산업인공지능학과

빅데이터 분석을 통한
포트홀 발생 위치 예측

데이터 분석 및 머신러닝 알고리즘을 활용한 포트홀 발생 원인 파악 및 발생 위치 예측

2024.12.09

2024254021 권준호

프로젝트
분석 배경

차량 사고 및 2차 사고 유발의 원인이 되는 도로 위의 포트홀

데이터 기반 관리를 통해 날씨, 도로 데이터, 공사장정보 등 을 활용하여
포트홀 발생 가능 지역을 예측하고 우선적으로 보수가 필요

프로젝트
분석 목표

포트홀 발생 주요 원인 파악 및 예측

포트홀 발생의 주요 원인을 통계적 분석과 머신러닝 기반 모델링을 통해 규명하고,
예측 모델을 활용하여 포트홀 발생 예방 및 도로 유지보수 효율성을 향상시키는 것을 목표로 함

프로젝트
분석 순서

분석 절차

데이터 수집

포트홀 위치 데이터 및
포트홀 발생과 관련이 있는
데이터 수집

EDA를 통한 분석

결측치 처리 및 데이터 시각화

모델 학습 및 예측

RF 모델을 통한
포트홀 위치 예측

데이터 소개

포트홀 원인 분석을 위한 데이터

No.	출처	데이터명	설명	기준 년도
1	지엔소프트(주)	포트홀 데이터.csv	대전시 유성구 포트홀 정보	24. 01. 01 ~ 24. 10. 31
2	기상청	강수량 데이터.csv	대전시 강수량 정보	24. 01. 01 ~ 24. 10. 31
3	기상청	기온 데이터.csv	대전시 기온 정보	24. 01. 01 ~ 24. 10. 31
4	대전 교통 빅데이터 플랫폼	교통량 데이터.csv	대전시 도로별 교통량 정보	24. 01. 01 ~ 24. 10. 31

포트홀 데이터

```python
import pandas as pd

df = pd.read_csv('./포트홀 데이터.csv')

# UID에서 연도, 월, 일을 추출하는 함수 정의
def extract_date(uid):
    # UID 형식은 'RAD01_YYYYMMDDHHMMSS_ID' 형식
    date_str = uid.split('_')[1]  # 'YYYYMMDDHHMMSS' 부분 추출
    year = '20' + date_str[:2]
    month = date_str[2:4]
    day = date_str[4:6]
    return int(year), int(month), int(day)

# 각 행에 대해 연도, 월, 일을 추출하여 새로운 컬럼 추가
df[['Year', 'Month', 'Day']] = df['DTCT_UID'].apply(lambda x: pd.Series(extract_date(x)))

# 필요한 컬럼만 선택하여 새로운 DataFrame 생성
new_df = df[['Year', 'Month', 'Day', 'PLC_LTTD', 'PLC_LGTD']]

# CSV 파일로 저장 (예: 'processed_pothole_data.csv')
new_df.to_csv('processed_pothole_data.csv', index=False, encoding='cp949')

p_df = pd.read_csv('./processed_pothole_data.csv', encoding='cp949')

result_df = p_df.groupby(['Year', 'Month', 'Day']).size().reset_index(name='Pothole_Count')

# 필요한 경우 CSV 파일로 저장
result_df.to_csv('processed_pothole_count_data.csv', index=False, encoding='cp949')

# 결과 확인
print(new_df.head(), end="\n########################################\n")
print(result_df.head())
```

```
   Year  Month  Day    PLC_LTTD    PLC_LGTD
0  2024      4    5  36.354964  127.344916
1  2024      4    5  36.357288  127.340732
2  2024      4    5  36.356196  127.332836
3  2024      4    5  36.355887  127.340315
4  2024      4    5  36.354569  127.350205
########################################
   Year  Month  Day  Pothole_Count
0  2024      1   19              7
1  2024      1   23              3
2  2024      2   13              3
3  2024      2   14              8
4  2024      2   19             10
```

포트홀 위치 데이터는 년/월/일/위도/경도의 데이터만 추출하여 저장 후,
각 날짜별 탐지된 포트홀 개수 집계한 데이터 추가 생성

데이터 정제

기온 데이터

```python
df = pd.read_csv('./기온 데이터.', encoding='cp949')

print(df.columns)

# '일시' 컬럼에서 연도, 월, 일 정보를 추출하여 새로운 컬럼 추가
df['Year'] = pd.to_datetime(df['일시']).dt.year.astype(int)
df['Month'] = pd.to_datetime(df['일시']).dt.month.astype(int)
df['Day'] = pd.to_datetime(df['일시']).dt.day.astype(int)

# 새로운 DataFrame 생성
new_df = df[['Year', 'Month', 'Day', '평균기온(℃)', '최고기온(℃)', '최저기온(℃)', '일교차']]

# CSV 파일로 저장
new_df.to_csv('processed_temperature_data.csv', index=False, encoding='cp949')

# 새로 생성된 테이블 확인
new_df.head()
```

	Year	Month	Day	평균기온(℃)	최고기온(℃)	최저기온(℃)	일교차
0	2024	1	1	3.8	9.0	-0.2	9.2
1	2024	1	2	4.0	6.4	1.5	4.9
2	2024	1	3	1.9	5.9	-1.2	7.1
3	2024	1	4	1.0	7.2	-3.7	10.9
4	2024	1	5	5.6	9.2	0.6	8.6

