1. 최적 물류센터 위치

개요

1. 문제 정의
    - 물류센터의 위치는 물류 운영의 효율성과 비용 절감에 매우 중요한 요소입니다. 현재의 물류센터 위치가 차량의 이동 경로와 효율적으로 연계되지 않으면 불필요한 비용이 발생하고, 물류 이동 시간이 길어질 수 있습니다. 이 보고서는 GPS 데이터를 기반으로 최적의 물류센터 위치를 추천하고, 이를 통해 물류 이동 비용을 절감하고 효율성을 높이는 것을 목표로 합니다.
2. 목표 설정
    - 물류 이동 비용 절감: 최적의 물류센터 위치를 찾아 물류 이동 경로를 최적화하고, 물류 이동에 소요되는 비용을 최소화합니다.
    - 효율성 증대: 물류센터와 배송 차량 간의 거리를 최소화하여 배송 효율성을 높입니다.

데이터 설명

1. GPS 데이터 소개
    - 데이터 포맷: GPS 데이터는 CSV 형식으로 제공되었으며, 각 행은 특정 시간대에 기록된 차량의 위치 좌표(위도 및 경도), 이동 속도, 물류센터 위치 등을 포함하고 있습니다.
    - 수집 주기: 데이터는 특정 시간 간격으로 수집되었습니다. 이 데이터를 통해 시간대별 및 요일별로 차량의 이동 경로를 분석할 수 있습니다.
    - 시간대별 및 요일별 분석 가능성: 데이터에는 시간이 포함되어 있어, 이를 바탕으로 특정 시간대나 요일별로 차량 이동 패턴을 분석할 수 있습니다.
2. 물류센터 좌표 데이터 설명
    - 물류센터 좌표: 데이터에는 현재 물류센터의 좌표가 포함되어 있으며, 이를 바탕으로 현재 위치의 효율성을 평가할 수 있습니다.

3. 외부 데이터 설명
    - OpenStreetMap 도로 네트워크 데이터: 외부 데이터로 OpenStreetMap(OSM) 도로 네트워크 데이터를 사용했습니다. 이 데이터를 통해 각 클러스터 중심점 주변의 도로망을 분석하고, 접근성을 평가하여 최적의 물류센터 위치를 도출했습니다.

데이터 전처리

1. 결측치 처리
    - GPS 데이터에서 결측값이 있는 열을 확인하고, 필요에 따라 해당 열을 제거하거나 결측값을 처리하였습니다. 예를 들어, Unnamed 열들은 의미 없는 데이터로 판명되어 삭제되었습니다.
2. 이상치 제거
    - 이상치 정의 및 처리: 이동 좌표와 속도 열에서 이상치를 정의하고 제거하는 과정을 거쳤습니다. 사분위수를 기반으로 한 IQR(Interquartile Range) 방법을 사용해 이상치를 처리하였습니다.

