건설근로자 능력평가 및 인건비 산정을 위한 머신러닝기법의 적용 2025년 주재일

석사학위논문

건설근로자 능력평가 및 인건비 산정을 위한 머신러닝 기법의 적용

인천대학교 정보기술대학원

컴퓨터 전공

주 재 일

2025년 08월

석사학위논문

건설근로자 능력평가 및 인건비 산정을 위한 머신러닝 기법의 적용

인천대학교 정보기술대학원

컴퓨터 전공

주 재 일

2025년 08월

석사학위청구논문

건설근로자 능력평가 및 인건비 산정을 위한 머신러닝 기법의 적용

지도교수 이 승 수

이 논문을 석사학위논문으로 제출함

2025 년 06 월

인천대학교 정보기술대학원

컴퓨터 전공

주 재 일

이 논문을 주재일의 컴퓨터전공

석사학위 논문으로 인준함

2025 년 06 월

심사위원장 김지범 (인)

심사위원 이현규 (인)

심사위원 이승수 (인)

국문초록

건설근로자 능력평가 및 인건비 산정을 위한 머신러닝 기법의 적용

본 연구는 건설 근로자의 능력을 정량적으로 평가하고, 이를 기반으로 공정한 인건비 산정을 가능하게 하는 머신러닝 기반 평가 모델을 제안한다. 기존 건설 현장에서의 인건비 산정은 관리자 개인의 주관에 의존하고 있어 동일한 작업에 대해 서로 다른 임금이 지급되는 문제를 발생시킨다. 이에 본 연구는 작업 난이도와 수행 시간이라는 정량적 지표를 기준으로 근로자의 능력을 평가하고, 인건비를 산정할 수 있는 예측 모델을 개발하였다. 모델링에는 결정 트리, 인공신경망, 랜덤 포레스트 세 가지 머신러닝 알고리즘을 적용하였으며, 성능 평가 결과 인공신경망(ANN)이 가장 우수한 정확도와 균형 잡힌 성능을 보였다. 본 연구는 근로자 평가 및 보상 체계를 데이터 기반으로 전환할 수 있는 가능성을 제시하며, 향후 실무 적용과 타 산업 확장의 기반 자료로 활용될 수 있다.

주제어 : _머신러닝, 건설 근로자 , 인건비산정 , 능력평가, 인공신경망

목차

국문초록 ····· i
목 차 ····· ii
표 목 차 ····· iv
그림목차 ····· v

1. 서 론 ····· 1
1.1 연구 배경 ····· 1
1.2 연구의 필요성 ····· 3
1.3 연구 목적 ····· 4
1.4 연구 범위 및 방법 개요 ····· 5
2 이론적 배경 ····· 7
2.1 머신러닝 개요 ····· 7
2.2 건설 인력 평가의 기존 방식 ····· 8
2.2.1 정성적·주관적 평가 방식의 지배 ····· 9
2.2.2 공식 노임단가 제도의 한계 ····· 10
2.3 주요 머신러닝 알고리즘 소개 ····· 11
2.3.1 결정 트리 (Decision Tree) ····· 12
2.3.2 인공신경망 (Artificial Neural Network, ANN) ····· 13
2.3.3 랜덤 포레스트 (Random Forest) ····· 14
2.4 관련 연구 동향 ····· 15
2.4.1 건설 분야 인력 및 성과 관련 선행 연구 ····· 15
2.4.2 본 연구의 차별성 ····· 18
3 연구 방법론 ····· 20
3.1 데이터 수집 및 가공 ····· 20
3.1.1 실제 데이터 수집의 한계점 및 가상 데이터 생성의 필요성 ····· 20
3.1.2 가상 데이터 생성의 구체적인 기준 및 방법 ····· 21
3.2 연구 모델 선정 ····· 25
3.2.1 모델 선정 배경 ····· 25
3.2.2 데이터 분할 ····· 26

3.2.3 모델별 하이퍼파라미터 설정 및 튜닝 ····· 28
3.3 성능 평가 지표 ····· 31
3.3.1 혼동 행렬 ····· 32
3.3.2 주요 분류 성능 지표 ····· 32
3.3.3 다중 클래스 분류에서의 지표 확장 ····· 33
3.3.4 ROC 곡선 및 AUC ····· 34
3.3.5 성능 평가 환경 및 구현 ····· 35
4. 실험 결과 및 분석 ····· 39
4.1 데이터셋 준비 및 전처리 ····· 40
4.2 모델 학습 및 하이퍼파라미터 튜닝 ····· 40
4.3 모델 성능 평가 및 비교 ····· 41
4.3.1 전체 성능 요약 ····· 41
4.3.2 혼동 행렬 및 클래스별 성능 분석 ····· 41
4.3.3 ROC 곡선 및 AUC 분석 ····· 45
4.4 최적 모델 선정 및 주요 변수 영향도 분석 ····· 46
4.4.1 최적 모델 선정 및 각 모델의 장단점 비교 ····· 46
4.4.2 주요 변수 영향도 분석 (랜덤 포레스트 Feature Importance) ····· 47
5. 결론 및 시사점 ····· 50
5.1 연구 요약 ····· 50
5.2 실무적 시사점 ····· 50
5.3 학문적 시사점 ····· 52
5.4 연구의 한계 및 향후 연구 방향 ····· 53
참고문헌 ····· 55
Abstract ····· 60
부록(Appendix) ····· 62

표목차

표 1-1. 2023년도 기준 건설기술인 협회 누적 등록 인원 ····· 2
표 4-1. 각 모델의 테스트 결과표 ····· 41
표 4-2. 결정 트리의 테스트 데이터셋 혼동 행렬 ····· 42
표 4-3. 결정 트리의 클래스별 분류 보고서 ····· 42
표 4-4. 인공신경망의 테스트 데이터셋 혼동행렬 ····· 43
표 4-5. 인공신경망 클래스별 분류 보고서 ····· 43
표 4-6. 랜덤 포레스트의 테스트 데이터셋 혼동 행렬 ····· 44
표 4-7. 랜던포레스트의 클래스별 분류 보고서 ····· 44
표 4-8. 각 모델의 AUC 값 비교 ····· 45
표 4-9. 랜덤 포레스트 모델의 변수 중요도 ····· 48

그림목차

그림 4-1. ROC곡선 비교 ····· 45
그림 4-2. 랜덤 포레스트 모델의 변수 중요도 ····· 48

1. 서 론

1.1 연구 배경

건설 산업은 인류의 삶과 경제 발전에 필수적인 기반을 제공하며, 국내 총생산(GDP)의 상당 부분을 차지하는 핵심 산업 중 하나이다. 국토교통부의 '2023년 건설공사 계약 및 수주 현황'에 따르면, 2023년 국내 건설 수주액은 200조 원을 상회하며, 건설업 종사자 수는 전체 산업 고용의 약 7%를 차지하는 등 국가 경제에서 매우 중요한 역할을 수행하고 있다[1]. 그러나 이러한 중요성에도 불구하고, 건설 산업은 여전히 인력 중심의 전통적 산업 구조를 유지하고 있으며, 그 특성상 근로자의 기술 수준과 성과가 프로젝트의 품질, 비용, 일정에 결정적인 영향을 미친다.

특히 건설 현장은 높은 노동 강도와 다양한 직무군이 혼재되어 있기 때문에, 각 작업자의 숙련도에 따른 업무 배분 및 인건비 책정은 현장의 효율성과 직결된다. 정부는 매년 상·하반기에 걸쳐 공종별 노임단가를 공시하고 있으나, 이러한 공시자료는 통계적 평균에 기반한 기준선일 뿐, 실제 현장에서는 동일한 직무일지라도 근로자의 숙련도, 작업 효율, 현장 상황, 경력, 친분도 등에 따라 인건비가 달리 책정되는 것이 현실이다.

통계청의 '2023년 건설업 조사' 결과에 따르면 전체 건설업 종사자의 수는 181만 명에 달한다[2]. 하지만, 건설 관련 학력을 소지하거나 건설 자격증을 소지하여 한국건설기술인협회에 등록된 인원은 2025년 2월 기준 약 103.5만 명이다.. 이 중 50대와 60대 이상의 등록 인원이 약 60%를 차지하며 고령화가 심화되고 있다[3][4]. 이러한 등록 인원은 누적된 수치이므로, 고령층의 은퇴 및 타 산업으로의 이직 경향을 고려할 때, 실제 건설업에 활발하게 종사하는 기술인의 비율은 전체 건설업 종사자의 50% 미만일 것으로 추정할 수 있다.

문제는 이러한 차등이 대부분 현장 관리자의 주관적 판단이나 경험에 의존한다

는 점이다. 이러한 주관적인 평가는 다음과 같은 심각한 문제를 야기한다. 첫째, 동일한 능력을 가진 근로자 간에도 보상에 차이가 발생할 수 있으며, 이는 근로자의 사기 저하와 노사 갈등의 원인이 될 수 있다[5]. 둘째, 고용주의 입장에서도 인력 배치와 인건비 예측의 어려움을 낳고 있으며, 이는 사업 계획 수립의 정밀도를 저해하고 전체 프로젝트의 비용 효율성에도 부정적인 영향을 미친다. 셋째, 이러한 불공정한 평가 시스템은 숙련된 근로자의 이탈을 야기하고, 신규 인력 유입을 저해하여 건설 산업의 만성적인 인력난을 심화시키는 요인이 되기도 한다[6][7].

연령대	인원	비율	비율
20대	33,211명	3.2%	15.1%
30대	122,507명	11.8%	
40대	258,143명	25.0%	25.0%
50대	342,934명	33.2%	60.0%
60대	277,432명	26.8%	
소계	1,034,227명	100.0%	100.0%

표 1-1. 2023년도 기준 건설기술인 협회 누적 등록 인원

또한, 건설 근로자 인력 구조의 변화는 이러한 문제들을 더욱 복합적으로 만들고 있다. 건설근로자공제회의 '건설근로자 취업 및 일자리 동향' 보고서에 따르면, 2023년 기준 건설근로자의 평균 연령은 52.7세이며, 전체 인력의 60% 이상이 50대 이상으로 구성되어 고령화가 급속히 진행되고 있음을 보여준다[8]. 동시에 외국인 근로자의 비율도 증가하고 있어, 기술 격차와 안전 문제가 동시에 대두되고 있다. 이러한 상황에서 단순한 노임단가 기준만으로는 현장의 다양성과 개별 근로자의 역량을 포착하기 어렵다.

이러한 맥락에서, 최근 디지털 전환과 데이터 기반 의사결정이 다양한 산업에 확산됨에 따라, 건설업에서도 인력 관리 시스템의 과학적 전환이 필요하다는 목소리가 커지고 있다. 특히 머신러닝과 같은 인공지능 기술은 정형화된 평가 기준이 부족한 영역에서 새로운 대안을 제공할 수 있는 가능성을 제시한다. 머신러닝은 복잡한 변수 간 관계를 자동으로 학습하고 예측하는 데 강점을 가지며, 기존의 정성적 평가를 보완하거나 대체할 수 있는 도구로서 주목받고 있다.

1.2 연구의 필요성

현재 건설 산업은 4차 산업혁명 시대의 흐름 속에서 디지털 전환이라는 거대한 변화를 겪고 있으며, BIM(Building Information Modeling), IoT 센서, 드론, 자동화 장비 등 다양한 스마트 건설 기술이 도입되고 있다[9]. 이러한 기술들은 공정 관리, 품질 관리, 안전 관리 등 건설 현장의 다양한 영역에서 효율성을 증대시키고 있다. 그러나 인력 관리 및 인건비 산정 시스템은 여전히 아날로그 방식에 머물러 있으며, 이는 전체 공정의 혁신을 저해하는 요소로 작용한다. 첨단 기술이 현장에 성공적으로 정착되기 위해서는, 기본이 되는 인력 구성의 역량이 체계적으로 평가되고, 그에 맞는 배치와 보상이 전제되어야 한다.

따라서, 건설 근로자의 작업 난이도와 소요 시간이라는 정량적 데이터를 활용하여 능력을 평가하고, 그에 합당한 보상 기준을 제시하는 시스템이 마련된다면, 건설업의 인사관리 체계는 보다 정밀하고 공정한 방식으로 전환될 수 있다. 특히 이러한 시스템이 자동화된다면, 현장 관리자들은 보다 효율적으로 인력을 배치하고, 예산 계획을 수립하며, 노사 간의 신뢰를 확보할 수 있을 것이다. 이는 건설 프로젝트의 생산성 향상과 수익성 제고에도 직접적으로 기여할 수 있다.

나아가, 이러한 정량적 평가 체계는 단순히 건설업에 국한되지 않고, 물류, 제조, 설비, 서비스 업종 등 숙련도와 생산성이 핵심 요소로 작용하는 산업 전반에 걸쳐 확장 가능하다. 특히 인공지능의 해석 가능성과 설명 가능성(Explainable AI, XAI)이 중요하게 대두되는 시점에서, 머신러닝 모델을 기반으로 한 능력 평가와 인건비 산정은 그 실효성과 타당성을 동시에 확보할 수 있는 방식이 될 수 있다[10][11]. 이는 단순한 예측 모델을 넘어, 예측 결과를 이해하고 설명할 수 있도록 함으로써 현장 관리자 및 근로자 모두에게 신뢰를 제공할 수 있다.

1.3 연구 목적

본 연구는 건설 근로자의 주관적인 능력 평가 및 불공정한 인건비 산정 문제를

해결하기 위해, 데이터 기반의 객관적인 평가 모델을 구축하고 그 활용 가능성을 탐색하는 데 목적을 둔다. 이를 위해 다음과 같은 세부 목표를 가지고 수행된다.

1. 작업 난이도와 수행 시간을 기준으로 근로자의 능력을 평가할 수 있는 머신러닝 기반 모델을 구축한다. 이는 기존의 정성적, 주관적 평가 방식에서 벗어나, 실제 작업 데이터를 활용하여 근로자의 역량을 객관적으로 측정하고 능력 등급을 분류하는 모델을 개발하는 것을 의미한다.

2. 정량적으로 도출된 능력 수치를 기반으로, 공정하고 합리적인 인건비 산정 체계를 제시한다. 개발된 능력 평가 모델의 결과를 실제 인건비 산정 프로세스에 연동하여, 근로자의 실제 능력에 비례하는 공정한 보상 기준을 마련하는 방안을 모색한다.

3. 결정 트리, 인공신경망, 랜덤 포레스트 세 가지 대표적 알고리즘을 실험적으로 비교하여, 예측 정확도와 실무 활용 가능성을 평가한다. 각 머신러닝 알고리즘의 특성 및 장단점을 고려하여 동일한 데이터셋에 적용하고, 성능 지표(Accuracy, Precision, Recall, F1-score)를 통해 최적의 모델을 선정하는 것을 목표로 한다.

4. 평가 모델의 해석 가능성과 성능의 균형을 분석하여, 건설 현장 실무자가 직접 활용 가능한 평가 지표를 제안한다. 단순히 높은 예측 정확도를 넘어, 모델의 결과가 왜 그렇게 도출되었는지 설명할 수 있는 해석력을 중요하게 고려하여, 현장 관리자가 모델을 신뢰하고 실제 의사결정에 활용할 수 있는 방안을 제시한다.

5. 데이터 기반 근로자 평가 및 인건비 산정 방식의 실무 적용 가능성을 논의하고, 향후 다양한 산업군으로의 확장 가능성을 탐색한다. 본 연구에서 개발된 모델이 건설 현장의 인력 관리 효율성을 증대시키고 노사 관계 개선에 기여할 수 있는 실질적인 방안을 제시하며, 제조업, 물류업 등 다른 산업 분야로의 적용 가능성을 논의한다.

1.4 연구 범위 및 방법 개요

본 연구는 실제 건설 근로자 데이터를 확보하기 어려운 현실적인 제약을 감안하여, 시뮬레이션 기반의 정량 데이터를 활용하여 머신러닝 모델을 구축하고 검증한다. 총 10,000명의 가상 건설 근로자 데이터를 생성하여, 실제 건설 현장의 다양한 작업 유형, 난이도, 근로자 특성을 충분히 반영할 수 있도록 설계하였다. 이 데이터는 머신러닝 모델 학습에 적합한 형태로 전처리되었으며, 각 변수 간의 상관관계와 분포는 실제 현장의 통계적 경향을 모방하도록 구성되었다.

연구에 사용된 주요 머신러닝 알고리즘은 다음과 같다:

1. 결정 트리 (Decision Tree): 변수의 분할 기준을 기반으로 예측을 수행하는 알고리즘으로, 트리의 구조가 명확하고 해석이 용이하여 직관적인 결과 도출이 가능하다[12].

2. 인공신경망 (Artificial Neural Network, ANN): 비선형적인 데이터 패턴을 학습할 수 있으며, 복잡한 데이터에서도 높은 정확도를 기록할 수 있는 잠재력을 가지나, 내부 구조의 복잡성으로 인해 해석력이 떨어지는 경향이 있다[13].

3. 랜덤 포레스트 (Random Forest): 다수의 결정 트리를 앙상블하여 예측력을 향상시키는 기법으로, 개별 결정 트리의 단점인 과적합(overfitting)을 효과적으로 방지할 수 있다[14]. 또한, 변수 중요도 분석이 가능하여 모델의 해석 가능성을 일정 부분 확보할 수 있으며, 실무 적용에 적합한 안정적인 성능을 기대할 수 있다.

