队伍编号	MC2406053
题号	C

论文标题

摘要

摘要内容

关键词: 随机森林算法

目录

1 问题重述

1.1 问题背景

在电商物流网络中，订单履约的过程包括多个环节，其中核心环节之一是分拣。分拣中心负责根据不同的目的地对包裹进行分类，并将它们发送到下一个目的地，最终交付给顾客。因此，提高分拣中心的管理效率对整个网络的订单履约效率和成本控制至关重要。

货量预测在电商物流网络中扮演着至关重要的角色。准确预测分拣中心的货物量是后续管理和决策的基础，有助于合理安排资源。通常，货量预测是根据历史货物量、物流网络配置等信息，来预测每个分拣中心每天和每小时的货物量。

分拣中心的货量预测与网络的运输线路密切相关。通过分析各线路的运输货物量，可以确定各分拣中心之间的网络连接关系。当线路关系发生变化时，可以根据调整信息来提高对各分拣中心货量的准确预测。

基于货量预测的人员排班是下一步需要解决的重要问题。分拣中心的人员包括正式员工和临时工两种类型。合理安排人员旨在完成工作的前提下尽可能降低人力成本。根据物流网络的情况，制定了人员安排的班次和小时人效指标。在确定人员安排时，优先考虑使用正式员工，必要时再增加临时工。

1.2 问题描述

1.2.1 问题一

建立货量预测模型，对 57 个分拣中心未来 30 天每天及每小时的货量进行预测。

1.3 总体思路分析

2 符号说明与基本假设

2.1 符号说明

表 1: 符号说明

符号	说明

2.2 基本假设

3 问题一的建模与求解

问题一给出了分拣中心在过去的四个月内每天的货量以及过去 30 天内每个小时的货量。在这些数据的基础上要对下一个月（30 天）的货量进行预测，我们先对给出的数据进行观察。我们选取，例如 SC

附录

A 附录 1

```python
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.axes import Axes
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from matplotlib.backends.backend_pdf import PdfPages
from matplotlib.animation import FuncAnimation
from scipy.integrate import odeint
from scipy import linalg as la
from scipy import optimize
import scipy

config = {
    "text.usetex": True,
    "text.latex.preamble": r"\usepackage{CJK}",  # 预先导入 CJK 宏包处理中文
}
plt.rcParams.update(config)
```

随机森林调参代码

```python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler

# 假设存在的 SCid 列表
data_for_sc = pd.read_csv("../../附件/附件1.csv", encoding="GB2312")
ALL_SC = list(set(data_for_sc["分拣中心"]))
existing_scs = list(map(lambda SC_: int(SC_[2:]), ALL_SC))
existing_scs.sort()


# 加载数据
def load_data(existing_scs):
    all_data = []
    for sc_id in existing_scs:
        try:
            data = pd.read_csv(f"SC{sc_id}.csv")
            data["center_id"] = sc_id
            all_data.append(data)
        except FileNotFoundError:
            print(f"File for center {sc_id} not found.")
```

```
    return pd.concat(all_data, ignore_index=True) if all_data else pd.DataFrame()


# 数据清洗和预处理
def preprocess_data(data):
    data["date"] = pd.to_datetime(data["date"], errors="coerce")
    data.dropna(subset=["date", "value"], inplace=True)

    data["year"] = data["date"].dt.year
    data["month"] = data["date"].dt.month
    data["day"] = data["date"].dt.day
    data["weekday"] = data["date"].dt.weekday

    scaler = StandardScaler()
    data[["year", "month", "day", "weekday"]] = scaler.fit_transform(
        data[["year", "month", "day", "weekday"]]
    )

    return data


# 模型训练和参数调整
def train_and_optimize_model(X_train, y_train):
    param_grid = {
        "n_estimators": [100 * i for i in range(1, 10)],
        "max_depth": [10 * i for i in range(1, 10)],
        "min_samples_split": [10 * i for i in range(1, 10)],
    }
    model = RandomForestRegressor(random_state=42)
    grid_search = GridSearchCV(
        model, param_grid, cv=5, scoring="neg_mean_squared_error", verbose=2
    )
    grid_search.fit(X_train, y_train)

    print("Best parameters:", grid_search.best_params_)
    return grid_search.best_estimator_


if __name__ == "__main__":
    data = load_data(existing_scs)
    if not data.empty:
        data = preprocess_data(data)
        features = data[["center_id", "year", "month", "day", "weekday"]]
        target = data["value"]

        X_train, X_test, y_train, y_test = train_test_split(
            features, target, test_size=0.2, random_state=42
        )
```

```
73
74          best_model = train_and_optimize_model(X_train, y_train)
75
76          y_pred = best_model.predict(X_test)
77          mse = mean_squared_error(y_test, y_pred)
78          print(f"Test MSE: {mse}")
79      else:
80          print("No data loaded, please check the data files.")
```