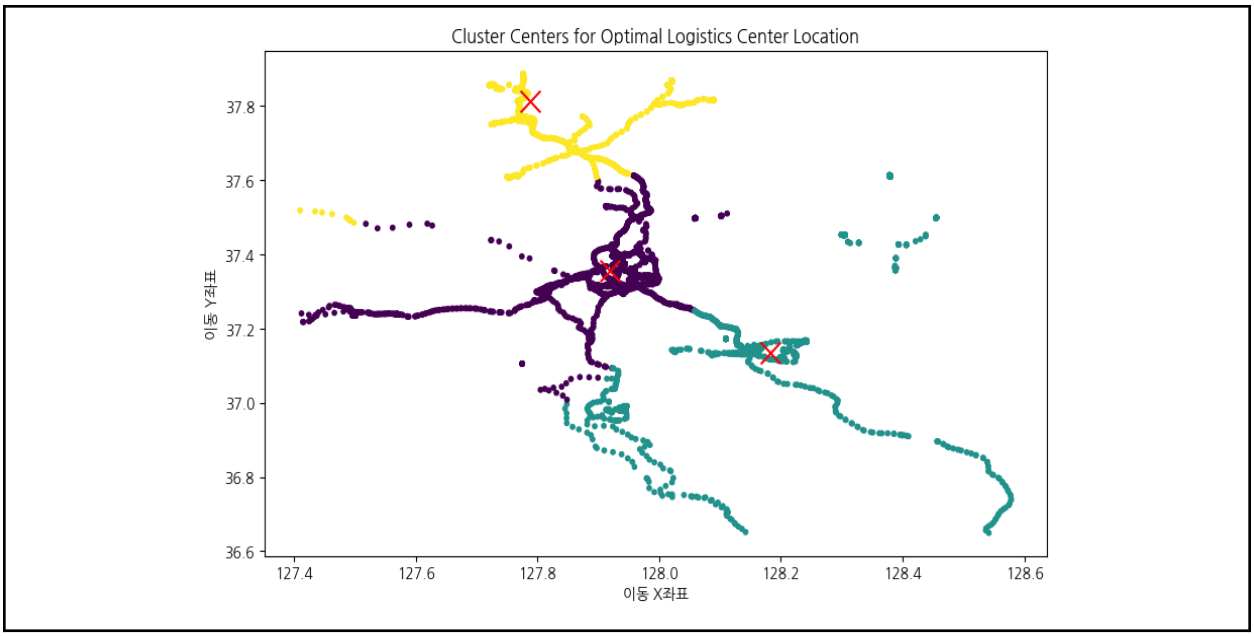
```
# Unnamed 열 제거 (필요 없는 열)
df_cleaned = df.dropna(axis=1, how='all')

# 이상치 제거를 위한 사분위수 계산
Q1_x = df_cleaned['이동 X좌표'].quantile(0.25)
Q3_x = df_cleaned['이동 X좌표'].quantile(0.75)
IQR_x = Q3_x - Q1_x
lower_bound_x = Q1_x - 1.5 * IQR_x
upper_bound_x = Q3_x + 1.5 * IQR_x

Q1_y = df_cleaned['이동 Y좌표'].quantile(0.25)
Q3_y = df_cleaned['이동 Y좌표'].quantile(0.75)
IQR_y = Q3_y - Q1_y
lower_bound_y = Q1_y - 1.5 * IQR_y
upper_bound_y = Q3_y + 1.5 * IQR_y

# 이상치 제거
df_filtered = df_cleaned[(df_cleaned['이동 X좌표'] >= lower_bound_x) &
                         (df_cleaned['이동 X좌표'] <= upper_bound_x) &
                         (df_cleaned['이동 Y좌표'] >= lower_bound_y) &
                         (df_cleaned['이동 Y좌표'] <= upper_bound_y)]
```

3. GPS 데이터에서 핵심 정보 추출
    - 차량 이동 경로 분석: 차량의 이동 좌표 데이터를 바탕으로 이동 경로를 분석하였습니다. 특정 경로를 자주 오가는 차량의 패턴을 파악할 수 있었습니다.
    - 속도 분석: 이동 속도 데이터를 활용해 물류 이동의 효율성을 평가하고, 특정 시간대에 이동 속도가 급격히 줄어드는 구간을 분석하였습니다.
    - 시간대별 분석: 시간대별로 차량 이동 데이터를 분류하여, 특정 시간대에 차량이 집중되는 구간을 파악하고 분석했습니다.

분석 방법론

1. 물류 이동 경로 및 비용 분석
    - GPS 데이터를 통해 차량의 이동 경로를 시각화하고, 이동 경로의 길이를 계산하여 물류 이동에 소요되는 비용을 분석하였습니다.
2. 거리 기반 분석: 중심점 탐색
    - K-Means 클러스터링: K-Means 알고리즘을 사용하여 차량이 자주 이동하는 위치를 클러스터링하고, 각 클러스터의 중심점을 도출하였습니다.
    - 엘보우 메소드: 클러스터 수를 결정하기 위해 엘보우 메소드를 사용해 최적의 클러스터 수를 자동으로 선택하였습니다.
3. 외부 데이터 결합을 통한 교통 혼잡도 및 시간 절약 고려
    - OpenStreetMap 도로 네트워크 분석: 각 클러스터 중심점 주변의 도로 네트워크를 OpenStreetMap 데이터를 통해 분석하고, 접근성을 평가하여 최적의 물류센터 위치를 도출했습니다.
4. 최적화 알고리즘 적용
    - K-Means 알고리즘: K-Means 클러스터링을 사용해 차량 이동 데이터를 클러스터링하였고, 각 클러스터의 중심점을 최적 물류센터 후보지로 설정하였습니다.
    - 접근성 평가: 각 클러스터 중심점의 도로 네트워크를 분석하여 접근성을 평가하고, 이를 기반으로 최적의 물류센터 위치를 선정했습니다.

