포트홀 발생 예측 모델 기반 보험 리스크 관리 전략

Sherlock Holes

고려대학교 박준희

송상현

이세은

1. 문제배경 및 리스크 정의

1-1. 포트홀 발생의 문제점 및 기존 관리체계

포트홀(pothole)은 도로 표면의 아스팔트가 파손되어 노면 일부가 움푹 패인 형태로 나타나는 도로 결함이다. 이는 주로 동절기나 장마철에 집중적으로 발생하며, 강우 및 결빙·해빙 과정에서 도로 노면의 균열 부위에 수분이 침투하고 반복되는 차량 하중에 의해 발생한다. 포트홀은 차량 주행 안정성을 크게 저해하여 사고 위험을 높이며, 나아가 운전자의 안전을 위협하는 심각한 문제로 부각되고 있다.[i]

최근 국민권익위원회가 발표한 『포트홀(도로파임) 관련 민원 분석』에 따르면, 2022년 1월부터 2024년 2월까지 약 2년여 기간 동안 포트홀 관련 민원이 총 52,262건 접수되었으며, 특히 2024년 1월에는 전년 동월 대비 약 5.8배 증가한 것으로 나타났다. 이 중 차량 파손에 따른 배상 요청이 전체 민원의 약 9.8%를 차지하여, 포트홀 사고의 우발성과 예측 불가능성을 여실히 드러내고 있다. 포트홀 발생은 특정 지역이나 계절에 한정되지 않고 전국적으로 광범위하게 발생하며, 이는 리스크의 동질성과 대량성 요건을 명백히 충족하고 있음을 시사한다.

포트홀로 인한 경제적 손실 규모도 상당한 수준이다. 국민 입장에서는 차량 손상 및 수리 비용뿐 아니라 사고에 따른 부상 및 인명 피해 가능성까지 부담해야 하며, 최근 한국도로공사 자료에 따르면 2020년부터 2023년까지 포트홀로 인한 배상 지급액이 약 44억 원에 달했다. 정부 및 지자체 차원에서도 지속적인 도로 보수 및 유지관리에 매년 수십억 원 이상의 비용을 투입하고 있으며, 보험사의 경우 포트홀 사고에 따른 영조물 배상책임보험 청구 증가로 인해 손해율 관리에 어려움을 겪고 있다.

현행 포트홀 리스크 관리체계는 대부분 사고 발생 후 대응하는 사후 관리 중심으로 운영되고 있다. 포트홀 사고 발생 시 피해자는 도로 관리 주체를 통해 손해 배상을 받을 수 있는데, 고속도로는 한국도로공사, 국도 및 지방도로는 국토교통부 또는 해당 지방자치단체가 관리한다. 관리 주체가 영조물 배상책임보험에 가입한 경우 피해자는 사고 후 보상 신청서를 제출하여 손해를 보전받을 수 있다. 영조물 배상책임보험은 국가 또는 지자체가 소유·관리하는 도로, 교량, 공원 등 공공시설에서 발생한 사고에 대한 배상책임을 보장하는 보험으로, 보험사가 사고 현장 조사 및 피해자 면담 등 손해사정 절차를 수행한 후 피해자에게 보험금을 지급하는 방식이다.[ii]

1-2. 이상기후로 인한 포트홀 발생 위험성 증가와 상관관계 설명

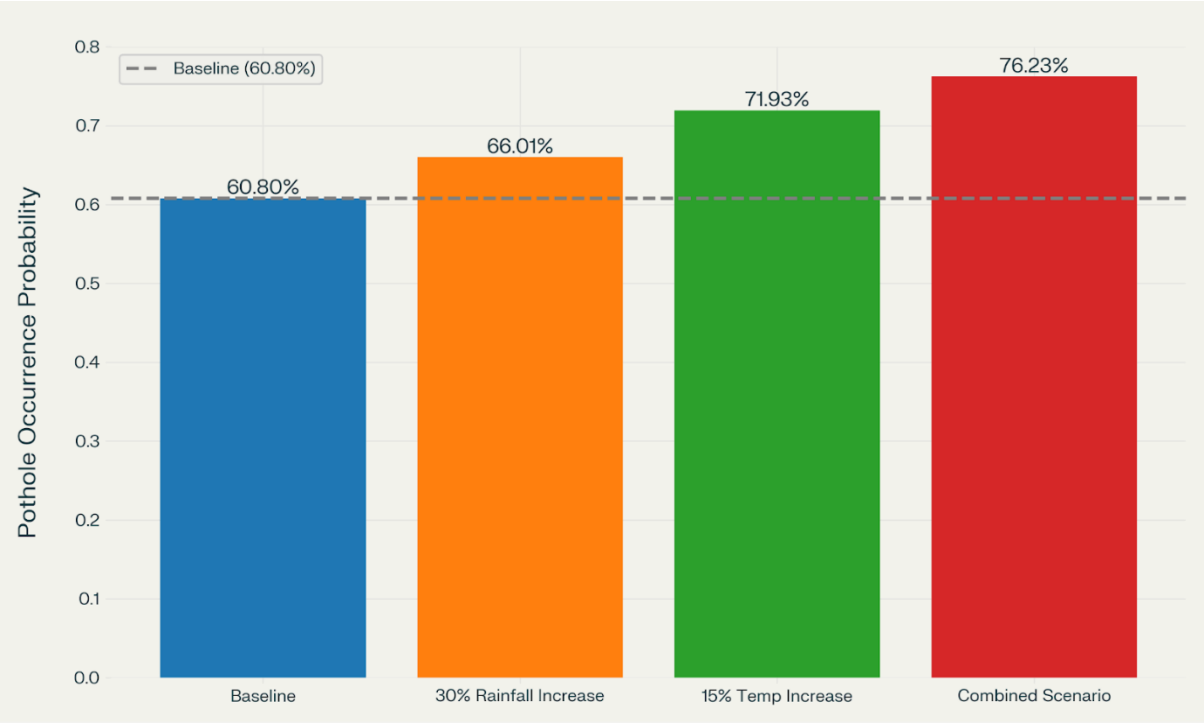
최근 기후변화로 인한 이상기후 현상이 빈번해지면서, 국내 주요 도시 및 고속도로에서 포트홀 발생 건수가 현저히 증가하고 있다. 서울시의 경우 2024년 1월 한 달간 포트홀 보수 건수는 4,527건, 2월 26일까지 추가로 2,540건이 발생해 두 달 동안 약 7,000건을 기록하였다. 이는 전년도 같은 기간(1월 2,271건, 2월 1,723건)과 비교해 두 배 이상 증가한 수치로, 기온 변동과 함께 강수일수 증가가 주된 원인으로 지목되었다.[iii] 한국도로공사 자료에 따르면, 고속도로의 포트홀 발생 건수 역시 지속적으로 증가하여 2020년부터 2024년 8월까지 총 22,692건이 보고되었으며, 2020년 4,440건에서 2023년에는 5,801건으로 약 31% 상승하였다. 특히 전국 평균 강수량이 2022년 1,140.5mm에서 2023년 1,740.4mm로 52.6% 급증하면서, 강우량과 포트홀 발생 간의 명확한 상관관계를 나타내고 있다.[iv]

변수	계수 (coef)	표준오차 (std err)	z 값	p 값	오즈비 (OR)	95% CI (하한)	95% CI (상한)
Intercept	2.3946	0.070	34.011	0.000	10.9636	9.5504	12.5858
누적강수량 (cumulative_rainfall)	0.7489	0.028	26.511	0.000	2.1147	2.0008	2.2351
평균습도 (avg_humidity)	-0.0241	0.040	-0.609	0.542	0.9762	0.9034	1.0548
연간 기온변동폭 (annual_temp_range)	3.3457	0.068	49.241	0.000	28.3795	24.8410	32.4220
일일 기온변동폭 (avg_daily_temp_range)	-0.3141	0.045	-7.033	0.000	0.7304	0.6692	0.7972

본 보고서는 이러한 이상기후가 포트홀 발생에 미치는 영향을 정량적으로 분석하기 위해 로지스틱 회귀모형을 사용하였다. 분석 결과, 누적 강수량과 연간 기온 변동폭은 포트홀 발생 확률에 통계적으로 유의한 영향을 미쳤다. 구체적으로, 누적 강수량이 증가할 때 포트홀 발생 확률은 약 2배 상승하며($p < 0.001$), 연간 기온 변동폭이 증가할 경우 발생 확률이 28배까지 급증하는 것으로 나타나, 누적 강수량 증가와 연간 기온차와 관련있는 동결-해동 주기가 포트홀 형성에 큰 양의 상관관계가 있음을 파악할 수 있다. 반면, 평균 일교차와 평균 습도는 통계적으로 유의하지 않거나($p>0.05$) 영향력이 제한적(OR가 1에 가까움)이었다.

향후 이상기후가 심화될 경우, 포트홀 발생 위험은 현재보다 비선형적으로 증가할 것으로 전망된다. 대한민국 기상청과 국립기상과학원이 제시한 온실가스 고배출 시나리오에 따르면, 한반도의 극한 강수량은 현재대비 약 30% 증가하며, 연간 기온 변동폭 또한 10%~20% 확대될 것으로 예측된다.[v] 위 시나리오에 대해 로지스틱 회귀분석에서 분석한 누적 강수량 오즈비와 연간 기온 변

동폭 오즈비를 이용해 이상기후 시 포트홀 발생 확률을 계산해보면, 강수량이 30% 증가하는 경우 포트홀 발생확률이 66.01%까지 증가하였으며, 연간 기온차가 15% 증가할 경우 71.93%까지 증가하였다. 특히 강수량과 기온창가 동시에 증가하는 복합 시나리오에서는 포트홀 발생 확률이 76.23%에 이르러, 기준선 대비 15.43%p(25.38%)나 높아지는 것으로 확인되었다. 이러한 결과는 이상기후가 포트홀 발생 위험을 실질적으로 증대시킴을 보여주는 통계적 근거이며, 향후 기후환경 변화의 심화에 따라 포트홀 리스크가 보다 광범위하고 만성적인 형태로 고착화될 가능성을 시사한다.



1-3. 포트홀 발생증가에 따른 보험사 배상책임 리스크의 증대와 관리 필요성

이상기후로 인한 포트홀 발생의 증가와 발생 양상의 불규칙성은 공공기관 및 보험사의 배상책임 리스크 확대로 이어진다. 실제로 최근 포트홀 사고가 급증하면서, 이에 따른 영조물손해배상보험 가입 및 피해배상 건수와 보험금 지급액이 매년 가파르게 증가하고 있다. 한국도로공사의 자료에 따르면, 포트홀 배상 건수는 2020년 707건에서 2022년 1,737건으로 늘었고, 2023년에는 2832건에 달했다. 이에 따른 연도별 배상 지급액도 2021년 약 19억원, 2022년 34억원, 2023년에는 41억원 이상을 기록하여 매년 최고치를 갱신하고 있다.[vi] 이러한 증가 추세는 도로 노후화와 이상기후에 따른 포트홀 발생 급증이 직접적 원인으로 지목된다. 국회 국토교통위원회 자료에서도 "도로 노후, 기후변화 등 다양한 원인으로 포트홀이 급증하고 있으며, 향후 노후화 진행을 고려하면 포트홀은 더욱 증가할 것으로 전망된다"고 지적한 바 있다. 이는, 기후변화로 인한 포트홀 피해의 확대가 일시적 현상을 넘어선, 구조적이고 중장기적인 리스크 요인으로 자리 잡고 있음을 의미한다.