기온데이터는 년/월/일 및 평균기온, 최고기온, 최저기온, 일교차 정보만 추출하여 저장

강수량 데이터

```python
df = pd.read_csv('./강수량 데이터.csv', encoding='cp949')

print(df.columns)

# '일시' 컬럼에서 연도, 월, 일 정보를 추출하여 새로운 컬럼 추가
df['Year'] = pd.to_datetime(df['일시']).dt.year.astype(int)
df['Month'] = pd.to_datetime(df['일시']).dt.month.astype(int)
df['Day'] = pd.to_datetime(df['일시']).dt.day.astype(int)

# 새로운 DataFrame 생성
new_df = df[['Year', 'Month', 'Day', '강수량(mm)']]

# CSV 파일로 저장
new_df.to_csv('processed_rain_data.csv', index=False, encoding='cp949')

# 새로 생성된 테이블 확인
new_df.head()
```

	Year	Month	Day	강수량(mm)
0	2024	1	1	NaN
1	2024	1	2	NaN
2	2024	1	3	0.2
3	2024	1	4	NaN
4	2024	1	5	NaN

강수량 데이터는 년/월/일 및 강수량 데이터만 추출 후 저장

데이터 정제

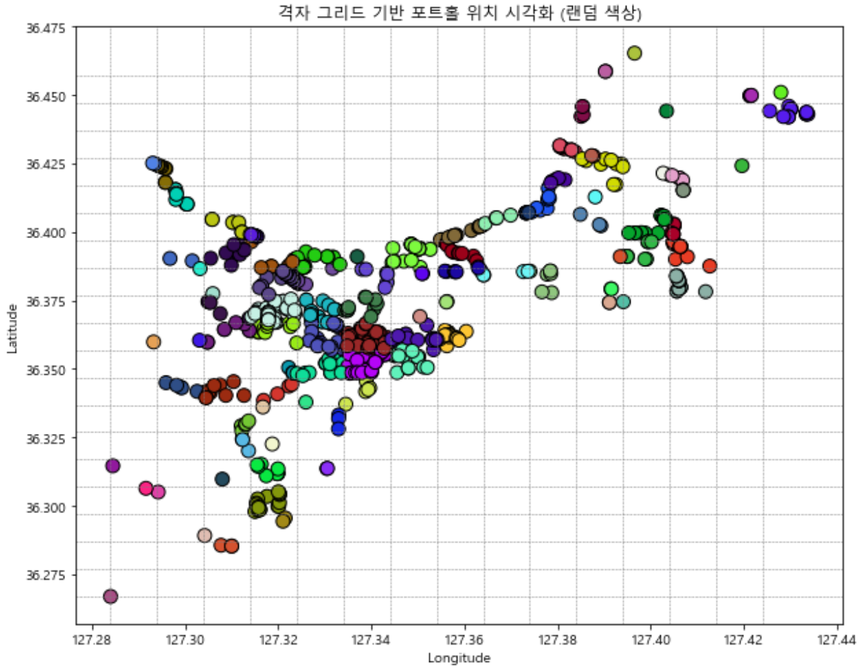
교통량 데이터

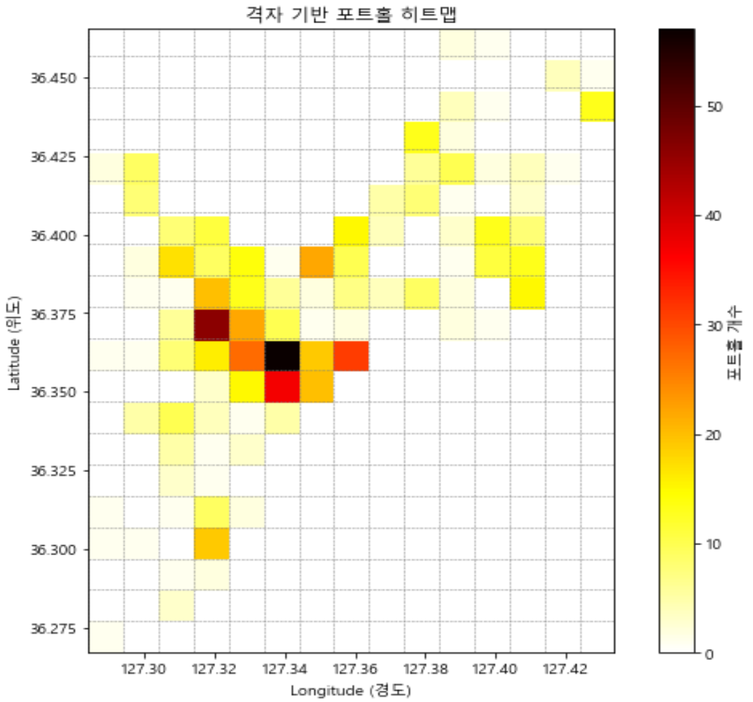
```python
# 교통량 데이터 처리
df = pd.read_csv('./교통량 데이터.csv', encoding='cp949')

# 유성구 도로 목록
yuseong_roads = [
    '가정로', '계룡로', '계백로', '노은길', '노은로', '문지로', '자양로',
    '도안대로', '도안동로', '도안중로', '엑스포로', '유성대로',
    '현충원로', '구즉세종로', '북유성대로', '월드컵대로', '구즉세종로',
    '대학로(1)', '배울1로', '동서대로2'
]

filtered_df = df[df['도로'].isin(yuseong_roads)]

# '월' 컬럼을 기준으로 데이터 피벗
pivot_df = filtered_df.pivot_table(
    index='월',                # 행: 월
    columns='도로',            # 열: 도로
    values='총교통량',          # 값: 교통량
    fill_value=None              # NaN은 그대로 유지
).reset_index()

# '월' 컬럼에서 년과 월 숫자로 분리
pivot_df['Year'] = pivot_df['월'].str.extract(r'(\d{4})').astype(int)  # '2024년'에
pivot_df['Month'] = pivot_df['월'].str.extract(r'(\d{2})월').astype(int)  # '01월'어

# '월' 컬럼 제거
pivot_df = pivot_df.drop(columns=['월'])

# 컬럼 순서를 변경하여 'Year'와 'Month'를 앞으로 이동
columns_order = ['Year', 'Month'] + [col for col in pivot_df.columns if col not in
pivot_df = pivot_df[columns_order]

# 월별 총교통량 계산
pivot_df['총교통량'] = pivot_df.drop(columns=['Year', 'Month']).sum(axis=1)

pivot_df.to_csv('./processed_traffic_data.csv', encoding='cp949', index=False)

pivot_df
```