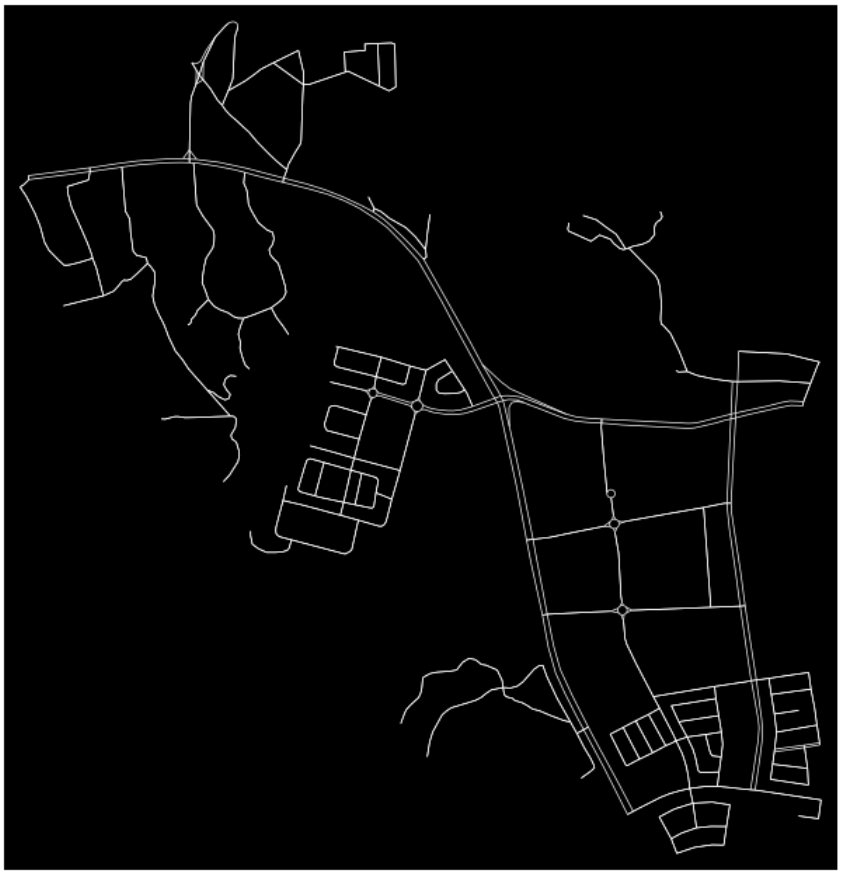
```
Cluster 1 Center: X좌표 = 127.91937420140448, Y좌표 = 37.35547730372297
Cluster 2 Center: X좌표 = 128.18263091269648, Y좌표 = 37.134549694830596
Cluster 3 Center: X좌표 = 127.7880656372177, Y좌표 = 37.8119615313008
```



최적 위치 선정 과정

- 도로 접근성 평가: OpenStreetMap 데이터를 활용하여 각 후보지의 도로 접근성을 분석하였습니다. 평균 이동 거리를 기준으로 접근성이 가장 좋은 클러스터 중심점을 최적 물류 센터 위치로 선정했습니다.

```
# 접근성 평가: 각 클러스터 중심점에서 주요 도로 노드까지의 평균 거리를 계산해 접근성 평가
accessibility_scores = []
for i, G in enumerate(road_networks):
    try:
        center_node = ox.distance.nearest_nodes(G, cluster_centers[i][0],
cluster_centers[i][1])

        # networkx의 shortest_path_length 사용
        lengths = nx.single_source_dijkstra_path_length(G, center_node,
weight='length')

        # 평균 거리를 접근성 지표로 사용
        mean_distance = np.mean(list(lengths.values()))
        accessibility_scores.append(mean_distance)
    except Exception as e:
        print(f"접근성 평가 오류: {e}. 클러스터 {i+1}의 접근성 평가를 건너뜁니다.")
        accessibility_scores.append(float('inf'))  # 무한대 값으로 접근성 평가
에서 제외
```