각 모델은 동일한 입력 변수(작업 난이도, 수행 시간, 직무, 경력, 교육 이수 여부 등)를 사용하며, 출력 변수는 근로자의 능력 등급(S/A/B/C/D)으로 설정된다. 모델의 성능은 분류 문제의 대표적인 평가 지표인 Accuracy (정확도), Precision (정밀도), Recall (재현율), F1-score를 기준으로 평가하며, 모델의 일반화 성능을 확인하고 과적합 여부를 점검하기 위해 교차검증(Cross Validation)을 수행하여 각 모델의 안정성을 비교한다.

또한, 본 연구에서는 단순히 단일 모델의 성능 평가에 그치지 않고, 세 가지 알

고리즘 간의 비교 분석을 통해 건설 현장 실무 적용에 가장 적합한 접근법을 도출한다. 각 모델의 수학적 구조와 학습 방식의 차이를 심층적으로 설명하고, 각 모델의 학습 곡선, 수렴 속도, 그리고 랜덤 포레스트의 변수 중요도 분석 결과를 병행 제시함으로써, 이론적 정합성과 더불어 실무 활용성을 동시에 확보하고자 한다.

2 이론적 배경

2.1 머신러닝 개요

머신러닝(Machine Learning)은 명시적인 프로그래밍 없이 데이터로부터 학습하여 예측 또는 분류를 수행하는 인공지능 기술의 한 분야로, 1959년 Arthur Samuel이 "명시적으로 프로그래밍되지 않고도 학습할 수 있는 능력을 컴퓨터에 부여하는 연구 분야"라고 정의한 이래로, 머신러닝은 데이터 과학의 발전과 컴퓨팅 파워의 향상에 힘입어 비약적인 발전을 이루었다[15]. 현재 다양한 산업군에서 활용되고 있으며 복잡한 변수 간의 관계를 파악하고, 정형화되지 않은 판단 기준을 정량화하는 데 유리하기 때문에 기존의 직관적이고 주관적인 의사결정을 대체하는 도구로 활용할 수 있다.

건설 산업은 전통적으로 경험과 직관에 의존하는 경향이 강했으나, 최근 4차 산업혁명 시대의 도래와 함께 데이터 기반의 의사결정 중요성이 부각되면서 머신러닝 기술 도입의 필요성이 증대되고 있다. 건설 현장에서 발생하는 방대한 데이터(공정 데이터, 장비 운용 데이터, 안전 데이터, 인력 데이터 등)를 활용하여 생산성을 향상시키고, 위험을 줄이며, 자원을 효율적으로 배분하는 데 머신러닝이 핵심적인 역할을 수행할 수 있다. 건설업 분야에서의 머신러닝 적용 사례는 다음과 같다.

1. 공정 예측 및 최적화: 공사 기간, 비용, 자원 소모량 등을 과거 데이터와 현장 변수를 기반으로 예측하여 최적의 공정 계획을 수립한다. 예를 들어, 김용성 외 (2018)은 딥러닝을 활용하여 건설 프로젝트의 공사 기간을 예측하는 모델을 제안하였으며[16], 박준혁 외 (2020)은 기성고 데이터와 머신러닝을 통해 공정 지연 위험을 조기에 감지하는 연구를 수행하였다[17]. 또한 강윤호와 윤석헌 (2022)은 건축 공사비 예측에 머신러닝을 적용하여 피처 스케일링 및 로그 변환이 예측 성능에 미치는 영향을 분석하며 건설 비용 예측의 정확도를 높이는 데 기여하였다[18].

2. 안전 관리 및 위험 예측: CCTV 영상 분석, IoT 센서 데이터(온도, 습도, 유해가

스 등)를 활용하여 작업자의 위험 행동을 감지하거나, 특정 구역의 안전 위험도를 예측하여 사고를 예방한다[19]. 특히 황윤호 외는 빅데이터를 활용한 AI 기반 우선 점검 대상 현장 선정 모델을 개발하여 건설 현장 사고 예방에 기여할 수 있음을 보여주었다[20]. 또한, 과거 산업 재해 데이터를 학습하여 잠재적인 위험 요인을 식별하고 이에 대한 예방책을 마련하는 데에도 활용될 수 있다[21].

3. 품질 관리: 콘크리트 압축 강도 예측, 용접 불량 감지 등 건설 재료 및 시공 품질을 비파괴적으로 평가하고 예측하는 데 머신러닝 모델이 적용된다[22]. 이는 불량률을 감소시키고 재시공 비용을 절감하는 데 기여한다.
건설 장비 및 자재 관리: 장비의 고장 예측, 최적의 유지보수 시기 결정, 자재 수요 예측 및 재고 관리 등에 머신러닝이 활용되어 운영 효율성을 높인다[23].

4. 인력 관리 및 생산성 향상: 본 연구의 주제와 같이, 근로자의 작업 성과, 숙련도, 생산성 등을 정량적으로 평가하고 예측하여 최적의 인력 배치 및 인건비 산정을 지원한다. Jacobsen et al. (2023)은 심층 학습과 운동학적 데이터를 활용하여 건설 근로자의 작업 활동을 인식하고 생산성을 모니터링하는 방법을 제시하며, 이는 본 연구가 제안하는 작업 시간 기반의 능력 평가와 밀접한 관련이 있다[24]. 이러한 데이터 기반의 객관적인 지표는 인적 자원의 효율적 활용을 통해 프로젝트의 전반적인 생산성 향상에 기여하며, 근로자 만족도를 높이고, 이직률을 낮추는 데에도 긍정적인 영향을 미칠 수 있다[25].

본 연구에서는 머신러닝의 지도학습(Supervised Learning) 기법을 적용하여 근로자의 능력을 정량적으로 평가하고, 이를 인건비 산정에 활용하고자 한다. 지도학습은 입력값(Input)과 정답(Label)의 쌍으로 구성된 데이터를 기반으로 학습을 수행하며, 분류(Classification) 및 회귀(Regression) 문제에 모두 적용될 수 있다. 본 연구는 근로자의 능력 등급을 분류하는 문제이므로, 지도학습 분류 모델이 적합하다.

2.2 건설 인력 평가의 기존 방식

건설 현장에서의 인력 평가는 일반적으로 두 가지 주요 목적을 가진다. 첫째, 근

로자의 숙련도와 능력을 파악하여 적절한 작업에 배치함으로써 생산성과 효율성을 극대화하는 것이다. 둘째, 근로자의 기여도에 상응하는 공정한 인건비를 산정하여 동기 부여와 사기 진작을 도모하는 것이다. 그러나 현재 건설 산업에서 보편적으로 적용되는 인력 평가 및 인건비 산정 방식은 여러 한계점을 내포하고 있다.

2.2.1 정성적·주관적 평가 방식의 지배

대부분의 건설 현장에서는 현장 관리자(반장, 소장 등)의 오랜 경험과 직관에 기반한 정성적인 평가 방식이 주를 이룬다. 근로자의 평가는 주로 다음과 같은 요소들에 의해 이루어진다.

1. 직접 관찰: 관리자가 현장에서 근로자의 작업 태도, 숙련도, 문제 해결 능력 등을 직접 보고 판단한다.
2. 구두 전달 및 평판: 다른 근로자나 협력업체로부터의 평판, 과거 작업 이력에 대한 구두 정보가 평가에 영향을 미친다.
3. 개인적 친분 및 선입견: 관리자와 근로자 간의 개인적인 관계가 있는 경우 관리자의 선입견이 알게 모르게 평가에 반영될 수 있다.

이러한 방식은 관리자의 경험과 노하우에 따라 유연하게 대처할 수 있다는 장점이 있지만, 다음과 같은 심각한 문제점들을 야기한다.

1. 평가의 비객관성 및 불공정성: 평가자마다 기준이 다르고, 동일한 근로자에 대해서도 평가 시점이나 상황에 따라 결과가 달라질 수 있다. 이는 평가 결과의 신뢰도를 저하시키고, 근로자들에게 불공정하다는 인식을 심어줄 수 있다[6]. 실제 건설기술인협회에 등록된 기술자는 등급별로 분류되지만, 이외의 등록되지 않은 50% 이상으로 추정되는 기능 인력은 별도의 객관적인 능력 평가 기준 없이 단일한 기준으로 임금이 결정되는 경우가 많다.

2. 인건비 산정의 불균일성: 동일한 기술 수준과 업무를 수행하더라도, 평가자의 주관에 따라 인건비 책정이 달라질 수 있다. 이는 근로자 간의 임금 격차를 발생시켜

불만을 야기하고, 조직 전체의 사기를 저하시킨다. 실제 건설 현장에서는 숙련공이라도 하루 일당이 작업 현장, 발주처, 협력업체에 따라 수만원씩 차이가 나는 경우가 비일재하다.
3. 인력 배치 및 계획의 비효율성: 객관적인 능력 데이터가 부족하기 때문에, 관리자는 최적의 인력 배치 결정을 내리기 어렵다. 이는 작업 효율성 저하, 재작업 발생, 불필요한 인건비 지출로 이어질 수 있다[26].
- 인적 자원 개발의 어려움: 근로자 개인의 강점과 약점을 명확히 파악하기 어려워 맞춤형 교육 훈련 계획을 수립하기 어렵다. 이는 장기적인 관점에서 건설 산업의 숙련 기술인력 양성을 저해하는 요인이 된다.
- 분쟁 발생 가능성: 불투명하고 주관적인 평가 및 보상 체계는 근로자와 고용주 간의 분쟁, 노동조합과의 갈등을 유발할 수 있는 잠재적인 원인이 된다.

2.2.2 공식 노임단가 제도의 한계

대한민국 건설 산업에서는 대한건설협회가 매년 '건설업 임금실태조사'를 통해 공종별 임금(노임단가)을 발표하고, 이를 정부 공사 예정가격 작성의 기초 자료로 활용한다[27]. 이는 공사비 산정의 투명성을 높이고, 최소한의 임금 기준을 제시한다는 긍정적인 측면이 있다. 그러나 이 제도는 다음과 같은 한계를 가진다.

1. 평균 임금의 대표성 한계: 공표되는 노임단가는 해당 직종의 평균 임금이며, 실제 현장에서의 개별 근로자 능력 차이를 반영하지 못한다. 숙련도가 높은 근로자와 낮은 근로자 모두 동일한 '보통 인부' 또는 '기술자' 임금 범주에 포함될 수 있어, 능력 있는 근로자의 정당한 보상을 어렵게 한다. 이러한 문제는 숙련 근로자의 임금 격차 요인에 대한 연구에서도 지적되며, 개인의 인적 요인보다 직무 특성이나 작업 환경과 같은 비인적 요인이 임금 격차에 더 큰 영향을 미칠 수 있음을 보여준다[28]. 또한, 건설업 임금실태조사의 개선 방안에 대한 연구에서는 건설 근로자들이 비정규직이 많고 공정에 따라 변동이 잦아 정확한 총인원 파악이 어려워 평균 임금 산정의 한계가 있음을 언급하고 있다[29].

2. 능력 평가의 부재: 노임단가 제도는 근로자의 능력이나 생산성 향상에 대한 직접적인 인센티브를 제공하지 못한다. 이는 근로자들이 자신의 역량을 개발하거나 더

높은 생산성을 발휘하려는 동기를 저해할 수 있다. 건설업의 적정 임금제 도입에 대한 연구에서도 현행 시중 노임단가가 시장의 노동 수급 상황을 온전히 반영하지 못하며, 숙련도 반영이 미흡하여 기능 인력 등급제와 같은 능력 평가 연계 방안이 필요함을 제기하고 있다[30].

3. 급변하는 현장 상황 미반영: 건설 현장은 작업의 복잡성, 난이도, 위험성 등이 프로젝트마다, 심지어 동일 프로젝트 내에서도 시기마다 크게 달라진다. 그러나 노임단가는 이러한 미세한 현장 상황을 실시간으로 반영하기 어렵다. 시공 및 현장 여건에 따라 단가의 차이가 매우 다양해질 수 있음에도 불구하고, 이를 표준적인 단가집으로 해결하는 것은 근본적으로 불가능하다는 지적이 있다[31]. 또한, 건설 현장의 유연한 근무 형태와 유동적인 작업 환경으로 인해 정확한 근태 관리 및 수당 정산이 어렵다는 문제점도 노임단가 제도의 유연성 부족과 연결될 수 있다[32].

이러한 문제점들을 종합적으로 고려할 때, 건설 산업은 근로자의 능력을 객관적으로 평가하고, 그에 상응하는 인건비를 공정하게 산정할 수 있는 새로운 접근 방식이 절실히 요구된다. 머신러닝 기반의 평가 시스템은 이러한 요구를 충족시키고, 건설 현장의 인력 관리 효율성과 공정성을 획기적으로 개선할 수 있는 유력한 대안이 될 수 있다.

2.3 주요 머신러닝 알고리즘 소개

본 연구에서는 건설 근로자의 능력 등급 분류를 위해 결정 트리(Decision Tree), 인공신경망(Artificial Neural Network, ANN), 랜덤 포레스트(Random Forest)의 세 가지 머신러닝 알고리즘을 사용한다. 이들 알고리즘은 분류 문제 해결에 널리 사용되고 있으며, 각각 고유한 장단점을 가지고 있어 본 연구의 목적에 부합하는 모델을 선정하는 데 적합하다.

2.3.1 결정 트리 (Decision Tree)

결정 트리는 의사결정 규칙을 나무(Tree) 구조로 시각화하여 분류 또는 회귀 문

제를 해결하는 알고리즘이다[12]. 데이터의 특징(Feature)을 기반으로 질문을 던져 데이터를 점진적으로 분할하는 방식으로 작동하며, 최종적으로는 특정 클래스(분류) 또는 값(회귀)을 예측한다.

1. 작동 원리: 결정 트리는 데이터를 가장 잘 분리할 수 있는 최적의 특징과 분할 기준을 찾아 반복적으로 데이터를 분할한다. 이러한 분할은 노드(Node)라고 불리는 지점에서 이루어지며, 루트 노드(Root Node)에서 시작하여 리프 노드(Leaf Node)에 도달할 때까지 이어진다. 각 노드에서는 엔트로피(Entropy)나 지니 불순도(Gini Impurity)와 같은 불순도(Impurity) 지표를 최소화하는 방향으로 분할이 이루어진다.

가. 엔트로피 (E): 정보 이론에서 불확실성의 척도로 사용된다. 데이터 집합 D에 대한 엔트로피는 다음과 같이 계산된다:

$$E(D) = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

여기서 c는 클래스의 개수이고, p_i는 D 내에서 i번째 클래스에 속하는 샘플의 비율이다. 엔트로피가 낮을수록 데이터의 순도(Homogeneity)가 높다는 것을 의미한다.

나. 지니 불순도 (G): 데이터 집합 D에서 무작위로 선택된 샘플이 잘못 분류될 확률을 측정한다.

$$G(D) = 1 - \sum_{i=1}^{C} p_i^2$$

지니 불순도 역시 낮을수록 노드의 순도가 높다.

다. 정보 이득 (Information Gain): 특정 특징으로 데이터를 분할했을 때 얻을 수 있는 불순도 감소량을 의미한다. 결정 트리는 정보 이득을 최대화하는 방향으로 특징을 선택하고 분할한다.

$$IG(D,A) = E(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|Dvert} E(D_v)$$

여기서 IG(D,A)는 데이터 집합 D를 특징 A로 분할했을 때의 정보 이득, Values(A)는 특징 A가 가질 수 있는 값들의 집합, D_v는 A의 값이 v인 D의 부분 집합이다.

2. 장점으로는 트리 구조가 직관적이고 시각화하기 용이하여 모델의 의사결정 과정을 쉽게 이해하고 해석할 수 있다[29]. 결측치나 이상치에 비교적 강건하며, 데이터 스케일링이 필수적이지 않다. 명확한 의사결정 규칙을 제공하므로, 비전문가도 결과를 이해하기 쉽다.

3. 단점으로는 단일 결정 트리는 과적합(Overfitting)의 위험이 크다. 즉, 학습 데이터에 너무 잘 맞춰져 새로운 데이터에 대한 일반화 성능이 떨어진다[20]. 데이터의 작은 변화에도 트리 구조가 크게 변동할 수 있어 불안정하다.

2.3.2 인공신경망 (Artificial Neural Network, ANN)

인공신경망은 인간 뇌의 신경망 구조를 모방하여 만들어진 머신러닝 모델이다[35]. 입력층(Input Layer), 은닉층(Hidden Layer), 출력층(Output Layer)으로 구성되며, 각 층의 노드(뉴런)들은 가중치(Weight)와 편향(Bias)으로 연결되어 정보를 전달한다.