도로	Year	Month	가정로	계룡로	계백로	구즉세종로	노은길	노은로	대학로(1)	도안대로	...
0	2024	1	25399	52420	51305	39810	14631	26371	19240	28994	...
1	2024	2	24446	52955	51753	40068	14758	25919	19200	27960	...
2	2024	3	25517	54980	52724	40356	15199	27525	20303	28761	...
3	2024	4	27109	56591	53013	41021	15510	28548	21344	28304	...
4	2024	5	26058	56097	52838	40712	15426	28025	20481	28157	...
5	2024	6	25220	55991	52330	40354	15683	27899	20003	26672	...
6	2024	7	26770	57175	50020	40724	15728	28211	20477	28643	...
7	2024	8	25746	55392	50020	40811	15729	27452	19900	29304	...
8	2024	9	24674	56240	50269	41317	15576	27138	19019	34443	...
9	2024	10	24766	57028	50208	41816	15353	27078	19063	35136	...

교통량 데이터는 도로별 교통량을 '월'을 기준으로 데이터를 피벗 테이블로 변환 후,
각 도로별 월별 교통량을 열로 재구성 후 년/월/일 및 월별 총 교통량 계산하여 저장

포트홀 데이터





포트홀 위치 정보 데이터를 활용하여 포트홀의 위치를 시각화

왼쪽은 각 포트홀의 위치를 점으로 표시하고 그리드별로 색상을 부여하여 시각화한 그래프이며,

오른쪽은 포트홀의 개수를 히트맵으로 표시

포트홀 데이터

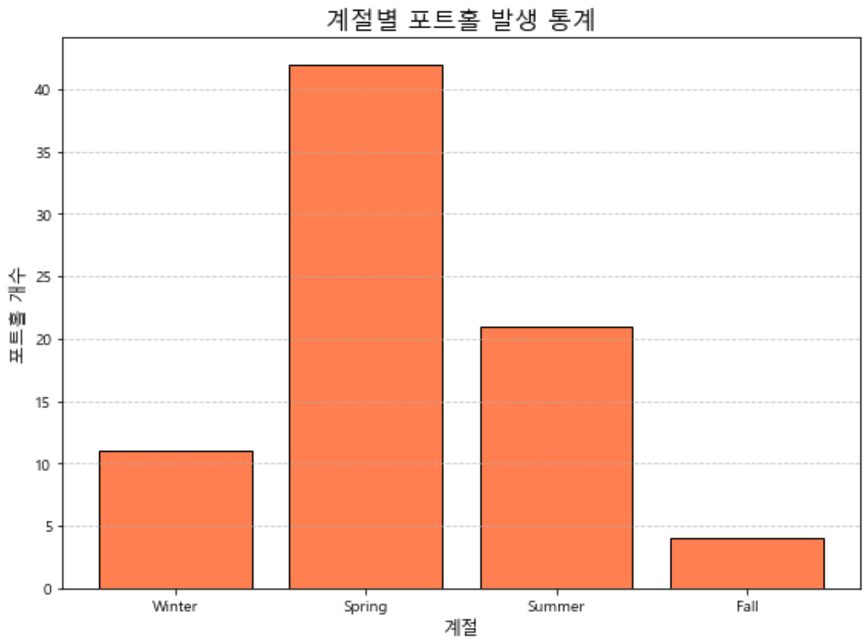
```
# 데이터 로드
p_df = pd.read_csv('./processed_pothole_data_with_grid.csv', encoding='cp949')

# 각 Grid_ID별 포트홀 개수 계산 (이미 저장된 데이터 활용)
grid_counts = p_df.groupby(['Grid_ID', 'Grid_X', 'Grid_Y']).size().reset_index(name

# 상위 5개 그리드 추출
top_5_grids = grid_counts.nlargest(5, 'Pothole_Count')

# 격자 중심 위경도 계산
grid_size = 0.01  # 격자 간격
min_lat, min_lon = p_df['PLC_LTTD'].min(), p_df['PLC_LGTD'].min()

top_5_grids['Center_Latitude'] = min_lat + (top_5_grids['Grid_X'] + 0.5) * grid_siz
top_5_grids['Center_Longitude'] = min_lon + (top_5_grids['Grid_Y'] + 0.5) * grid_si

# 결과 출력
top_5_centers = top_5_grids[['Grid_ID', 'Center_Latitude', 'Center_Longitude', 'Pothole_Count']]
top_5_centers
```