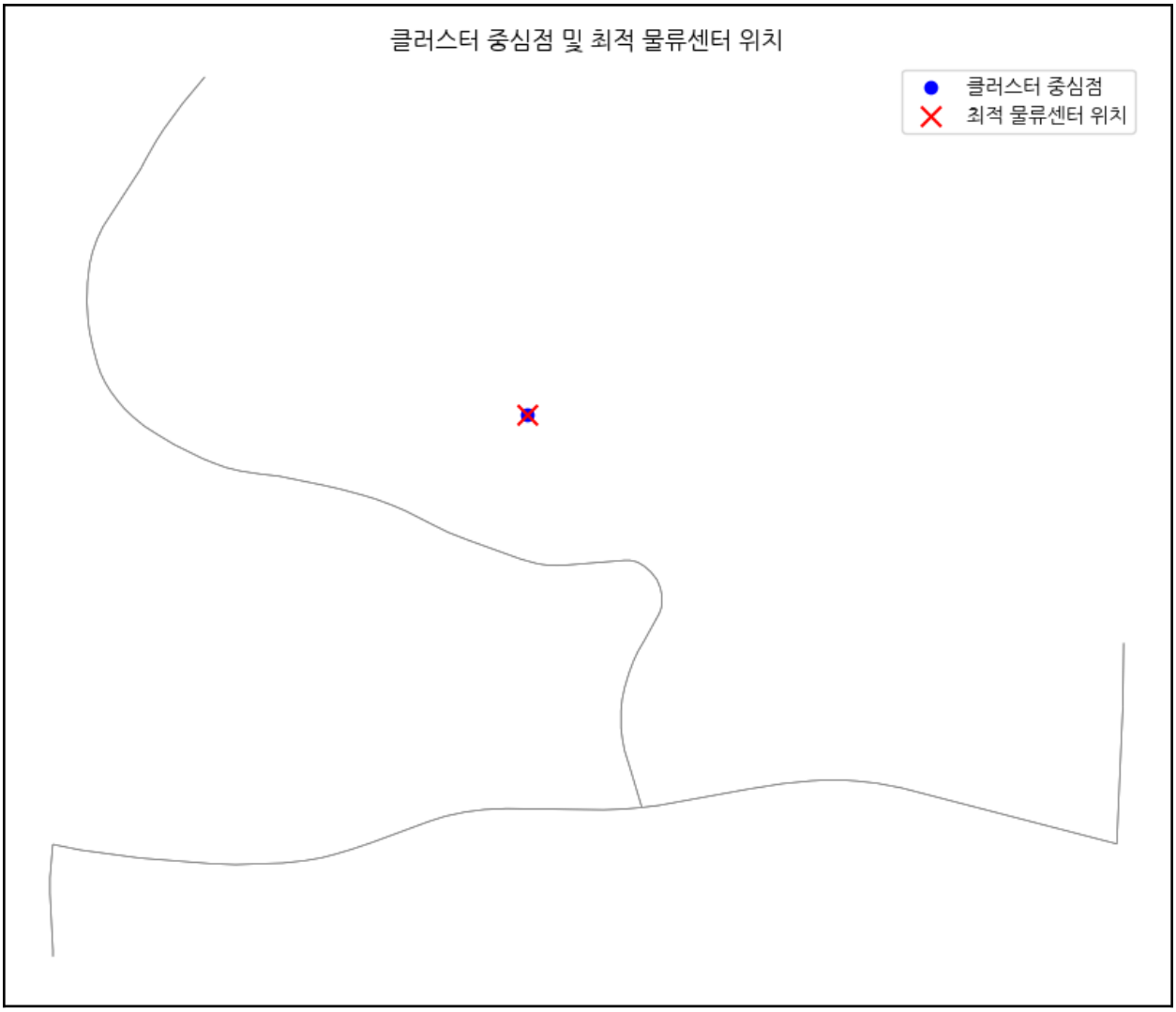


결론

1. 최적 위치 선정 결과
    - 최적 물류센터 위치: 최적의 물류센터 위치는 클러스터 중심점 중 접근성이 가장 우수한 클러스터 2의 중심점으로 선정되었습니다.
    - 최적 위치 좌표: 최적 위치의 좌표는
      X좌표 = 128.18263091269648,
      Y좌표 = 37.1345496948305961입니다.

```
# 접근성이 가장 좋은 클러스터 중심점을 최적 위치로 선택
if accessibility_scores:
    best_cluster_idx = np.argmin(accessibility_scores)
    best_location = cluster_centers[best_cluster_idx]

    print(f"최적의 물류센터 위치는 클러스터 {best_cluster_idx+1}의 중심점입니다.")
    print(f"최적 위치 좌표: X좌표 = {best_location[0]}, Y좌표 =
{best_location[1]}")

    # 최적 위치의 도로망 시각화
```

```
최적의 물류센터 위치는 클러스터 2의 중심점입니다.
최적 위치 좌표: X좌표 = 128.18263091269648, Y좌표 = 37.134549694830596
```



2. 향후 과제 및 개선 방안

- 데이터의 시간대별/요일별 세분화 분석: 차량의 이동 패턴을 더 정밀하게 분석하기 위해 시간대 및 요일별로 데이터를 세분화하여 분석할 필요가 있습니다.
- 교통 혼잡도 데이터 반영: 현재 도로 네트워크만을 고려한 접근성 평가에서 나아가, 교통 혼잡도 데이터를 결합하여 보다 정밀한 최적 물류센터 위치 분석이 필요합니다.
- 환경적 요인 반영: 물류센터 위치를 선정할 때, 물류 수요뿐만 아니라 환경적 요인(예: 날씨, 자연재해 가능성)도 함께 고려하여 분석할 수 있는 추가적인 데이터 분석이 필요합니다.

