用于在验证损失不再下降时提前停止训练，以防止模型过拟合。
4. **def create\_model()：定义一个函数用于创建LSTM模型。**
   * def关键字用于定义函数，create\_model是函数名，该函数将创建并返回一个LSTM神经网络模型。
5. **model = Sequential()：创建一个顺序模型。**
   * **Sequential()：**初始化一个顺序模型对象。
6. **model.add(LSTM(64, input\_shape=(1, 5)))：**添加一个LSTM层，设置隐含层神经元数量为64，输入形状为(1, 5)。
   * **model.add(LSTM(64, input\_shape=(1, 5)))：**向模型中添加一个LSTM层。
   * **64：**隐含层神经元的数量。
   * **input\_shape=(1, 5)：**输入数据的形状为（时间步数，特征数量），这里时间步数为1，特征数量为5。
7. **model.add(Dense(1))：**添加一个全连接层，输出一个值。
   * **model.add(Dense(1))：**向模型中添加一个全连接层（Dense层）。
   * 1：输出神经元的数量，这里表示输出一个值。
8. **return model：**函数返回创建的LSTM模型。



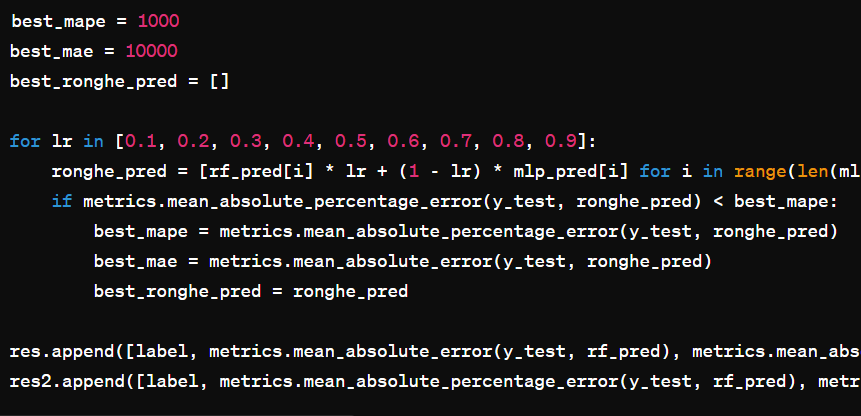
**解释：**

1. **model = create\_model()：**创建LSTM模型。
   * **create\_model()：**调用create\_model函数，创建一个LSTM模型。
   * **model：**存储创建的LSTM模型对象。
2. **model.compile(loss='mse', optimizer='sgd', metrics=['mae'])：**编译模型，设置损失函数为均方误差（MSE），优化器为随机梯度下降（SGD），评价指标为平均绝对误差（MAE）。
   * **model.compile(loss='mse',optimizer='sgd',metrics=['mae'])：**编译模型，设置训练过程中使用的损失函数、优化器和评价指标。
   * **loss='mse'：**损失函数为均方误差（MSE）。
   * **optimizer='sgd'：**优化器为随机梯度下降（SGD）。
   * **metrics=['mae']：**评价指标为平均绝对误差（MAE）。
3. **X\_train = X\_train.reshape(-1, 1, 5)：**将训练数据调整为LSTM所需的输入形状。
   * **X\_train.reshape(-1, 1, 5)：**将训练数据调整为三维形状（样本数量，时间步数，特征数量）。
   * **-1：**自动计算样本数量。
   * **1：**时间步数为1。
   * **5：**特征数量为5。
4. **X\_test = X\_test.reshape(-1, 1, 5)：**将测试数据调整为LSTM所需的输入形状。
   * **X\_test.reshape(-1, 1, 5)：**将测试数据调整为三维形状（样本数量，时间步数，特征数量）。
   * 这一步确保训练数据和测试数据的形状一致。
5. **callbacks=[EarlyStopping(monitor='val\_loss',patience=3,verbose=2)]：**设置早停回调函数，监控验证损失（val\_loss），如果连续3次没有改善，则停止训练。
   * **EarlyStopping(monitor='val\_loss', patience=3, verbose=2)：**初始化一个EarlyStopping对象。
   * **monitor='val\_loss'：**监控的指标为验证损失（val\_loss）。
   * **patience=3：**如果验证损失连续3次没有改善，则停止训练。
   * **verbose=2：**打印详细日志信息。
6. **hist = model.fit(X\_train, y\_train, batch\_size=128, epochs=500, validation\_data=(X\_test, y\_test), callbacks=callbacks)：**训练模型，设置批次大小为128，最大训练轮数为500，使用验证数据进行验证，并使用早停回调函数。
   * **model.fit(X\_train, y\_train, batch\_size=128, epochs=500, validation\_data=(X\_test, y\_test), callbacks=callbacks)：**训练LSTM模型。
