方法说明

路线安排

由于稻飞虱具有迁飞性，因此当天的气象数据和虫情数据匹配作为模型的训练集并不精准，因此我们考虑把气象数据分别前移5天，10天和15天，在计算前移后的气象数据与下游地区虫数的相关性。我们提出了一迭代一模型的说法，因为不同地区的气候条件， 地形影响和人文因素对虫情的发生都会有影响，所以我们不能把整个路径数据的气象数据拿来用作为下游虫数的训练数据，因此我们只考虑上游的气象因素和下游的虫情数量，这样确保了模型的准确性。

我们把整个稻飞虱迁飞的分为了9各阶段：

第一个阶段：丘北-锦屏
第二个阶段：锦屏-洪江
第三个阶段：洪江-秀山
第四个阶段：秀山-桂阳
第五个阶段：桂阳-监利
第六个阶段：监利-固始
第七个阶段：固始-徽州
第八个阶段：徽州-盐都
第九个阶段：盐都-奉贤

前面是使用的气象数据的地区，后面就是下游的虫情数据，假如我们想要预测第三个阶段秀山的虫情数据，我们就要用秀山上游的洪江的气象数据作为匹配的数据集来对秀山的虫情数据进行预测。

但是考虑到用于训练的数据集的数据比较小，一迭代一模型的方法可能会出现预测不准确的情况，因此我们决定在同一稻区内，实行数据的累加，例如第一阶段，第二阶段和第三阶段用的虫情数据都是西南稻区的数据，第一阶段的训练集就只有丘北的气象数据和 锦屏的虫情数据，而第二阶段的训练就是丘北，锦屏的气象数据和锦屏，洪江的虫情数据，这样不断的迭代累加，就能提高模型预测精准度。

数据处理

我们在每个阶段都设置了四个文件，分别是当天的气象数据和当天下游虫数的匹配数据集，气象数据前移5天与下游虫数的匹配数据集，气象数据前移10天与下游虫数的匹配数据集，气象数据前移15天与下游虫数的匹配训练集。

为了实现对照，我们把预测数据集分为了两组，一组累加数据，一组没有累加数据，用来对照预测的精准度是否有变化。

代码实现

我们使用文件名为`preiction`的py文件作为预测的代码

导入文件

导入的文件或文件夹需要与py文件在同一目录下。 在下面的示例中，我们的训练集是固始到徽州这一段的气象数据和虫情数据，如果想要预测其他地方的虫情数据，只需要更改文件名就可以了。

```
# 导入数据
#导入训练数据集
df = pd.read_excel('气象地点+虫情地点数据/Step7_固始-徽州/Step7_固始-徽州.xlsx')
df_5 = pd.read_excel('气象地点+虫情地点数据/Step7_固始-徽州/Step7_固始-徽州(气象数据前移5天).xlsx')
df_10 = pd.read_excel('气象地点+虫情地点数据/Step7_固始-徽州/Step7_固始-徽州(气象数据前移10天).xlsx')
df_15 = pd.read_excel('气象地点+虫情地点数据/Step7_固始-徽州/Step7_固始-徽州(气象数据前移15天).xlsx')

df_pred = pd.read_csv('cmip_Nor/SSP1/江淮固始ssp126.csv') #导入预测数据
```

处理数据

由于我们迭代数据集中包含日期列，而大多机器学习的模型并不能处理日期格式的数据，因此要把日期格式的数据转化为时间戳类型

```
missing_dates = df[df['date'].isna()]
print(missing_dates)
df['date'] = df['date'].apply(lambda x: x.timestamp())
# 查找 NaT 值

missing_dates = df_5[df_5['date'].isna()]
print(missing_dates)
df_5['date'] = df_5['date'].apply(lambda x: x.timestamp())

df_10['date'] = df_10['date'].apply(lambda x: x.timestamp())
missing_dates = df_10[df_10['date'].isna()]
print(missing_dates)
df_15['date'] = df_15['date'].apply(lambda x: x.timestamp())
missing_dates = df_15[df_15['date'].isna()]
print(missing_dates)
```