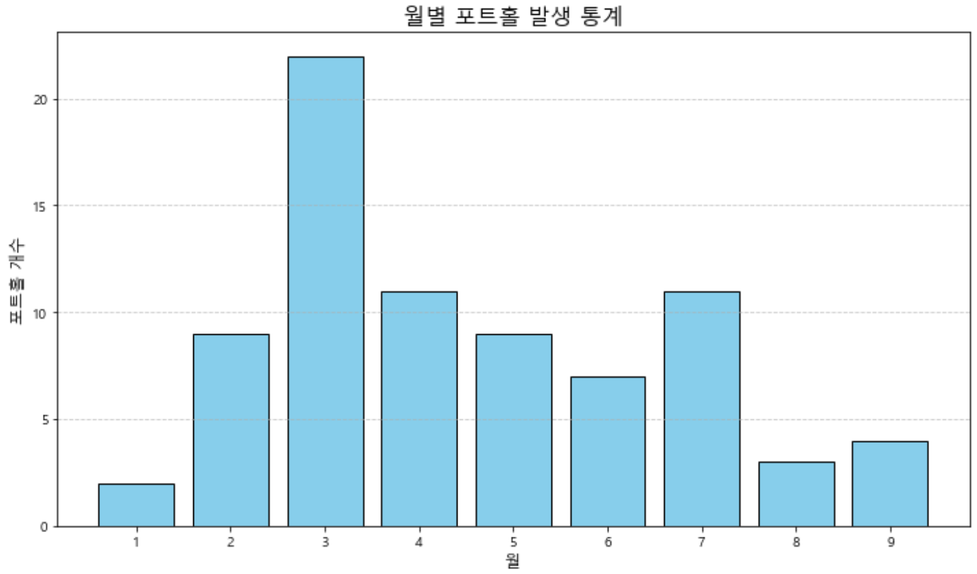
	Grid_ID	Center_Latitude	Center_Longitude	Pothole_Count
30	30	36.361938	127.339043	57
34	34	36.371938	127.319043	46
23	23	36.351938	127.339043	37
32	32	36.361938	127.359043	31
29	29	36.361938	127.329043	27