이러한 현상은 보험사 입장에서 볼 때, 기존에는 국지적 관리 이슈로 간주되던 포트홀 리스크가 기후 변화의 영향에 따라 전국적인 차원에서 빈발하는 대규모 위험으로 전환되고 있음을 보여준다. 포트홀 사고로 인한 보험금 지급 규모가 지속적으로 확대되고 있는 현 상황은, 해당 리스크가 전통적인 보험계리 모델로는 설명이 어려운 기후 비정상성(climate non-stationarity)에 기반하고 있음을 시사한다.

따라서 보험산업은 이러한 거대화된 리스크에 대응하기 위해, 보다 과학적이고 선제적인 리스크 예측 및 관리 전략을 수립해야한다. 본 보고서는 기존의 사후 대응 중심의 리스크 관리 체계를 넘어, 포트홀 발생에 영향을 미치는 다양한 변수들을 활용하여 예측 모델을 수립하고자 한다. 특히, 이상기후와 같은 불확실한 환경에서도 높은 예측 정확도와 강건성을 유지할 수 있는 모델을 구축함으로써, 기존 방식과 차별화된 새로운 유형의 리스크 측정 기법과 관리 전략을 제안하는데 목적이 있다.

2. 포트홀 발생 예측 모델 구축

2-1. 사용데이터 및 데이터 전처리 과정

모델 학습을 위해 사용한 설명변수는 교통량 데이터를 포함하여 총 10개의 변수를 사용하였으며, 예측변수는 "포트홀 발생 여부"이다. 아래는 사용한 변수들의 데이터 가공 및 전처리과정을 요약한 내용이다.

1. 포트홀 발생 여부

모델의 종속변수로 활용된 포트홀 발생 여부의 경우, 서울 열린데이터 광장의 "서울시 포트홀 보수 위치 정보" 데이터를 활용하였다. 총 22985개의 데이터가 존재하며, 포트홀이 발생한 행정구역, 도로의 노선, 발생 도로명 및 지번주소가 포함되어 있다.

2. 도로별 교통량

포트홀이 발생한 도로의 도로별 교통량을 구하기 위해 국가교통데이터베이스의 "도로구간별 차량 교통량" 데이터와 ITS 국가교통정보센터의 "전국표준노드링크" 데이터를 사용하였다. 우선 도로구간별 차량 교통량의 경우 8501개 도로의 데이터를 포함하고 있으며, 각 데이터는 ITS LINK ID, 도로등급, 도로명, 차선수, 승용차, 트럭, 버스의 각 시간대별 교통량을

포함한다. 이 중 승용차, 트럭, 버스의 각 시간대별 교통량을 더하여 전일 승용차 교통량, 트럭 교통량, 버스 교통량 파생변수를 생성하였으며, 이를 통해 각 차종별 교통량을 더한 총교통량 변수를 생성, (트럭 교통량 + 버스 교통량) / 총교통량을 계산하여 중대형차량 교통량 파생변수를 생성하였다.

도로별 교통량과 포트홀 발생 도로를 정확하게 매칭시키기 위해, 포트홀 발생 주소를 경위도로 변환한 후, 파이썬의 geopandas 라이브러리를 활용하여 각 포트홀 발생 경위도 별 가장 가까운 도로링크를 매칭하고 도로링크를 기준으로 포트홀 발생 도로에 교통량 데이터를 join하였다. 이때, 포트홀이 발생하지 않았지만 교통량이 존재하는 도로를 "정상도로" 데이터로 구분하여(포트홀 발생 여부 : 0) 모델학습에 진행하였다. 또한 일부 도로에 대해 교통량 데이터가 결측된 문제가 존재하였는데, 다차선 도로(2차선 이상)의 경우 교통량이 있는 가장 가까운 도로의 교통량으로 보간하였다. 1차선 도로의 경우 차선수 기반 상대비율 보간 방식을 적용하였는데, 먼저 전체 도로망에서 차선수별 평균 교통량을 산출하고 이를 이용해 차선수 간 교통량 비율(예 : 1차선:2차선 =1:2, 1차선:3차선 = 1:3 등)을 계산하였다. 이후, 교통량 정보가 결측된 1차선 도로에 대하여 가장 인접한 교통량이 존재하는 다차선 도로를 탐색하고, 해당 도로의 차선수를 기준으로 사전에 계산된 차선비율을 적용하였다. 예를 들어, 인접한 4차선 도로의 교통량이 400대이고, 1:4 비율이 도출되었다면, 해당 1차선 도로의 교통량은 100대로 보간하는 방식이다. 이를 통해 공간적으로 인접하면서 구조적으로 유사한 도로의 교통 흐름을 반영하여 교통량 보간의 신뢰성을 높였다.

3. 배수등급

배수등급 변수는 국토교통부 브이월드(VWorld) 공간정보 플랫폼에서 제공하는 배수등급 데이터셋(shapefile)을 기반으로 구축하였다. 해당 데이터는 도로 구간별 배수 상태를 폴리곤(Polygon) 형태로 표현하고 있으며, 각 구간은 '매우불량', '불량', '약간불량', '약간양호', '양호', '매우양호', '기타'의 등급으로 구분되어 있다. 본 분석에서는 포트홀 보수 이력 데이터의 위·경도 정보를 기준으로, 각 포인트 좌표가 어느 배수등급 폴리곤 내에 포함되는지를 판별하여 공간 조인을 수행하였다. 이로써 개별 포트홀 발생 지점에 대응되는 배수등급 값을 추출할 수 있었다.

한편, 원자료 내 '기타'로 표기된 등급은 해석이 어렵고 분석상 누락값과 유사하게 기능하므로, 이를 보완하기 위해 K-최근접 이웃 분류기(K-Nearest Neighbors Classifier, K=3)를 적용하였다. 보간 시에는 동일 자치구 내에서 지리적으로 인접한 세 좌표의 배수등급을 참조하여 최빈값(mode) 기준으로 대체함으로써, 지리적 일관성과 현실성을 동시에 확보하였다.

4. 건물연령

도로 노후도를 간접적으로 평가하기 위한 대체 지표로 건축물 평균 연령을 활용하였다. 이는 포트홀 발생이 주변 기반시설의 물리적 노후도와 밀접한 상관관계를 가진다는 가정에

기반한 접근이며, 직접적인 도로 포장 연차 정보가 부재한 상황에서의 현실적인 대안으로 적용되었다.

건물연령 변수는 국토교통부 브이월드(VWorld) 공간정보 플랫폼에서 제공하는 건축물 연령정보(shapefile) 데이터셋을 기반으로 구성하였다. 해당 데이터는 건축물대장 상의 사용승인일자를 기준으로 건물의 준공 연도를 산출하고, 이를 5년 단위의 연령대로 분류한 후, 각 건축물의 위치가 속한 행정동 단위로 평균을 산출하였다. 이후 포트홀 보수 이력 데이터셋과의 통합을 위해, '자치구' 및 '행정동' 컬럼을 기준으로 조인 작업을 수행하였다. 이를 통해 개별 포트홀 발생 지점에 대해 해당 지역의 **건축물 평균 연령(=간접적 도로 노후도)**을 효과적으로 매칭할 수 있도록 처리하였다.

5. 토양경사도

국토교통부 브이월드 공간정보에서 토양경사도 데이터셋을 다운로드하였다. shp 파일을 로드하여, 기존의 포트홀 보수 데이터셋과 공간 조인을 하였다. 즉, 포트홀 보수 데이터셋에서 위도, 경도로 이루어진 point 좌표가 토양경사도 데이터셋의 폴리곤 좌표 안에 포함되면 조인하였다. 이를 통해 포트홀 발생 위치별 토양의 경사도가 파악 가능하다. 경사도 값은 '0-2%', '2-7%', '7-15%', '15-30%', '30-60%', '60-100%', '기타'로 구성되어 있고, 이 중 기타는 KNeighborsClassifier 알고리즘을 사용하여 위도·경도, 그리고 자치구 기준으로 가장 가까운 3개의 위치 중 토양경사도 최빈값으로 대체했다.

6. 인구 수

KOSIS 국가통계포털에서 서울특별시 자치구별 인구 수 자료를 다운로드하였다. KNeighborsClassifier 알고리즘을 사용하여 위도·경도 상 가장 가까운 3개의 위치 중 배수등급 최빈값으로 대체했다. 포트홀 보수 데이터셋에서 예측한 '발생일' 범위에 따라 2022년 12월부터 2024년 01월까지의 인구 수를 활용했으며, 자치구별로 평균을 계산하여 자치구 기준으로 포트홀 보수 데이터셋과 조인하였다.

7. 강수량

기상청의 기상자료 개방포털에서 서울특별시 자치구별 일일 강수량 자료를 다운로드하였다. 결측값의 경우, 주변 데이터 중 비어있지 않은 데이터를 가져와 보간하였으며, 보간에 실패한 데이터의 경우 강수량이 없는 것으로 판단하여 0으로 보간하였다. 이후, 포트홀 데이터에서의 '등록번호' 기준으로 '발생일' 열을 생성하고, 발생일을 기준으로 하여 그 전 1년동안의 누적 강수량 값을 계산한 뒤 기존 포트홀 보수 데이터셋과 조인하였다.

8. 습도

습도 변수는 기상청의 기상자료 개방포털을 통해 수집된 서울특별시 자치구별 시간 단위 습도 데이터를 기반으로 구축하였다. 데이터 전처리 과정에서 일부 시점 및 지역의 습도 데이터에 결측치가 존재하였으며, 이에 대해서는 시계열 기준으로 인접한 시간대의 유효 값을 활용하여 **선형 보간(linear interpolation)**을 수행하였다. 보간이 불가능한 경우에는 해당 자치구의 1년치 평균 습도값으로 대체하였다. 포트홀 발생지점별 습도 정보를 매핑하기 위해, 포트홀 보수 데이터셋의 고유 식별자인 '등록번호'를 기준으로 '발생일' 컬럼을 생성하고, 각 발생일을 기준으로 이전 1년간의 시간 단위 습도 데이터를 평균 산출하였다. 이렇게 산출된 연평균 습도값은 '평균 습도' 열로 명명되어 기존 포트홀 보수 데이터셋에 병합(조인)되었다.

특기할 점은, 누적 강수량과 달리 습도는 상대적인 백분율(예: 23%)로 표현되기 때문에, 단순 누적 방식으로는 '17000%'와 같은 비직관적이고 왜곡된 결과가 도출될 수 있다. 따라서 본 변수는 누적이 아닌 산술 평균 방식을 적용하여 수치 해석의 직관성과 비교 가능성을 확보하였다.