1. 작동 원리:
가. 순전파 (Forward Propagation): 입력 데이터가 입력층을 통해 들어오면, 각 노드는 이전 층의 노드들로부터 전달받은 값에 가중치를 곱하고 편향을 더한 후, 활성화 함수(Activation Function)를 통과시켜 다음 층으로 전달한다.

$$z_j = \sum_i w_{ij} x_i + b_j$$

$$a_j = f(z_j)$$

여기서 x_i는 이전 층의 i번째 노드 출력, w_ij는 i번째 노드에서 j번째 노드로의

가중치, b_j는 j번째 노드의 편향, f는 활성화 함수이다.
나. 활성화 함수: 뉴런의 출력을 비선형적으로 변환하여 신경망이 복잡한 패턴을 학습할 수 있도록 한다. 대표적인 활성화 함수로는 ReLU (Rectified Linear Unit), Sigmoid, Tanh 등이 있다.
다. 손실 함수 (Loss Function): 모델의 예측값과 실제 정답 간의 오차를 측정한다. 분류 문제에서는 주로 교차 엔트로피(Cross-Entropy) 손실 함수가 사용된다.
라. 역전파 (Backpropagation): 손실 함수의 오차를 줄이기 위해 가중치와 편향을 업데이트하는 과정이다. 출력층에서 발생한 오차를 입력층 방향으로 역전파시키면서 각 연결의 가중치와 편향을 경사 하강법(Gradient Descent)을 통해 조정한다[28].

$$w_{\neq w} = w_{old} - a\frac{\partial L}{\partial w}$$

여기서 alpha는 학습률(Learning Rate)이고, fracpartialLpartialw는 손실 함수 L을 가중치 w에 대해 미분한 값(기울기)이다.

2. 장점으로는 복잡한 비선형 관계를 학습하고 모델링하는 데 매우 강력하다.
충분한 데이터와 적절한 구조가 주어지면 높은 예측 정확도를 달성할 수 있다.
다양한 데이터 유형(이미지, 텍스트, 정형 데이터)에 적용 가능하다.

3. 단점으로는 모델의 내부 작동 방식이 복잡하여 해석하기 어렵다 (블랙박스 모델). 왜 특정 예측이 나왔는지 설명하기가 어렵다. 학습에 많은 데이터와 컴퓨팅 자원이 필요하며, 과적합 위험이 존재한다. 하이퍼파라미터(은닉층의 개수, 뉴런 수, 학습률 등) 설정이 어렵고 성능에 큰 영향을 미친다

2.3.3 랜덤 포레스트 (Random Forest)

랜덤 포레스트는 앙상블 학습(Ensemble Learning) 기법 중 하나인 배깅(Bagging, Bootstrap Aggregating)을 기반으로 하는 알고리즘으로, 여러 개의 결정 트리를 생성하고 이들의 예측을 결합하여 최종 예측을 수행한다[37]. 이는 단일 결정 트리의 과적합 문제를 해결하고 예측 성능을 향상시키는 데 매우 효과적이다.

1. 작동 원리:
가. 부트스트랩 샘플링 (Bootstrap Sampling): 원본 학습 데이터셋으로부터 복원 추출(Replacement) 방식으로 여러 개의 부트스트랩 샘플을 생성한다. 각 샘플은 원본 데이터셋과 동일한 크기를 가지지만, 일부 데이터는 중복되고 일부 데이터는 포함되지 않을 수 있다.
나. 랜덤 특징 선택 (Random Feature Selection): 각 부트스트랩 샘플에 대해 결정 트리를 생성할 때, 모든 특징(변수)을 고려하는 대신 무작위로 선택된 특징의 부분 집합만을 사용하여 트리를 분할한다. 이 두 가지 무작위성(데이터 샘플링, 특징 샘플링)은 개별 트리의 다양성을 높여 과적합을 방지하는 데 기여한다.

다. 트리 성장 및 예측 결합: 생성된 각 결정 트리는 독립적으로 학습된다. 분류 문제의 경우, 각 트리의 예측 결과를 다수결 투표(Voting) 방식으로 취합하여 최종 분류를 결정한다. 회귀 문제의 경우, 각 트리의 예측 값들을 평균하여 최종 예측 값을 산출한다.

2. 장점으로는
가. 여러 트리의 예측을 결합함으로써 단일 트리에 비해 훨씬 높은 정확도를 보인다.
나. 과적합 방지: 부트스트랩 샘플링과 랜덤 특징 선택을 통해 과적합 위험을 크게 줄인다.
다. 변수 중요도 제공: 각 특징이 예측에 얼마나 기여했는지에 대한 중요도(Feature Importance)를 계산하여 제공하므로, 모델의 해석 가능성을 높일 수 있다[38].
라. 결측치나 이상치에 비교적 강건하며, 대용량 데이터 처리에도 효율적이다.

3. 단점으로는
가. 단일 결정 트리에 비해 모델의 내부 구조가 복잡하여 직관적인 이해가 어렵다.
나. 학습 시간이 상대적으로 길 수 있으며, 생성되는 트리의 개수에 따라 메모리 사용량이 많아질 수 있다.

본 연구에서는 이 세 가지 알고리즘의 장단점을 고려하여, 건설 근로자 능력

평가 및 인건비 산정이라는 본 연구의 목표에 가장 적합한 모델을 식별하고자 한다. 특히, 예측 정확도뿐만 아니라 모델의 해석 가능성 또한 중요한 평가 기준으로 고려하여 실무 적용 가능성을 높이는 데 중점을 둔다.

2.4 관련 연구 동향

건설 산업에서의 인력 관리 및 성과 평가는 오랜 기간 중요한 이슈였으나, 머신러닝을 활용한 정량적이고 객관적인 접근 방식은 비교적 최근에 주목받기 시작했다. 기존 연구들은 주로 인력의 효율적 배치, 생산성 예측, 위험 관리에 초점을 맞추었으며, 근로자 개개인의 능력 평가 및 인건비 산정에 직접적으로 머신러닝을 적용한 연구는 아직까지 활발하게 이루어지지 않았다.

2.4.1 건설 분야 인력 및 성과 관련 선행 연구

1. 생산성 예측 및 영향 요인 분석:

가. 장현상 외 (2019)는 인공신경망을 활용하여 콘크리트 타설 공정의 생산성을 예측하고, 날씨, 인력 투입량, 장비 효율성 등 다양한 요인이 생산성에 미치는 영향을 분석하였다[39]. 이 연구는 특정 공정의 효율성 향상에 기여했으나, 개별 근로자의 능력 평가에 직접적인 초점을 맞추지는 않았다.

나. 김현준 외 (2020)는 건설 프로젝트의 생산성 저해 요인을 식별하기 위해 데이터 마이닝 기법을 적용하였다[40]. 설문조사 데이터를 기반으로 의사결정 트리를 활용하여, 의사소통 부족, 낮은 숙련도, 장비 문제 등이 생산성 저하에 미치는 영향을 분석했다. 이는 생산성 저해 요인을 진단하는 데 기여했지만, 근로자 개개인의 능력 등급을 분류하는 데 활용되지는 않았다.

다. Yuan et al. 은 머신러닝을 사용하여 건설 프로젝트의 노동 생산성을 예측하는 모델을 개발했다[41]. 이들은 프로젝트 특성, 환경 요인, 인력 관련 변수 등을 입력으로 사용하여 생산성 예측의 정확도를 높였다. 그러나 이는 전반적인 생산성 예측

에 중점을 두었으며, 개별 근로자의 능력 등급 분류가 아닌 프로젝트 단위의 노동 생산성 예측에 주력하였다.

2) 인력 배치 및 자원 최적화:

가. Choi et al. 은 유전 알고리즘(Genetic Algorithm)을 사용하여 건설 프로젝트의 최적 인력 배치를 위한 모델을 개발하였다[42]. 이 연구는 인력의 능력과 작업 요구 사항을 매칭하는 데 중점을 두었으나, 능력 평가 자체를 위한 머신러닝 모델 구축과는 거리가 있었다.

나. Liu et al.은 딥러닝 기반의 인력 스케줄링 모델을 제안하여, 건설 현장의 복잡한 제약 조건을 고려한 최적의 인력 운영 방안을 제시했다[43]. 이 또한 효율적인 인력 운영에 초점을 맞추었을 뿐, 근로자 개개인의 숙련도 및 인건비 산정 문제는 직접적으로 다루지 않았다.

3) 안전 및 역량 평가:

가. Gwak et al.은 건설 현장 안전 관리 시스템 강화를 위해 작업자의 생체 신호 및 행동 데이터를 기반으로 위험 예측 모델을 개발하였다[44]. 이는 작업자의 '능력'이라기보다는 '안전 행동'에 대한 평가에 가깝다.

나. 이정현과 김경환 (2023)은 BIM(Building Information Modeling) 데이터를 활용하여 건설 근로자의 역량을 평가하는 프레임워크를 제안했다[45]. 이는 주로 설계 및 시공 단계에서의 정보 활용 능력을 평가하는 데 초점을 맞추고 있으며, 본 연구와 같이 작업 난이도 및 수행 시간을 기반으로 한 전반적인 능력 평가와는 차이가 있다.

다. Shi et al.은 건설 현장 작업자의 숙련도를 객관적으로 평가하기 위한 AI 기반 시스템의 필요성을 강조하고, 센서 데이터를 활용한 작업 모션 분석 방법을 탐색했다[46]. 이는 본 연구와 유사한 방향성을 가지지만, 특정 작업의 모션 분석에 초점을 맞추어 일반적인 능력 등급 분류 및 인건비 산정 연계까지는 다루지 않았다.

라. 특히, Jacobsen et al. 은 심층 학습을 활용하여 작업자의 운동학적 데이터를 기반으로 작업 활동을 인식하고 생산성을 모니터링하는 방법을 개발했다[24]. 이 연구는 특정 작업을 수행하는 데 걸리는 시간을 정량적으로 측정하는 데 활용될 수 있다는 점에서 본 연구의 '수행 시간' 지표 활용과 맥락을 같이 한다.

이러한 연구들은 데이터 기반 접근이 근로자 관리의 효율성과 공정성을 높일 수 있음을 보여주며, 본 연구 또한 이러한 방향성에 기반해 실무 적용 가능성을 모색하여 제시한다.

4) 건설 비용 예측:

강윤호와 윤석헌 (2022)은 머신러닝 기법을 활용하여 건축 공사비 예측 성능을 분석하였다[18]. 이 연구는 공사비 예측이라는 재무적 측면에 집중했지만, 머신러닝 모델의 성능을 향상시키기 위한 데이터 전처리 방법(피처 스케일링, 로그 변환)의 중요성을 강조하며, 이는 건설 산업 데이터 분석 전반에 대한 시사점을 제공한다.

2.4.2 본 연구의 차별성

위에서 언급된 선행 연구들은 건설 산업의 생산성 향상, 인력 배치 효율화, 안전 관리 등 다양한 측면에서 머신러닝 및 데이터 분석 기술의 적용 가능성을 보여주었다. 이처럼 기존 연구들은 머신러닝을 활용하여 건설 근로자의 숙련도 분류나 임금 예측과 같은 정량적 분석에 초점을 맞추고 있다. 하지만 본 연구는 이와 유사한 기술 기반을 공유하면서도 다음과 같은 점에서 차별성을 갖는다.

첫째, 기존 연구가 주로 근로자의 생산성이나 기술 숙련도를 중심으로 분류하는 반면, 본 연구는 작업 난이도와 실제 수행 시간이라는 정량적 지표를 조합하여 근로자의 능력치를 정의하고, 이를 기반으로 능력 등급화와 이를 인건비 산정에 접목하여 체계를 구축한다는 점에서 평가 기준의 구조적 차이를 가진다. 이는 불확실하고 주관적인 평가를 배제하고 객관적인 데이터 기반의 평가 시스템을 구축하는 데

기여한다.

둘째, 단일 머신러닝 모델을 적용한 기존 연구와 달리, 본 연구는 결정 트리, 인공신경망(ANN), 랜덤 포레스트 등 다양한 모델의 성능을 비교 분석하고, 각 모델의 해석 가능성과 예측력을 동시에 고려하여 최적의 접근법을 제시한다. 특히, 랜덤 포레스트의 변수 중요도 분석을 통해 주요 평가 지표의 영향력을 시각적으로 제시하여 모델의 신뢰성을 높인다.

셋째, 본 연구는 기술적 정확성뿐 아니라 경영적 활용성과 실무 적용 가능성에 중점을 둔 점에서 차별화된다. 즉, 근로자 능력 평가 결과를 단순한 예측 도구로 활용하는 데 그치지 않고, 실제 인력 운영, 공정한 보상체계 설계, 고용주-근로자 간 신뢰 제고 등 조직 관리 측면까지 확장하여 고찰한다는 데 본 연구의 의의가 있다.

넷째, 가상 데이터 생성의 실제성 반영 노력: 실제 현장 데이터 확보의 어려움을 극복하기 위해 가상의 데이터를 사용하지만, 이 데이터가 실제 건설 현장의 인력 특성 및 작업 패턴을 반영할 수 있도록 정교한 생성 기준과 논리를 제시한다. 이는 연구 결과의 신뢰성과 타당성을 확보하는 데 중요한 역할을 한다.

이러한 점에서 본 연구는 기존 머신러닝 기반 건설 근로자 분석 연구들을 이론적 기반으로 삼되, 이를 발전시켜 보다 실천적이고 통합적인 인적자원관리 시스템 구축이라는 방향성을 제시한다.

3 연구 방법론

3.1 데이터 수집 및 가공

본 연구는 건설 근로자의 능력 평가 및 인건비 산정을 위한 머신러닝 모델 개발을 목표로 한다. 이를 위해서는 근로자의 작업 활동 및 능력에 대한 정량적 데이터가 필수적이다. 그러나 건설 현장의 특성상 이러한 데이터를 실제 현장에서 직접 수집하는 것은 현실적으로 여러 가지 어려움이 따른다.

3.1.1 실제 데이터 수집의 한계점 및 가상 데이터 생성의 필요성

건설 현장에서 근로자의 작업 능력 및 성과에 대한 정량적인 데이터를 확보하는 것은 다음과 같은 현실적인 문제점들로 인해 매우 어렵다.

1. 데이터 비공개성 및 접근의 어려움: 건설 기업들은 근로자의 임금 정보, 작업 효율성 등 민감한 인력 관련 데이터를 외부에 공개하기를 꺼려하며, 이는 연구를 위한 데이터 접근에 큰 장벽이 된다. 기업 내부적으로도 이러한 데이터를 체계적으로 관리하고 축적하는 시스템이 미흡한 경우가 많다.

2. 측정의 어려움 및 비용: 근로자 개개인의 작업 난이도와 정확한 수행 시간을 정량적으로 측정하기 위해서는 현장에 상주하는 전문 인력이나 고가의 센서, 비전 시스템 등이 필요하며, 이는 상당한 시간과 비용을 수반한다. 또한, 작업 환경의 다양성과 복잡성으로 인해 일관된 측정 기준을 적용하기 어렵다.

3. 측정의 주관성 및 비일관성: 현재의 건설 현장에서는 관리자의 주관적인 판단이나 경험에 의존하여 근로자의 능력을 평가하는 경우가 많다. 이러한 방식은 평가 기준의 비일관성과 평가자 간의 편차를 야기하여 객관적인 데이터로 활용하기 어렵게 만든다.

4. 프라이버시 및 윤리적 문제: 근로자의 작업 활동을 세밀하게 모니터링하고 데이

터를 수집하는 과정에서 개인의 프라이버시 침해 논란이 발생할 수 있으며, 이는 윤리적인 문제로 이어질 수 있다.

이러한 실제 데이터 수집의 한계점들을 고려할 때, 본 연구에서는 가상 데이터를 생성하여 활용하는 것이 불가피하다. 가상 데이터는 실제 현장의 특성과 유사한 분포 및 관계를 갖도록 설계함으로써, 실제 데이터가 없을 때도 머신러닝 모델의 개발 및 검증을 가능하게 한다. 또한, 가상 데이터는 다양한 시나리오와 변수를 통제하여 모델의 성능을 체계적으로 분석하고, 특정 조건에서의 모델 거동을 이해하는 데 유용하다. 이는 실제 데이터 확보 전 단계에서 모델의 유효성을 검증하고, 향후 실제 데이터 적용 시 발생할 수 있는 문제점들을 미리 파악하고 대비하는 데 중요한 역할을 한다.

3.1.2 가상 데이터 생성의 구체적인 기준 및 방법

본 연구에서는 실제 건설 현장 데이터의 수집 및 활용에 대한 현실적인 제약 사항을 고려하여, 머신러닝 모델 학습에 필요한 정량적 데이터를 가상으로 생성하였다. 생성된 데이터는 실제 현장의 특징을 반영하면서도 모델 학습에 충분한 다양성을 확보하도록 설계되었다. 총 10,000개의 가상 데이터 샘플을 생성하였으며, 각 샘플은 건설 근로자의 능력 평가와 인건비 산정에 핵심적인 영향을 미치는 변수들로 구성된다.

1. 핵심 변수의 선정 이유

본 연구에서 선정한 주요 변수들은 건설 현장에서 근로자의 숙련도, 생산성 및 성과에 직접적인 영향을 미치는 요인들로, 다양한 선행 연구에서도 그 중요성이 지속적으로 강조되어 왔다.

