队伍编号	123456
题号	A

基于物流数据的货量预测与调度优化算法的设计与实现

摘 要

摘要

关键词：Prophet 时序预测

目录

一、问题的提出

1.1 问题的背景

随着互联网经济的蓬勃发展，电商销售模式也愈发受到欢迎。电商模式中，买家无需亲自前往门店购买商品，而是通过互联网下单。商家接单后通过物流网络将商品寄送给买家。在电商的运营中，物流网络的建设，维护与优化尤为重要。物流网络的效率直接决定了商品送达客户的时间，能后显著影响客户的满意度。一个稳定的网络系统决定了电商模式的可靠性，确保电商系统能应对高峰订单量，服务更广大的商家和买家群体。

物流网络的基本元素有物流场地（接货仓，分拣中心，营业部等）和场地之间的运输线路。物流场地负责存放，分拣，收发货物，是货物运输的枢纽。物流线路连接不同场地，具有方向性和承载上限。众多站点和线路构成了庞大的物流网络，承载着巨大的货量，是电商模式的基石。如何维护和优化如此庞大的物流网络，是一项前所未有的挑战。

在物流网络的维护和优化问题中，货量预测是一个重要的方向。经过各场地和线路的货量往往与日期和电商促销时间点（“双十一”，“618”等）密切相关。如果能通过历史数据预测场地和线路的货量，将能显著降低管理成本，提高物流网络的运营效率。特别地，物流场地的临时或永久关停会对整个物流网络产生显著影响。如果能基于货量的预测数据，在物流场地关停时重新调整线路网，则能有效降低不利影响，维持物流网络的效率和稳定。进一步，如果能对每个站点和线路进行重要性评级，并通过线路动态调整来优化调度，将大大提升整个物流网络的效率，更好服务日益繁荣的电商经济

1.2 问题的提出

围绕物流货量的预测与物流网络调度优化，本文依次解决这些问题：

问题一：货量预测模型的建立与应用

物流网络的历史货量数据非常繁杂。如何建立合适的数学模型，经由历史数据训练后，预测未来指定时间段每条线路的货量。同时，给出 2023-01-01 至 2023-01-31 之间线路 DC14→DC10、DC20→DC35、DC25→DC62 的预测结果。

二、问题的分析

2.1 问题一的分析

首先需要用 Python **读取**附件中的历史货量数据并进行**预处理**，以便后续的模型训练与预测。其次需要选择符合条件的模型进行训练和预测。本题中货量为**因变量**，日期为**自变量**，两者存在**强相关性**。其次，日期具有以周，年为单位的**周期变动**，并且**特殊节日**（“618”，“双十一”等）对物流量的影响非常显著。因此，我们选择了 **Prophet 时序预测工具库**进行预测，使用预处理好的数据对模型进行**拟合和训练**。最终，使用训练好的模型进行**预测**，得到 2023-01 时间段相关线路的预测结构。

2.2 问题二的分析

2.3 问题三的分析

三、模型的假设

假设 1：题目给出的历史货量数据真实可靠，不存在采集的人工误差；
假设 2：2022 年的数据相比 2021 年受疫情影响较小，近似为正常数据；
假设 3：节日窗口期假定为前 5 天到后 10 天；
假设 4：受周期性供应链调整，运货量应有小周期性。

四、符号说明

符号	符号说明
t	时间变量
$y(t)$	Prophet 模型中的预测结果项
$g(t)$	Prophet 模型中的趋势项
$s(t)$	Prophet 模型中的季节项
$h(t)$	Prophet 模型中的节假日项
$\epsilon_{(t)}$	Prophet 模型中的误差项

五、线路货量预测模型的建立与应用

5.1 问题描述

问题一要求基于历史物流数据建立线路货量预测模型，预测 2023-01 期间每条线路的货量吗，并给出线路 DC14→DC10、DC20→DC35、DC25→DC62 的预测结果。

5.2 数据预处理

题目给出的数据集中，一个数据项包含 4 个元素：日期，起点，终点（共 81 个站点）和货量。我们先使用 Python 中的 pandas 模块读取数据。再使用 numpy 模块建立 $365 \times 81 \times 81$ 的 tensor 单元, 三个维度分别为日期，起点，终点，存放货量数据。考虑到货量变化的最大周期为 1 年，且 2021 年和 2022 年受疫情的影响显著不同，我们建立了两个上述的 tensor 单元来分别存储这两年的数据，以便后续的模型训练和预测。

5.3 模型的选择

首先，我们认为货量数据的特征为（1）强时间相关性；（2）具有年，月，周等不同周期；（3）受特定节日（“618”，“双十一”等）影响大。为了验证上述观点，我们绘制了所有线路每天总货量随日期变化的图像。如下图所示：