9. 기온

기상청의 기상자료 개방포털에서 서울특별시 자치구별 일일 기온 자료를 다운로드하였다. 원 데이터에는 일부 날짜 및 지역 단위의 결측치가 존재하였으며, 이에 대해서는 인접한 날짜의 유효 데이터를 활용하여 시계열 보간을 우선 적용하였다. 보간이 불가능한 경우에는 해당 자치구의 연간 평균 기온 값으로 대체하여 데이터의 연속성과 대표성을 확보하였다. 포트홀 발생 시점과의 연계 분석을 위해 기온 변수는 두 가지 방식으로 처리되었다. 첫 번째는, 발생일 기준 직전 1년 동안의 최고기온과 최저기온 간의 차이를 산출한 '1년 기온차' 변수로, 이는 여름철 고온과 겨울철 한파가 반복되는 계절 간 온도 격차가 도로 포장재의 수축·팽창에 미치는 영향을 반영하기 위한 것이다. 해당 변수는 동결-해빙 주기로 인한 도로 물성 변화와 구조적 취약성 심화를 설명할 수 있는 중요한 지표로 판단된다. 두 번째는, 발생일 기준 직전 1개월 간의 일별 기온 차(최고–최저)의 평균값을 산출한 '평균 일교차' 변수이다. 이는 포트홀이 실제로 발생한 시기 전후의 단기적 기온 변동성이 국지적인 도로 손상에 미치는 영향을 반영하기 위해 도입되었다. 이와 같이 구성된 두 개의 기온 변수는 각각 '1년 기온차', '평균 일교차'로 명명되어, 포트홀 보수 이력 데이터와 발생일 기준으로 조인되었다.

최종적으로, 완성된 데이터셋은 다음과 같다.

	차선수	승용차	버스	트럭	총교통량	중대형차량 교통량	포트홀 존재여부	평균_건물연령	인구 수	누적 강수량	평균 습도	1년 기온차	평균 일교차	배수등급	경사도
0	1.0	15805.000000	687.000000	1722.000000	18214.000000	0.132261	0	30.813623	237377.882353	1198.392531	61.716769	41.984124	9.605253	4.0	2.0
1	1.0	15805.000000	687.000000	1722.000000	18214.000000	0.132261	0	30.813623	237377.882353	1198.392531	61.716769	41.984124	9.605253	4.0	4.0
2	1.0	17325.560756	350.444889	1556.192196	19232.197841	0.099138	1	30.813623	237377.882353	1354.779114	61.372140	45.300000	9.564516	4.0	3.0
3	1.0	17325.560756	350.444889	1556.192196	19232.197841	0.099138	1	30.813623	237377.882353	1354.779114	61.676781	45.300000	9.867742	3.0	1.0
4	2.0	9638.000000	83.000000	710.000000	10431.000000	0.076023	1	30.813623	237377.882353	1354.779114	61.287654	45.300000	9.506452	4.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
15476	6.0	20971.000000	1022.000000	2866.000000	24859.000000	0.156402	1	34.518054	187936.000000	1358.948734	57.691826	45.000000	8.725806	2.0	0.0
15477	6.0	23283.000000	1980.000000	3129.000000	28392.000000	0.179945	0	34.518054	187936.000000	1202.080838	57.757131	41.658824	8.442201	2.0	0.0
15478	6.0	23332.000000	1573.000000	2880.000000	27785.000000	0.160266	0	34.518054	187936.000000	1202.080838	57.757131	41.658824	8.442201	2.0	0.0
15479	7.0	25349.000000	1770.000000	3119.000000	30238.000000	0.161684	0	34.518054	187936.000000	1202.080838	57.757131	41.658824	8.442201	1.0	0.0
15480	7.0	25349.000000	1770.000000	3119.000000	30238.000000	0.161684	1	34.518054	187936.000000	1358.948734	57.035419	44.300000	7.580645	1.0	0.0

15481 rows × 15 columns

2-2. 모델링 과정

도로 및 환경 데이터를 활용하여 포트홀 발생 여부를 예측하는 분류 모델을 구축했다. 전체적인 모델링 과정은 데이터 전처리, 불균형 보정, 학습-검증 데이터 분할, 그리고 XGBoost 기반 분류 모델 학습의 흐름으로 구성된다.

1. 데이터 전처리

먼저, 2-1에서 구축한 데이터셋을 불러온 후, 포트홀의 존재 여부를 이진 변수로 정의하였다. 포트홀이 발생한 적이 없는 경우 '포트홀 존재여부'를 0으로 설정하고, 나머지는 1로 설정하여 타겟 변수 y를 구성했다.

데이터셋에서 포트홀 예측에 불필요한 변수(자치구, 행정동, 경도, 위도, 등록번호, 발생일)를 제거하였다. 또한, 기후 관련 변수(누적 강수량, 평균 습도, 1년 기온차, 평균 일교차)를 제거하여 입력 변수 x를 구성하였다.

모델 설계 과정에서 기후 관련 변수는 포트홀 발생과의 상관성이 높아 보였으나, 최종 모델에서는 제거하였는데, 이는 다음 이유에 기반한다.

첫째, 앞으로 기온 및 강수량 등의 기후 요소를 예측하는 것이 매우 불확실해질 가능성이 크다고 판단하였다. '침수흔적정보를 활용한 기후변화 피해비용 추정 연구'에 따르면, 지난 106년 동안 강한 강수는 증가하고 약한 강수는 감소하는 강수의 양극화 현상이 나타났으며, 1일, 5일 최대강수량과 95th 퍼센타일, 99th 퍼센타일의 강수일수가 다소 증가한 것으로 나타났다. 이러한 극한 이상기후 현상이 미래에는 더 빈번하게 발생할 것으로 예측되고 있다. 따라서 특정한 이상기후 상황에 과도하게 종속된 모델보다는, 기후 변동성과 무관하게 작동 가능한 강건한 모델을 설계하는 것이 더 유의미하다고 판단하였다. 게다가 필요 시, 기후 API를 통해 보완할 수 있는 구조이기 때문에, 기후 변수를 제거하는 결정이 분석력에 치명적인 손실로 이어지지 않는다는 점도 고려되었다.

둘째, 기후 요소는 도로 관리자가 통제할 수 없는 변수라는 한계가 있다. 본 모델의 핵심 목적 중 하나는 예측 결과를 바탕으로 도로 관리자에게 실행 가능한 관리 방안을 제시하는 것이다. 이후 제시될 '예측 리포트'에서는 도로 관리자에게 포트홀 위험이 높은 구간에 대한 사전 조치 방향을 안내하는데, 이때 관리자가 직접 조작하거나 개선할 수 없는 기후 변수보다는 차선 수, 교통량, 경사도, 배수등급 등 통제 가능한 물리적, 환경적 요인 위주

로 분석 결과를 제시하는 것이 실효성이 높다. 이처럼 모델은 단순 예측 정확도를 넘어서, 현장 적용 가능성과 실천 가능성을 중요하게 고려한 방향으로 구성되었다.

2. 학습 데이터 구성 및 불균형 보정

전체 데이터는 학습용(80%)과 테스트용(20%)로 분리하였으며, 이때 클래스 비율을 유지하기 위해 stratify=y 옵션을 사용했다. 이후 학습 데이터에 대해 SMOTE를 적용하여 포트홀 발생 클래스(1)에 대한 샘플을 증강하였다. SMOTE는 소수 클래스의 대표성을 보완해 모델의 예측 편향을 줄이는 데 효과적이다.

3. 모델 학습

기본 모델로는 XGBoostClassifier가 사용되었으며, 모델 학습 후에는 분류 평가 지표들을 활용해 성능을 측정하였다: 정확도(Accuracy), 정밀도(Precision), 재현율(Recall), F1 스코어, ROC AUC Score, FNR (False Negative Rate). 여기서 FNR은 실제로 포트홀이 발생했음에도 이를 감지하지 못한 비율이다.

4. 하이퍼파라미터 튜닝

모델의 성능을 극대화하기 위해 RandomizedSearchCV를 활용한 하이퍼파라미터 튜닝이 진행되었다. 튜닝 대상 파라미터는 다음과 같다:

- n_estimators : 트리 개수
- learning_rate : 학습률
- max_depth : 트리 최대 깊이
- min_child_weight : 최소 자식 노드 가중치
- gamma : 분할 최소 손실 감소
- subsample : 샘플 비율
- colsample_bytree : 트리 분할 시 사용할 피처 비율
- scale_pos_weight : 클래스 불균형을 반영하는 가중치

이러한 튜닝 과정은 총 1000회의 랜덤 탐색으로 이루어졌으며, 최적의 파라미터 조합을 도출하여 최종 모델 성능을 개선하였다.

2-3. 모델 해석 및 평가

2.3.1 모델 해석

XGBoost의 feature_importances_ 속성을 활용하여 도출한 주요 변수의 영향력은 다음과 같다.

순위	Feature	중요도
1	총교통량	0.147
2	차선수	0.128
3	인구수	0.121
4	트럭 교통량	0.108
5	평균_건물연령	0.088
6	버스 교통량	0.087
7	승용차 교통량	0.083
8	경사도	0.080
9	배수등급	0.080
10	중대형차량 교통량	0.079

해석 결과, 총교통량이 포트홀 발생에 가장 큰 영향을 미치는 변수로 확인되었다. 이는 차량 통행이 많은 도로일수록 포장 손상 가능성이 높기 때문으로 분석된다. 차선 수, 인구 수, 트럭 및 대형차량 통행량 역시 높은 중요도를 보이며, 도로 하중 및 주변 도시 밀도와 관련된 요인들이 포트홀 발생의 핵심 결정 변수로 작용하고 있음을 시사한다. 또한, 평균 건물 연령과 경사도, 배수등급 등도 도로 주변 환경 및 배수 성능과 관련된 변수로, 물리적 취약성과 구조적 특성을 반영한다.

2.3.2 모델 성능

모델 성능 평가는 테스트 데이터셋을 대상으로 수행되었다. 모델 성능 지표는 재현율, F1 score, FNR이 사용되었고, 모델의 도입효과 및 비용을 확인하기 위해 ROI(투자 수익률)를 사용하였다.

모델의 혼동행렬은 아래 표와 같다.

	실제: 포트홀 없음 (0)	실제: 포트홀 있음 (1)
예측: 포트홀 없음 (0)	1,176 (True Negative)	244 (False Negative)
예측: 포트홀 있음 (1)	520 (False Positive)	1,156 (True Positive)

1. **재현율 (Recall): 82.57%**

   실제로 포트홀이 발생한 사례 중 약 83%를 모델이 정확히 포착하였다. 이는 포트홀 발생을 놓치지 않는 감지 능력이 우수하다는 의미로, 공공 안전 및 사고 예방 목적의 활용 측면에서 중요한 지표다.

2. **F1 Score: 0.752**

   정밀도와 재현율의 조화 평균인 F1 점수는 0.75로, 클래스 불균형 상황에서도 안정적인 성능을 보였음을 의미한다.

3. **False Negative Rate (FNR): 17.43%**

   실제 포트홀이 존재했음에도 불구하고 이를 탐지하지 못한 비율은 약 17%이다. 본 분석에서는 False Positive Rate (FPR) 보다 FNR을 더 중요하게 고려하였다. 이는 후속 투자수익률 분석에서 제시될 바와 같이, 포트홀을 탐지하지 못해 사고로 이어지는 경우의 비용이, 포트홀이 없음에도 정비를 수행하여 발생하는 불필요한 정비 비용보다 현저히 크다는 판단에 기반한다.