포트홀이 가장 많이 발생한 상위 5개의 그리드의 중심 좌표를 구한 결과,
장대동, 노은동, 봉명동, 어은동 순으로 나타남

데이터 시각화

월별/계절별 포트홀 통계



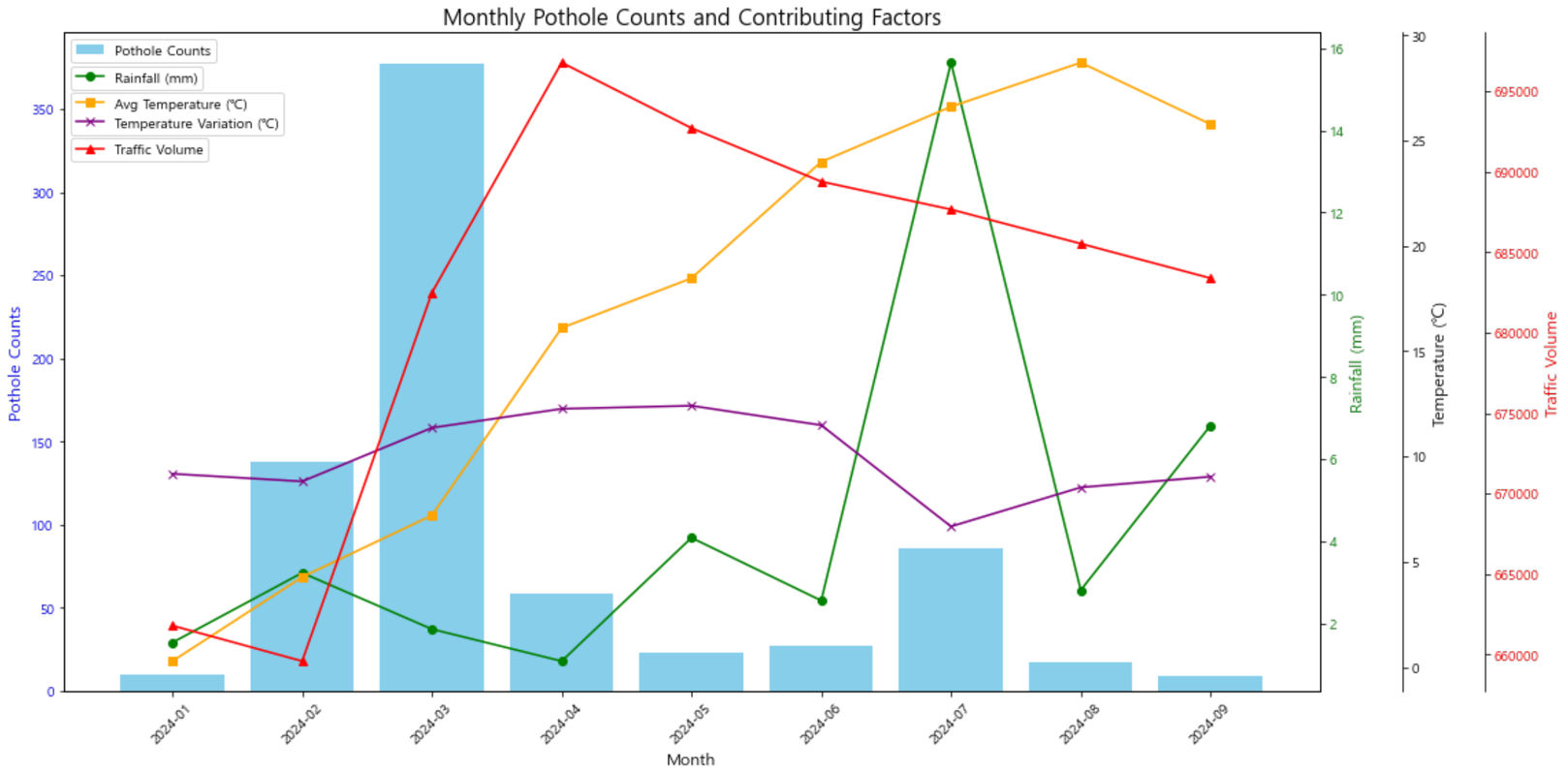


포트홀 발생 개수를 월별/계절별로 통계 그래프 생성

해빙기인 3월에 가장 많이 발생함

데이터 시각화

월별 포트홀 발생 개수와 주요 요인의 변화 시각화

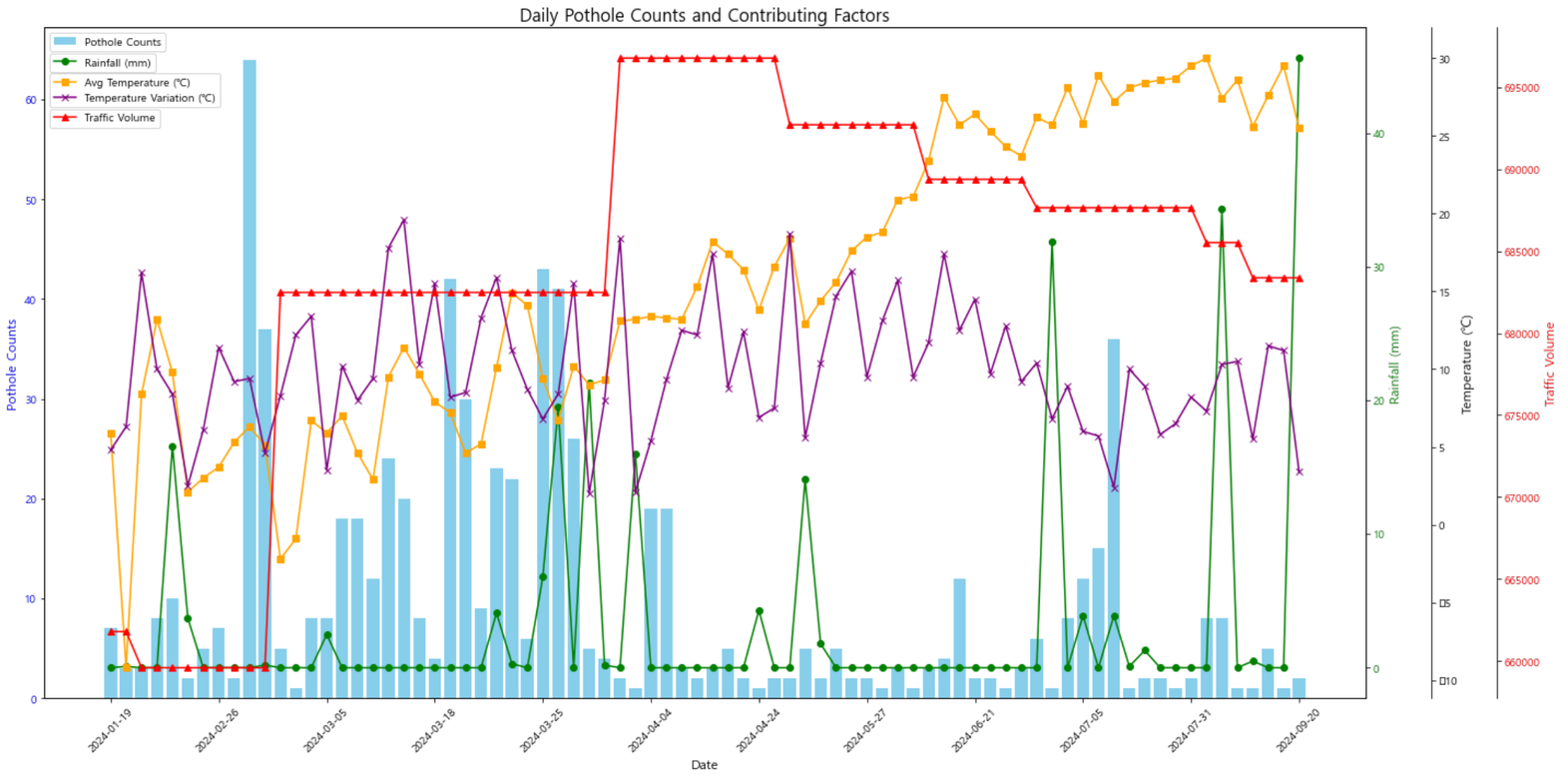


강수량과 기온은 포트홀 발생에 중요한 요인으로 작용하며,

특히 장마철(강수량이 높은 여름철)과 겨울철(일교차가 큰 시기)에 포트홀이 많이 발생하는 경향이 있음

데이터 시각화

일별 포트홀 발생 개수와 주요 요인의 변화 시각화

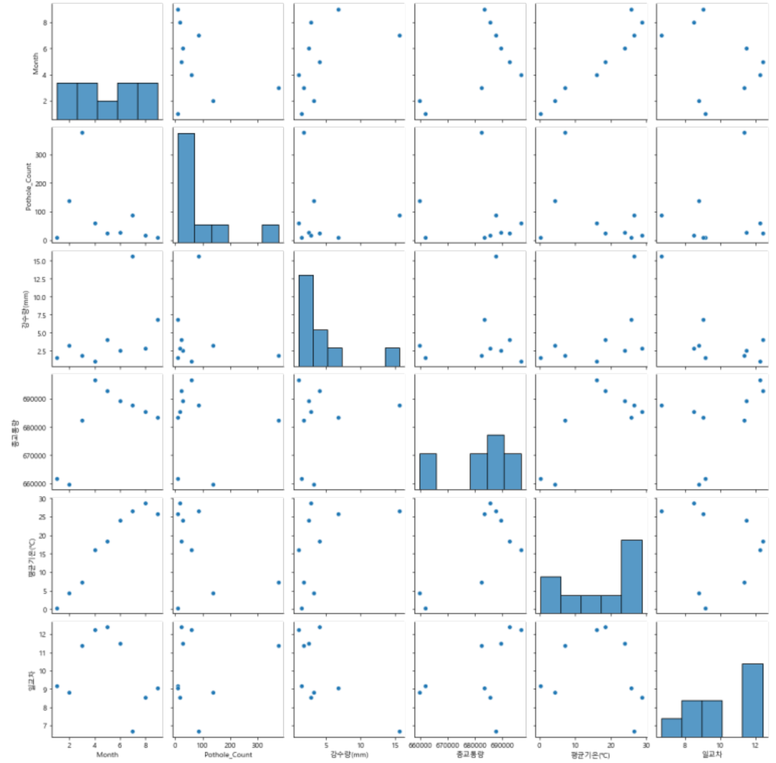


강수량과 기온은 포트홀 발생에 중요한 요인으로 작용하며,

특히 장마철(강수량이 높은 여름철)과 겨울철(일교차가 큰 시기)에 포트홀이 많이 발생하는 경향이 있음

데이터 시각화

포트홀 개수에 대한 상관관계 분석



포트홀과 일교차가 약한 양의 상관계수(0.15)를 보이며,
각 특성에서 선형적인 관계를 보이는 특성은 발견되지 않음

모델 학습

데이터 전처리

```
# 데이터 로드
pothole_df = pd.read_csv('./processed_pothole_data.csv', encoding='cp949')
rainfall_df = pd.read_csv('./processed_rain_data.csv', encoding='cp949')
temperature_df = pd.read_csv('./processed_temperature_data.csv', encoding='cp949')