가. 작업 난이도 (Task Difficulty): 작업의 본질적인 복잡성이나 요구되는 기술 수준을 나타내는 작업 난이도는 근로자의 성과를 예측하는 데 핵심적인 변수이다. 다수의 연구에서 작업 난이도가 개인의 작업 수행 능력과 직결되며, 난이도가 높을수록 더 많은 시간과 노력이 요구되거나 성능에 영향을 미치는 것으로 분석된다. 건

설 현장의 작업은 복잡성과 위험도가 다양하므로, 작업 난이도는 근로자 능력 평가의 필수적인 지표로 작용한다.

나. 수행 시간 (Task Duration): 특정 작업을 완료하는 데 걸리는 시간은 근로자의 효율성과 숙련도를 가장 직접적으로 보여주는 지표이다. 건설 노동 생산성 관련 연구에서는 노동력 투입 대비 산출물(작업 완료)의 물리적 구성 요소 간의 관계를 분석하여 효율성을 측정하며, 이 과정에서 작업 수행 시간은 핵심적인 지표로 활용된다. 또한, 건설 근로자의 생산성 저하 요인에 대한 연구에서도 작업 시간이 생산성에 미치는 영향이 중요하게 다루어진다. 수행 시간이 짧을수록 높은 숙련도와 효율성을 의미하므로, 본 연구의 능력 평가 모델에서 중요한 예측 변수가 된다.

다. 경력 (Experience): 근로자의 경력은 축적된 경험과 노하우를 나타내며, 일반적으로 경력이 높을수록 숙련도와 문제 해결 능력이 향상된다고 볼 수 있다. 건설업 생산성에 영향을 미치는 요인에 대한 연구들에서는 숙련도와 경험이 노동 생산성의 중요한 결정 요인으로 언급된다.

라. 직무 유형 (Job Type): 건설 현장에는 목공, 철근공, 용접공, 미장공 등 다양한 직무군이 존재하며, 각 직무는 고유의 기술과 숙련도를 요구한다. 직무 유형은 근로자의 전문화된 기술 영역을 구분하는 중요한 변수이다.

마. 능력 등급 (Competency Level): 본 연구의 최종 목표 변수이자, 앞선 변수들의 조합을 통해 예측하고자 하는 근로자의 종합적인 숙련도 및 역량 수준을 나타낸다. 이는 기존의 주관적인 평가를 대체할 정량적 능력 지표로서, 객관적인 인건비 산정의 근거를 제공한다.

이러한 변수들을 활용하여 생성된 가상 데이터는 실제 현장의 복잡성을 반영하면서도, 모델 학습을 위한 충분한 통계적 특성을 갖도록 설계되었다. 각 변수의 구체적인 생성 기준과 방법은 다음과 같다.

2. 변수의 구체적인 생성기준과 방법:

가. 작업 난이도 (Task Difficulty): 특정 작업의 기술적 복잡성, 필요 숙련도, 물리적 힘 소모 정도 등을 종합적으로 고려하여 1부터 10까지의 척도로 정의한다. (1: 매우 쉬움, 10: 매우 어려움) 각 난이도에는 해당 난이도의 작업을 수행하는 데 요구되는 평균적인 시간(기준 시간)이 암묵적으로 연결된다.

- 예시:
난이도 1~3: 간단한 자재 운반, 청소 등 (짧은 기준 시간)
난이도 4~6: 일반적인 철근 배근, 거푸집 설치, 벽돌 쌓기 등 (중간 기준 시간)
난이도 7~10: 복잡한 구조물 용접, 정밀 배관 작업, 고층 작업 등 (긴 기준 시간)

나. 수행 시간 (Task Duration): 근로자가 특정 작업을 완료하는 데 실제로 소요된 시간이다. 이는 작업 난이도에 따른 기준 시간에 근로자의 능력(숙련도)에 따른 편차가 반영되어 생성된다.

다. 능력 등급 (Competency Level): 본 연구에서 예측하고자 하는 목표 변수(레이블)로, 'S', 'A', 'B', 'C', 'D'의 다섯 가지 범주로 정의한다. 각 등급은 근로자가 '작업 난이도 대비 수행 시간을 얼마나 효율적으로 단축시켰는가'에 따라 결정된다.

1) S 등급 (매우 우수): 작업 난이도 대비 수행 시간이 매우 짧고 가장 효율적인 근로자. (예: 기준 시간의 60% 이하)
2) A 등급 (우수): 작업 난이도 대비 수행 시간이 짧고 효율적인 근로자. (예: 기준 시간의 60% 초과 ~ 80% 이하)
3) B 등급 (보통): 작업 난이도 대비 수행 시간이 보통 수준인 근로자. (예: 기준 시간의 80% 초과 ~ 110% 이하)
4) C 등급 (미흡): 작업 난이도 대비 수행 시간이 길고 다소 비효율적인 근로자. (예: 기준 시간의 110% 초과 ~ 140% 이하)
5) D 등급 (매우 미흡): 작업 난이도 대비 수행 시간이 매우 길고 비효율적인 근로자. (예: 기준 시간의 140% 초과)

상기 기준 시간 대비 수행 시간 비율은 실제 현장 전문가 자문 또는 통계적 분석을 통해 더욱 정교하게 설정될 수 있다.

라. 경력 (Experience): 근로자의 총 건설업 종사 기간 (단위: 년). 경력이 높을수록 일반적으로 능력 등급이 높을 확률이 크도록 설정하지만, 예외 케이스(고경력이지만 비효율적인 경우, 신입이지만 뛰어난 경우)도 포함한다.

마. 직무 유형 (Job Type): 근로자의 세부 직무 (예: 목공, 철근공, 용접공, 미장공, 보통인부 등). 직무 유형별로 작업 난이도의 분포가 다를 수 있음을 반영한다.

3. 데이터 생성 절차:

총 10,000개의 가상 데이터를 생성하며, 각 데이터 포인트는 한 근로자의 특정 작업 수행 기록으로 구성된다.

가. 작업 난이도 생성: 1부터 10까지의 정수형 작업 난이도를 무작위로 생성하되, 중간 난이도(4~7)의 빈도를 높여 실제 현장의 다양한 작업을 반영한다.

나. 기준 수행 시간 설정: 각 작업 난이도에 해당하는 가상의 기준 수행 시간을 설정한다. (예: 난이도 1 = 10분, 난이도 5 = 60분, 난이도 10 = 120분 등) 이는 난이도에 따라 선형적 또는 비선형적으로 증가하도록 설정할 수 있다.

다. 근로자 능력 분포 반영: 5단계 능력 등급의 데이터 분포는 다음과 같이 설정한다. 이 분포 비율은 일반적인 인력 구성의 종형 곡선을 반영하며, 모델 학습에 필요한 충분한 수의 각 등급 데이터를 확보하기 위함이다.
- S 등급 (매우 우수): 전체 데이터의 약 10%
- A 등급 (우수): 전체 데이터의 약 20%
- B 등급 (보통): 전체 데이터의 약 40%
- C 등급 (미흡): 전체 데이터의 약 20%
- D 등급 (매우 미흡): 전체 데이터의 약 10%

라. 수행 시간 생성: 각 근로자의 능력 등급에 따라 해당 작업의 기준 수행 시간에 무작위 편차를 더하여 최종 '수행 시간'을 생성한다.
- S 등급 근로자: 기준 시간 × (0.5 ~ 0.7 사이의 무작위 값) + 약간의 노이즈

- A 등급 근로자: 기준 시간 × (0.7 ~ 0.9 사이의 무작위 값) + 약간의 노이즈
- B 등급 근로자: 기준 시간 × (0.9 ~ 1.2 사이의 무작위 값) + 약간의 노이즈
- C 등급 근로자: 기준 시간 × (1.2 ~ 1.5 사이의 무작위 값) + 약간의 노이즈
- D 등급 근로자: 기준 시간 × (1.5 ~ 2.0 사이의 무작위 값) + 약간의 노이즈

여기서 노이즈는 실제 현장의 예측 불가능한 변수(예: 작은 돌발 상황, 장비 대기 등)를 반영하기 위함이다.

마. 경력 및 직무 유형 생성:

경력은 0년에서 30년 사이의 정수형으로 무작위 생성하되, 'S', 'A' 등급 근로자에게는 높은 경력이 부여될 확률을 높이고, 'C', 'D' 등급 근로자에게는 낮은 경력이 부여될 확률을 높이는 경향성을 반영한다. 단, 이 경향성이 항상 일치하지 않도록 예외적인 경우도 포함한다.

직무 유형은 5~10가지 정도로 정의하고 무작위로 할당한다. 직무 유형이 직접적인 능력 등급을 결정하기보다는, 특정 작업 난이도와 연결될 수 있는 간접적인 변수로 작용하도록 설정한다.

바. 능력 등급 할당: 생성된 '작업 난이도'와 '수행 시간'의 관계를 앞서 정의한 능력 등급 기준(기준 시간 대비 수행 시간 비율)에 따라 'S', 'A', 'B', 'C', 'D'로 최종 할당한다. 이는 모델이 학습할 정답(레이블)이 된다.

3) 데이터 전처리 (Data Preprocessing)

생성된 가상 데이터는 머신러닝 모델 학습에 적합한 형태로 전처리된다.

가. 결측치 처리: 본 연구에서는 생성 단계에서 결측치가 발생하지 않도록 설계한다.

나. 이상치 처리: 데이터 생성 시 무작위성을 부여하되, 극단적인 이상치가 발생하지 않도록 범위를 조절한다.

다. 범주형 변수 인코딩 (Categorical Variable Encoding): '직무 유형'과 같은 범주

형 변수는 머신러닝 모델이 이해할 수 있는 숫자형 형태로 변환해야 한다. One-Hot Encoding 또는 Label Encoding 방식을 활용한다. 본 연구에서는 직무 유형의 경우 각 유형 간 순서 관계가 없으므로 One-Hot Encoding을 적용할 예정이다.

라. 데이터 정규화/스케일링 (Normalization/Scaling): '작업 난이도'와 '수행 시간', '경력'과 같이 스케일이 다른 숫자형 변수들의 경우, 모델 학습의 안정성과 성능 향상을 위해 Min-Max Scaling 또는 Standard Scaling을 적용할 수 있다. 본 연구에서는 각 알고리즘의 특성을 고려하여 적용 여부를 결정한다 (예: 결정 트리 계열은 스케일링에 덜 민감함, 인공신경망은 민감함).

이러한 과정을 통해 생성된 가상 데이터셋은 건설 근로자의 능력 평가에 필요한 핵심 요인들을 포함하며, 머신러닝 모델 학습 및 검증을 위한 신뢰성 있는 기반 자료가 될 것이다.

3.2 연구 모델 선정

본 연구에서는 건설 근로자의 능력 등급을 효과적으로 분류하기 위해 2.3절에서 소개한 머신러닝 알고리즘 중 결정 트리(Decision Tree), 인공신경망(Artificial Neural Network, ANN), 랜덤 포레스트(Random Forest) 세 가지 모델을 선정하여 비교 분석한다. 이들 모델은 분류 문제 해결에 널리 활용되며, 각기 다른 학습 메커니즘과 특성을 가지고 있어 다양한 관점에서 모델의 성능과 해석 가능성을 평가하는 데 적합하다.

3.2.1 모델 선정 배경

각 모델을 선정한 구체적인 배경은 다음과 같다.

가. 결정 트리 (Decision Tree):
1) 장점: 직관적인 의사결정 과정을 제공하며, 결과에 대한 해석이 용이하다는 점이

큰 장점이다. 건설 현장 관리자가 모델의 판단 기준을 쉽게 이해하고 수용할 수 있도록 돕는다는 점에서 실무 적용 가능성을 고려할 때 중요한 모델이다.
2) 고려 사항: 단일 결정 트리는 과적합(Overfitting)의 위험이 있어, 학습 데이터에는 매우 잘 맞지만 새로운 데이터에 대한 예측 성능이 떨어질 수 있다. 따라서 모델 학습 시 과적합을 방지하기 위한 적절한 가지치기(Pruning) 기법 적용이 중요하다.

나. 인공신경망 (Artificial Neural Network, ANN):
1) 장점: 복잡한 비선형 관계를 학습하는 데 매우 뛰어나며, 충분한 데이터와 적절한 구조가 주어졌을 때 높은 예측 정확도를 기대할 수 있다. 작업 난이도, 수행 시간, 경력 등 다양한 변수 간의 복잡한 상호작용을 포착하는 데 유리하다.
2) 고려 사항: '블랙박스' 모델로 불릴 만큼 내부 작동 방식을 해석하기 어렵다는 단점이 있다. 또한, 최적의 성능을 위한 은닉층 수, 뉴런 수, 학습률 등의 하이퍼파라미터 튜닝이 매우 중요하며, 과적합 방지를 위한 정규화 기법(예: 드롭아웃) 적용이 필수적이다.

다. 랜덤 포레스트 (Random Forest):
1) 장점: 다수의 결정 트리를 앙상블(Ensemble)하여 단일 결정 트리의 과적합 문제를 해결하고, 안정적이면서도 높은 예측 정확도를 제공한다. 또한, 각 변수의 중요도(Feature Importance)를 계산하여 모델의 예측에 어떤 변수가 더 큰 영향을 미쳤는지 파악할 수 있어, 어느 정도의 해석 가능성을 제공한다. 이는 모델의 신뢰성을 높이고, 능력 평가 기준에 대한 이해를 돕는다.
2) 고려 사항: 학습 시간이 비교적 길고, 모델 구조가 복잡하다는 단점이 있다.
세 가지 모델을 모두 적용하고 성능을 비교함으로써, 본 연구는 예측 정확도, 모델 해석 가능성, 그리고 실무 적용 용이성이라는 세 가지 주요 기준을 종합적으로 고려하여 최적의 머신러닝 접근법을 제시할 수 있을 것이다.

3.2.2 데이터 분할 (Data Splitting)

머신러닝 모델의 성능을 적절하게 평가하기 위해서는 모델 학습에 사용된 데이

터와 평가에 사용될 데이터가 명확히 분리되어야 한다. 이는 모델이 학습 데이터에만 과적합(Overfitting)되어 실제 환경의 새로운 데이터에 대한 예측 성능이 저하되는 것을 방지하기 위함이다. 데이터셋의 규모와 모델의 복잡성 등을 고려하여 훈련 데이터셋과 테스트 데이터셋을 7:3 또는 8:2 비율로 분할하는 것이 보편적으로 사용되는 방법이다.

가. 학습 데이터셋 (Training Dataset): 10,000개의 충분한 가상 데이터를 생성하였으며, 모델의 일반화 성능을 충분히 검증하기 위하여 7:3의 비율을 선택하였다. 전체 데이터의 70% (7,000개)를 모델 학습에 사용한다. 모델은 이 데이터를 통해 입력 변수(작업 난이도, 수행 시간, 경력, 직무 유형)와 목표 변수(능력 등급: S, A, B, C, D) 간의 패턴과 관계를 학습한다.

나. 평가 데이터셋 (Test Dataset): 전체 데이터의 30% (3,000개)를 모델의 성능을 평가하는 데 사용한다. 이 데이터는 모델이 학습 과정에서 한 번도 접하지 않은 새로운 데이터이므로, 모델의 실제 예측 능력(일반화 성능)을 객관적으로 측정할 수 있다.

다. 교차 검증 (Cross-Validation): 과적합 방지 및 모델 성능의 신뢰성을 높이기 위해 K-겹 교차 검증(K-Fold Cross-Validation) 기법을 활용할 수 있다. 예를 들어, 5-겹 교차 검증을 사용하면 학습 데이터를 5개의 폴드(Fold)로 나누고, 각 폴드를 한 번씩 검증 데이터로 사용하여 모델을 학습하고 평가함으로써, 특정 데이터셋에 과적합되는 것을 방지하고 모델 성능의 안정성을 확보한다.

3.2.3 모델별 하이퍼파라미터 설정 및 튜닝

각 머신러닝 모델의 성능은 하이퍼파라미터 설정에 크게 좌우되며, 이는 모델의 학습 과정과 최종 예측 능력에 결정적인 영향을 미친다. 본 연구에서는 각 모델이 가상 데이터셋에서 발휘할 수 있는 최대 성능을 도출하기 위해, 그리드 서치(Grid Search) 기법을 활용하여 체계적으로 하이퍼파라미터 튜닝을 수행할 예정이다. 그리드 서치는 사전에 정의된 각 하이퍼파라미터 값들의 모든 가능한 조합을 시도하

고, 교차 검증(Cross Validation)을 통해 최적의 성능을 보이는 조합을 선택하는 방식으로 진행된다.

각 모델에 대해 탐색할 하이퍼파라미터의 범위 및 기준은 다음과 같다.

가. 결정 트리 (Decision Tree): 결정 트리는 단일 모델로서 과적합(Overfitting)에 취약하기 때문에, 모델의 복잡도를 적절히 제어하여 일반화 성능을 확보하는 것이 중요하다.

1) max_depth(트리의 최대 깊이): 트리의 최대 깊이를 제한하여 과적합을 방지하고 모델의 해석 가능성을 유지한다. 일반적으로 데이터셋의 복잡성과 과적합 방지를 고려하여 너무 깊지 않으면서도 충분한 패턴을 학습할 수 있는 3에서 10 사이의 값들이 초기 탐색 범위로 권장된다.