4. **ROI (투자수익률)**

   모델 도입의 실질적 타당성을 검토함에 있어, 단순한 예측 성능 외에도 경제적 효과 대비 비용을 고려하는 것이 필수적이다. 아무리 예측력이 뛰어난 모델이라 하더라도, 실제 현장에 적용했을 때 비용이 편익을 초과한다면 실무적 활용 가능성은 낮다. 이에 따라 본 보고서에서는 모델 도입의 수익성을 평가하기 위해 투자수익률(ROI: Return on Investment) 지

표를 활용하였다. ROI는 다음과 같은 수식으로 산출된다.

$$\text{ROI} = \left(\frac{\text{절감액} - \text{추가비용}}{\text{추가비용}}\right) \times 100$$

사고 1건당 평균 손해액(절감편익 기준)

고속도로 기준 2020년부터 2024년까지의 총 배상 금액은 136억 원, 사고 건수는 22,692건으로, 건당 평균 배상액은 약 599,000원이었다. 또한, 언론사 포트홀 사고 보고서에 따르면, 2017년 기준 자차보험의 평균 지급액은 1,160,000원이었다. 두 수치를 종합적으로 고려해, 본 분석에서는 사고 1건당 평균 배상액을 880,000원으로 설정하였다.[vii]

정비 1건당 평균 비용(FP 비용)

고속도로 보수 데이터를 보면, 2019년부터 2023년까지 총 보수비용은 약 68억 9,500만 원, 총 보수건수는 12,504건으로, 1건당 평균 정비비용은 약 551,000원으로 추정된다.[viii] 지방소도시(2024년 제주도 기준)에서는 보수 1건당 아스콘 20kg × 233포대 사용으로 345,000원이 소요되었으며[ix], 이를 감안하여 종합적으로 건당 평균 정비비용은 약 445,000원으로 설정하였다.

모델의 예측 혼동행렬 값을 기준으로 전체 사례 수를 1,000건이라 설정할 때, True Positive(TP)는 0.826×1000=826건, False Positive(FP)는 826×0.450≈372건이다. 이에 따라 편익 및 비용과 ROI는 다음과 같이 산출된다.

- 절감액 (TP로 인한 사고 탐지 및 배상비용 절감): 826×880,000원=약725,000,000원
- 추가비용 (FP로 인한 불필요한 정비 비용): 372×445,000원=약165,540,000원
- ROI = (725,000,000-165,540,000)/165,540,000×100 ≈ 338%

즉, 본 포트홀 예측 모델을 적용할 경우 False Positive로 인해 발생하는 추가적인 정비 비용보다 포트홀로 인한 실제 사고를 사전에 탐지하여 절감할 수 있는 배상 비용이 훨씬 더 큰 것으로 나타났으며, ROI 338%의 높은 투자 수익률을 기록하였다. 이는 본 모델이 단순한 예측 정확도뿐 아니라 경제적 비용 효율성 측면에서도 뛰어난 성과를 보였음을 시사한다. 따라서 본 모델은 실무 현장에 적용할 충분한 경제적 타당성을 확보한 것으로 평가할 수 있다.

2.3.3 모델 활용

개별 위치를 대상으로 학습된 위 모델은 도로 단위로도 확장하여 활용될 수 있다. '차선수', '총교통량', '중대형차량 교통량'의 세 가지 도로 특성을 기준으로 데이터를 그룹화한 후, 각 그룹에 대해 실제 포트홀이 발생한 총합, 예측된 포트홀 수의 총합, 해당 그룹에 포함된 데이터 수를 계산한다. 이 과정을 통해 도로별 포트홀 발생 위험률과 예측 성능을 비교할 수 있다.

그 결과는 다음과 같다.

인덱스	차선수	총 교통량	중대형차량 교통량	actual_count	predicted_sum	error
0	1	735	1.0	729	726.824890	-2.175110
1	2	1952	2.0	24	24.309074	0.309074
2	2	1840	2.0	22	22.380312	0.380312
3	6	5665	6.0	20	20.043438	0.043438
4	6	5559	6.0	19	19.343739	0.343739
5	2	2140	2.0	17	16.718311	-0.281689
6	3	2877	3.0	17	16.907480	-0.092520

위의 표는 도로별로 실제 포트홀 발생갯수와 예측값을 기반으로 산출된 값을 비교한다. 이를 통해 포트홀 예측 정확도 낮은 도로 유형을 선별하여 모델의 개선 대상을 식별할 수 있다. 뿐 만 아니라, 이후의 각 도로별 포트홀 발생 리스크를 측정할 때 predicted_sum 값을 이용하여 발생빈도로 사용할 수 있다.

3. 보험사의 포트홀 리스크 측정 및 관리방안

3.1 포트홀 배상 책임리스크 측정

보험사의 포트홀 배상 책임리스크는 다음의 손실빈도와 손실심도의 곱으로 정의된다.

$$\text{리스크}(R_i) = \text{발생빈도}(\lambda_i) \times \text{배상전환율}(\pi_i) \times \text{예상청구액}(\mu_i)$$

여기서 손실빈도는 앞선 예측모델로 산정된 도로별 포트홀 예상 발생빈도와 도로구분별 포트홀 발생 대비 배상 전환율의 결합으로 추정하였으며, 손실심도는 도로구분별 평균 청구액으로 정의

하였다.

이때, 각 도로 유형별 배상전환율과 평균 청구액이 상이하다는 것을 확인하여, 각 도로유형별 리스크를 측정하고자 대표적으로 고속도로와 지자체 관리도로의 배상 실적과 청구액을 분석하였다.

3.1.1 고속도로 구간 배상 전환율 및 평균 청구액 분석

연도	배상 전환율	평균 청구액(백만원)
2020	17.99%	1.79
2021	28.66%	1.59
2022	38.77%	2.01
2023	44.65%	1.72
2024	71.59%	1.16

위의 표는 한국도로공사가 제공한 2020-2024년 고속도로 노선별 포트홀 발생건수, 배상건수 및 평균 청구액을 나타낸다. 분석 결과, 고속도로의 포트홀 발생 대비 배상 전환율은 2020년 18.3%에서 2024년 71.6%로 지속적인 증가 추세(연평균 34.7%p 상승)를 보였는데, 이는 운전자의 손해 인식 수준 향상, 배상 프로세스 간소화 및 블랙박스 장착률 증가에 따른 손해 입증 용이성 개선 때문으로 해석된다.

반면 평균 청구액은 2022년 201.3만원으로 최고치를 기록한 뒤, 2024년 115.9만원으로 하락세를 보였는데, 이는 경미한 손상에 대한 청구 증가와 전자 청구 시스템 도입으로 인한 효율성 제고에 따른 결과로 분석된다.

따라서, 특정 고속도로 i에서의 포트홀 배상 책임 리스크는 다음과 같은 수식을 통해 근사적으로 산정할 수 있다:

$$\text{리스크}\left(R_{\text{고속도로}\,i}\right) = \text{발생빈도}\left(\lambda_{\text{고속도로}\,i}\right) \times 0.7159\left(\pi_{\text{고속도로}}\right) \times 116\left(\mu_{\text{고속도로}}\right)$$

3.1.2 지자체 관리도로 및 시군구도의 배상 실적 분석

지자체 관리도로는 고속도로와 달리 배상 실적이 현저히 낮다. 서울시 자동차전용도로의 경우 최근 3년간 포트홀 발생 18,820건 중 배상 청구 건수는 376건(2.0%), 승인 건수는 108건(0.57%)으로 총 지급액은 3,271만원(건당 평균 30.3만원)에 그쳤다. 용인시는 2021-2023년 11,240건 중 배상 전환율 7.3%, 평균 청구액 89만원으로 나타났고, 전북 등 지방 소도시는 2016-2018년 72,838건 중 전환율이 불과 0.07%였다. 즉 결론적으로 위 값을 가중평균한 0.96을 지자체 도로의 배상 전환율, 79.4만원을 평균 청구액으로 설정할 수 있다.[xxi]

이러한 격차는 관리주체의 조직적·기술적 차이에서 기인하는 것으로 판단된다. 한국도로공사는 단일 기관이 전 구간을 체계적으로 관리하고 CCTV, ITS 등 표준화된 인프라를 구축하여 손해 입증

이 용이하지만, 지자체 도로는 관리기관의 분산과 낮은 CCTV 설치율 등으로 인해 입증책임이 운전자에게 전가되어 청구 포기 및 배상 기각 비율이 높기 때문이다.

위와 동일하게, 특정 지자체 도로 j의 배상 책임 리스크는 아래의 수식으로 산정된다.

$$\text{리스크}\left(R_{\text{지자체도로 }j}\right) = \text{발생빈도}\left(\lambda_{\text{지자체도로 }j}\right) \times 0.01\left(\pi_{\text{지자체도로}}\right) \times 79.4\left(\mu_{\text{지자체도로}}\right)$$

이때 유의할 사항은, 상기 리스크 산정식은 각 도로별 배상 전환율과 평균 청구액의 수준에 민감하게 반응하므로, 보험사는 해당 지표에 대한 정기적인 조사 및 체계적인 데이터 축적을 통해 리스크 평가의 정밀도를 높일 수 있다.

3.2 포트홀 배상 책임리스크 관리방안 및 보험상품 개발

보험사는 3-1에서 산출한 도로별 포트홀 배상 책임 리스크를 기반으로 다양한 리스크관리 전략 수립 및 상품개발이 가능하다.

3.2.1 영조물손해배상보험 도로구간별 차등요율 적용

현재 전국 대부분의 지자체 및 공공기관은 행정적 효율성과 재정적 안정성을 위해 영조물손해배상보험에 가입하고 있다. 그러나 일반적으로 요율이 시설물 유형과 규모만을 기준으로 일률적으로 산정되어 있어 포트홀 발생 리스크를 충분히 반영하지 못한다. 예를 들어, 도로 포트홀 발생에 대한 영조물손해배상보험의 경우 "도로환경"이라는 시설물로 구분되어, 모든 도로에 동등하게 1인당 보상한도와 보험료가 설정되어 있다. 이처럼 각 도로별 위험도를 고려하지 않은 경우, 보험료의 효율적인 배분과 보험의 지속가능성을 감소시킬 수 있다.[xii]

따라서 본 보고서는 3-1에서 제시한 "포트홀 배상책임 리스크"를 기반으로 도로구간별 리스크를 각각 측정하고 이를 보험료 요율 산정에 반영한 차등 요율 체계를 제안한다. 도로별 위험도에 따른 보험료 차등화는 관리기관의 관리 책임성을 높이고 도덕적 해이를 예방한다. 또한 장기적으로 포트홀 발생 및 사고율을 감소시켜 보험사의 손해율을 낮추고, 보험료 인상 압력을 완화하는 등의 이점을 기대할 수 있다.