随机森林训练代码

```
1  import pandas as pd
2  import numpy as np
3  from sklearn.ensemble import RandomForestRegressor
4  from sklearn.model_selection import train_test_split
5  from sklearn.metrics import mean_squared_error
6
7  # 假设数据已经按照分拣中心编号和日期排列好
8  # 每个文件名格式为'SCi.csv'，其中 i 是分拣中心编号
9  # 假设存在的 SCid 列表
10 data_for_sc = pd.read_csv("../../附件/附件1.csv", encoding="GB2312")
11 ALL_SC = list(set(data_for_sc["分拣中心"]))
12 existing_scs = list(map(lambda SC_: int(SC_[2:]), ALL_SC))
13 existing_scs.sort()
14
15 # 加载数据
16 def load_data(existing_scs):
17     all_data = []
18     for i in existing_scs:
19         data = pd.read_csv(f"SC{i}.csv")
20         data["SC_id"] = i
21         all_data.append(data)
22     return pd.concat(all_data, ignore_index=True)
23
24
25 # 数据预处理
26 def preprocess(data):
27     # 假设数据包含日期和货量两列，日期列名为'date'，货量列名为'value'
28     data["date"] = pd.to_datetime(data["date"])
29     data["year"] = (pd.to_datetime(data["date"])).dt.year
30     data["month"] = (pd.to_datetime(data["date"])).dt.month
31     data["day"] = (pd.to_datetime(data["date"])).dt.day
32     data["weekday"] = (pd.to_datetime(data["date"])).dt.weekday
33     return data
34
35
36 # 训练模型
37 def train_model(data):
38     features = data[["SC_id", "year", "month", "day", "weekday"]]
```

```
    target = data["value"]
    X_train, X_test, y_train, y_test = train_test_split(
        features,
        target,
        test_size=0.2,
        random_state=50,

    )

    model = RandomForestRegressor(n_estimators=300, random_state=42, min_samples_split=20)
    model.fit(X_train, y_train)

    # 预测和评估
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    print(f"Mean Squared Error: {mse}")
    return model


# 预测未来的货量
def predict_future(model, start_date, num_days, existing_scs):
    future_dates = pd.date_range(start_date, periods=num_days)
    future_data = pd.DataFrame(
        {
            "date": np.repeat(future_dates, len(existing_scs)),
            "SC_id": np.tile(existing_scs, num_days),
        }
    )
    future_data = preprocess(future_data)
    features = future_data[["SC_id", "year", "month", "day", "weekday"]]
    predictions = model.predict(features)
    future_data["predicted_volume"] = predictions
    return future_data


# 保存结果到 CSV
def save_predictions_to_csv(predictions, file_name):
    predictions["date"] = predictions["date"].dt.strftime("%Y/%m/%d")
    predictions.to_csv(file_name, index=False)
    print(f"Saved predictions to {file_name}")


# 主函数
def main():
    data = load_data(existing_scs)
    data = preprocess(data)
    model = train_model(data)
    future_predictions = predict_future(model, "2023-08-01", 153, existing_scs)
```

```
87      save_predictions_to_csv(future_predictions, "predicted_volumes.csv")
88
89
90  if __name__ == "__main__":
91      main()
```