   * **X\_train, y\_train：**训练数据和标签。
   * **batch\_size=128：**批次大小为128。
   * **epochs=500：**最大训练轮数为500。
   * **validation\_data=(X\_test, y\_test)：**验证数据和标签。
   * **callbacks=callbacks：**早停回调函数。
7. **mlp\_pred = model.predict(X\_test)：**使用训练好的模型对测试数据进行预测。
   * **model.predict(X\_test)：**使用训练好的LSTM模型对测试数据进行预测，得到预测结果mlp\_pred。
   * 预测结果是模型对测试数据中的流量进行预测的值。
8. **print('mlp\_predmean\_absolute\_percentage\_error:',metrics.mean\_absolute\_percentage\_error(y\_test, mlp\_pred))：**计算并打印预测结果的均绝对百分比误差（MAPE）。
   * **metrics.mean\_absolute\_percentage\_error(y\_test, mlp\_pred)：**计算测试数据真实值y\_test与预测值mlp\_pred之间的均绝对百分比误差。
   * MAPE是评估回归模型性能的一种常用指标，表示预测值与真实值之间的平均百分比误差。
9. **print('mlp\_predmse:',metrics.mean\_squared\_error(y\_test, mlp\_pred))：**计算并打印预测结果的均方误差（MSE）。
   * **metrics.mean\_squared\_error(y\_test, mlp\_pred)：**计算测试数据真实值y\_test与预测值mlp\_pred之间的均方误差。
   * MSE是评估回归模型性能的一种常用指标，表示预测值与真实值之间的平均平方误差。
10. **print('mlp\_predmae:',metrics.mean\_absolute\_error(y\_test, mlp\_pred))：**计算并打印预测结果的平均绝对误差（MAE）。
    * **metrics.mean\_absolute\_error(y\_test, mlp\_pred)：**计算测试数据真实值y\_test与预测值mlp\_pred之间的平均绝对误差。
    * MAE是评估回归模型性能的一种常用指标，表示预测值与真实值之间的平均绝对误差。

# 四．模型融合

## 1.加权平均融合

为了提升模型的预测性能，我们可以将多个模型的预测结果进行加权平均，从而得到更好的预测结果。这里我们通过调整不同的权重系数，找到使误差最小的权重组合。



**解释：**

1. **best\_mape = 1000：**初始化最小均绝对百分比误差（MAPE）值为1000。
   * best\_mape是用于存储最小MAPE值的变量，初始值设为1000，这是一个较大的数值，用于后续比较更新。
2. **best\_mae = 10000：**初始化最小平均绝对误差（MAE）值为10000。
   * best\_mae是用于存储最小MAE值的变量，初始值设为10000，这也是一个较大的数值，用于后续比较更新。
3. **best\_ronghe\_pred = []：初始化最优融合预测结果为空列表。**
   * best\_ronghe\_pred用于存储当前找到的最优融合模型的预测结果，初始为空列表。
4. **for lr in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]：**遍历不同的权重系数（lr）值。
   * for lr in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]表示遍历一组权重系数lr，这些权重系数用于加权平均融合模型的预测结果。
   * lr的范围从0.1到0.9，分别表示不同的权重比例，以0.1为步长。
5. **ronghe\_pred = [rf\_pred[i] \* lr + (1 - lr) \* mlp\_pred[i] for i in range(len(mlp\_pred))]：**计算随机森林和LSTM模型预测结果的加权平均值，权重为lr。
   * **ronghe\_pred：**用于存储融合模型的预测结果。
   * **[rf\_pred[i] \* lr + (1 - lr) \* mlp\_pred[i] for i in range(len(mlp\_pred))]**：对每个预测值进行加权平均。