3.2.2 XAI 기반 포트홀 발생 예측 리포트 제공 및 위험관리 인센티브 보험

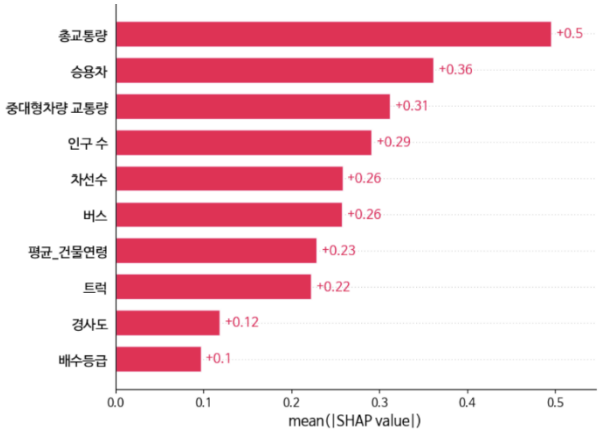
보험사는 자체 개발한 포트홀 발생 예측 모델을 기상 및 교통량 API와 연동하여, 보험에 가입한 도로 관리기관에게 실시간 예측 리포트를 제공할 수 있다. 특히, XAI(설명가능한 인공지능) 기술을 적용함으로써 각 도로 구간의 위험도 및 주요 원인 요인을 명확히 도출할 수 있으며, 이를 통해 도로 관리기관은 구체적인 개선 방향성과 전략적 인사이트를 확보할 수 있다. 이러한 리포트는

단순한 예측을 넘어, 도로 유지보수의 우선순위 결정 및 예산 배분의 효율성 향상에 기여할 수 있다.

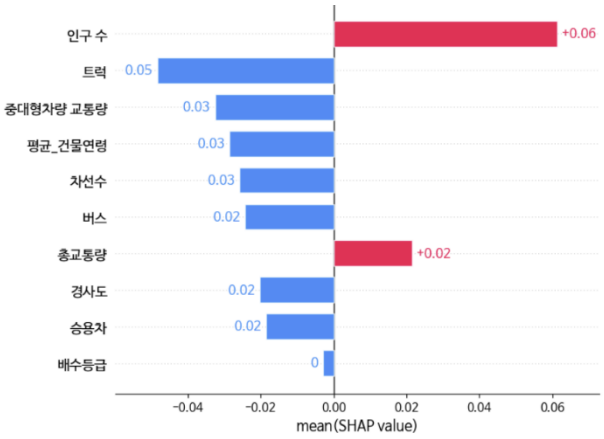
리포트는 XAI 기법을 주로 활용하며, 크게 네 가지 핵심 기능을 중심으로 설계하였다: 예측에 영향을 준 주요 변수 파악, 해당 지자체의 예측 결과 확인, 예측 결과를 결정지은 핵심 요인 분석, 그리고 예측 결과를 바꾸기 위한 개선 방향 제시. 각 기능들을 예시와 함께 살펴보기로 한다.

1. 예측에 영향을 준 주요 변수 파악

   첫 번째 기능은 모델이 예측 결과를 도출하는 과정에서 어떤 변수들이 핵심적인 기여를 했는지를 시각적으로 파악하는 것이다. 아래 그래프들의 변수별 값들은 개별 데이터를 대상으로 분석이 이루어지며, XAI 기법 중 SHAP을 활용하였다.



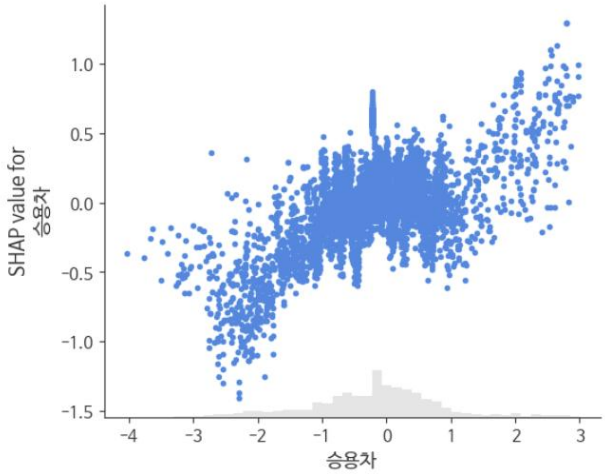
   위 그래프는 전체 데이터에서 변수들이 예측에 얼마나 영향을 주었는지를 시각화한 것이다. 변수별로 평균적인 기여도를 나타내며, 방향과 무관한 고유한 영향력을 나타낸다. 이를 통해 모델이 포트홀 발생 여부를 판단하는 데 있어 주로 어떤 요인에 기반하고 있는지를 확인할 수 있다.



   해당 그래프는 각 변수의 평균 SHAP 값을 기반으로 산출된 것으로, 변수별 예측 기여도의 방향성과 크기를 직관적으로 확인할 수 있도록 구성되어 있다. 시각화에서 빨간색으로 표시된 변수는 포트홀 발생(예측값 1) 방향으로 예측에 기여한 항목이며, 반대로 파란색은

포트홀 미발생(예측값 0) 방향으로 영향을 미친 변수들을 나타낸다.

SHAP 값의 크기와 순위는 분석에 사용된 데이터셋의 특성에 따라 달라질 수 있으므로, 분석 목적에 따라 다양한 조건의 데이터를 적용하여 비교 분석이 가능하다. 예를 들어, 특정 자치구의 과거 데이터를 입력하여 분석을 수행할 경우, 해당 지역에서는 어떤 요인이 포트홀 발생 예측에 가장 큰 영향을 미치는지를 지역 맞춤형으로 파악할 수 있으며, 이는 실무적으로 지자체 단위의 리스크 인자 진단 및 대응 전략 수립에 활용될 수 있다.



마지막으로, 특정 변수의 값 변화가 예측 결과에 미치는 영향을 시각화한 그래프를 통해 변수의 예측 민감도 및 영향력의 방향성을 확인할 수 있다. 사용자가 분석 대상 변수를 지정하면, 해당 변수의 값이 변화함에 따라 모델의 예측값(SHAP Value)이 어떻게 반응하는지를 정량적으로 파악할 수 있다.

예를 들어, '승용차' 변수의 경우, 변수 값이 증가할수록 y축의 SHAP 값도 함께 상승하는 패턴을 보이는데, 이는 해당 변수의 값이 커질수록 포트홀 발생(예측값 1) 가능성을 높이는 방향으로 모델에 영향을 미친다는 것을 의미한다. 이러한 시각화는 단순한 변수 중요도를 넘어, 예측 경향의 형태(예: 선형성, 임계값, 비선형 구간 등)를 파악할 수 있어, 실무적으로는 정책 개입의 임계 범위 설정이나 요율 조정 기준 마련 등에 유용하게 활용될 수 있다.

2. 해당 지자체의 예측 결과 확인

	날짜	주소	예측	예측 확률
0	2024-07-28	서울특별시 강남구 대치동 507	1	0.628422
1	2023-06-23	서울특별시 동대문구 제기동 137-418	1	0.797912
2	2021-10-29	서울특별시 서초구 방배동 756-4	0	0.136294
3	2024-01-05	서울특별시 용산구 동빙고동 90-1	0	0.114550
4	2022-04-29	서울특별시 성북구 보문동5가 235	0	0.397331

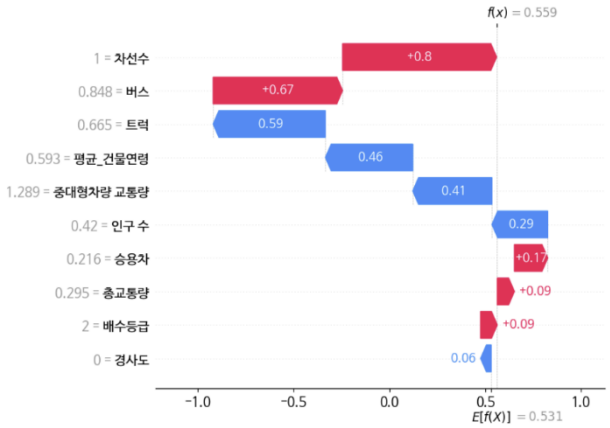
해당 기능은 사용자가 입력한 주소 및 날짜 정보를 기반으로, 해당 지점·시점에서의 포트홀 발생 예측 결과를 직관적으로 확인할 수 있도록 설계되었다. 예측 결과는 표 형태로 출력되며, **이진 예측값(발생 여부)**과 함께 **발생 확률(예측 확률)**을 병행 제시하여, 사용자가 포트홀 위험 수준을 수치적으로 이해할 수 있도록 구성되어 있다.

특히 사용 편의성을 높이기 위해, 사용자가 단순히 주소와 날짜만 입력하면, 해당 시점·지역에 필요한 입력 변수들(예: 승용차 및 버스 교통량, 평균 건물 연령, 경사도, 인구 수 등)을 자동으로 불러와 모델의 입력값으로 구성하고, 예측 결과를 실시간으로 산출하도록 구현하였다.

3. 예측 결과를 결정지은 핵심 요인 분석

이 기능은 특정 지점과 시점에 대한 예측 결과가 모델 내부에서 어떻게 도출되었는지를 설명하는 것으로, 앞서 제시한 리포트의 '예측 결과 확인(2번 기능)'과 연계하여 활용된다. 사용자가 입력한 주소와 날짜 정보를 바탕으로 모델이 왜 해당 지점에서 포트홀이 발생할 것으로(또는 발생하지 않을 것으로) 판단했는지를 변수 수준에서 세부적으로 추적할 수 있도록 구성하였다.

이 분석은 1번 기능과 유사하게 SHAP 값을 기반으로 변수 중요도를 시각화하지만, 여러 샘플에 대한 평균적 영향력을 보여주는 1번과 달리, 본 기능은 특정 개별 샘플에 대한 변수 기여도를 정밀하게 파악하는 데 목적이 있다.



시각화 그래프는 SHAP Value의 누적 구조를 통해, 예측값이 모델의 평균 예측값(E[f(X)])로부터 어떻게 변화하여 최종 확률에 도달했는지를 단계별로 설명한다. 그래프는 하단에서부터 시작하여, 각 변수의 기여도가 양(+) 방향이면 예측을 '포트홀 발생(y=1)' 쪽으로, 음(–) 방향이면 '미발생(y=0)' 쪽으로 이끈다는 방식으로 누적된다.

예를 들어, 특정 샘플에 대해 '경사도' 변수가 예측값을 y=0 방향으로 0.06만큼 이동시킨 후, '배수등급' 변수가 다시 y=1 방향으로 0.09만큼 이동시킴으로써, 전체적인 예측값은 0.559에 도달하게 된다. 이 예측값은 로그 오즈(log-odds) 공간에서 계산된 확률값으로, 0을 기준으로 양(+)의 값을 가지면 포트홀 발생(y=1), 음(–)의 값을 가지면 포트홀 미발생

(y=0)으로 해석된다.

따라서 본 기능은 모델이 특정 사례에 대해 어떤 변수의 어떤 값 때문에 발생 여부를 예측했는지를 시각적으로 명확하게 보여주는 결정 해석 기능으로서, 실무에서의 설명 가능성 확보, 위험요인 피드백, 그리고 리스크 대응방안 제안 등 다양한 활용이 가능하다.