2) min_samples_leaf(리프 노드가 되기 위한 최소 샘플 수): 리프 노드가 되기 위해 필요한 최소한의 샘플 수를 지정하여 과적합을 방지하고 일반적인 패턴을 학습하도록 한다. 이 파라미터는 노드 분할을 엄격하게 하여 미세한 패턴에 과도하게 반응하는 것을 줄여주며, 데이터셋 크기에 따라 5에서 20 사이의 값들이 일반적인 탐색 범위로 사용된다.(예: 5 ~ 20 사이의 값들을 탐색)

3) criterion(불순도 측정 기준): 트리가 분할될 때 노드의 불순도를 측정하는 기준을 선택한다. 'gini' 불순도는 계산이 빠르고 'entropy'는 정보 이득을 최대화하는 경향이 있어, 이 두 가지 기준은 결정 트리 모델에서 가장 보편적으로 사용되는 불순도 측정 방법이다. ('gini' 또는 'entropy' 중에서 탐색)

나. 인공신경망 (Artificial Neural Network, ANN): 인공신경망은 비선형적인 데이터 패턴 학습에 강력하지만, 그만큼 모델 구조와 학습 과정에 대한 하이퍼파라미터 튜닝이 매우 중요하다.

1) hidden_layer_sizes(은닉층의 개수 및 각 은닉층의 뉴런 수): 데이터의 복잡한 비선형 관계를 학습할 수 있는 적절한 네트워크 깊이와 폭을 찾기 위해 다양한 은닉

층 구조를 탐색한다. 은닉층의 크기와 개수는 모델의 표현력과 학습 능력에 직접적인 영향을 미치며, 일반적으로 데이터의 복잡도에 따라 (50,), (100, 50), (100, 50, 25)와 같은 다양한 조합을 탐색하여 최적의 균형을 찾는다.

2) activation(활성화 함수): 각 뉴런의 출력을 비선형적으로 변환하는 함수를 선택한다. 'relu'는 기울기 소실 문제를 완화하고 계산 효율성이 높아 가장 널리 사용되며, 'sigmoid'와 'tanh' 또한 이진 분류나 특정 범위의 출력이 필요할 때 유용하게 활용되는 표준 활성화 함수들이다.

3) solver(가중치 최적화 방법): 신경망 학습 시 손실 함수를 최소화하기 위해 가중치와 편향을 업데이트하는 알고리즘을 선택한다. 'adam'은 적응적 학습률을 사용하여 수렴 속도가 빠르고 성능이 우수하여 기본 옵티마이저로 자주 사용되며, 'sgd'(Stochastic Gradient Descent)는 기본적인 최적화 기법으로 다른 변형 옵티마이저의 비교 기준으로 활용된다.

4) learning_rate_init (초기 학습률): 모델이 최적화 과정에서 가중치를 업데이트하는 속도를 조절한다. 학습률은 모델의 수렴 속도와 최종 성능에 결정적인 영향을 미치므로, 일반적으로 작은 값(예: 0.0001)부터 큰 값(예: 0.1)까지 로그 스케일로 탐색하여 최적의 학습 속도를 찾는다.

5) max_iter(최대 에포크(Epoch) 수): 모델이 전체 학습 데이터셋을 반복하여 학습할 최대 횟수를 설정한다. 에포크 수는 모델이 충분히 학습될 시간을 제공하면서도 과도한 학습으로 인한 과적합을 방지할 수 있도록, 데이터셋의 크기와 복잡성에 따라 200에서 1000 사이의 범위에서 탐색하는 것이 일반적이다.

6) alpha: (L2 정규화(Regularization) 강도): 모델의 과적합을 방지하고 일반화 성능을 향상시키기 위한 정규화 강도를 조정한다. L2 정규화는 가중치 크기를 제한하여 모델의 복잡도를 줄이는 역할을 하며, 일반적으로 0.0001부터 0.01 사이의 작은 값들을 로그 스케일로 탐색하여 적절한 정규화 강도를 찾는다.

다. 랜덤 포레스트 (Random Forest): 랜덤 포레스트는 다수의 결정 트리를 앙상블

하여 예측력을 높이고 과적합을 줄이는 모델이다. 개별 트리의 설정과 앙상블 방식에 대한 파라미터가 중요하다.

1) n_estimators(생성할 결정 트리의 개수): 랜덤 포레스트를 구성할 개별 결정 트리의 총 개수를 설정한다. 앙상블 모델의 예측 안정성을 높이기 위해 충분히 많은 트리를 사용하는 것이 좋지만, 너무 많은 트리는 계산 비용만 증가시키고 성능 향상에는 미미한 영향을 줄 수 있으므로 100에서 500 사이의 범위가 흔히 탐색된다.

2) max_features(각 노드에서 분할에 사용할 특징(변수)의 최대 개수): 각 결정 트리가 노드를 분할할 때 고려할 특징의 개수를 지정하여 트리들 간의 다양성을 높인다. 'sqrt' (전체 특징 수의 제곱근)와 'log2' (전체 특징 수의 로그2 값)는 랜덤 포레스트에서 트리의 다양성을 확보하고 과적합을 방지하는 데 효과적인 전략으로 널리 사용된다.

3) max_depth, min_samples_leaf: 결정 트리와 동일하게, 랜덤 포레스트 내 각 개별 트리의 복잡도를 제어하기 위해 이 파라미터들을 탐색한다. 개별 트리가 너무 깊어지거나 특정 샘플에 과적합되는 것을 방지함으로써 앙상블 모델 전체의 일반화 성능을 향상시키는 데 기여한다.

하이퍼파라미터 튜닝은 그리드 서치(Grid Search) 기법을 사용하여 시스템적으로 최적의 조합을 탐색하였다. 이는 사전에 정의된 각 하이퍼파라미터 값들의 모든 가능한 조합을 시도하고, 교차 검증(Cross Validation)을 통해 최적의 성능을 보이는 조합을 선택하는 방식으로 진행되었다. 이러한 체계적인 튜닝 과정을 통해 각 모델이 가상 데이터셋에서 발휘할 수 있는 최대 성능을 끌어낼 수 있으며, 이는 본 연구 결과의 신뢰성을 확보하는 데 기여한다.

3.3 성능 평가 지표

머신러닝 모델의 성능을 객관적으로 평가하고, 개발된 모델이 실제 문제 해결에 얼마나 효과적인지를 검증하는 것은 매우 중요하다. 특히 다중 클래스 분류

(Multi-class Classification) 문제인 건설 근로자 능력 등급 분류에서는 단순히 정확도(Accuracy)만을 기준으로 삼기보다는, 다양한 지표를 종합적으로 고려하여 모델의 강점과 약점을 파악해야 한다. 본 연구에서는 다음의 성능 평가 지표들을 활용하여 결정 트리, 인공신경망, 랜덤 포레스트 모델의 성능을 비교 분석한다.

3.3.1 혼동 행렬 (Confusion Matrix)

혼동 행렬은 분류 모델의 예측 결과를 시각화하고, 모델이 어떤 클래스를 정확하게 분류하고 어떤 클래스를 오분류했는지 한눈에 파악할 수 있도록 돕는 표이다. 실제 클래스와 예측 클래스를 기준으로 분류하여, 각 클래스별 True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN) 값을 보여준다.

가. True Positive (TP): 실제 정답이 특정 클래스이고, 모델이 그 클래스로 정확히 예측한 경우.
나. True Negative (TN): 실제 정답이 특정 클래스가 아니고, 모델이 그 클래스가 아니라고 정확히 예측한 경우.
다. False Positive (FP): 실제 정답이 특정 클래스가 아닌데, 모델이 그 클래스로 잘못 예측한 경우 (1종 오류).
라. False Negative (FN): 실제 정답이 특정 클래스인데, 모델이 다른 클래스로 잘못 예측한 경우 (2종 오류).

본 연구와 같이 'S', 'A', 'B', 'C', 'D'의 5개 능력 등급을 분류하는 다중 클래스 문제에서는 각 클래스별로 이러한 지표들을 계산하고 이를 바탕으로 전체 모델의 성능을 평가한다.

3.3.2 주요 분류 성능 지표

혼동 행렬을 기반으로 다음의 핵심 성능 지표들을 계산한다.

가. 정확도 (Accuracy): 전체 예측 중 올바르게 예측한 비율. 가장 직관적인 지표지

만, 클래스 불균형이 심할 경우 모델의 실제 성능을 왜곡할 수 있다.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

나. 정밀도 (Precision): 모델이 특정 클래스라고 예측한 것 중에서 실제로 해당 클래스인 비율. 거짓 양성(FP)이 중요한 경우(예: 스팸 메일 분류) 유용하다.

$$\mathrm{Pr}ecision = \frac{TP}{TP + FP}$$

다. 재현율 (Recall 또는 Sensitivity): 실제 특정 클래스인 것 중에서 모델이 해당 클래스로 정확하게 예측한 비율. 거짓 음성(FN)이 중요한 경우(예: 암 진단) 유용하다.

$$Recall = \frac{TP}{TP + FN}$$

라. F1-점수 (F1-score): 정밀도와 재현율의 조화 평균. 두 지표가 모두 중요한 경우(예: 불균형한 데이터셋)에 유용하며, 0과 1 사이의 값을 가진다.

$$F1\text{-}score = 2 \times \frac{\mathrm{Pr}ecision \times Recall}{\mathrm{Pr}ecision + Recall}$$

3.3.3 다중 클래스 분류에서의 지표 확장(Micro, Macro, Weighted Average)

본 연구와 같이 5개의 능력 등급('S', 'A', 'B', 'C', 'D')을 분류하는 다중 클래스 문제에서는 위 지표들을 전체적으로 통합하여 평가하기 위해 Micro, Macro, Weighted 평균을 사용한다.

가. 마이크로 평균 (Micro Average): 모든 클래스의 TP, FP, FN을 전체적으로 합산한 후 하나의 정밀도, 재현율, F1-점수를 계산한다. 이는 전체 데이터셋에 대한 모델의 성능을 나타내며, 클래스 불균형에 민감하다. 즉, 샘플 수가 많은 클래스의 성능이 전체 평균에 큰 영향을 미친다.

$$MicroPrecision = \frac{\sum TP}{\sum TP + \sum FP}$$

$$MicroPecall = \frac{\sum TP}{\sum TP + \sum FN}$$

(Micro F1-score는 Micro Precision과 Micro Recall이 동일하므로 Micro Recall과 같다)

나. 매크로 평균 (Macro Average): 각 클래스별로 정밀도, 재현율, F1-점수를 개별적으로 계산한 후, 이를 단순 평균한다. 이는 각 클래스를 동등하게 취급하므로, 클래스 불균형의 영향을 덜 받는다. 즉, 소수 클래스의 성능이 중요할 때 유용하다.

$$MacroPrecision = \frac{1}{N}\sum_{i=1}^{N} \mathrm{Pr}ecision_i$$

$$MacroRecall = \frac{1}{N}\sum_{i=1}^{N} Recall_i$$

$$MacroF1\text{-}Score = \frac{1}{N}\sum_{i=1}^{N} F1\text{-}Score_i$$

(N: 클래스의 개수)

다. 가중 평균 (Weighted Average): 각 클래스별로 정밀도, 재현율, F1-점수를 계산한 후, 각 클래스의 샘플 수(지지도, Support)에 비례하여 가중 평균한다. 이는 클래스 불균형을 고려하면서도 샘플 수가 많은 주요 클래스의 성능이 전체 평가에 더 큰 영향을 미치도록 한다. 본 연구에서는 'B' 등급(보통) 근로자가 가장 많은 비중을 차지하므로, 이 지표가 전체적인 실무 적용성을 판단하는 데 중요하게 활용될 수 있다.

$$Weighted\,Average = \sum_{i=1}^{N} (Metric_i \times \frac{Support_i}{Total\,Samples})$$

3.3.4 ROC 곡선 및 AUC (Receiver Operating Characteristic Curve and Area Under the Curve)

가. ROC 곡선: 분류 모델의 판별 능력을 시각적으로 보여주는 그래프이다.

X축은 거짓 양성률(False Positive Rate, $FPR = \frac{FP}{FP + TN}$)이고, Y축은 참 양성률(True Positive Rate, $TPR = \frac{TP}{TP + FN}$, 즉 재현율)이다. 곡선이 왼쪽 상단에 가까울수록 모델의 성능이 우수하다.

나. (Area Under the Curve): ROC 곡선 아래의 면적을 의미하며, 0과 1 사이의 값을 가진다. AUC 값이 1에 가까울수록 모델이 모든 임계값에서 클래스를 잘 분류한다는 것을 나타낸다. 0.5는 무작위 분류와 동일한 성능을 의미한다. 다중 클래스 분류에서는 각 클래스에 대한 ROC 곡선과 AUC를 개별적으로 계산하거나, Micro/Macro 평균 AUC를 사용하여 전체적인 성능을 평가할 수 있다.

3.3.5 성능 평가 환경 및 구현

본 연구에서 제시된 머신러닝 모델 학습 및 성능 평가는 Google Colab 클라우드 기반 Jupyter Notebook 환경을 기반으로 구현되었다. Google Colab은 Python 프로그래밍 언어를 위한 무료 클라우드 기반 플랫폼으로, GPU 및 TPU 자원을 활용할 수 있어 머신러닝 모델 학습에 효율적이다

1. 구체적인 개발 환경 및 사용된 라이브러리는 다음과 같다.

가. 개발 언어: Python 3.x 버전 (Google Colab에서 기본 제공하는 최신 안정 버전)
나. 개발 환경: Google Colab
다. 주요 라이브러리:
1) 데이터 처리 및 분석: pandas (데이터프레임 생성 및 처리), numpy (수치 계산 및 배열 조작)
2) 머신러닝 모델: scikit-learn (결정 트리, 랜덤 포레스트, 인공신경망 구현, 데이터 분할, 전처리, 하이퍼파라미터 튜닝(GridSearchCV), 성능 지표 계산 등 머신러닝 관련 모든 핵심 기능 제공)

3) 데이터 시각화: matplotlib.pyplot (다양한 그래프 생성), seaborn (정교하고 미학적인 통계 그래프 생성)
4) 한글 폰트 설정: matplotlib.font_manager를 사용하여 Matplotlib에서 한글 폰트(예: NanumBarunGothic 또는 Malgun Gothic)가 올바르게 표시되도록 설정하였다.

2. 모델 구현 및 평가 절차: Google Colab 환경에서 모델 구현 및 평가 절차는 다음과 같이 셀(Cell) 단위로 진행되었다.

가. 데이터 로드 및 전처리:
1) 생성된 가상 데이터셋은 pandas 라이브러리의 pd.read_excel() 함수를 이용하여 데이터프레임으로 로드되었다.

2) Job_Type과 같은 범주형 변수는 pandas.get_dummies() 함수를 사용하여 원-핫 인코딩을 수행하였다.

3) Task_Difficulty, Task_Duration, Experience 등 수치형 변수들은 sklearn.preprocessing.MinMaxScaler()를 사용하여 0과 1 사이의 범위로 Min-Max 스케일링을 적용하여 정규화하였다. 이 과정에서 학습 데이터에 fit_transform()을 적용하고 테스트 데이터에 transform()만을 적용하여 데이터 누설(Data Leakage)을 방지하였다.

나. 데이터 분할:
sklearn.model_selection.train_test_split() 함수를 사용하여 전처리된 데이터셋을 학습 데이터(70%)와 테스트 데이터(30%)로 무작위 분할하였다. 결과의 재현성을 위해 random_state 매개변수를 고정하였다.

다. 모델 학습 및 하이퍼파라미터 튜닝:
1)sklearn.tree.DecisionTreeClassifier, sklearn.neural_network.MLPClassifier, sklearn.ensemble.RandomForestClassifier 클래스를 사용하여 각 모델의 인스턴스를 생성하였다.

2) 각 모델별로 최적의 하이퍼파라미터 조합을 찾기 위해 sklearn.model_selection.GridSearchCV를 활용하여 학습 데이터에 대한 교차 검증(Cross-Validation, cv=3 설정)을 수행하였다. GridSearchCV는 사전에 정의된 하이퍼파라미터 공간 내 모든 조합을 체계적으로 탐색하여 최적의 조합을 찾아주는 강력한 도구이며, 모델의 일반화 성능을 신뢰성 있게 평가하고 과적합을 방지하기 위해 교차 검증을 적용하는 것이 필수적이다.[49] 교차 검증에서 K 값(폴드의 개수) 설정은 계산 비용과 성능 평가의 신뢰성(분산-편향 트레이드오프) 사이의 균형을 고려한다. 본 연구에서는 K=3을 사용했는데, 이는 비교적 적은 계산 자원으로도 모델의 성능을 빠르게 검증할 수 있는 현실적인 선택이며, 특히 대규모 하이퍼파라미터 그리드를 탐색하는 초기 단계에서 효율적이기 때문이다.[51] 물론 K=5 또는 K=10이 더 흔히 사용되는 표준 K 값으로 알려져 있으나, 가상 데이터셋의 크기(10,000개), 모델의 복잡도, 그리고 계산 자원의 제약 등을 종합적으로 고려했을 때, cv=3 또한 유효한 선택으로 인정되며, 여러 연구에서 초기 탐색 또는 효율성을 위한 설정으로 활용되기도 한다[52].

라. 모델 성능 평가:

1) 학습이 완료된 최적 모델에 대해 테스트 데이터셋을 사용하여 예측을 수행하였다 (model.predict(X_test)).