因为预测稻飞虱的具体数量十分困难，因此我们根据历史数据的虫数，进行`log(x+1)`计算,将稻飞虱的发生分为了5各等级

稻飞虱暴发等级	稻飞虱灯诱数量(lg)形式
Level1	$0 \leq x \leq 1$
Level2	$1 \leq x \leq 2$
Level3	$2 \leq x \leq 3$
Level4	$3 \leq x \leq 4$
Level5	$4 \leq x \leq 5$

代码的计算步骤如下:

```
df['白背飞虱'] = np.log(df['白背飞虱'] + 1)
df_5['白背飞虱'] = np.log(df_5['白背飞虱'] + 1)
df_10['白背飞虱'] = np.log(df_10['白背飞虱'] + 1)
df_15['白背飞虱'] = np.log(df_15['白背飞虱'] + 1)

df['褐飞虱'] = np.log(df['褐飞虱'] + 1)
df_5['褐飞虱'] = np.log(df_5['褐飞虱'] + 1)
df_10['褐飞虱'] = np.log(df_10['褐飞虱'] + 1)
df_15['褐飞虱'] = np.log(df_15['褐飞虱'] + 1)
```

绘制相关性矩阵

通过绘制一个热力图来直观地展示我们的特征值和目标值中各特征之间的相关性，帮助我们识别哪些特征之间可能存在强烈的线性关系。

```
# 相关性矩阵
correlation_matrix = df.corr()
# 绘制热力图，查看各特征与目标变量的相关性
selected_rows_cols = correlation_matrix.loc[['date', 'year', 'month', 'hou',
'1000hPa_air', '850hPa_air', '1000hPa_rhum', '850hPa_rhum', '1000hPa_wind',
'850hPa_wind', '1000hPa_azimuth', '850hPa_azimuth', '1000hPa_number',
'850hPa_number', '850hPa_omega'], ['白背飞虱', '褐飞虱']]
plt.figure(figsize=(20, 12))
sns.heatmap(selected_rows_cols, annot=True, cmap='coolwarm', fmt='.2f',
linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()

correlation_matrix1 = df_5.corr()
# 绘制热力图，查看各特征与目标变量的相关性
selected_rows_cols = correlation_matrix1.loc[['date', 'year', 'month', 'hou',
'1000hPa_air', '850hPa_air', '1000hPa_rhum', '850hPa_rhum', '1000hPa_wind',
'850hPa_wind', '1000hPa_azimuth', '850hPa_azimuth', '1000hPa_number',
'850hPa_number', '850hPa_omega'], ['白背飞虱', '褐飞虱']]
plt.figure(figsize=(20, 12))
sns.heatmap(selected_rows_cols, annot=True, cmap='coolwarm', fmt='.2f',
linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()


correlation_matrix2 = df_10.corr()
# 绘制热力图，查看各特征与目标变量的相关性
selected_rows_cols = correlation_matrix2.loc[['date', 'year', 'month', 'hou',
'1000hPa_air', '850hPa_air', '1000hPa_rhum', '850hPa_rhum', '1000hPa_wind',
'850hPa_wind', '1000hPa_azimuth', '850hPa_azimuth', '1000hPa_number',
'850hPa_number', '850hPa_omega'], ['白背飞虱', '褐飞虱']]
plt.figure(figsize=(20, 12))
sns.heatmap(selected_rows_cols, annot=True, cmap='coolwarm', fmt='.2f',
linewidths=0.5)
```

```
plt.title("Correlation Matrix")
plt.show()

correlation_matrix3 = df_15.corr()
# 绘制热力图，查看各特征与目标变量的相关性
selected_rows_cols = correlation_matrix3.loc[['date', 'year', 'month', 'hou',
'1000hPa_air', '850hPa_air', '1000hPa_rhum', '850hPa_rhum', '1000hPa_wind',
'850hPa_wind', '1000hPa_azimuth', '850hPa_azimuth', '1000hPa_number',
'850hPa_number', '850hPa_omega'], ['白背飞虱', '褐飞虱']]
plt.figure(figsize=(20, 12))
sns.heatmap(selected_rows_cols, annot=True, cmap='coolwarm', fmt='.2f',
linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()
```

我们分别绘制不同的前移的气象数据，可以对比前移天数对于稻飞虱虫数的影响。

特征强化选择

在之前的实验中，由于未对特征相关性进行筛选，预测过程中混入了过多的噪声，导致预测结果偏差较大。因此，有必要进行特征选择，筛选出具有较强相关性的关键因素，去除相关性较弱甚至存在负相关的因素，从而提升预测的准确性，获得更加可靠的预测结果。