4. 예측 결과를 바꾸기 위한 개선 방향 제시

	차선수	승용차	버스	트럭	총교통량	중대형차량 교통량	평균_건물연령	인구 수	배수등급	경사도	발생여부
0	2.0	19029.999986	259.999991	1220.999997	20510.999911	0.072206	30.813624	237377.882668	양호	7-15%	1

	차선수	승용차	버스	트럭	총교통량	중대형차량 교통량	평균_건물연령	인구 수	배수등급	경사도	발생여부
0	2.0	13069.474317	289.367623	1221.000001	20511.000001	0.072206	30.813624	237377.882354	양호	7-15%	0

본 기능은 모델이 특정 데이터에 대해 포트홀 발생으로 예측한 결과를 미발생으로 전환하기 위해 필요한 변수 조정 방향 및 수준을 제시하는 것으로, XAI 기법 중 하나인 LIME(Local Interpretable Model-agnostic Explanations)을 활용하였다. 이는 단순한 예측을 넘어, 리스크 완화를 위한 개입 시나리오를 제시할 수 있다는 점에서 정책적 활용도가 높은 기능이다.

예시로 제시된 두 건의 샘플 중 첫 번째는 실제 데이터를 기반으로 모델이 '포트홀 발생(발생여부 = 1)'로 예측한 사례이며, 두 번째는 동일한 데이터에서 일부 변수 값만을 미세하게 조정하여 예측 결과를 '포트홀 미발생(발생여부 = 0)'로 전환한 사례이다. 이 과정에서 가장 예측에 큰 영향을 미친 변수로는 '승용차 교통량'이 확인되었으며, 해당 값이 약 6,000대 감소했을 때 예측 결과가 반전되었다. 이는 모델 관점에서 해당 변수가 포트홀 발생에 중요한 기여 요인으로 작용하고 있음을 시사한다.

그러나 이러한 결과를 단편적으로 해석하여 "승용차 교통량을 단순히 줄여야 한다"는 식의 개선책을 제시하는 것은 현실적 실행 가능성과 정책 적합성 측면에서 한계가 있다. 보다 실무적으로는 교통량을 직접 줄이는 대신, 통행 분산 정책, 도로 포장 강도 강화, 혹은 배수 시스템 개선을 통한 리스크 완화 전략이 병행되어야 한다. 즉, 본 기능은 단순한 변수 조정 시뮬레이션을 넘어, 정책적 대안 수립을 위한 가이드라인 제공이라는 점에서 의미를 갖는다.

본 리포트는 지자체가 포트홀 발생을 효과적으로 관리할 수 있도록 현재 상황과 개선 방향을 제시하는 매뉴얼로서 작용할 수 있다. 파악하고자 하는 장소와 날짜를 입력하게 되면, 관련 데이터를 자동으로 확인할 수 있어 실질적인 정책 개선 방향 수립에 도움을 받을 수 있으며, 데이터를 기반으로 한 도로 유지보수 방안을 확립할 수 있다. 더불어, 보험사의 경우 해당 예측 리포트를 바탕으로 도로 관리기관이 위험 구간에 대한 개선 조치를 이행한 경우, 차년도 보험료 할인 등의 성과 기반 인센티브 제도를 마련할 수 있다. 이를 통해 도로 관리자의 능동적인 위험관리 행동을 유도하고, 포트홀 리스크에 대한 선제적 관리체계 구축을 지원함으로써 보험사와 도로관리기관 간의 위험공유 및 공동관리 모델을 실질적으로 구현할 수 있다.

3.2.3 일반 운전자 대상 포트홀 위험정보 제공 서비스

보험사는 포트홀 발생 예측 모델을 기반으로 운전자에게 고위험 도로 구간에 대한 정보를 실시간 제공함으로써 운전자의 사전적 대응을 유도하고 포트홀 사고 및 피해를 최소화할 수 있다. 예를 들어, 운전자의 이동 경로 중 예측 모델이 지목한 고위험 포트홀 구간이 있다면, 이를 운전자에게 위험구간 회피 경로안내나 서행 유도 메시지를 제공하는 방식을 고려할 수 있다. 뿐만 아니라 이러한 서비스는 보험사의 브랜드 이미지 향상 및 소비자 신뢰 제고에도 기여하여, 보험사의 마케팅 경쟁력을 강화하는 부가적 효과를 기대할 수 있다.

부록 : AI 사용 코드

1. 로지스틱 회귀 분석 python code

```
# 라이브러리 불러오기
import pickle
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf

# 데이터 불러오기
pothole = pd.read_csv("/content/drive/MyDrive/DATA_완선/pothole_예측용.csv")

# 포트홀 발생 여부 이진화 (1개 이상 발생 시 1)
pothole['포트홀 존재여부'] = (pothole['포트홀 갯수'] >= 1).astype(int)

# 변수명 영문화 및 일관성 정리
pothole.columns = [
    'index',                      # Unnamed: 0
    'district',                   # 자치구
    'neighborhood',               # 행정동
    'longitude',                  # 경도
    'latitude',                   # 위도
    'num_lanes',                  # 차선수
    'num_cars',                   # 승용차
    'num_buses',                  # 버스
    'num_trucks',                 # 트럭
    'total_traffic',              # 총교통량
    'mid_large_vehicle_traffic',  # 중대형차량 교통량
    'num_potholes',               # 포트홀 갯수
    'avg_building_age',           # 평균 건물연령
    'road_id',                    # 등록번호
    'occurrence_date',            # 발생일
    'population',                 # 인구 수
    'cumulative_rainfall',        # 누적 강수량
```

```
    'avg_humidity',              # 평균 습도
    'annual_temp_range',         # 연간 기온 변동폭
    'avg_daily_temp_range',      # 평균 일교차
    'drainage_grade',            # 배수등급
    'slope',                     # 경사도
    'district_enc',              # 자치구 인코딩
    'pothole_presence'           # 포트홀 존재여부 (목표변수)
]

# 주요 변수 정의
climate_vars = ['cumulative_rainfall', 'avg_humidity', 'annual_temp_range',
'avg_daily_temp_range']
model_vars = climate_vars + ['total_traffic', 'slope']

# 로지스틱 회귀 모델 적합
logit_model = smf.logit(
    'pothole_presence ~ ' + ' + '.join(climate_vars),
    data=pothole
).fit()

# 회귀 결과 출력
print("\n[Logistic Regression Results]")
print(logit_model.summary())

# 오즈비 및 95% 신뢰구간 계산
odds_ratios = pd.DataFrame({
    'OR': np.exp(logit_model.params),
    'Lower CI': np.exp(logit_model.conf_int()[0]),
    'Upper CI': np.exp(logit_model.conf_int()[1])
})

# 오즈비 출력
print("\n[Odds Ratios]")
print(odds_ratios)
```

2. 포트홀 발생 예측 모델 : Xgboost model code

```
import xgboost as xgb
from sklearn.metrics import classification_report, roc_auc_score
import pickle
import pandas as pd
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
# 시각화
import seaborn as sns
import matplotlib.pyplot as plt
!apt-get -qq install fonts-nanum

pothole_original     =     pd.read_csv('/content/drive/MyDrive/DATA_ᄋᆞᄂᆞᄉᆞᄀᆡᆼ
/dataset.csv')
pothole = pothole_original

# 경사도 '기타' 전처리
slope_encoding = {
    '0-2%': 0,
    '2-7%': 1,
    '7-15%': 2,
    '15-30%': 3,
    '30-60%': 4,
    '60-100%': 5}
pothole['경사도'] = pothole['경사도'].map(slope_encoding)

# 배수등급 '기타' 전처리
drain_encoding = {
    '매우양호': 5,
    '양호': 4,
    '약간양호': 3,
    '약간불량': 2,
    '불량': 1,
```

```
    '매우불량': 0}
pothole['배수등급'] = pothole['배수등급'].map(drain_encoding)

# '기타'인 데이터는 NaN으로 인코딩됨 !
pothole.isna().sum() # NaN - 2533

# 자치구도 반영하기!
gu_encoder = LabelEncoder()
pothole['자치구_enc'] = gu_encoder.fit_transform(pothole['자치구'])

# 채워넣을 데이터 분리
knn_train = pothole[pothole['배수등급'].notna()]
knn_test = pothole[pothole['배수등급'].isna()]


### 모델 - 경사도
# 모델 fitting
knn_slope = KNeighborsClassifier(n_neighbors = 3)
knn_slope.fit(knn_train[['경도', '위도', '자치구_enc']], knn_train['경사도'])
# prediction
pred_slope = knn_slope.predict(knn_test[['경도', '위도', '자치구_enc']])
pothole.loc[knn_test.index, '경사도'] = pred_slope

### 모델 - 배수등급
# 모델 fitting
knn_drain = KNeighborsClassifier(n_neighbors = 3)
knn_drain.fit(knn_train[['경도', '위도', '자치구_enc']], knn_train['배수등급'])
# prediction
pred_drain = knn_drain.predict(knn_test[['경도', '위도', '자치구_enc']])
pothole.loc[knn_test.index, '배수등급'] = pred_drain


scaling_vars = ['승용차', '버스', '트럭', '총교통량', '중대형차량 교통량', '평균_건
물연령',
               '누적 강수량', '평균 습도', '1년 기온차', '평균 일교차']

scaler = StandardScaler()
pothole[scaling_vars] = scaler.fit_transform(pothole[scaling_vars])


# 전처리 완료 후 저장
pothole.to_pickle('pothole_예측용.pickle')

with open(r"pothole_예측용.pickle","rb") as f:
    dataset = pickle.load(f)
dataset
```

```
dataset['포트홀 존재'] = 1  # 기본값 1로 초기화
dataset.loc[dataset['포트홀 갯수'] == 0, '포트홀 존재'] = 0

#x = dataset.drop(['자치구', '행정동', '경도', '위도', '등록번호', '발생일', '포트홀 갯수', '자치구_enc', '포트홀 존재'], axis=1)

# 기후 관련 컬럼 제거
x = dataset.drop(['자치구', '행정동', '경도', '위도', '등록번호', '발생일', '포트홀 갯수', '자치구_enc', '포트홀 존재',
                  '누적 강수량', '평균 습도', '1년 기온차', '평균 일교차'], axis=1)
y = dataset['포트홀 존재']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, stratify=y, test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape)

from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
X_train_resampled.to_csv('X_train.csv', index=False)
print("원래 학습 데이터 클래스 분포:\n", y_train.value_counts(), '\n')
print("SMOTE 적용 후 클래스 분포:\n", pd.Series(y_train_resampled).value_counts())

# 모델 평가
def get_clf_eval(y_test, pred=None, pred_proba=None):

    cm = confusion_matrix(y_test, pred)
    tn, fp, fn, tp = cm.ravel()  # 혼동 행렬 해체

    print('오차행렬 \n', cm)
    print('정확도 :', accuracy_score(y_test, pred))
    print('정밀도 : ',precision_score(y_test, pred))
    print('재현율 :', recall_score(y_test, pred))
    print('f1 score :', f1_score(y_test, pred))
    print('roc auc score :', roc_auc_score(y_test, pred_proba))

    # False Negative Rate. 실제 포트홀이 있는데 없다고 예측한 경우
    fnr = fn / (fn + tp)
    print(f'False Negative Rate (FNR): {fnr}')

def get_model_train_eval(model, X_train = None, X_test = None, y_train=None, y_test=None):
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    pred_proba = model.predict_proba(X_test)[:,1]
    get_clf_eval(y_test, pred, pred_proba)
```

```
    return pred_proba

# 기후 제거. 기본 모델.
model = xgb.XGBClassifier(random_state=42)

pred_proba    =    get_model_train_eval(model,    X_train_resampled,    X_test,
y_train_resampled, y_test)

# 하이퍼파라미터 후보군 설정
param_dist = {
    'n_estimators': [100, 200, 300, 400, 500],
    'learning_rate': [0.001, 0.005, 0.01, 0.05, 0.1, 0.2],
    'max_depth': [3, 4, 5, 6, 7, 8, 9, 10],
    'min_child_weight': [1, 2, 3, 4, 5],
    'gamma': [0, 0.1, 0.3, 0.5],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'scale_pos_weight': [1, 2, 3]  # 불균형 데이터일 경우 유용
}

model = xgb.XGBClassifier(random_state=42)

# RandomizedSearchCV 설정
random_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_dist,
    n_iter=1000,
    scoring='f1',
    n_jobs=-1,
    cv=5,
    verbose=1,
    random_state=42
)

# 모델 학습 및 튜닝
random_search.fit(X_train_resampled, y_train_resampled)

# 결과 출력
print("Best Parameters:", random_search.best_params_)
print("Best CV Accuracy:", random_search.best_score_)

# 최적 모델로 예측 및 평가
best_model = random_search.best_estimator_
y_pred = best_model.predict(X_test)
print("Test Accuracy:", accuracy_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = cm.ravel()  # 혼동 행렬 해체

print('오차행렬 \n', cm)
print('정확도 :', accuracy_score(y_test, y_pred))
print('정밀도 : ',precision_score(y_test, y_pred))
```

```
print('재현율 :', recall_score(y_test, y_pred))
print('f1 score :', f1_score(y_test, y_pred))
print('roc auc score :', roc_auc_score(y_test, pred_proba))

# False Negative Rate. 실제 포트홀이 있는데 없다고 예측한 경우
fnr = fn / (fn + tp)
print(f'False Negative Rate (FNR): {fnr}')

model = xgb.XGBClassifier(subsample=0.6,
                          scale_pos_weight= 2,
                          n_estimators= 400,
                          min_child_weight= 3,
                          max_depth= 8,
                          learning_rate= 0.05,
                          gamma= 0.1,
                          colsample_bytree= 0.8,
                          random_state=42)
pred_proba    =    get_model_train_eval(model,    X_train_resampled,    X_test,
y_train_resampled, y_test)

importances = model.feature_importances_
importance_df = pd.DataFrame({
    'Feature': X_train_resampled.columns,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

print(importance_df)

with open("model.pickle", "wb") as fw:
    pickle.dump(model, fw)
# 전체 데이터로 모델 재학습
final_model = xgb.XGBClassifier(subsample=0.6,
                          scale_pos_weight= 2,
                          n_estimators= 400,
                          min_child_weight= 3,
                          max_depth= 8,
                          learning_rate= 0.05,
                          gamma= 0.1,
                          colsample_bytree= 0.8,
                          random_state=42)
final_model.fit(x, y)

# 전체 데이터에 대한 예측 확률
result = dataset.copy()

result['예측값'] = final_model.predict_proba(x)[:, 1]

road_groups = result.groupby(['차선수', '총교통량', '중대형차량 교통량']).agg(
    actual_count=('포트홀 존재', 'sum'),
    predicted_sum=('예측값', 'sum'),
    count=('포트홀 존재', 'size')    # number of records in this group
).reset_index()

# Calculate prediction error per road (difference and absolute difference)
```

```
road_groups['error'] = road_groups['predicted_sum'] - road_groups['actual_count']
road_groups['abs_error'] = road_groups['error'].abs()

road_groups.sort_values('actual_count', ascending=False, inplace=True)
print(road_groups[['차선수','총교통량','중대형차량                                    교통량
','actual_count','predicted_sum','error']].head(10))
```