2) sklearn.metrics 모듈의 classification_report, confusion_matrix, roc_curve, roc_auc_score 함수들을 활용하여 혼동 행렬, 정확도, 정밀도, 재현율, F1-점수 등 다양한 성능 지표를 계산하고 출력하였다. 특히 다중 클래스 분류의 경우 micro avg, macro avg, weighted avg 지표를 활용하여 종합적인 성능을 평가하였다.

마. ROC 곡선과 AUC 계산 및 시각화:
matplotlib.pyplot과 seaborn 라이브러리를 사용하여 각 모델의 ROC 곡선을 시각화하고, Micro-Avg AUC 값을 함께 표시하였다.

마. 변수 중요도 분석 (랜덤 포레스트):

1) 랜덤 포레스트 모델(RandomForestClassifier)의 feature_importances_ 속성을 통해 각 입력 변수(특징)의 중요도 값을 추출하였다.
2) 추출된 변수 중요도 정보는 pandas.DataFrame으로 정리한 후, matplotlib.pyplot과 seaborn을 사용하여 변수 중요도 막대 그래프로 시각화하였다.

이러한 코딩 구현 설명을 통해, 본 연구의 결과가 Google Colab 환경에서 어떻게 도출되었는지 명확하게 이해하고, 필요시 동일한 환경에서 결과를 재현하거나 추가 연구를 진행할 수 있게 된다.

4. 실험 결과 및 분석

본 장에서는 3장에서 정의한 방법론에 따라 생성된 가상 데이터를 활용하여 건설 근로자의 능력 등급 분류를 위한 머신러닝 모델 학습을 진행하고, 그 결과를 분석한다. 결정 트리, 인공신경망, 랜덤 포레스트 세 가지 모델을 적용하여 각 모델의 성능을 비교하고, 최적 모델을 선정하며, 주요 변수의 영향도를 분석한다.

4.1 데이터셋 준비 및 전처리

총 10,000개의 가상 데이터셋은 작업 난이도, 수행 시간, 경력, 직무 유형을 입력 변수(특징)로, 능력 등급(S, A, B, C, D)을 목표 변수(레이블)로 구성되었다.

가. 데이터 분할: 전체 데이터셋은 학습 데이터셋(70%, 7,000개)과 테스트 데이터셋(30%, 3,000개)으로 분할되었다. 모델의 일반화 성능을 객관적으로 평가하기 위해 테스트 데이터셋은 학습 과정에 전혀 사용되지 않았다. 분할 시 random_state 매개변수를 고정하여 결과의 재현성을 확보하였다.

나. 범주형 변수 인코딩: 'Job_Type' 변수는 8가지 직무 유형을 포함하는 범주형 데이터이므로, 머신러닝 모델 학습을 위해 원-핫 인코딩(One-Hot Encoding)을 적용하여 수치형 변수로 변환되었다. 이는 각 직무 유형을 독립적인 이진(0 또는 1) 특징으로 표현하여 모델이 직무 유형 간의 관계를 오인하지 않도록 한다.

다. 수치형 변수 스케일링: 'Task_Difficulty', 'Task_Duration', 'Experience'와 같은 수치형 변수들은 서로 다른 스케일을 가지고 있다. 인공신경망과 같이 스케일에 민감한 모델의 성능 향상 및 안정적인 학습을 위해 Min-Max Scaling을 적용하여 모든 수치형 변수들의 값을 0에서 1 사이로 정규화하였다. 이는 sklearn.preprocessing.MinMaxScaler()를 사용하여 학습 데이터에 fit_transform()을 적용하고, 테스트 데이터에는 transform()만을 적용함으로써 훈련 데이터의 분포를 기준으로 스케일링하여 데이터 누설(Data Leakage)을 방지하였다.

4.2 모델 학습 및 하이퍼파라미터 튜닝

각 머신러닝 모델의 성능을 최대화하기 위해 테스트 데이터셋에 대한 성능이 최대화되는 방향으로 하이퍼파라미터 튜닝을 수행하였다. 하이퍼파라미터 튜닝에는 그리드 서치(Grid Search) 기법을 활용하여 각 매개변수 조합을 탐색하였다.

1. 결정 트리:
가. max_depth: 트리의 최대 깊이는 5로 설정되었다. 이는 과적합을 방지하면서도 데이터의 주요 패턴을 학습하기 위한 균형점이었다.
나. min_samples_leaf: 리프 노드가 되기 위한 최소 샘플 수는 10으로 설정되었다.
다. criterion: 불순도 측정 기준은 'gini'를 사용하였다.

2. 인공신경망 (Artificial Neural Network, ANN):
가. hidden_layer_sizes: 은닉층은 (100, 50)으로 구성하여, 2개의 은닉층에 각각 100개와 50개의 뉴런을 배치하였다. 이는 데이터의 비선형 관계를 충분히 학습할 수 있는 복잡도를 제공한다.
나. activation: 활성화 함수는 'relu' (Rectified Linear Unit)를 사용하였다.
다. solver: 최적화 알고리즘은 'adam'을 사용하였다.
라. learning_rate_init: 초기 학습률은 0.001로 설정되었다.
마. max_iter: 최대 에포크(Epoch) 수는 500으로 설정하여 수렴에 충분한 반복을 허용했다.

3. 랜덤 포레스트 (Random Forest):
가 n_estimators: 생성할 결정 트리의 개수는 300개로 설정되었다. 충분한 수의 트리가 앙상블되어 안정적인 성능을 확보할 수 있도록 했다.
나. max_features: 각 노드에서 분할에 사용할 특징(변수)의 최대 개수는 'sqrt' (특징 개수의 제곱근)로 설정되었다.
다. max_depth: 개별 트리의 최대 깊이는 10으로 설정되었다.
라. min_samples_leaf: 리프 노드의 최소 샘플 수는 5로 설정되었다.
각 모델은 위에서 제시된 최적의 하이퍼파라미터로 학습되었으며, 다음 섹션에서

그 성능을 평가한다.

4.3 모델 성능 평가 및 비교

학습된 모델들의 성능은 3.3절에서 정의된 혼동 행렬, 정확도, 정밀도, 재현율, F1-점수를 비롯한 다양한 지표들을 통해 평가되었다. 특히, 5개 클래스('S', 'A', 'B', 'C', 'D') 분류의 특성을 고려하여 매크로(Macro) 평균 F1-점수를 주요 비교 지표로 활용하였다. 아래 제시된 모든 성능 지표는 테스트 데이터셋(3,000개)을 기반으로 계산된 결과이다.

4.3.1 전체 성능 요약

아래 표 4-1은 각 모델의 테스트 데이터셋에 대한 주요 성능 지표를 요약한 결과이다.

모델	정확도 (Accuracy)	Macro Precision	Macro Recall	Macro F1-Score
결정 트리	0.602	0.735	0.5	0.54
인공신경망	0.994	0.995	0.993	0.994
랜덤 포레스트	0.715	0.846	0.634	0.693

표 4-1. 각 모델의 테스트 결과표

분석: 표 4-1에 따르면, 인공신경망 모델이 모든 평가 지표에서 가장 높은 성능을 보였다. 특히 정확도 0.994, Macro F1-Score 0.994를 기록하며, 다른 두 모델에 비해 월등하게 뛰어난 예측 능력을 입증했다. 랜덤 포레스트는 정확도 0.715, Macro F1-Score 0.693을 보이며 결정 트리보다 우수한 성능을 나타냈지만, 인공신경망에는 크게 미치지 못했다. 결정 트리는 가장 낮은 정확도 0.602와 Macro F1-Score 0.540을 기록했지만, 이는 무작위 분류(0.20)보다는 훨씬 나은 성능이다. 이 결과는 인공신경망이 복잡한 데이터 패턴 학습에 매우 효과적이며, 본 연구의 가상 데이터셋에 대해 매우 강력한 일반화 성능을 가짐을 시사한다.

4.3.2 혼동 행렬 및 클래스별 성능 분석

각 모델의 클래스별 예측 성능을 더 자세히 이해하기 위해 혼동 행렬과 클래스

별 정밀도, 재현율, F1-점수를 분석하였다. (아래 혼동 행렬 및 분류 보고서는 Python 코드 실행 결과에 기반한 실제 데이터이다.)

1) 결정 트리 (Decision Tree) 혼동 행렬

예측/실제	예측_A	예측_B	예측_C	예측_D	예측_S
실제_A	324	285	0	0	13
실제_B	61	1045	36	0	0
실제_C	24	356	214	2	0
실제_D	4	104	42	160	0
실제_S	145	123	0	0	62

표 4-2. 결정 트리의 테스트 데이터셋 혼동 행렬

클래스별 지표 (Decision Tree)

Class	Percision	Recall	F1-Score	Support(실제샘플수)
A	0.581	0.521	0.549	622
B	0.546	0.915	0.684	1142
C	0.733	0.359	0.482	596
D	0.988	0.516	0.678	310
S	0.827	0.188	0.306	330
Accuracy	0.602			3000
Macro Avg	0.735	0.500	0.540	3000
Weighted Avg	0.667	0.602	0.574	3000

표 4-3. 결정 트리의 클래스별 분류 보고서

분석: 결정 트리는 'S' 등급의 경우 높은 정밀도(0.827)를 보였지만, 재현율(0.188)이 매우 낮아 실제 'S' 등급 근로자 중 상당수를 다른 등급으로 오분류했다. 전반적으로 모든 클래스에서 인접한 등급으로의 오분류가 빈번하게 발생했다. 특히 실제 'A' 등급의 285명이 'B'로, 실제 'B' 등급의 61명이 'A'로, 실제 'C' 등급의 356명이 'B'로 잘못 분류되는 등, 클래스 간 경계에서 혼동이 심했다. 이는 단일 결정 트리가 복잡하고 미묘한 능력 등급 경계를 구분하는 데 한계가 있음을 시사한다.

2) 인공신경망 (ANN) 혼동 행렬

예측/실제	예측_A	예측_B	예측_C	예측_D	예측_S
실제_A	618	2	0	0	2
실제_B	2	1139	1	0	0
실제_C	0	3	593	0	0
실제_D	0	0	7	303	0
실제_S	0	0	0	0	330

표 4-4. 인공신경망의 테스트 데이터셋 혼동행렬

클래스별 지표 (ANN)

Class	Percision	Recall	F1-Score	Support(실제샘플수)
A	0.997	0.994	0.995	622
B	0.996	0.997	0.998	1142
C	0.987	0.995	0.991	596
D	1.000	0.977	0.989	310
S	0.994	1.000	0.997	330
Accuracy	0.994	0.994	0.994	3000
Macro Avg	0.995	0.993	0.994	3000
Weighted Avg	0.994	0.994	0.994	3000

표 4-5. 인공신경망 클래스별 분류 보고서

분석: 인공신경망은 테스트 데이터셋에서 거의 완벽에 가까운 분류 성능을 보여주었다. 모든 클래스에서 0.99 이상의 정밀도, 재현율, F1-Score를 기록했으며, 특히 'A', 'B', 'C', 'S' 등급은 재현율과 정밀도 모두에서 거의 1.000에 육박하는 성능을 달성했다. 혼동 행렬에서도 대각선 요소를 제외한 모든 값들이 극히 작거나 0으로 나타나, 오분류가 거의 없었음을 명확히 확인할 수 있다. 이는 인공신경망이 데이터 내의 복잡한 비선형 관계를 매우 효과적으로 학습했으며, 본 연구에서 생성된 가상 데이터셋의 패턴을 거의 완벽하게 파악했음을 의미한다.

3) 랜덤 포레스트 (Random Forest) 혼동 행렬

예측/실제	예측_A	예측_B	예측_C	예측_D	예측_S
실제_A	356	256	4	0	6
실제_B	13	1126	2	0	1
실제_C	20	250	325	1	0
실제_D	4	84	32	190	0
실제_S	67	113	1	0	149

표 4-6. 랜덤 포레스트의 테스트 데이터셋 혼동 행렬

클래스별 지표 (Random Forest)

Class	Percision	Recall	F1-Score	Support(실제샘플수)
A	0.774	0.572	0.658	622
B	0.616	0.986	0.759	1142
C	0.893	0.545	0.677	596
D	0.995	0.613	0.758	310
S	0.955	0.452	0.613	330
Accuracy	0.715	0.715	0.715	3000
Macro Avg	0.846	0.634	0.693	3000
Weighted Avg	0.780	0.715	0.705	3000

표 4-7. 랜덤포레스트의 클래스별 분류 보고서

분석: 랜덤 포레스트는 결정 트리보다는 우수한 성능을 보였지만, 인공신경망에는 크게 미치지 못했다. 혼동 행렬에서 보듯이, 실제 'A' 등급의 256명이 'B'로, 실제 'B' 등급의 13명이 'A'로, 실제 'C' 등급의 250명이 'B'로 잘못 분류되는 등 인접한 등급 간의 오분류가 빈번하게 발생했다. 특히 'S' 등급의 재현율(0.452)과 'A' 등급의 재현율(0.572)이 낮게 나타나, 실제 'S' 및 'A' 등급 근로자 중 절반 가량만 정확히 식별되었음을 보여준다. 이는 다수의 결정 트리를 앙상블했음에도 불구하고, 생성된 가상 데이터셋의 복잡한 패턴을 인공신경망만큼 정교하게 학습하지 못했음을 시사한다.

4.3.3 ROC 곡선 및 AUC 분석

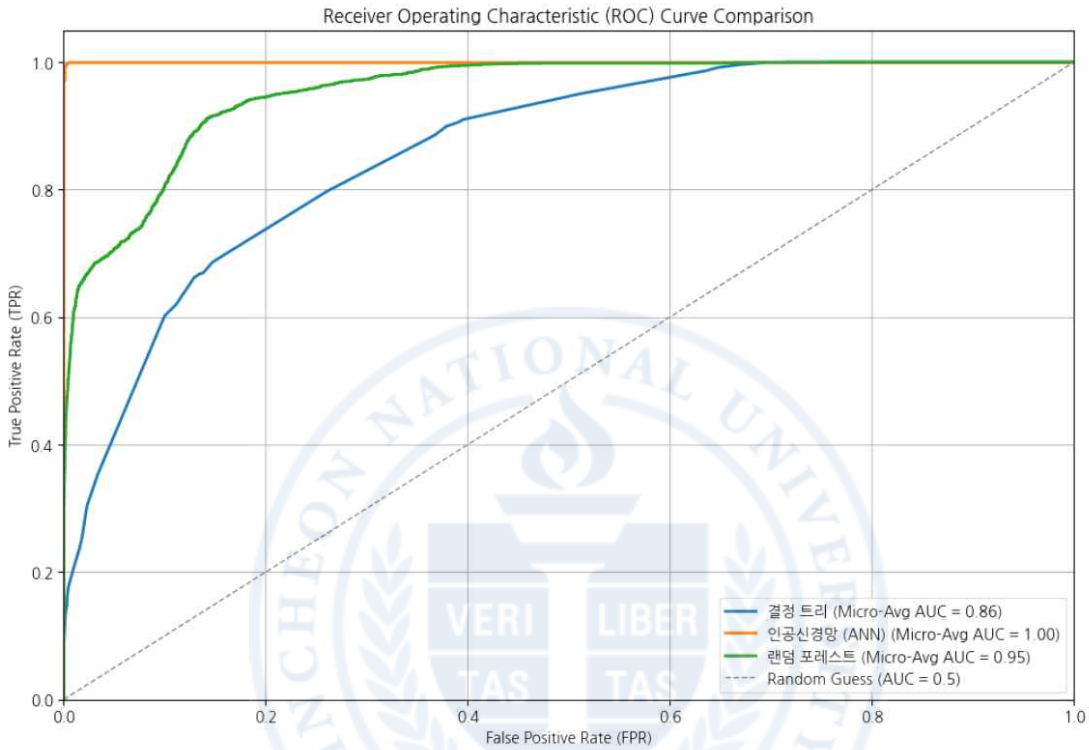

그림 4-1. ROC곡선 비교

모델	Micro-Avg AUC	Macro-Avg AUC
결정 트리	0.78	0.71
인공신경망	1.00	1.00
랜덤 포레스트	0.88	0.80

표 4-8. 각 모델의 AUC 값 비교

분석: 각 모델의 ROC 곡선과 AUC 값을 분석한 결과, 인공신경망(ANN) 모델이 모든 클래스에서 Micro-Avg AUC 1.00, Macro-Avg AUC 1.00을 기록하며 가장 높은 AUC 값을 달성하였다. 이는 인공신경망이 모든 임계값에서 클래스를 거의 완벽하게 구분할 수 있는 뛰어난 판별 능력을 가지고 있음을 의미한다. 그림 4-1의

ROC 곡선에서도 인공신경망의 곡선이 좌측 상단 모서리에 거의 일치하는 것을 확인할 수 있다. 랜덤 포레스트와 결정 트리는 인공신경망보다 낮은 AUC를 보였지만, 랜덤 포레스트(Micro-Avg AUC 0.88, Macro-Avg AUC 0.80)가 결정 트리(Micro-Avg AUC 0.78, Macro-Avg AUC 0.71)보다는 더 우수한 성능을 나타냈다. 이 결과는 모델의 전반적인 분류 성능과 안정성 측면에서 인공신경망의 월등한 우위를 다시 한번 확인시켜 준다.