通过上面的相关性热力图我们可以看出，与白背飞虱和褐飞虱两种稻飞虱相关性强的特征值并不完全一样，因此我需要把白背飞虱和褐飞虱两种稻飞虱分开了进行特征选择。

```
# 选择和目标变量"白背飞虱"和"褐飞虱"相关性较强的特征
target = "白背飞虱"
cor_target = correlation_matrix[target].sort_values(ascending=False)
print(cor_target)

# 选择和目标变量相关性较高的特征
# 可以选择相关性高于某个阈值的特征，假设阈值为0.1
# 直接在筛选相关性绝对值大于0.1的特征索引时，排除指定的特征
selected_features = cor_target[(cor_target > 0.1) & (~cor_target.index.isin(['白背
飞虱', '褐飞虱']))].index.tolist()
print("Selected features:", selected_features)

                                    .
                                    .
                                    .
                                    .
                                    .

target1 = "褐飞虱"
cor_target1 = correlation_matrix[target1].sort_values(ascending=False) #这里需要修
改！！！！！！！！！！！！！！！！！
print(cor_target1)

# 选择和目标变量相关性较高的特征
# 可以选择相关性高于某个阈值的特征，假设阈值为0.1
```

```
# 直接在筛选相关性绝对值大于0.1的特征索引时，排除指定的特征
selected_features1 = cor_target1[(cor_target1 > 0.1) &
(~cor_target1.index.isin(['白背飞虱', '褐飞虱']))].index.tolist()
print("Selected features:", selected_features1)
```

在上述相关性矩阵中，已计算出每个特征与目标变量之间的相关性。在统计学中，相关性系数大于 0.1 表示变量之间存在轻微的线性关系，通常用于初步筛选，以保留可能对目标变量有一定影响的特征，避免遗漏潜在的重要因素。而相关性系数大于 0.3 则表明变量之间存在较明显的线性关系，此类特征对目标变量的影响更显著，适合用于进一步分析或模型训练。基于此，我们筛选出相关性大于 0.1 的特征，用于模型预测，以提高预测效果。

白背飞虱的特征选择：

```
白背飞虱              1.000000
褐飞虱               0.739762
850hPa_air         0.497016
1000hPa_air        0.484501
1000hPa_rhum       0.380256
850hPa_rhum        0.206195
month              0.118283
850hPa_wind        0.020190
hou                0.002761
1000hPa_number     0.001677
1000hPa_azimuth    0.000384
850hPa_number     -0.025145
850hPa_azimuth    -0.029493
date              -0.160079
year              -0.165182
1000hPa_wind      -0.184793
850hPa_omega      -0.249286
Name: 白背飞虱, dtype: float64
Selected features: ['850hPa_air', '1000hPa_air', '1000hPa_rhum', '850hPa_rhum',
'month']
```

褐飞虱的特征选择：

```
褐飞虱               1.000000
白背飞虱              0.739762
850hPa_air         0.452913
1000hPa_air        0.436401
1000hPa_rhum       0.335472
month              0.201087
850hPa_rhum        0.185345
hou                0.032039
850hPa_wind       -0.020004
1000hPa_number    -0.053197
1000hPa_azimuth   -0.059478
850hPa_number     -0.099943
850hPa_azimuth    -0.105851
```

```
1000hPa_wind      -0.118340
date              -0.181696
year              -0.190455
850hPa_omega      -0.195034
Name: 褐飞虱, dtype: float64
Selected features: ['850hPa_air', '1000hPa_air', '1000hPa_rhum', 'month',
'850hPa_rhum']
```