# 데이터 병합 (교통량 데이터 제외)
merged_df = pd.merge(pothole_df, rainfall_df, on=['Year', 'Month', 'Day'], how='outer')
merged_df = pd.merge(merged_df, temperature_df, on=['Year', 'Month', 'Day'], how='outer')

# NaN 값을 0으로 대체 (필요한 경우)
merged_df = merged_df.dropna()

# 통합 데이터프레임 저장
merged_df.to_csv('./merged_pothole_locati
```

	Year	Month	Day	PLC_LTTD	PLC_LGTD	강수량(mm)	평균기온(℃)	최고기온(℃)	최저기온(℃)	일교차
0	2024	4	5	36.354964	127.344916	0.0	13.3	19.2	9.9	9.3
1	2024	4	5	36.357288	127.340732	0.0	13.3	19.2	9.9	9.3
2	2024	4	5	36.356196	127.332836	0.0	13.3	19.2	9.9	9.3
3	2024	4	5	36.355887	127.340315	0.0	13.3	19.2	9.9	9.3
4	2024	4	5	36.354569	127.350205	0.0	13.3	19.2	9.9	9.3
...	...	...	...	...	...	...	...	...	...	...
741	2024	1	19	36.427741	127.387724	0.0	5.9	9.3	4.5	4.8
742	2024	1	19	36.427899	127.387323	0.0	5.9	9.3	4.5	4.8
743	2024	1	19	36.398783	127.358005	0.0	5.9	9.3	4.5	4.8
744	2024	1	19	36.398713	127.357864	0.0	5.9	9.3	4.5	4.8
745	2024	1	19	36.384627	127.343459	0.0	5.9	9.3	4.5	4.8