3. XAI 예측 리포트 : SHAP, LIME, DICE code

```
###----- 모델 로드 -----###
model_path = model_path = ###### 모델 경로 ######
xgb_model = joblib.load(model_path)

# 모델 만들 때 사용한 x
x_train_up = pd.read_csv(###### 모델 학습 시 사용한 train set 경로 ######)

# 살펴볼 데이터
x_test = pd.read_csv(###### 중요도를 확인할 데이터 경로 #######)
x_test.drop(['Unnamed: 0'], axis = 1, inplace = True)

###----- SHAP -----###
shap_explainer = shap.Explainer(xgb_model, x_train_up)
shap_values_ex = shap_explainer(x_test) # 각 SHAP value에 대한 정보를 담은
Explainer

# 보고서 예시 1 - 전체 feature importance
shap.plots.bar(shap_values_ex)

# 보고서 예시 2 - 평균 feature importance
shap.plots.bar(shap_values_ex.mean(axis = 0))

# 보고서 예시 3 - 특정 feature 하나 (scatter)
shap.plots.scatter(shap_values_ex[:, "승용차"]) # '승용차' 변수

###----- 함수 정의 -----###
### 지오코딩 - 위도, 경도, 자치구, 행정동 찾아오기
def geo_coding(df):
    ### 위도, 경도
    gk.add_coordinates_to_dataframe(df, '주소')
    df = df.rename(columns = {"decimalLatitude" : "위도", "decimalLongitude" : "
경도"})

    ### 자치구, 행정동
    # 동 : ~동, ~가 부분 찾아오기
```

```
    def get_dong(address):
        find = re.search(r"\b(\w+[동가])\b", address)
        if find:
            dong = find.group(1)
            return(dong)
        else:
            return(np.nan)

    # 구 : ~구 부분 찾아오기
    def get_gu(address):
        find = re.search(r"\b(\w+구)\b", address)
        if find:
            gu = find.group(1)
            return(gu)
        else:
            return(np.nan)

    # DataFrame 추가
    def add_dong_gu_to_dataframe(df):
        dongs = []
        gus = []

        for i in range(len(df)):
            address = df.loc[i, '주소']

            if address:
                try:
                    dong = get_dong(address)
                except:
                    dong = np.nan
                try:
                    gu = get_gu(address)
                except:
                    gu = np.nan
            else:
                dong = np.nan
                gu = np.nan

            dongs.append(dong)
            gus.append(gu)

        df["행정동"] = dongs
        df["자치구"] = gus
        return df

    df = add_dong_gu_to_dataframe(df)
    return df

### ------------------------------------------------------------------------
------

### 교통량 데이터 추가하기
def traffic(df):
    # 도로 데이터
```

```
    gdf_links = gpd.read_file(###### 도로 데이터 경로 ######)
    gdf_links['LINK_ID'] = gdf_links['LINK_ID'].astype(int)

    # 새로운 데이터 공간 데이터로 변환
    gdf_potholes = gpd.GeoDataFrame(
        df,
        geometry = gpd.points_from_xy(df['경도'], df['위도']),
        crs="EPSG:4326")  # 위/경도 WGS84 좌표계로 설정

    # 새로운 데이터 좌표계 통일
    if gdf_links.crs is not None:
        gdf_potholes = gdf_potholes.to_crs(gdf_links.crs)

    # 새로운 데이터의 위치와 가까운 도로 번호(+도로정보) 매칭
    gdf_nearest = gpd.sjoin_nearest(
        gdf_potholes, gdf_links,
        how='left',
        distance_col='distance')  # 계산한 거리(m)

    ### 교통량 데이터
    traffic_d = pd.read_excel(###### 교통량 데이터 경로 ######)

    # 전처리
    traffic_d.columns = [
        f"{upper}" if 'Unnamed' in str(lower) else f"{upper}_{lower}"
        for upper, lower in traffic_d.columns]
    traffic_d.rename(columns={
        'ITS LINK ID': 'ITS_LINK_ID',
        '승용차-평일_전일': '승용차',
        '버스-평일_전일': '버스',
        '트럭-평일_전일': '트럭'}, inplace=True)

  traffic_d['ITS_LINK_ID'] = traffic_d['ITS_LINK_ID'].astype(str).str.split(',')
    traffic_d = traffic_d.explode('ITS_LINK_ID')
    traffic_d['LINK_ID'] = traffic_d['ITS_LINK_ID'].str.strip().astype(int)
    traffic_df = traffic_d[['LINK_ID', '도로명', '차선수', '승용차', '버스', '트럭
']].copy()
                                                traffic_df                      =
(traffic_df.dropna(subset=['LINK_ID']).drop_duplicates('LINK_ID').reset_index(dr
op=True).astype({'LINK_ID': 'int'}))

    # 교통량 join
    pothole_traffic = gdf_nearest.merge(traffic_df, on='LINK_ID', how='left')
    pothole_output = pothole_traffic[['날짜', '주소', '위도', '경도', '행정동', '
자치구', 'LINK_ID', '도로명', '차선수', '승용차', '버스', '트럭']]
    pothole_output['총교통량'] = pothole_output['승용차'] + pothole_output['버스']
+ pothole_output['트럭']
         pothole_output['중대형차량    교통량']    =    (pothole_output['버스']    +
```

```
pothole_output['트럭']) / pothole_output['총교통량']

    return pothole_output

### ------------------------------------------------------------------------
------

### 건물 평균 연령, 배수등급, 경사도 추가
def nature(new_pothole_traffic):

    ### 건물별 평균 연령
    old = pd.read_csv(###### 건물별 연령 데이터 경로 ######)
    bup = pd.read_csv(###### 법정동 코드 데이터 경로 ######)
    bup['법정동명'] = bup['법정동명'].astype(str)

    # 법정동 만들기
    new_pothole_traffic['법정동명'] = '서울특별시 ' + new_pothole_traffic['자치구
'] + ' ' + new_pothole_traffic['행정동']
    new_pothole_traffic['법정동명'] = new_pothole_traffic['법정동명'].astype(str)

    # join
    new_pothole_building = new_pothole_traffic.merge(bup[['법정동코드', '법정동명
']], on = '법정동명', how = 'left')
    new_pothole_building = new_pothole_building.merge(old[['법정동코드', '평균_건
물연령']], on = '법정동코드', how = 'left')

    ### 배수등급
    # 공간 데이터로 변환
    gdf_pothole2 = gpd.GeoDataFrame(
        new_pothole_building,
                        geometry=gpd.points_from_xy(new_pothole_building['경도'],
new_pothole_building['위도']),
        crs="EPSG:4326")

    # 포트홀 좌표계 → EPSG:5174 로 변환 (shp에 맞추기)
    gdf_pothole2 = gdf_pothole2.to_crs("EPSG:5174")

    # 배수등급 로드
    soil_df = gpd.read_file(###### 배수등급 데이터 경로 ######)

    # join
    new_pothole_soil = gpd.sjoin(gdf_pothole2, soil_df[['SOILDRA', 'geometry']],
how='left', predicate='within')

    ### 토양 경사도
    # 데이터 로드
```

```
    slope_df = gpd.read_file(###### 토양 경사도 데이터 경로 ######)
    new_pothole_soil.drop(['index_right'], axis=1, inplace=True)

    # join
       new_pothole_slope  =  gpd.sjoin(new_pothole_soil,  slope_df[['SOILSLOPE',
'geometry']], how='left', predicate='within')

    # 열 이름 수정
    new_pothole_slope.rename(columns = {'SOILDRA' : '배수등급', 'SOILSLOPE' : '경
사도'}, inplace = True)
    new_pothole_done = new_pothole_slope[['날짜', '주소', '위도', '경도', '자치구
', '행정동', '도로명', '차선수', '승용차', '버스', '트럭', '총교통량', '중대형차량 교
통량', '평균_건물연령', '배수등급' ,'경사도']]

    return new_pothole_done

### ----------------------------------------------------------------------------
------

def join_gu(new_pothole_done):
    ### 자치구별 데이터 로드
    people = pd.read_pickle(###### 자치구별 인구 수 데이터 경로 ######)
    people['자치구'] = people['자치구'].str.replace("\u3000","",regex = False)

    new_pothole_done.rename(columns = {'날짜' : '발생일'}, inplace = True)
    new_pothole_done['발생일'] = pd.to_datetime(new_pothole_done['발생일'])

    ### 인구 수
    new_pothole = pd.merge(new_pothole_done, people, on = '자치구', how = 'left')

    return new_pothole

### ----------------------------------------------------------------------------
------

def prediction(new_pothole, transformer_path, scaler_path, model_path):
    ## x 할당
    new_x = new_pothole[['차선수', '승용차', '버스', '트럭', '총교통량', '중대형차량
교통량', '평균_건물연령', '인구 수', '배수등급', '경사도']]

    # 저장된 변환기, 스케일러 불러오기
    transformer = joblib.load(transformer_path)
    scaler = joblib.load(scaler_path)
    # 새로운 데이터 변환
    cols = ['승용차', '버스', '트럭', '총교통량', '중대형차량 교통량', '평균_건물연령
', '인구 수']
    arr = new_x[cols].values + 1e-6
    bc = transformer.transform(arr)
```

```
    bc_std = scaler.transform(bc)
    new_x[cols] = bc_std

    ### 배수등급, 경사도 변수 처리
    # 경사도
    slope_encoding = {
        '0-2%': 0,
        '2-7%': 1,
        '7-15%': 2,
        '15-30%': 3,
        '30-60%': 4,
        '60-100%': 5}
    # 배수등급
    drain_encoding = {
        '매우양호': 5,
        '양호': 4,
        '약간양호': 3,
        '약간불량': 2,
        '불량': 1,
        '매우불량': 0}
    # 매핑
    new_x['경사도'] = new_x['경사도'].map(slope_encoding)
    new_x['배수등급'] = new_x['배수등급'].map(drain_encoding)

    ### 모델 로드
    xgb_model = joblib.load(model_path)
    y_pred = xgb_model.predict(new_x)
    y_pred_prob = xgb_model.predict_proba(new_x)[:, 1]

    return new_pothole, new_x, y_pred, y_pred_prob ## 원본 데이터, x를 살펴보기 위
해 따로 받아옴 !

def prediction_without_env(new_data, transformer_path, scaler_path, model_path):
    ### 마지막에 출력할 output
    output_df = new_data.copy()
    ### x 만들기
    geo = geo_coding(new_data)
    traffic_df = traffic(geo)
    nature_df = nature(traffic_df)
    new_potholes = join_gu(nature_df)
    ### 예측
       new_data_org,  new_x,  y_pred,  y_pred_prob  =  prediction(new_potholes,
transformer_path, scaler_path, model_path)
    ### output
    output_df['예측'] = y_pred
    output_df['예측 확률'] = y_pred_prob
    return new_data_org, new_x, output_df
```

```
### 새롭게 확인해볼 데이터 목록 (임의로 생성)
input = ["서울특별시 강남구 대치동 507",
         "서울특별시 동대문구 제기동 137-418",
         "서울특별시 서초구 방배동 756-4",
         "서울특별시 용산구 동빙고동 90-1",
         "서울특별시 성북구 보문동5가 235",
         ]
date = ['2024-07-28', '2023-06-23', '2021-10-29', '2024-01-05', '2022-04-29']
new_data = pd.DataFrame({'날짜' : date, '주소' : input})

### Model, BoxCox Transformer, Scaler Path
model_path = ###### 모델 경로 ######
transformer_path = ###### BoxCox Transformer 경로 ######
scaler_path = ###### Standard Scaler 경로 ######

### Prediction
new_data_org,     new_x,     output_df     =     prediction_without_env(new_data,
transformer_path, scaler_path, model_path)

# 리포트 예시 - 최종 데이터 프레임
output_df

###----- 모델 로드 -----###
model_path = model_path = ###### 모델 경로 ######
xgb_model = joblib.load(model_path)

# 모델 만들 때 사용한 x
x_train_up = pd.read_csv(###### 모델 학습 시 사용한 train set 경로 ######)

# 살펴볼 데이터 로드
new_places_x = pd.read_csv(###### 새롭게 입력한 데이터의 x 변환 결과 경로 #######)
new_places_x.drop(['Unnamed: 0'], axis = 1, inplace = True)

###----- SHAP -----###
shap_explainer = shap.Explainer(xgb_model, x_train_up)
shap_values_ex = shap_explainer(new_places_x) # 각 SHAP value에 대한 정보를 담은
Explainer

# 보고서 예시 - 특정 샘플(0번째 데이터) 설명
shap.plots.waterfall(shap_values_ex[0])

###----- 역변환 함수 -----###
def inverse_scaling_boxcox(transformed_data, transformer_path, scaler_path):

    # 역변환 대상 변수
    cols = ['승용차', '버스', '트럭', '총교통량', '중대형차량 교통량', '평균_건물연령
', '인구 수']
```

```
    cols_idx = [transformed_data.columns.get_loc(col) for col in cols]

    # scaler, transformer
    scaler = joblib.load(scaler_path)
    transformer = joblib.load(transformer_path)

    # 변환된 5개 변수만 추출
    transformed = transformed_data.iloc[:, cols_idx]

    # 역표준화
    un_scaled = scaler.inverse_transform(transformed)

    # 역변환
        un_transformed      =      np.array([inv_boxcox(un_scaled[:,      i],
transformer.lambdas_[i]) for i in range(un_scaled.shape[1])]).T

    # 전체 복원
    original = transformed_data.copy()
    for j, col in enumerate(cols):
        original[col] = un_transformed[:,j]

    ### 배수등급, 경사도 변수 처리
    # 경사도
    slope_encoding = {'0-2%': 0, '2-7%': 1, '7-15%': 2, '15-30%': 3, '30-60%': 4,
'60-100%': 5}
    # 배수등급
    drain_encoding = {'매우양호': 5, '양호': 4, '약간양호': 3, '약간불량': 2, '불량
': 1, '매우불량': 0}
    # 인코딩 딕셔너리 뒤집기
    slope_decoding = {v: k for k, v in slope_encoding.items()}
    drain_decoding = {v: k for k, v in drain_encoding.items()}

    # 역매핑
    original['경사도'] = original['경사도'].map(slope_decoding)
    original['배수등급'] = original['배수등급'].map(drain_decoding)

    return original

###----- DiCE 활용, Counterfactual 생성 -----###
def dice(x_test, model_path, transformer_path, scaler_path): # x_test : 예측이 1
로 나오는 예시 샘플 하나
    xgb_model = joblib.load(model_path)

    # 혹시 모르니까 한 번 더 변환해주고
    x_test = x_test.astype('float64')

    ### DiCE
    continuous = ['승용차', '버스', '트럭', '총교통량', '중대형차량 교통량', '평균_
건물연령', '인구 수']
```

```
    d = dice_ml.Data(dataframe = x_test.assign(발생여부 = 0),  # dummy target
                     continuous_features = continuous,
                     outcome_name = "발생여부")
    m = dice_ml.Model(model = xgb_model, backend = "sklearn")
    exp = dice_ml.Dice(d, m)

    ### counterfactual 생성
    # 값을 바꿀 변수들만 지정
    features_to_change = ['승용차', '버스', '트럭', '총교통량', '중대형차량 교통량',
'평균_건물연령']
    query_instance = x_test
    query_instance = query_instance.astype('float64')
          cf    =    exp.generate_counterfactuals(query_instance,    total_CFs=1,
desired_class="opposite",features_to_vary=features_to_change)

    ### 데이터프레임 저장
    # 기존 데이터
    original_one = cf.cf_examples_list[0].test_instance_df
    # 바뀐 데이터
    changed_zero = cf.cf_examples_list[0].final_cfs_df

       original_one   =   inverse_scaling_boxcox(original_one,   transformer_path,
scaler_path)
       changed_zero   =   inverse_scaling_boxcox(changed_zero,   transformer_path,
scaler_path)

    return original_one, changed_zero

### Model, BoxCox Transformer, Scaler Path
model_path = ###### 모델 경로 ######
transformer_path = ###### BoxCox Transformer 경로 ######
scaler_path = ###### Standard Scaler 경로 ######

### DiCE로 Counterfactual 생성
new_x = pd.read_csv(###### 새롭게 확인하고자 하는 데이터 ######)
new_x.drop(['Unnamed: 0'], axis = 1, inplace = True)
new_x_first = new_x.loc[[3]] # 세 번째 데이터 확인
new_x_first = new_x_first.astype('float64')

original_one,  changed_zero  =  dice(new_x_first,  model_path,  transformer_path,
scaler_path)
# 리포트 예시 - original_one과 changed_zero 데이터프레임 출력
display(original_one)
display(changed_zero)
```