选择机器学习的模型进行预测

由于各个机器学习的模型的性能各不相同，我们使用不同的机器学习的模型来训练模型，并用不同的训练模型对预测值进行预测。

```
# 目标和特征值
# X = df_5[selected_features].drop(['白背飞虱', '褐飞虱'], axis=1)
X = df[selected_features] #这里需要修改！！！！！！！！ 11
y = df[target] #这里需要修改！！！！！！！！！！！

# 特征标准化
# ss = StandardScaler()
# X_std = ss.fit_transform(X)

# 分割数据集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

mes_white = []

# 随机森林回归
rf_white = RandomForestRegressor(n_estimators=150, max_depth=10, random_state=0)
rf_white.fit(X_train, y_train)

y_pred_rf = rf_white.predict(X_test)
mes_white.append(mean_squared_error(y_test, y_pred_rf))
print("Random Forest R2:", r2_score(y_test, y_pred_rf))
print("Random Forest MSE:", mean_squared_error(y_test, y_pred_rf))
print('\n')


from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

# 初始化KNN模型
knn_model_white = KNeighborsRegressor(n_neighbors=5)
knn_model_white.fit(X_train, y_train)

# 预测并评估
y_pred_knn = knn_model_white.predict(X_test)
mes_white.append(mean_squared_error(y_test, y_pred_knn))
print("KNN R2:", r2_score(y_test, y_pred_knn))
print("KNN MSE:", mean_squared_error(y_test, y_pred_knn))
```

```
print('\n')

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# 初始化线性回归模型
lr_model_white = LinearRegression()
lr_model_white.fit(X_train, y_train)

# 预测并评估
y_pred_lr = lr_model_white.predict(X_test)
mes_white.append(mean_squared_error(y_test, y_pred_lr))
print("Linear Regression R2:", r2_score(y_test, y_pred_lr))
print("Linear Regression MSE:", mean_squared_error(y_test, y_pred_lr))
print('\n')
```

示例的预测结果如下：

用随机森林对目标值为白背飞虱的性能：

```
Random Forest R2: 0.20891842099003755
Random Forest MSE: 1.7698542350437143
```

用KNN模型对目标值为白背飞虱的性能:

```
KNN R2: 0.17562113203945606
KNN MSE: 1.8443488882227372
```

用线性回归模型对目标值为白背飞虱的性能:

```
Linear Regression R2: 0.23587116854082457
Linear Regression MSE: 1.7095539630306509
```

用随机森林对目标值为褐飞虱的性能：

```
Random Forest R2: 0.22218255509526075
Random Forest MSE: 1.6248412958275005
```

用KNN对目标值为褐飞虱的性能：

```
KNN R2: 0.1297408578925927
KNN MSE: 1.81794970199043
```

用线性回归模型对目标值为褐飞虱的性能:

```
Linear Regression R2: 0.181041865058425
Linear Regression MSE: 1.7107831740260238
```

上述展示的示例中随机森林模型的性能是最好的，但是在其他的数据集的测试中，存在一些情况是KNN模型或线性回归模型的性能更好，因此我们通过一个数组来存储每个模型的性能并选择最好的模型作为后面整数预测的模型。

```
# 预测白背飞虱
X_white = df_pred[selected_features]
print(X_white.columns)

if mes_white[0] < mes_white[1] and mes_white[0] < mes_white[2]:
    white_num = rf_white.predict(X_white)
elif mes_white[1] < mes_white[2] and mes_white[1] < mes_white[0]:
    white_num = knn_model_white.predict(X_white)
elif mes_white[2] < mes_white[0] and mes_white[2] < mes_white[1]:
    white_num = lr_model_white.predict(X_white)

df_pred['白背飞虱'] = white_num

# 预测褐飞虱
X_brown = df_pred[selected_features1]
print(X_brown.columns)

if mes_white[0] < mes_white[1] and mes_white[0] < mes_white[2]:
    brown_num = rf.predict(X_brown)
elif mes_white[1] < mes_white[0] and mes_white[1] < mes_white[2]:
    brown_num = knn_model.predict(X_brown)
elif mes_white[2] < mes_white[0] and mes_white[2] < mes_white[1]:
    brown_num = lr_model.predict(X_brown)

df_pred['褐飞虱'] = brown_num
```

数据的导出

在预测完成后，我们把数据作为dataframe的性质导出到excel文件中：

```
df_pred.to_excel('pred_SSP126_Step7_固始-徽州(气象数据前移15天).xlsx', index=False)
```

代码的其他说明

因为我们有四个不同的气象数据文件，分别为不前移，前移5天，前移10天和前移15天，为了分别得到这下时间错开的气象数据和虫情数据之间的关系，我们分别对这写数据进行上述的重复操作。