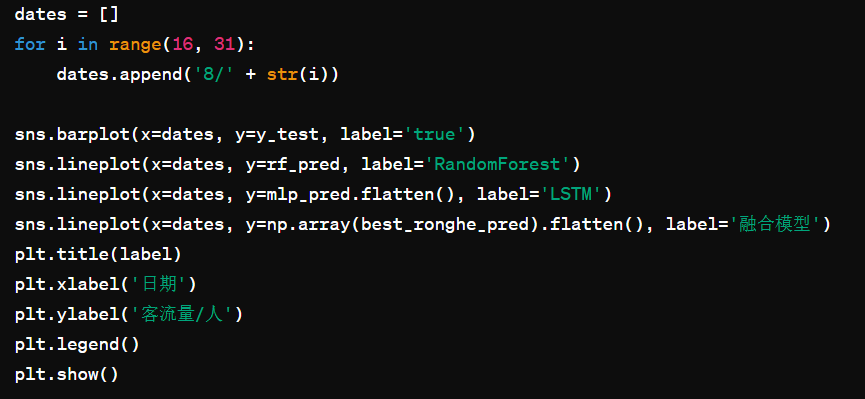
* **rf\_pred[i] \* lr：**随机森林模型的预测值乘以权重系数lr。
* **(1 - lr) \* mlp\_pred[i]：**LSTM模型的预测值乘以权重系数1 - lr。
* **for i in range(len(mlp\_pred))：**遍历所有预测值。

1. **if metrics.mean\_absolute\_percentage\_error(y\_test, ronghe\_pred) < best\_mape：**如果当前融合预测结果的MAPE小于最小MAPE值，则更新最小MAPE值和最小MAE值，并保存当前最优融合预测结果。
   * **metrics.mean\_absolute\_percentage\_error(y\_test, ronghe\_pred)：**计算当前融合模型的MAPE。
   * **if metrics.mean\_absolute\_percentage\_error(y\_test, ronghe\_pred) < best\_mape：**如果当前计算的MAPE小于之前记录的最小MAPE值，则更新最小MAPE值。
2. **best\_mape=metrics.mean\_absolute\_percentage\_error(y\_test,ronghe\_pred)：**更新最小MAPE值。
   * **best\_mape：**更新为当前融合模型的MAPE值。
3. **best\_mae = metrics.mean\_absolute\_error(y\_test, ronghe\_pred)：**更新最小MAE值。
   * **best\_mae：**更新为当前融合模型的MAE值。
4. **best\_ronghe\_pred = ronghe\_pred：**保存当前最优融合预测结果。
   * **best\_ronghe\_pred：**更新为当前计算的融合预测结果。
5. **res.append([label, metrics.mean\_absolute\_error(y\_test, rf\_pred), metrics.mean\_absolute\_error(y\_test,mlp\_pred),metrics.mean\_absolute\_error(y\_test, best\_ronghe\_pred)])：**将当前路线的随机森林、LSTM和融合模型的MAE添加到结果列表。
   * **res.append([label,metrics.mean\_absolute\_error(y\_test, rf\_pred),metrics.mean\_absolute\_error(y\_test,mlp\_pred),metrics.mean\_absolute\_error(y\_test, best\_ronghe\_pred)])：**将当前路线标签和三个模型的MAE结果添加到res列表中。
   * **metrics.mean\_absolute\_error(y\_test, rf\_pred)：**计算随机森林模型的MAE。
   * **metrics.mean\_absolute\_error(y\_test, mlp\_pred)：**计算LSTM模型的MAE。
   * **metrics.mean\_absolute\_error(y\_test, best\_ronghe\_pred)：**计算最优融合模型的MAE。
6. **res2.append([label, metrics.mean\_absolute\_percentage\_error(y\_test, rf\_pred), metrics.mean\_absolute\_percentage\_error(y\_test, mlp\_pred), metrics.mean\_absolute\_percentage\_error(y\_test,best\_ronghe\_pred)])：**将当前路线的随机森林、LSTM和融合模型的MAPE添加到结果列表。
   * **res2.append([label,metrics.mean\_absolute\_percentage\_error(y\_test,rf\_pred),metrics.mean\_absolute\_percentage\_error(y\_test,mlp\_pred),metrics.mean\_absolute\_percentage\_error(y\_test, best\_ronghe\_pred)])：**将当前路线标签和三个模型的MAPE结果添加到res2列表中。
   * **metrics.mean\_absolute\_percentage\_error(y\_test, rf\_pred)：**计算随机森林模型的MAPE。
   * **metrics.mean\_absolute\_percentage\_error(y\_test, mlp\_pred)：**计算LSTM模型的MAPE。
   * **metrics.mean\_absolute\_percentage\_error(y\_test,best\_ronghe\_pred)：**计算最优融合模型的MAPE。