참고문헌

[i] 국민권익위원회,「포트홀(도로파임) 관련 민원 분석」, 2024.4.

[ii] 아투포커스, 도로 지뢰 '포트홀' 밟았는데...피해 보상까지 '하세월', 2024.10.10

[iii] 뉴데일리, "1~2월 서울 포트홀 1만개 육박 ... 전년 대비 2배", 2024.03.14.

[iv] HRC Opinion, "[기획] 2023년 여름, 어떻게 지내셨나요? – 여름날씨에 대한 인식", 2024.04.17.

[v] 국립기상과학원, <한반도 기후변화 전망보고서 2020>.

[vi] 부산일보, "5년간 고속도로 포트홀 2만건...피해 배상액 136억원", 2024.10.21.

[vii] 아이뉴스24, "고속도로 '포트홀' 5년간 2만건 넘어... 배상액도 135억원", 2024.10.21.

[viii] 한국도로공사, "고속도로 포트홀 피해 배상 및 보수 현황", 2024.

[ix] 제주특별자치도, "2024년 도로 포트홀 보수 단가 공고", 2024.

[x] 김영호, "서울시 자동차전용도로 13곳, 최근 3년간 포트홀 1만8820건 발생…피해 배상은 108건, 3271만원에 그쳐", 브릿지경제, 2024.9.4., https://news.nate.com/view/20240904n17771

[xi] 김광호, "용인지역 최근 3년간 포트홀 사고 827건…7억4천만원 배상", 매일경제, 2025.5.6., https://stock.mk.co.kr/news/view/738935

[xii] 한국지방재정공제회, "영조물배상책임보험 안내,"