2. 수요예측 모델

개요

1. 문제 정의
    - 물류 및 재고 관리의 핵심은 적절한 수요 예측입니다. 수요 예측이 잘못되면 재고가 과잉되거나 부족할 수 있으며, 이는 물류 비용 증가 또는 판매 손실로 이어집니다. 본 프로젝트의 목표는 각 업체별로 적절한 수요 예측 모델을 개발하여 물류 및 재고 관리를 최적화하는 것입니다. 특히, 단기적인 수요 예측에 중점을 두어 비용 절감과 운영 효율성을 높이는 것이 주된 목표입니다.
2. 목표 설정
    - 각 업체별 단기(일 단위, 주 단위) 수요를 예측하여 물류와 재고 관리의 효율성을 극대화합니다.
    - Prophet 및 pmdarima를 사용하여 경량화된 시스템을 구축하며, 이를 통해 실시간 운영에 적합한 성능을 달성합니다.

데이터 설명

1. 데이터 개요
    - 사용된 데이터는 물류 시계열 데이터로, 각 업체별 입고량, 출고량, 재고량 등이 포함되어 있습니다. 데이터는 '날짜', '코드'(업체 구분), '수량' 등의 필드를 포함하고 있으며, 각 필드는 다음과 같은 역할을 합니다:

        - 날짜: 일 단위의 시계열 데이터를 기록.

        - 코드: 업체별 구분 코드.

        - 수량: 입고량 또는 출고량을 의미하며, 예측할 목표 변수입니다.

2. 외부 데이터 활용 가능성
    - 이번 프로젝트에서는 외부 데이터를 활용하지 않았지만, 향후 날씨 데이터, 경제 지표, 또는 경쟁사의 시장 상황 등을 추가하여 예측 성능을 더욱 향상시킬 수 있습니다. 이러한 외부 데이터는 특히 경제 변동성이나 특정 이벤트(예: 세일 기간)에 대한 수요 변동성을 더 정확히 예측할 수 있도록 도와줍니다.

데이터 전처리

1. 결측치 처리
    - 결측치는 시계열 데이터에서 자주 발생할 수 있습니다. 그러나 본 데이터셋에서는 결측치가 발견되지 않아 별도의 결측치 처리가 필요하지 않았습니다. 향후 결측치가 발생할 경우, 선형 보간법 또는 이동 평균을 활용한 보간법을 통해 처리할 수 있습니다.
2. 이상치 제거
    - 이상치는 모델의 성능에 영향을 미치는 중요한 요소입니다. 본 프로젝트에서는 IQR 방법과 Z-Score 방법을 사용하여 이상치를 제거하였습니다.

        IQR 방법

```
Q1 = grouped_data['수량'].quantile(0.25)
Q3 = grouped_data['수량'].quantile(0.75)
IQR = Q3 - Q1
grouped_data = grouped_data[(grouped_data['수량'] >= (Q1 - 1.5 *
IQR)) & (grouped_data['수량'] <= (Q3 + 1.5 * IQR))]
```