# 结果可视化

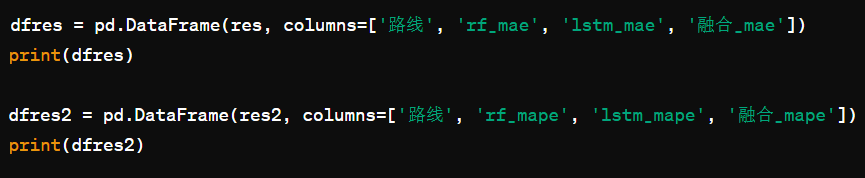
## 1.汇总结果

为了更直观地展示模型的预测结果，我们可以使用可视化工具对实际值和预测值进行对比。我们将绘制真实值、随机森林预测值、LSTM预测值和融合模型预测值的图表，以便直观地比较这些模型的预测性能。



**解释：**

1. **dates = []：**初始化日期列表。
2. **for i in range(16, 31)：**遍历16到31之间的数字。
3. **dates.append('8/' + str(i))：**将每个数字转换为‘8/数字’的格式并添加到日期列表中。
4. **sns.barplot(x=dates, y=y\_test, label='true')：**绘制真实值的条形图。
   * **sns.barplot(x=dates, y=y\_test, label='true')：**使用seaborn库绘制真实值的条形图。
   * **x=dates：**设置X轴为日期列表。
   * **y=y\_test：**设置Y轴为测试数据的真实值。
   * **label='true'：**设置图例标签为‘true’。
5. **sns.lineplot(x=dates, y=rf\_pred, label='RandomForest')**：绘制随机森林预测值的折线图。
   * **sns.lineplot(x=dates, y=rf\_pred, label='RandomForest')：**使用seaborn库绘制随机森林预测值的折线图。
   * **x=dates：**设置X轴为日期列表。
   * **y=rf\_pred：**设置Y轴为随机森林模型的预测值。
   * **label='RandomForest'：**设置图例标签为‘RandomForest’。
6. **sns.lineplot(x=dates, y=mlp\_pred.flatten(), label='LSTM')：**绘制LSTM预测值的折线图。
   * **sns.lineplot(x=dates, y=mlp\_pred.flatten(), label='LSTM')：**使用seaborn库绘制LSTM预测值的折线图。
   * **x=dates：**设置X轴为日期列表。
   * **y=mlp\_pred.flatten()：**设置Y轴为LSTM模型的预测值，并将其展平为一维数组。
   * **label='LSTM'：**设置图例标签为‘LSTM’。
7. **sns.lineplot(x=dates,y=np.array(best\_ronghe\_pred).flatten(), label='融合模型')：**绘制融合模型预测值的折线图。
   * **sns.lineplot(x=dates,y=np.array(best\_ronghe\_pred).flatten(), label='融合模型')：**使用seaborn库绘制融合模型预测值的折线图。
   * **x=dates：**设置X轴为日期列表。
   * **y=np.array(best\_ronghe\_pred).flatten()：**设置Y轴为融合模型的预测值，并将其展平为一维数组。
   * **label='融合模型'：**设置图例标签为‘融合模型’。
8. **plt.title(label)：**使用matplotlib库设置图表标题为当前路线标签。
9. **plt.xlabel('日期')：**使用matplotlib库设置X轴标签为‘日期’。
10. **plt.ylabel('客流量/人')：**使用matplotlib库设置Y轴标签为‘客流量/人’。
11. **plt.legend()：**使用matplotlib库显示图例。
12. **plt.show()：**使用matplotlib库显示图表。