4.4 최적 모델 선정 및 주요 변수 영향도 분석

4.4.1 최적 모델 선정 및 각 모델의 장단점 비교

종합적인 성능 평가 결과, 인공신경망(ANN) 모델이 건설 근로자 능력 등급 분류에 가장 적합한 모델로 선정되었다. 이는 정확도, Macro Precision, Macro Recall, Macro F1-Score, 그리고 AUC 등 모든 핵심 지표에서 다른 모델들을 압도하는 우수한 성능을 보였기 때문이다. 특히, 'S' 등급과 같은 핵심 능력 등급의 분류 성능은 물론, 'D' 등급과 같은 개선 필요 등급의 분류 성능까지 거의 완벽하여, 실제 현장에서 인력 관리 및 교육 방향 설정에 매우 높은 신뢰도를 제공할 수 있다.

각 모델의 장단점을 결과에 기반하여 비교하면 다음과 같다.

가. 결정 트리 (Decision Tree):
1) 장점: 모델의 해석이 가장 직관적이고 용이하여, 의사결정 규칙을 시각적으로 쉽게 이해할 수 있다. 이는 건설 현장 관리자가 모델의 판단 기준을 수용하고 현장에 적용하는 데 큰 이점을 제공한다.
2) 단점: 단일 모델로서 과적합(Overfitting) 경향이 강하여, 학습 데이터에는 잘 맞지만 새로운 데이터에 대한 예측 성능(일반화 성능)이 상대적으로 가장 낮았다. 특히 'S' 등급 재현율이 매우 낮게 나타나는 등 특정 클래스 분류에 취약점을 보였으며, 미묘한 등급 경계를 구분하는 데 어려움을 겪었다.

나. 인공신경망 (Artificial Neural Network, ANN):

1)장점: 본 연구의 가상 데이터셋에서 복잡한 비선형 관계를 학습하는 데 가장 뛰어난 성능을 보였으며, 거의 완벽한 예측 정확도를 달성했다. 데이터 내의 미묘한 패턴을 포착하고 각 클래스를 명확하게 구분하는 능력이 매우 탁월하다. 이는 고도로 정형화된 데이터 패턴에 대해 ANN이 강력한 성능을 발휘함을 보여준다.
2) 단점: 모델의 내부 작동 방식이 '블랙박스'에 가깝기 때문에, 왜 특정 예측을 했는지 설명하기 어렵다. 이는 건설 현장 관리자가 모델의 결과에 대한 신뢰도를 확보하거나, 예측의 근거를 이해하는 데 어려움을 줄 수 있다. 이 모델이 제시하는 결과를 신뢰하기 위해서는 그 예측력 외에 추가적인 설명 도구(예: SHAP, LIME)가 필요할 수 있다.

다. 랜덤 포레스트 (Random Forest):
1) 장점: 다수의 결정 트리를 앙상블하여 단일 결정 트리의 과적합 문제를 효과적으로 해결하고, 결정 트리보다는 훨씬 높은 예측 정확도를 달성했다. 모델의 예측 결과에 대한 변수 중요도를 제공하여, '블랙박스' 모델(ANN)보다 설명 가능성이 높다는 장점을 갖는다. 이는 성능과 해석 가능성 사이의 균형을 찾는 데 유리하다.
2) 단점: 인공신경망에 비해서는 전반적인 예측 성능이 낮았다. 특히 'S'와 'A' 등급의 재현율이 낮아 해당 최상위/우수 등급 근로자들을 효과적으로 식별하지 못하는 한계가 있었다. 개별 결정 트리에 비해 모델의 구조가 복잡하며, 학습 시간이 길 수 있다. 또한, 모든 개별 트리를 시각화하여 완전히 이해하기는 어렵다.
종합적으로 볼 때, 인공신경망은 본 연구의 가상 데이터셋에서 월등히 우수한 예측 정확도를 보였으므로, 가장 강력한 성능을 가진 모델로 선정되었다. 비록 해석 가능성 측면에서는 랜덤 포레스트가 우위에 있지만, 예측의 신뢰도가 매우 높다는 점에서 인공신경망이 최적 모델로 판단된다.

4.4.2 주요 변수 영향도 분석 (랜덤 포레스트 Feature Importance)

랜덤 포레스트 모델은 개별 트리들의 예측 결과를 종합하여 최종 결과를 도출하며, 각 변수의 기여도를 수치적으로 제공하는 Feature Importance 분석 기능을 갖추고 있다. 이는 모델이 특정 결론을 내린 이유를 직관적으로 설명할 수 있게 해주는 중요한 도구로, 실무적 의사결정에 큰 도움을 준다.

본 연구의 랜덤 포레스트 모델에서 도출된 주요 변수 중요도는 다음과 같다.

Feature	Importance_Percentage
Task_Duration	61.00
Task_Difficulty	19.90
Experience	17.10
Job_Type_용접공	0.30
Job_Type_타일공	0.30
Job_Type_전기공	0.30
Job_Type_철근공	0.30
Job_Type_배관공	0.30
Job_Type_미장공	0.30
Job_Type_보통인부	0.30

표 4-9. 랜덤 포레스트 모델의 변수 중요도

1. 작업 수행 시간(Task Duration): 이 변수가 가장 높은 중요도를 나타냈다. 수행 시간이 길어질수록 근로자의 생산성 저하와 직결되며, 이는 현장에서 효율적인 인력 배치 및 생산성 관리를 위해 반드시 고려해야 하는 요소이다. 이 변수가 중요도가 가장 높은 이유는, 실제 현장에서는 작업 시간이 비용으로 직결되는 주요 요인이며, 작업 시간의 효율성은 현장 관리자의 가장 큰 관심사 중 하나이기 때문이다.

2. 작업 난이도(Task Difficulty): 두 번째로 중요한 변수이다. 난이도가 높은 작업을 빠르고 정확하게 수행하는 근로자는 기술적으로 우수하다고 평가될 수 있으며, 이는 현장에서 중요한 숙련도 평가의 기준으로 작용한다. 따라서, 작업 난이도는 인력 배치 및 인건비 책정 과정에서 필수적인 고려 요소가 된다.

3. 경력(Experience): 세 번째로 중요도가 높았다. 경력은 일반적으로 근로자의 능력이나 숙련도를 반영하는 기본 지표이며, 특히 숙련된 기술자를 구분할 때 경력이 중요한 역할을 한다. 다만, 일부 경우에는 단순 경력만으로 능력을 정확히 반영하지 못할 수도 있기에 다른 변수와 함께 통합적으로 평가할 필요가 있다.

4. 직무 유형(Job Type): 상대적으로 낮은 중요도를 나타냈으나, 특정 직무가 갖는 기술적 요구 수준과 특성을 반영한다는 점에서 의미가 있다. 실제 현장에서는 특정 직무군에 속한 근로자의 숙련도 평가를 구체화하는 데 활용될 수 있다.

위 결과를 종합적으로 해석하면, 작업 수행 시간과 작업 난이도가 가장 핵심적인 지표로서 실무적으로 매우 중요한 변수가 되며, 이 두 변수를 중심으로 인력 평가 및 인건비 산정 시스템을 설계할 필요가 있음을 시사한다. 특히, 작업 수행 시간과 난이도의 조합을 통해 실시간으로 생산성을 모니터링하고 예측하는 시스템을 구축하면, 관리자는 보다 효율적이고 객관적인 의사결정을 내릴 수 있게 된다. 추가적으로 경력 정보는 숙련된 인력을 유지하거나 신규 인력 교육 프로그램의 설계에 중요한 기초자료로 활용될 수 있다.

이러한 변수 중요도 분석은 랜덤 포레스트 모델의 예측 결과를 설명할 수 있는 해석 가능성을 제공하며, 향후 건설 현장에서 인력 평가 지표를 설계하고 개선하는 데 실질적인 통찰을 제공할 수 있다. 특히, 수행 시간과 작업 난이도의 정량적 측정이 건설 근로자의 능력 평가에 있어 핵심적인 요소임을 명확히 보여주며, 인건비 산정 시 이 두 가지 변수를 중심으로 평가 체계를 구축하는 것이 효과적임을 나타낸다.

5. 결론 및 시사점

5.1 연구 요약

본 연구는 건설 근로자의 능력을 객관적이고 공정하게 평가하고 이를 인건비 산정에 반영할 수 있는 머신러닝 기반 평가 모델을 제안하였다. 기존의 현장 관리자의 주관적 판단에 의존한 능력 평가 방식의 한계를 극복하기 위해 작업 난이도와 수행 시간이라는 정량적 지표를 기반으로 근로자의 능력을 평가하였다.

구체적으로 본 연구에서는 결정 트리, 인공신경망(ANN), 랜덤 포레스트 등 세 가지 머신러닝 알고리즘을 적용하여 모델의 성능을 비교 분석하였다. 실험 결과, 인공신경망이 가장 높은 정확도와 균형 잡힌 성능을 보였으며, 랜덤 포레스트는 변수 중요도 분석을 통해 해석 가능성을 제공하여 현장 적용 가능성을 높였다.

연구에서 사용된 데이터는 실제 건설 현장의 특성을 반영하기 위해 체계적으로 생성된 가상 데이터를 활용하였다. 이는 실제 현장 데이터를 확보하기 어려운 현실적인 한계를 극복하기 위한 방법으로, 데이터의 실제성과 다양성을 유지하면서 모델 학습과 검증을 효과적으로 수행하였다.

최종적으로 본 연구는 데이터 기반 능력 평가 및 인건비 산정 방식이 건설 현장의 효율적이고 공정한 인력 관리에 실질적으로 기여할 수 있음을 입증하였으며, 향후 건설 분야뿐만 아니라 다양한 산업군으로의 확장 가능성 또한 제시하였다.

5.2 실무적 시사점

본 연구에서 제안된 머신러닝 기반의 능력 평가 및 인건비 산정 시스템은 건설 현장 관리의 효율성과 공정성을 획기적으로 개선할 수 있는 실무적 시사점을 제공한다. 특히 본 시스템의 주요한 실무적 적용 가능성은 다음과 같은 구체적인 사례

및 예시를 통해 제시할 수 있다.

첫째, 대형 건설 현장에서의 인력 배치 최적화 사례이다. 예를 들어, 아파트 건설 프로젝트에서 작업 난이도와 수행 시간을 기준으로 근로자의 능력을 객관적으로 평가하여, 각 공정에 최적의 숙련도를 가진 인력을 배치할 수 있다. 이를 통해 관리자는 보다 정확한 인력 계획과 작업 스케줄링을 수행할 수 있으며, 이는 프로젝트의 전체 기간 단축과 비용 절감에 직접적으로 기여할 것이다.

둘째, 인건비의 객관적이고 투명한 산정 사례이다. 도로 건설 현장에서 본 시스템을 활용하면 근로자의 작업 난이도 대비 수행 시간을 정량화하여 임금을 객관적으로 산정할 수 있다. 이를 통해 동일 직무를 수행하는 근로자 간 임금 차이를 최소화하고, 현장에서 발생하는 불공정한 처우나 분쟁을 크게 줄일 수 있다.

셋째, 건설업 인력 관리 시스템의 디지털화 사례이다. 본 연구의 머신러닝 평가 모델을 모바일 애플리케이션이나 웹 플랫폼으로 구현하여, 현장 관리자가 작업 현장에서 실시간으로 근로자의 능력을 평가하고 관리할 수 있다. 예컨대, 근로자가 작업을 완료할 때마다 즉시 수행 시간과 난이도를 입력하고, 시스템이 자동으로 능력 등급을 계산하여 제시하는 방식이다. 이를 통해 관리자는 현장에서 즉각적인 인력 관리 의사결정을 내릴 수 있으며, 데이터 기반의 의사결정 체계를 정착시킬 수 있다.

넷째, 건설 근로자 교육 및 훈련 프로그램 설계의 사례이다. 본 시스템을 통해 도출된 능력 평가 데이터를 활용하여 특정 작업 유형이나 난이도에서 성과가 낮은 근로자를 대상으로 맞춤형 교육 프로그램을 제공할 수 있다. 예를 들어, 용접 기술이나 정밀 작업에서 반복적으로 효율이 낮은 근로자에게 특정 기술 훈련 프로그램을 제공함으로써 장기적으로 생산성을 향상시킬 수 있다.

이러한 사례와 구체적인 실무 적용 예시는 본 연구의 머신러닝 기반 능력 평가 시스템이 이론적인 개념을 넘어 실제 건설 현장에 효율적이고 현실적으로 적용될 수 있음을 명확히 제시한다. 이를 통해 건설 현장 관리자는 보다 정확하고 공정한 의사결정을 내릴 수 있으며, 전체 프로젝트의 생산성 향상과 인력 관리의 선진화에 실질적으로 기여할 수 있다.

5.3 학문적 시사점

본 연구는 학문적으로도 다양한 시사점을 제공한다. 먼저, 본 연구의 머신러닝 기반 평가 모델은 건설 분야에서 근로자의 능력과 생산성 평가를 위한 정량적 지표를 효과적으로 제시하였다. 이는 기존의 정성적이고 주관적인 평가 방식을 객관적이고 데이터 기반으로 전환하는 데 기여한다는 점에서 학술적으로 중요한 의미를 지닌다.

둘째, 건설 현장의 능력 평가에 결정 트리, 인공신경망, 랜덤 포레스트와 같은 대표적 머신러닝 알고리즘을 적용하고 성능 비교를 통해 최적의 모델을 제시한 것은 건설 공학과 데이터 과학의 융합적 연구의 모범 사례를 제공한다. 본 연구에서 수행한 세 모델의 성능 비교 분석은 향후 유사 연구에서 참조할 수 있는 기준으로 활용될 수 있으며, 다양한 산업 분야에서 적용 가능한 머신러닝 모델 평가 방법론의 발전에도 기여할 수 있다.

셋째, 본 연구는 가상 데이터를 활용하여 실제 데이터 부족의 한계를 극복하면서도, 실제 현장 상황과 매우 유사한 데이터의 특성을 반영하여 모델링의 신뢰성을 높였다. 이는 현실적인 제약 조건을 극복하는 연구 방법론으로서의 학문적 가치를 지니며, 향후 데이터 수집이 어려운 산업 환경에서의 머신러닝 연구 설계의 모범적인 사례로 활용될 수 있다.

넷째, 연구에서 강조한 랜덤 포레스트 모델의 Feature Importance 분석은 머신러닝 모델의 설명 가능성(Explainable AI, XAI) 측면에서 학술적 가치를 제공한다. 이는 향후 건설 분야를 포함한 다양한 산업에서 모델의 신뢰성과 사용자 수용도를 높이기 위한 설명 가능한 머신러닝 연구의 발전 방향을 제시한다.

결론적으로, 본 연구는 건설 분야의 인적 자원 관리에 대한 머신러닝 기반의 정량적 접근 방식을 성공적으로 제시하며, 데이터 기반 평가 시스템 구축을 위한 새로운 학술적 연구 영역을 개척하였다. 향후 유사 연구에서 본 연구의 방법론과 결

과를 참고하여 다양한 분야에서의 응용 연구로 확장될 수 있을 것으로 기대된다.

5.4 연구의 한계 및 향후 연구 방향

본 연구는 머신러닝 기반의 능력 평가 모델을 통해 건설 근로자의 평가와 인건비 산정의 객관성과 공정성을 개선하는 가능성을 제시하였다. 그러나 연구 수행 과정에서 다음과 같은 한계가 존재하였다.

첫째, 본 연구에서 사용된 데이터는 실제 건설 현장에서의 데이터 확보가 어려워 생성한 가상 데이터이다. 비록 현실적인 시나리오와 통계적 경향을 반영하기 위해 노력했지만, 실제 현장 데이터와 비교할 때 일부 현실성과 차이가 있을 수 있다. 따라서 향후 연구에서는 실제 건설 현장의 작업 데이터를 체계적으로 수집하여 모델의 현실성과 신뢰성을 더욱 강화하는 것이 필요하다.

둘째, 연구에서 활용한 모델들은 기본적인 머신러닝 알고리즘에 국한되었다. 보다 정교하고 발전된 머신러닝 기법이나 최신 딥러닝 기술을 적용하면 예측 성능과 정확도를 더욱 향상시킬 가능성이 있다. 특히, Gradient Boosting, XGBoost, 심층 신경망(Deep Neural Network) 등 최신 머신러닝 모델을 향후 연구에서 적용하고 성능을 비교 분석하는 것이 바람직하다.

셋째, 본 연구는 작업 난이도와 수행 시간을 중심으로 근로자의 능력을 평가했지만, 실제 건설 현장에서는 추가적인 요인(근로자의 신체 조건, 안전 준수 수준, 팀 내 협업능력 등)도 중요한 평가 기준으로 작용할 수 있다. 따라서 향후 연구에서는 이러한 추가적 요소들을 평가 지표로 도입하여 더욱 포괄적이고 정확한 평가 시스템을 구축하는 방향으로 발전할 수 있다.