포트홀 위치 정보, 강수량, 기온 데이터를 날짜별로 결합하여 데이터 프레임으로 만든 후,
결측치는 모두 제거하여 저장

모델 학습

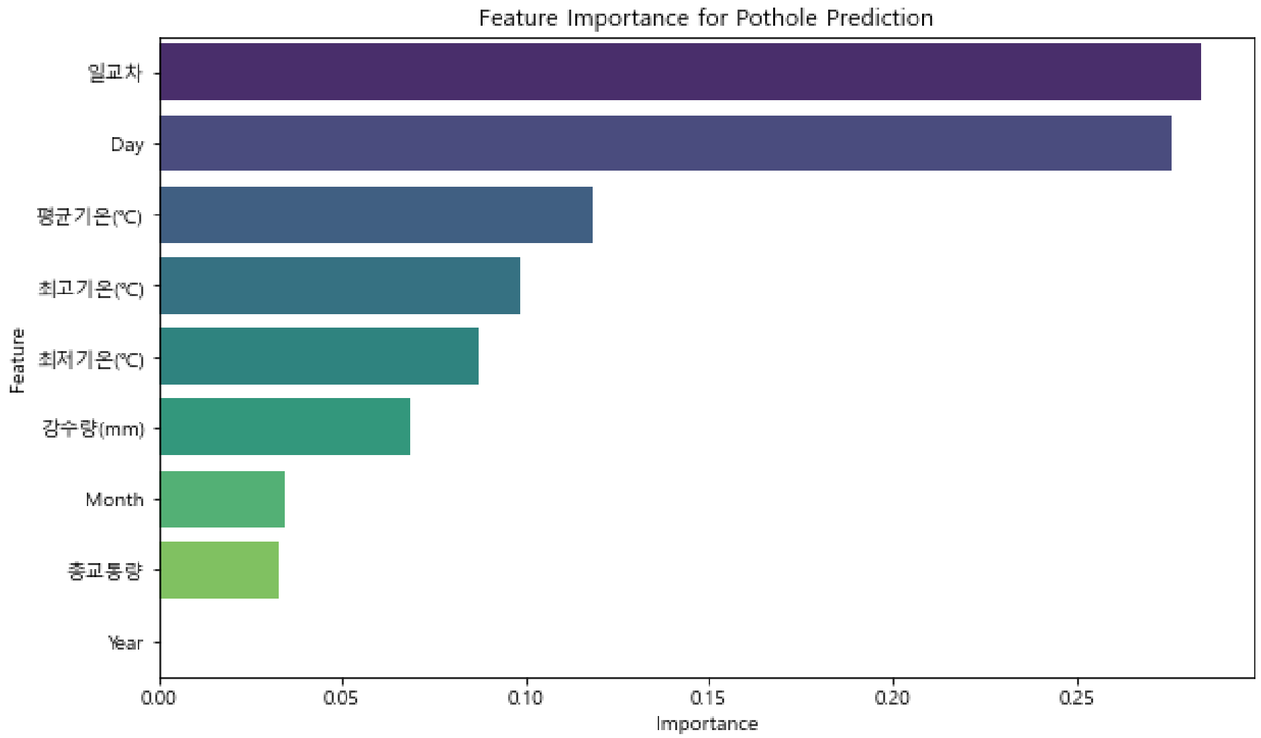
랜덤 포레스트 모델을 통한 학습

```python
# 데이터 로드
pothole_df = pd.read_csv('./processed_pothole_data.csv', encoding='cp949')
rainfall_df = pd.read_csv('./processed_rain_data.csv', encoding='cp949')
temperature_df = pd.read_csv('./processed_temperature_data.csv', encoding='cp949')
traffic_df = pd.read_csv('./processed_traffic_data.csv', encoding='cp949')

tmp_traffic_df = traffic_df[['Year', 'Month', '총교통량']]

# 데이터 병합 (교통량 데이터 제외)
merged_df = pd.merge(pothole_df, rainfall_df, on=['Year', 'Month', 'Day'], how='outer')
merged_df = pd.merge(merged_df, temperature_df, on=['Year', 'Month', 'Day'], how='outer')
merged_df = pd.merge(merged_df, tmp_traffic_df, on=['Year', 'Month'], how='outer')

# NaN 값을 0으로 대체 (필요한 경우)
merged_df = merged_df.dropna()

# 통합 데이터프레임 저장
merged_df.to_csv('./merged_pothole_location_weather_data.csv', index=False, encoding='cp949')

# 입력 데이터(X)와 출력 데이터(y) 분리
X = merged_df.drop(columns=['PLC_LTTD', 'PLC_LGTD'])  # 입력 변수에서 위도와 경도를 제거
y = merged_df[['PLC_LTTD', 'PLC_LGTD']]  # 출력 변수는 위도(LTTD)와 경도(LGTD)

# 데이터를 8:2로 분할 (학습 데이터 80%, 테스트 데이터 20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Random Forest 모델 초기화
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# 모델 학습
rf_model.fit(X_train, y_train)

# 테스트 데이터 예측
y_pred = rf_model.predict(X_test)

# 테스트 데이터에 대한 평가
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# 출력
print("테스트 데이터에 대한 평가:")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"R-squared (R2 Score): {r2:.4f}")
```

```
테스트 데이터에 대한 평가:
Mean Squared Error (MSE): 0.0006
R-squared (R2 Score): 0.4184
```