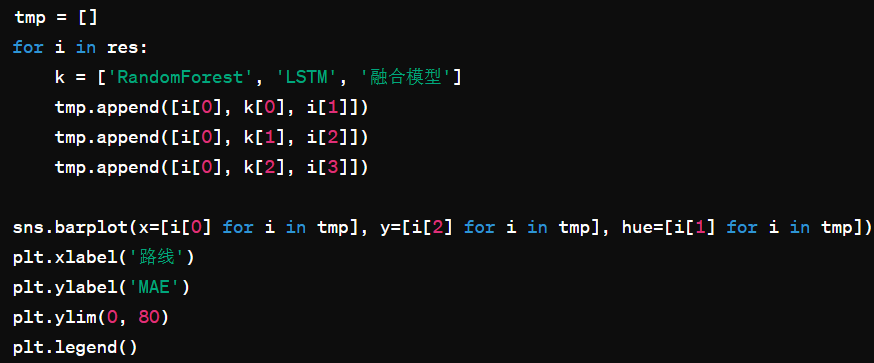


**解释：**

1. **dfres = pd.DataFrame(res, columns=['路线', 'rf\_mae', 'lstm\_mae', '融合\_mae'])：**将MAE结果转换为DataFrame，并设置列名。
   * **pd.DataFrame(res, columns=['路线', 'rf\_mae', 'lstm\_mae', '融合\_mae'])：**使用pandas库将MAE结果转换为DataFrame对象，并设置列名为‘路线’，‘rf\_mae’，‘lstm\_mae’，‘融合\_mae’。
2. **print(dfres)：**使用print函数打印MAE结果的DataFrame，方便查看和分析。
3. **dfres2 = pd.DataFrame(res2, columns=['路线', 'rf\_mape', 'lstm\_mape', '融合\_mape'])：**将MAPE结果转换为DataFrame，并设置列名。
   * **pd.DataFrame(res2, columns=['路线', 'rf\_mape', 'lstm\_mape', '融合\_mape'])：**使用pandas库将MAPE结果转换为DataFrame对象，并设置列名为‘路线’，‘rf\_mape’，‘lstm\_mape’，‘融合\_mape’。
4. **print(dfres2)：**使用print函数打印MAPE结果的DataFrame，方便查看和分析。

## 最终可视化

最后，我们可以使用条形图对不同模型的MAE进行可视化对比，以直观展示模型的性能差异。



**解释：**

1. **tmp = []：**初始化临时列表。
   * **[]：**创建一个空列表，用于存储模型的MAE结果。
2. **for i in res：**遍历MAE结果列表。
   * for循环用于遍历res中的每个元素（每条路线的MAE结果）。
3. **k = ['RandomForest', 'LSTM', '融合模型']：**创建包含模型名称的列表。
   * **['RandomForest', 'LSTM', '融合模型']：**一个包含三个模型名称的列表。
4. **tmp.append([i[0], k[0], i[1]])：**将路线名称、模型名称（RandomForest）和对应的MAE结果添加到临时列表。
5. **tmp.append([i[0], k[1], i[2]])：**将路线名称、模型名称（LSTM）和对应的MAE结果添加到临时列表。
6. **tmp.append([i[0], k[2], i[3]])：**将路线名称、模型名称（融合模型）和对应的MAE结果添加到临时列表。
7. **sns.barplot(x=[i[0] for i in tmp], y=[i[2] for i in tmp], hue=[i[1] for i in tmp])：**使用seaborn库绘制MAE结果的条形图，按照路线分组，并使用不同颜色表示不同模型。
   * **x=[i[0] for i in tmp]：**设置X轴为路线名称。
   * **y=[i[2] for i in tmp]：**设置Y轴为MAE结果。
   * **hue=[i[1] for i in tmp]：**使用不同颜色表示不同模型。
8. **plt.xlabel('路线')：**使用matplotlib库设置X轴标签为‘路线’。
9. **plt.ylabel('MAE')：**使用matplotlib库设置Y轴标签为‘MAE’。
10. **plt.ylim(0, 80)：**使用matplotlib库设置Y轴的范围为0到80。
11. **plt.legend()：**使用matplotlib库显示图例。