        Z-Score 방법

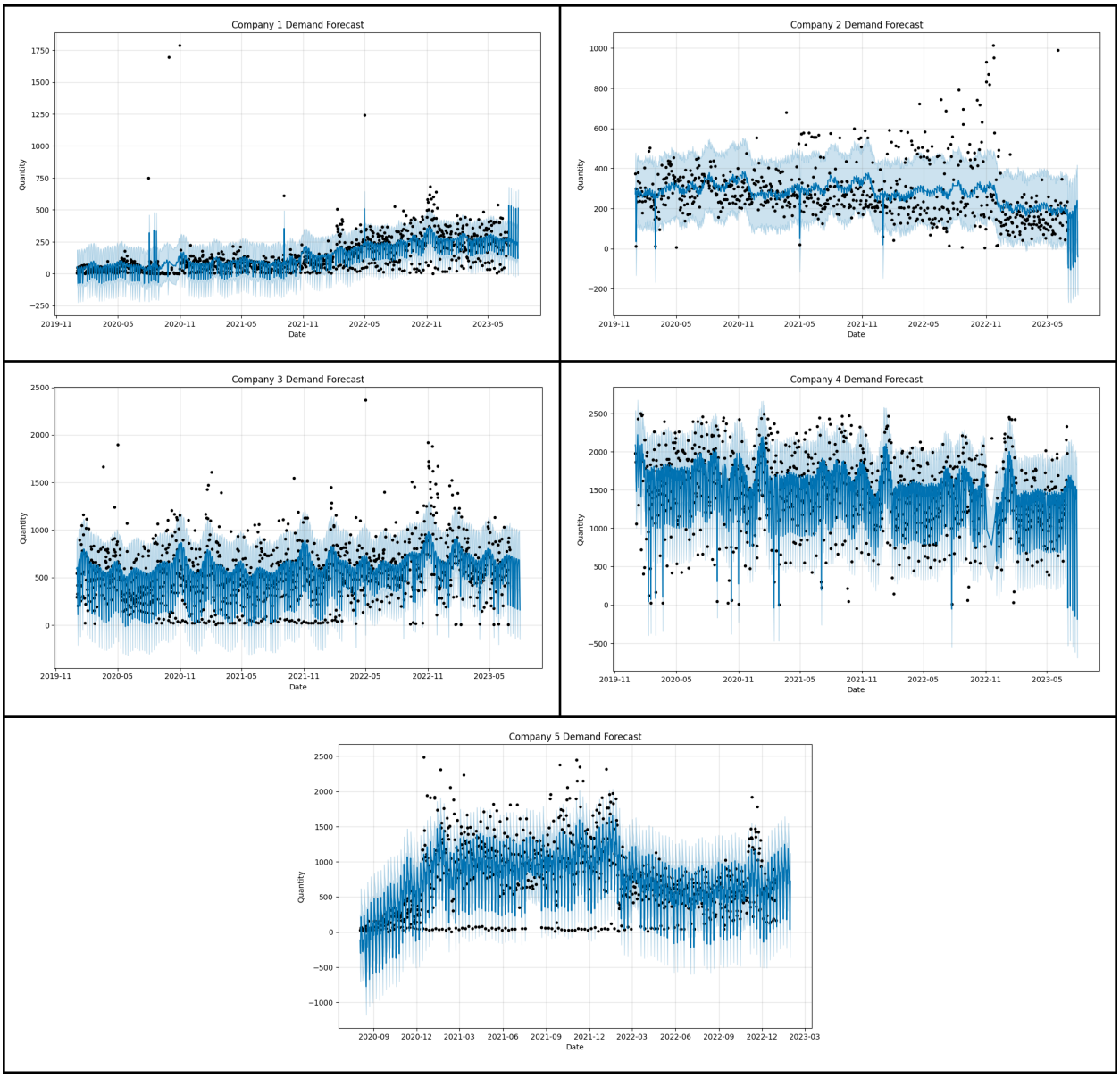
```
from scipy.stats import zscore
grouped_data['zscore'] = zscore(grouped_data['수량'])
grouped_data_clean = grouped_data[(grouped_data['zscore'].abs() <=
3)]
grouped_data_clean = grouped_data_clean.drop('zscore', axis=1)
```

예측 모델 개발

본 섹션에서는 Prophet과 ARIMA 모델을 활용하여 예측을 수행하였습니다. 각각의 모델은

데이터의 시계열 특성을 반영하여 최적의 예측 결과를 도출하는 데 중점을 두었습니다.