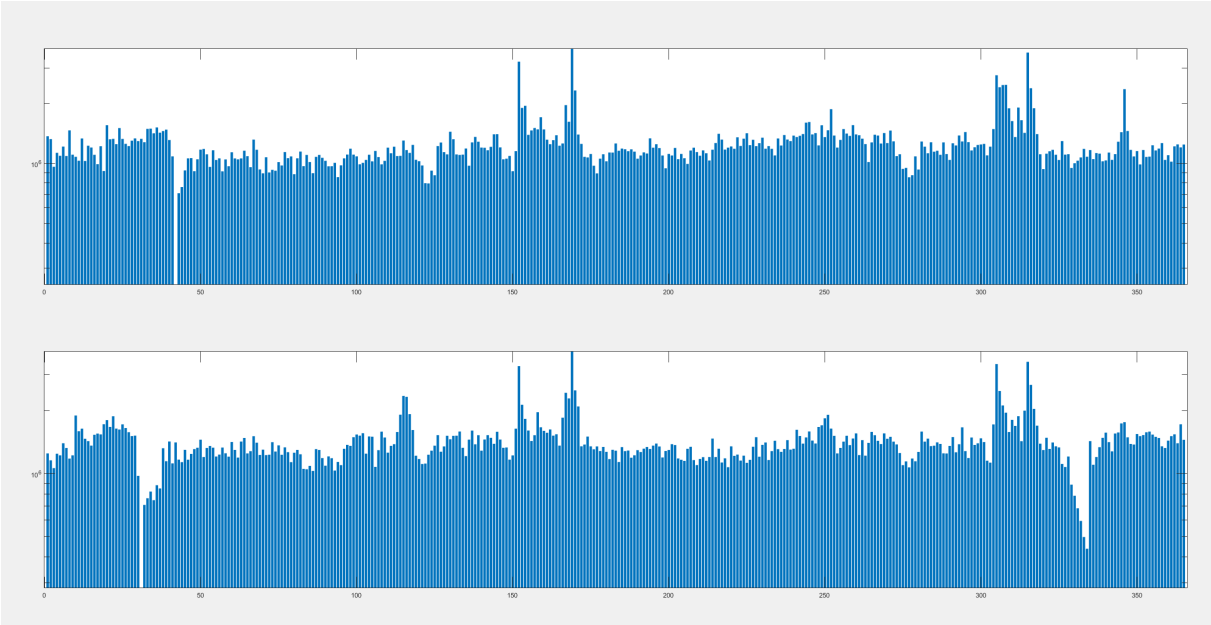

图 1　2021,2022 两年物流网络每日总货量随时间变化图

可以看出，两年内物流网络运输总量随时间的的变化趋势基本一致，且在“618”，“双十一”等时间节点有明显的大爆发。由此，我们明确了我们选择模型的要求：（1）能处理变化具有强周期性的数据；（2）需针对时间序列优化；（3）能把节日作为特殊因素考虑。

Prophet 是 Facebook 开源的时间序列预测算法，可以**有效处理节假日信息**，并**按周、月、年对时间序列数据的变化趋势进行拟合**。其算法模型如下：

$$y(t) = g(t) + s(t) + h(t) + \epsilon_{(t)}$$

其中：

- 趋势项 $g(t)$ 表示时间序列在非周期上面的变化趋势；
- 季节项 $s(t)$ 一般来说是以周或者年为单位；
- 节假日项 $h(t)$ 表示时间序列中那些潜在的具有非固定周期的节假日对预测值造成的影响；
- 误差项 $\epsilon(t)$ 表示模型未预测到的波动，服从高斯分布；