기존의 시각화한 모든 데이터를 년/월/일 기준으로 병합하고,
RandomForestRegressor 모델을 사용하여 학습 진행

모델의 특성 중요도 분석



일교차가 가장 중요한 특성으로 나타났으며,

년도는 24년 1개 년도밖에 없어 중요도는 0으로 나온 것으로 예상

모델 최적화

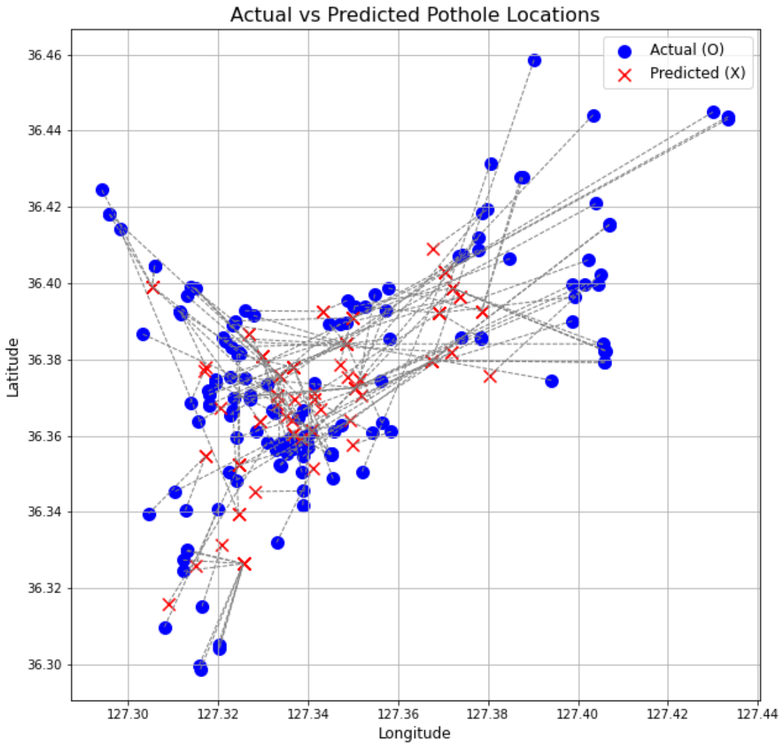
```python
from sklearn.model_selection import GridSearchCV

# 하이퍼파라미터 후보군 설정
param_grid = {
    'n_estimators': [50, 100, 200],  # 트리 개수
    'max_depth': [10, 20, 30],  # 최대 깊이
    'min_samples_split': [10, 20, 30],  # 노드를 나누는 최소 샘플 수
    'min_samples_leaf': [10, 20, 40]  # 리프 노드의 최소 샘플 수
}

# RandomForestRegressor 모델 초기화
rf_model = RandomForestRegressor(random_state=42)

# GridSearchCV 초기화
grid_search = GridSearchCV(
    estimator=rf_model,
    param_grid=param_grid,
    scoring='r2',  # 평가 지표로 R-squared 사용
    cv=3,  # 3-fold 교차 검증
    verbose=2,
    n_jobs=-1  # 모든 CPU 코어 사용
)

# 그리드 탐색 실행
grid_search.fit(X_train, y_train)

# 최적의 파라미터와 성능 출력
print("Best Parameters:", grid_search.best_params_)
print("Best R2 Score:", grid_search.best_score_)

# 테스트 데이터에서 최적 모델 평가
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\n테스트 데이터에 대한 평가:")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"R-squared (R2 Score): {r2:.4f}")
```

```python
    # 개별 모델 학습
    rf_pipeline.fit(X_train, y_train)
    gb_pipeline.fit(X_train, y_train)
    xgb_pipeline.fit(X_train, y_train)

    # 개별 모델 평가
    rf_pred = rf_pipeline.predict(X_test)
    gb_pred = gb_pipeline.predict(X_test)
    xgb_pred = xgb_pipeline.predict(X_test)

    print("\n개별 모델 성능:")
    print(f"Random Forest R2: {r2_score(y_test, rf_pred):.4f}")
    print(f"Gradient Boosting R2: {r2_score(y_test, gb_pred):.4f}")
    print(f"XGBoost R2: {r2_score(y_test, xgb_pred):.4f}")

    # 앙상블 모델 정의
    voting_model = VotingRegressor(
        estimators=[
            ('rf', rf_pipeline),
            ('gb', gb_pipeline),
            ('xgb', xgb_pipeline)
        ]
    )

    # 앙상블 모델 학습
    voting_model.fit(X_train, y_train)

    # 앙상블 모델 평가
    voting_pred = voting_model.predict(X_test)
    mse = mean_squared_error(y_test, voting_pred)
    r2 = r2_score(y_test, voting_pred)

    print("\n앙상블 모델 성능:")
    print(f"Mean Squared Error (MSE): {mse:.4f}")
    print(f"R-squared (R2 Score): {r2:.4f}")

    return voting_model

# 위도(PLC_LTTD) 모델 학습
print("위도(PLC_LTTD) 모델 학습")
ensemble_model_lat = train_ensemble_model_with_scaling(X_train, y_train_lat, X_test, y_test_lat)

# 경도(PLC_LGTD) 모델 학습
print("\n경도(PLC_LGTD) 모델 학습")
ensemble_model_lon = train_ensemble_model_with_scaling(X_train, y_train_lon, X_test, y_test_lon)
```

파라미터 변경 및 RF 모델을 포함한 다른 모델과 앙상블 학습을 시도함

데이터를 정규화하여도 기존 RF 모델보다 성능이 저하되는 현상 발생

최종 모델 성능 평가



실제와 예측의 차이가 많이 나는 것을 확인할 수 있음

결론

1. 통계적 분석 결과
   1) 주요 원인: 일교차
   2) 분석 방법: 데이터 시각화를 통한 분석

2. 머신러닝 모델을 통한 분석 결과
   1) 주요 원인: 일교차
   2) 분석 방법: RandomForest 기반의 비선형 패턴 분석

3. 결론
   1) 일교차가 포트홀 발생에 가장 큰 영향을 미침

향후 계획 및 개선점

1. 계획
   1) 모델 고도화 - 새로운 머신러닝, 딥러닝 모델로 학습
   2) GN-RAD 시스템에 포트홀 위치 예측 서비스 추가

2. 개선점
   1) 학습용 데이터 수집
   2) 포트홀의 위치와 포트홀 개수와의 연결점 분석