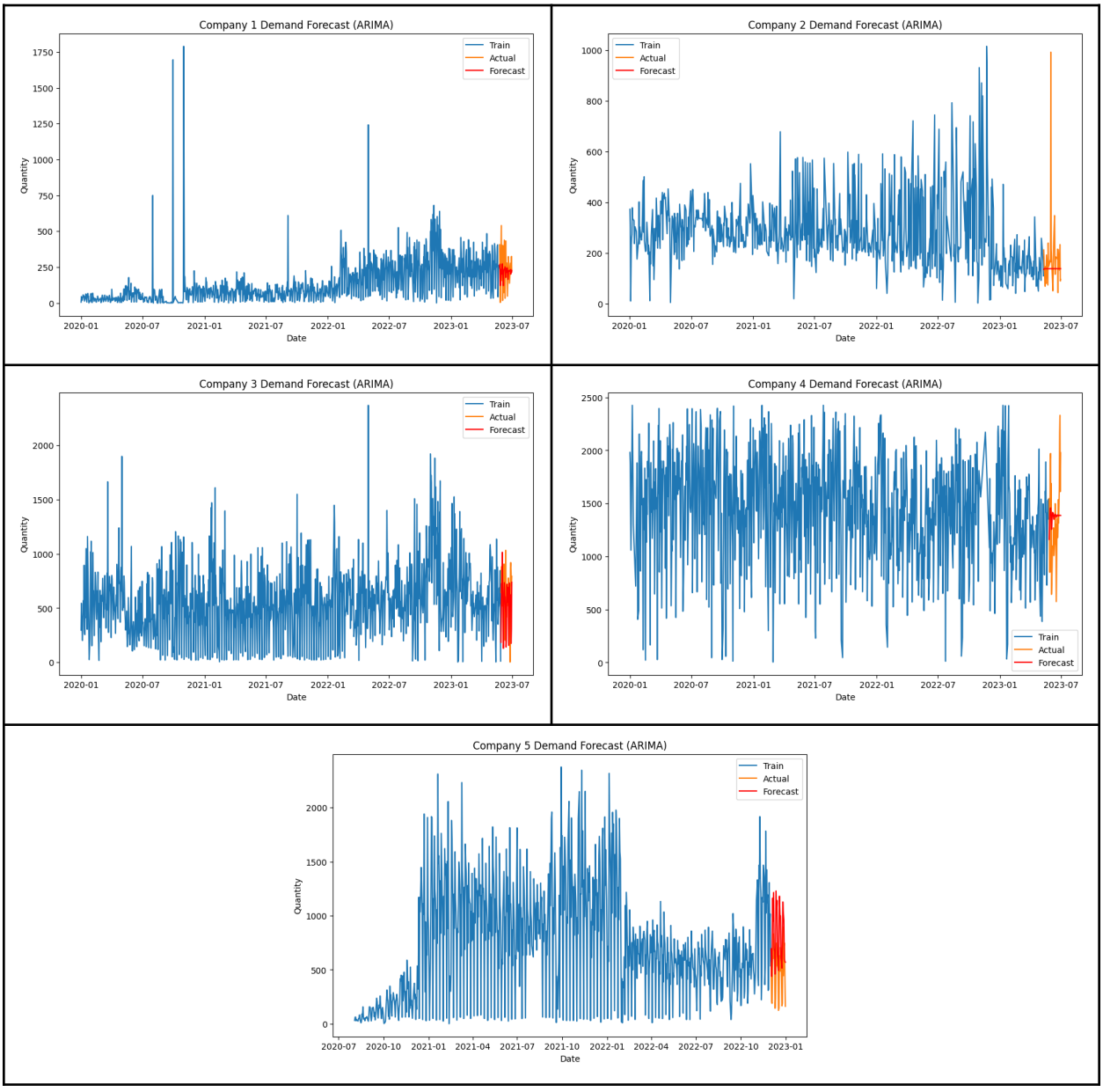
1. Prophet 모델
    - Prophet은 페이스북에서 개발한 시계열 예측 모델로, 데이터의 계절성, 추세, 주기성을 반영하여 예측을 수행합니다. 특히, Prophet은 복잡한 하이퍼파라미터 튜닝 없이도 빠르고 정확한 예측을 수행할 수 있는 장점이 있습니다. Prophet 모델을 사용하여 각 업체별로 30일간의 단기 수요 예측을 수행하였으며, 일 단위의 계절성을 반영하기 위해 daily_seasonality=True 옵션을 설정하였습니다.



2. Prophet 모델 예측 결과 분석
    - Prophet 모델은 각 업체의 수요 패턴에 따라 예측을 수행하였으며, 장기적인 추세와 계절성을 잘 반영하였습니다. 예측 결과 그래프에서, 파란색 선은 예측된 수요를 나타내며, 주위의 음영은 95% 신뢰 구간을 나타냅니다. 대부분의 경우 예측된 수요가 실제 수요와 유사하게 나타났습니다. 일부 그래프에서는 급격한 수요 변화가 발생했으나, Prophet 모델은 이러한 변동을 반영하여 예측 범위를 넓게 설정하는 경향을 보였습니다.

3. ARIMA 모델

- ARIMA 모델은 자기회귀, 차분, 이동평균을 조합한 시계열 모델로, 본 프로젝트에서는 로그 변환을 통해 데이터의 스케일을 조정한 후 ARIMA 모델을 적용했습니다. 로그 변환은 데이터의 분포를 정규화하고, 극단적인 값의 영향을 줄이는 데 효과적입니다. 또한, 월간 계절성을 반영하여 모델 성능을 개선하였습니다.



4. ARIMA 모델 예측 결과 분석

    - 로그 변환: 데이터에 로그 변환을 적용하여 데이터 분포를 정규화했습니다. 로그 변환은 특히 급격한 수요 변동이 있는 데이터를 다룰 때 유용하며, 이를 통해 모델이 더 안정적으로 작동할 수 있었습니다.
    - 월간 계절성 반영: m=12로 설정하여 월간 계절성을 반영하였으며, 이는 특히 계절성이 강한 데이터에서 유리하게 작용했습니다.
    - 예측 결과: ARIMA 모델은 예측 결과를 원래 스케일로 되돌렸을 때, 비교적 실제 수요와 일치하는 결과를 도출했습니다. 로그 변환 덕분에 데이터 변동성이 완화되었으며, 모델 성능이 개선되었습니다. 그러나 급격한 변동이 있는 일부 업체에서는 여전히 예측 오차가 발생했습니다.
    - 각 업체별 예측 그래프에서 파란색 선은 훈련 데이터, 주황색 선은 실제 수요, 빨간색 선은 예측된 수요를 나타냅니다. ARIMA 모델은 대체로 예측된 값이 실제 값과 잘 일치하였으며, 로그 변환 덕분에 큰 변동에도 대응할 수 있었습니다. 다만, 급격한 변동이 있는 경우에는 예측이 다소 부정확할 수 있습니다.