Prophet 算法就是通过拟合这几项，然后相加，得到时间序列的预测值。

参考文献

附 录

附录 1： 问题一的 Python 代码

```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt
from data_proc import new

'''
This program uses the Prophet algorithm (which forecasts time-dependent sequences)
    to predict the freight volume of a specific route in 2023.

Algorithm principle:
   y(t) = g(t) + s(t) + h(t) + epsilon(t)

y(t): outcome
g(t): growth trend
s(t): seasonality trend
h(t): holidays effects
epsilon(t): errors

Noted that the data from 2021 was significantly affected by the pandemic, making
   it less representative and resulting in larger error terms.
Using data from 2021 and combined data of 2021 and 2022 for prediction can easily
   lead to unrealistic results,
as the model may learn incorrect patterns.
It is better to use the prediction results based on 2022 data alone.
'''

#remove outliers
def outlier_filter(df):
    q1 = df['y'].quantile(0.25)
    q3 = df['y'].quantile(0.75)
    IQR = q3 - q1
    lower_bound = q1 - 1.5 * IQR
    upper_bound = q3 + 1.5 * IQR
    return df[(df['y'] >= lower_bound) & (df['y'] <= upper_bound)]

#generate legal data format for prediction
def data_selection(sfrom, sto, year):
    #Select and format data for specific route and year
    specific_routes = new[
        (new["site_from"] == sfrom) &
```

```
        (new["site_to"] == sto) &
        (new["date_code"] // 1000 == year)
    ]

    #process data format
    specific_routes = specific_routes.drop(["site_from", "site_to"], axis = 1)
    specific_routes["date_code"] = pd.to_datetime(
        specific_routes["date_code"].astype(str),
        format = '%Y%j'
    )
    specific_routes.rename(
        columns = {"date_code": "ds", "shipment_volume": "y"},
        inplace = True
    )

    return specific_routes

#initialization
predictions = {
    "DC14-DC10": {2021: {}, 2022: {}, "Both Years": {}},
    "DC20-DC35": {2021: {}, 2022: {}, "Both Years": {}},
    "DC25-DC62": {2021: {}, 2022: {}, "Both Years": {}},
}

#train model and make predictions
for route in predictions:
    sites = route.split('-')
    sfrom, sto = sites[0], sites[1]

    for year_key in predictions[route]:
        if year_key != "Both Years":
            data = data_selection(sfrom, sto, year_key)
        else:
            df1 = data_selection(sfrom, sto, 2021)
            df2 = data_selection(sfrom, sto, 2022)
            data = pd.concat([df1, df2], ignore_index = True)

        #remove outlier
        data = outlier_filter(data)

        #set limit
        historical_peak = data['y'].quantile(0.85)
        cap_value = historical_peak * 1.8 if historical_peak > 0 else 1
        data["cap"] = cap_value
```

```
data["floor"] = data['y'].min() * 0.7

#model construction
model = Prophet(
    growth = "logistic",
    seasonality_mode = "multiplicative",
    changepoint_prior_scale = 0.002,
    seasonality_prior_scale = 25,
    yearly_seasonality = 18,
    weekly_seasonality = 2,
    daily_seasonality = False,
    holidays_prior_scale = 18,
    mcmc_samples = 0
)

#traditional holidays and custom holidays
model.add_country_holidays(country_name = "CN")
promotion_dates = [
    "2021-06-18", "2021-11-11", "2021-12-12",
    "2022-06-18", "2022-11-11", "2022-12-12",
    "2023-06-18", "2023-11-11", "2023-12-12",
]
custom_holidays = pd.DataFrame({
    "holiday": 'promotion',
    "ds": pd.to_datetime(promotion_dates),
    "lower_window": -5,
    "upper_window": 10
})
model.holidays = pd.concat([model.holidays, custom_holidays])

#strengthen seasonality
model.add_seasonality(
    name='quarter',
    period = 91.25,
    fourier_order = 10,
    prior_scale = 15
)
model.fit(data)

#predictions generation
future = model.make_future_dataframe(periods = 365)
future["cap"] = cap_value
future["floor"] = data['floor'].min()
forecast = model.predict(future)
```

```
        min_volume = data['y'].min() * 0.5
        forecast[["yhat", "yhat_lower", "yhat_upper"]] = forecast[
            ["yhat", "yhat_lower", "yhat_upper"]
        ].clip(lower=min_volume)

        #store results
        predictions[route][year_key] = {
            "model": model,
            "forecast": forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail
                (365)
        }

'''
Chart explanation:
Black dots: Observed values (actual values)
Light blue shaded area: Uncertainly interval
Dark blue line: Predicted/Forecast values (median value of uncertainly interval)
'''

if __name__ == "__main__":
    #result visualization
    for route in predictions:
        for year_key in predictions[route]:
            #escape empty records
            if not predictions[route][year_key]:
                continue

            #get results
            model = predictions[route][year_key]["model"]
            forecast = predictions[route][year_key]["forecast"]

            #generate graphs
            fig = model.plot(forecast)

            current_ax = plt.gca()

            forecast_start = forecast['ds'].max() - pd.DateOffset(days = 365)
            current_ax.axvline(x = forecast_start, color = 'purple',
                               linestyle = '--', alpha = 0.7, label = 'Forecast Start')

            current_ax.axhline(y = data['floor'].min(), color = 'grey',
                               linestyle = ':', alpha = 0.5, label = 'Minimum Limit')

            plt.title(f"{route} Volume Prediction ({year_key} Basis)")
```

```
        plt.xlabel("Date")
        plt.ylabel("Shipment Volume")
        plt.legend()
        plt.show()
```

附录 2：问题一的 MATLAB 代码

```
clear;
tiledlayout(2,2);
nexttile([1,2]);
T=readtable("附件1：物流网络历史货量数据.xlsx","VariableNamingRule","preserve");
days=[31 28 31 30 31 30 31 31 30 31 30 31];
total=[];
for i=1:12
    for j=1:days(i)
        if j<=9
            index=find(T.("日期")==sprintf("2021-0%d-0%d",i,j));
        else
            index=find(T.("日期")==sprintf("2021-0%d-%d",i,j));
        end
        total=[total,sum(T.("货量")(index))];
    end
end
bar(total);
set(gca,"YScale","log");
nexttile([1,2]);
total=[];
for i=1:12
    for j=1:days(i)
        if j<=9
            index=find(T.("日期")==sprintf("2022-0%d-0%d",i,j));
        else
            index=find(T.("日期")==sprintf("2022-0%d-%d",i,j));
        end
        total=[total,sum(T.("货量")(index))];
    end
end
bar(total);
set(gca,"YScale","log");
```