결론

1. 단기 수요 예측 결과
    - 본 프로젝트에서는 Prophet과 수정된 ARIMA 모델을 활용하여 각 업체별 단기 수요 예측을 수행했습니다. 특히 ARIMA 모델은 전처리 과정(로그 변환)을 통해 개선된 성능을 보여주었으며, 수요 예측의 정확성을 높이는 데 중요한 역할을 했습니다.

      [Prophet 모델]

        - Prophet 모델은 장기적인 트렌드와 계절성을 반영하여 각 업체의 30일 단기 수요를 예측했습니다. Prophet 모델은 시계열 데이터의 복잡한 패턴을 자동으로 학습할 수 있어 간편하게 예측을 수행할 수 있습니다.
        - 그래프 분석: Prophet 예측 결과에서 파란색 선은 예측된 수요를, 음영은 신뢰 구간을 나타냅니다. 예측 결과가 실제 데이터와 일치하는 경우가 많으며, 장기적인 패턴과 계절성을 반영한 예측이 가능했습니다.

      [수정된 ARIMA 모델]

        - ARIMA 모델은 전처리 과정을 통해 로그 변환을 적용한 후 학습이 이루어졌습니다. 로그 변환을 통해 데이터 분포를 정규화함으로써 모델의 성능

이 향상되었습니다. 또한 월간 계절성을 반영하여 보다 정확한 예측을 수행했습니다.

- 그래프 분석: 로그 변환을 적용한 후의 ARIMA 예측 결과는 이전보다 훨씬 안정적이며, 실제 수요와 비교했을 때 큰 차이를 보이지 않았습니다. 파란색 선은 훈련 데이터를, 주황색 선은 실제 데이터를, 빨간색 선은 예측 데이터를 나타냅니다. ARIMA 모델은 특히 급격한 변동이 적은 업체에서 예측 성능이 뛰어났으며, 로그 변환 덕분에 데이터의 급격한 변동을 더 잘 처리할 수 있었습니다.

2. 최종 시스템 구축 결과 및 향후 개선 방안

- 본 프로젝트에서 구축한 수요 예측 시스템은 물류 및 재고 관리의 효율성을 높이는 데 기여할 수 있습니다. Prophet과 수정된 ARIMA 모델을 사용하여 각 업체별로 맞춤형 예측 결과를 제공함으로써 재고 부족과 과잉을 방지하고, 비용 절감 및 물류 운영의 효율화를 이끌어낼 수 있었습니다.

향후 개선 방안으로는 다음과 같은 방법을 고려할 수 있습니다:

1. 외부 데이터 활용: 경제 지표, 기후 데이터 등 외부 요인을 추가하여 수요 예측의 정확성을 더욱 높일 수 있습니다. 외부 데이터는 예측 모델에 추가적인 설명력을 제공하여, 예측 성능을 향상시킬 수 있습니다.
2. 모델 앙상블 기법 적용: Prophet과 수정된 ARIMA 모델의 장점을 결합한 앙상블 모델을 구축함으로써, 예측 성능을 더욱 개선할 수 있습니다. 이를 통해 예측 결과의 신뢰성을 높이고, 다양한 데이터 패턴에 대응할 수 있습니다.
3. 실시간 운영 시스템 구축: 실시간 데이터 흐름을 반영한 예측 시스템을 구축하여, 변화하는 시장 상황에 유연하게 대응할 수 있는 시스템을 개발할 수 있습니다.

결론

본 프로젝트는 Prophet과 수정된 ARIMA 모델을 활용한 수요 예측 시스템을 구축하여, 각 업체별로 최적화된 수요 예측을 수행했습니다. 이를 통해 물류 및 재고 관리의 효율성을 극대화하고, 비용 절감 및 운영 효율성을 높이는 데 기여할 수 있었습니다. 향후에는 외부

데이터를 추가하고, 더 다양한 모델을 사용하여 시스템을 확장할 수 있는 가능성을 열어 두고 있습니다.