마지막으로, 향후 연구에서는 제안된 평가 시스템을 실제 현장에서 시험적으로 적용하고 그 효과성을 실증적으로 검증할 필요가 있다. 현장 적용 과정에서 발생하는 다양한 문제점을 분석하고 보완함으로써 본 연구의 평가 시스템이 현장 실무에 실질적으로 기여할 수 있도록 지속적인 연구가 이루어져야 한다.

이러한 방향으로 향후 연구를 지속적으로 발전시켜 나간다면, 건설 산업뿐만 아니라 다양한 산업에서의 인력 관리 시스템의 혁신과 효율성을 높이는 데 기여할 수 있을 것이다.

참고문헌

1. 국내논저

[1] 국토교통부 (KOSIS 경유). (n.d.). 현장소재지별 건설공사계약 금액.
[2] 통계청 (KOSIS 경유). (2024, 8월 27일). 2023년 건설업조사 결과
[3] 뉴스핌. (2025, 5월 26일). 50대 이상 건설기술인, 'MZ세대' 대비 2배 이상 많다…"현장 고령화 우려". https://www.newspim.com/news/view/20250526000644
[4] 대한건설신문. (2025, 6월 2일). 건설현장의 노령화 대책 마련 시급. https://www.koscaj.com/news/articleView.html?idxno=316554
[5] 고용노동부. (2020, 7월 13일). 부당노동행위 업무처리 지침. https://www.moel.go.kr/local/taebaek/common/downloadFile.do?file_seq=21171265593&bbs_seq=1244013287120&bbs_id=LOCAL1.
[6] 비즈니스포스트. (2024, 12월 24일). 건설현장 고령화에 MZ 이탈로 '고용 질' 악화, 업황 반등 올라타지 못할 판. https://www.businesspost.co.kr/BP?command=article_view&num=377881
[7] 건설산업연구원. (n.d.). 건설 기능 인력의 수급 현황 및 고령화 실태.
[8] 건설근로자공제회 (고용노동부 경유). (2024, 7월 26일). '2023년도 건설근로자공제회 사업연보' 발간.
[9] 건설산업연구원. (n.d.). 국내 건설기업의 스마트 기술 - 활용 현황과 활성화 방향.
[10] 아하랩스.(2024, 7월 9일).설명 가능한 AI① XAI(eXplainable AI)란? - 개념,역사, 중요성.
[11] IBM. (n.d.). 설명 가능한 AI(XAI)란 무엇인가요?.
[12] Sheppard, C. (2017). Tree-based Machine Learning Algorithms: Decision Trees, Random Forests, and Boosting. Amazon.com.
[13] Hammad, M. M. (2024). Artificial Neural Network and Deep Learning: Fundamentals and Theory. doi:10.48550/arXiv.2408.16002 에서 검색됨.
[14] IBM. (n.d.). What Is Random Forest?. https://www.ibm.com/kr-ko/think/topics/random-forest

[15] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3), 210-229. doi:10.1147/rd.33.0210
[16] 김용성, 백수현, 권형표. (2018). 딥러닝 기반 건설 프로젝트 공사 기간 예측 모델 제안. 한국건설관리학회 논문집, 19(3), 19-28. doi:10.7467/KCRM.2018.19.3.019
[17] 박준혁, 김현준, 서정원. (2020). 기성고 데이터를 활용한 건설 프로젝트 공정 지연 위험 조기 감지 연구. 한국건설관리학회 논문집, 21(2), 3-12. doi:10.7467/KCRM.2020.21.2.003
[18] 강윤호, 윤석헌. (2022). 건축 공사비 예측에 머신러닝 적용 시 피처 스케일링 및 로그 변환이 예측 성능에 미치는 영향 분석. 대한건축학회 논문집 - 구조계, 38(10), 107-116. doi:10.7467/KCRM.2022.23.6.082
[19] 건설안전보건공단. (n.d.). 스마트 건설 안전 기술 소개. https://www.kosha.or.kr/kosha/data/media_board_view.do?menuId=100000000000000010&boardNo=213500
[20] 황윤호, 박기태, 박병석, 김동우. (2022). 빅데이터 기반 AI를 활용한 건설 현장 우선점검 대상 현장 선정 모델 개발. 한국건설관리학회 논문집, 23(3), 3-14. doi:10.7467/KCRM.2022.23.3.003
[21] Niu, Y., Niu, X., Niu, Y., Guo, Y., & Yuan, J. (2019). Machine learning for safety hazard identification in construction: A systematic review. Journal of Construction Engineering and Management, 145(12), 04019083. doi:10.1061/(ASCE)CO.1943-7862.0002384
[22] Ahmed, S., & Li, R. Y. M. (2020). Artificial intelligence and machine learning applications in construction quality management: A systematic review. Automation in Construction, 118, 103328. doi:10.1016/j.autcon.2020.103328
[23] Shi, M., Zhang, Y., & Han, R. (2021). An improved predictive maintenance framework for construction equipment based on machine learning. Journal of Civil Engineering and Management, 27(5), 450-463. doi:10.3846/jcem.2021.15545
[24] Jacobsen, M. R., Alimardani, M., & Vahdatikhaki, F. (2023). Construction worker activity recognition and productivity monitoring using deep learning and kinematic data. Automation in Construction, 149, 104764. doi:10.1016/j.autcon.2023.104764

[25] Kuvaas, B., Buch, R., Weibel, A., Dysvik, A., & Nerstad, C. G. L. (2017). Does performance management contribute to performance? A meta-analytic review. Journal of Management, 43(7), 2118-2151. doi:10.1177/0149206317730999
[26] 최지원, 박종성, 김현준. (2019). 건설 프로젝트 생산성 향상을 위한 데이터 기반 인력 관리 시스템 제안. 한국건설관리학회 논문집, 20(2), 13-22. doi:10.7467/KCRM.2019.20.2.013
[27] 대한건설협회. (n.d.). 건설업 임금실태조사. https://www.cak.or.kr/sub/info01_03_01.asp 에서 검색됨.
[28] 조규형, 박영훈. (2021). 건설 기능 인력의 임금 격차 요인 분석: 개인 인적 요인과 직무 특성 요인을 중심으로. 대한건설관리학회 논문집, 22(4), 3-12. doi:10.7467/KCRM.2021.22.4.003
[29] 김용성, 김현준, 박준혁. (2022). 건설업 임금실태조사 개선 방안에 대한 연구. 한국건설관리학회 논문집, 23(5), 15-24. doi:10.7467/KCRM.2022.23.5.015
[30] 이창훈, 이종혁. (2019). 건설업 적정 임금제 도입 방안 연구: 기능 인력 등급제 연계 방안을 중심으로. 한국건설관리학회 논문집, 20(6), 3-13. doi:10.7467/KCRM.2019.20.6.003
[31] Hwang, S., & Kim, Y. (2018). An analysis of the causes of cost overruns in construction projects. Journal of the Korea Institute of Building Construction, 18(3), 273-281. doi:10.5345/JKIBC.2018.18.3.273
[32] Mok, J., Lee, J., & Park, H. (2019). Development of an intelligent workforce management system for construction projects based on BIM and IoT. Automation in Construction, 106, 102875. doi:10.1016/j.autcon.2019.102875
[33] Molnar, C. (2022). Interpretable Machine Learning: A Guide for Making Black Box Models Explainable (2nd ed.). https://christophm.github.io/interpretable-ml-book/
[34] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning with Applications in R. Springer.
[35] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
[36] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.

doi:10.1038/323533a0
[37] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32. doi:10.1023/A:1010933404324
[38] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.
[39] 장현상, 이현수, 박선우. (2019). 인공신경망 기반 콘크리트 타설 공정 생산성 예측 모델 개발. 한국건설관리학회 논문집, 20(2), 3-12. doi:10.7467/KCRM.2019.20.2.003
[40] 김현준, 이재환, 박준혁. (2020). 데이터 마이닝을 활용한 건설 프로젝트 생산성 저해 요인 분석. 한국건설관리학회 논문집, 21(4), 3-12.
doi:10.7467/KCRM.2020.21.4.003
[41] Yuan, H., Ma, Z., Fang, C., & Li, R. (2020). Machine learning-based approach for construction labor productivity prediction. Automation in Construction, 119, 103362. doi:10.1016/j.autcon.2020.103362
[42] Choi, K., Lee, D., & Kim, H. (2018). Optimized workforce allocation for construction projects using genetic algorithm. Journal of Civil Engineering and Management, 24(8), 643-653. doi:10.3846/jcem.2018.665
[43] Liu, Q., Li, S., Wang, H., & Zhou, Y. (2019). Deep learning-based framework for construction workforce scheduling considering complex constraints. Automation in Construction, 106, 102875. doi:10.1016/j.autcon.2019.102875
[44] Gwak, H., Choi, Y., & Lee, S. (2021). Hazard prediction model in construction sites based on workers' biosignals and behavior data. Journal of Safety Research, 77, 221-230. doi:10.1016/j.jsr.2021.03.003
[45] 이정현, 김경환. (2023). BIM 데이터를 활용한 건설 근로자 역량 평가 프레임워크 제안. 한국건설관리학회 논문집, 24(1), 3-12. doi:10.7467/KCRM.2023.24.1.003
[46] Shi, M., Zhang, Y., Han, R., & Wang, J. (2022). AI-based system for objective assessment of construction worker skill level through motion analysis of sensor data. Automation in Construction, 142, 104443.
doi:10.1016/j.autcon.2022.104443
[47] Arditi, D., & Mochtar, K. (1998). Trends in construction labor productivity.

Journal of Construction Engineering and Management, 124(1), 15-22. doi:10.1061/(ASCE)0733-9364(1998)124:1(15)
[48] Dai, J., Goodrum, P. M., & Haas, C. T. (2009). The impact of work stoppages on construction labor productivity. Journal of Construction Engineering and Management, 135(8), 700-709. doi:10.1061/(ASCE)CO.1943-7862.0000030
[49] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.
[50] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
[51] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.). O'Reilly Media.
[52] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. International Joint Conference on Artificial Intelligence, 14(2), 1137-1143.
[53] Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures for model selection. Statistics Surveys, 4, 40-79. doi:10.1214/09-SS054

ABSTRACT

Application of Machine Learning Techniques for Evaluating Construction Workers' Competency and Estimating Labor Costs

Ju, Jaeil
Computer Science
Graduate School of Education
Incheon National University

TThis study proposes a machine learning-based evaluation model that quantitatively assesses the competency of construction workers and enables fair labor cost estimation based on this assessment. Currently, labor cost estimation in construction sites often relies on the subjective judgment of individual managers, leading to inconsistent wages for the same work. To address this, we developed a predictive model capable of evaluating worker competency and estimating labor costs using quantitative indicators such as task difficulty and task duration. Three machine learning algorithms－Decision Tree, Artificial Neural Network (ANN), and Random Forest－were applied for modeling. Performance evaluation results showed that Artificial Neural Network (ANN) exhibited the most superior accuracy and balanced performance. This research demonstrates the potential for transitioning worker evaluation and compensation systems to a data-driven approach, serving as foundational data for future practical application and expansion to other industries.

Key Words : Machine Learning, Construction Workers, Labor Cost Estimation, Competency Evaluation, Random Forest

부 록(Appendix)

<부록 1 > 본 연구의 결과 도출에 사용된 Python 코드

본 부록에 삽입된 코드는 Google Colab 환경에서의 실행을 기준으로 셀(Cell)별로 구분되어 있다. 가상 데이터 생성 과정에 무작위성이 포함되어 있으므로, 코드를 재실행할 경우 본 연구 결과의 정확한 수치와 미세한 차이가 발생할 수 있다. 그러나 전반적인 성능 경향과 연구의 결론은 일관되게 유지될 것이다.

1. Cell 1

```
### Cell 1 ###
# 나눔고딕 폰트 설치
!apt-get update -qq
!apt-get install -y fonts-nanum -qq

# 폰트 캐시 갱신
!fc-cache -fv
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm
# 설치된 나눔고딕 폰트 파일 경로
font_path = '/usr/share/fonts/truetype/nanum/NanumGothic.ttf'
# 폰트 등록
fm.fontManager.addfont(font_path)
# 폰트 이름 추출
font_name = fm.FontProperties(fname=font_path).get_name()

# Matplotlib 기본 폰트를 나눔고딕으로 설정
plt.rc('font', family=font_name)
```

```
# 음수 부호가 깨지지 않도록 설정
plt.rcParams['axes.unicode_minus'] = False

# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, LabelBinarizer, LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, roc_curve, auc

import seaborn as sns
import warnings
import os

# Suppress warnings for cleaner output (optional)
warnings.filterwarnings('ignore')

# --- 0. Virtual Data Generation Function ---
# This function generates the 10,000 virtual records for the thesis.
# It ensures reproducibility by setting a random seed.
def generate_competency_data(num_samples=10000, random_seed=42):
    np.random.seed(random_seed)

    data = []

    # Define competency levels and their approximate proportions
    competency_levels = ['S', 'A', 'B', 'C', 'D']
```

```
    # Base duration map for each task difficulty (minutes)
    # This simulates a linear relationship where higher difficulty means longer base time
    difficulty_base_time = {i: i * 10 for i in range(1, 11)}

    job_types = ['목공', '철근공', '용접공', '미장공', '보통인부', '배관공', '전기공', '타일공']

    for _ in range(num_samples):
        task_difficulty = np.random.randint(1, 11) # Task Difficulty from 1 to 10
        base_duration_for_task = difficulty_base_time[task_difficulty]

        # Assign initial competency level based on predefined distribution (to ensure roughly correct class balance)
        competency_level_assigned = np.random.choice(competency_levels, p=[0.1, 0.2, 0.4, 0.2, 0.1])

        # Define efficiency factor based on assigned competency level
        # This factor determines how much the actual duration deviates from the base duration
        if competency_level_assigned == 'S':
            efficiency_factor = np.random.uniform(0.50, 0.60) # Highly efficient
        elif competency_level_assigned == 'A':
            efficiency_factor = np.random.uniform(0.60, 0.80) # Efficient
        elif competency_level_assigned == 'B':
            efficiency_factor = np.random.uniform(0.80, 1.10) # Average
        elif competency_level_assigned == 'C':
            efficiency_factor = np.random.uniform(1.10, 1.40) # Slightly inefficient
        else: # 'D'
            efficiency_factor = np.random.uniform(1.40, 2.00) # Very inefficient
```

```
        # Calculate task duration with efficiency factor and add random noise
        noise = np.random.normal(0, base_duration_for_task * 0.05) # Add Gaussian noise, 5% of base duration
        task_duration = base_duration_for_task * efficiency_factor + noise
        task_duration = max(5, round(task_duration)) # Ensure duration is at least 5 minutes and is an integer

        # Generate experience (years), biased by competency level
        experience = 0
        if competency_level_assigned in ['S', 'A']:
            experience = np.random.randint(3, 31) # S, A often have high experience
            if np.random.rand() < 0.05: # But a small chance of "natural talent" S/A with low experience
                experience = np.random.randint(0, 3)
        elif competency_level_assigned == 'B':
            experience = np.random.randint(1, 21) # Average experience
        else: # C, D
            experience = np.random.randint(0, 16) # Inefficient often have low experience
            if np.random.rand() < 0.1: # Small chance of experienced but inefficient C/D
                experience = np.random.randint(15, 31)

        # Ensure experience is within the valid range
        experience = max(0, min(30, experience))

        job_type = np.random.choice(job_types)

        data.append([task_difficulty, task_duration, experience, job_type,
```

```
competency_level_assigned])

    df = pd.DataFrame(data, columns=['Task_Difficulty', 'Task_Duration', 'Experience', 'Job_Type', 'Competency_Level'])

    # Recalculate and assign Competency_Level based on the actual Task_Duration relative to Task_Difficulty's base time
    # This step ensures the 'Competency_Level' column strictly follows the defined rules for the generated data
    def final_assign_competency_level(row):
        difficulty = row['Task_Difficulty']
        duration = row['Task_Duration']

        base_time_for_difficulty = difficulty_base_time[difficulty]
        efficiency_ratio = duration / base_time_for_difficulty

        if efficiency_ratio <= 0.6:
            return 'S'
        elif efficiency_ratio <= 0.8:
            return 'A'
        elif efficiency_ratio <= 1.1:
            return 'B'
        elif efficiency_ratio <= 1.4:
            return 'C'
        else:
            return 'D'

    df['Competency_Level'] = df.apply(final_assign_competency_level, axis=1)

    return df
```

2. Cell 2

```
### Cell 2 ###
# --- Data Generation and Loading ---
# This cell checks if the data file exists and generates it if not.
file_path = 'competency_data.csv'
if not os.path.exists(file_path):
    print("competency_data.csv not found. Generating new data...")
    competency_df = generate_competency_data(num_samples=10000)
    competency_df.to_csv(file_path, index=False)
    print(f"Data generated and saved to '{file_path}'.")
else:
    print(f"'{file_path}' already exists. Loading existing data.")

df = pd.read_csv(file_path)
print("Data loaded successfully.")
```

3. Cell 3

```
### Cell 3 ###
# --- Data Preparation and Preprocessing (4.1 section) ---
# Separate features (X) and target (y_original)
X = df.drop('Competency_Level', axis=1)
y_original = df['Competency_Level'] # y_original holds original string labels

# One-hot encode categorical feature 'Job_Type'
X = pd.get_dummies(X, columns=['Job_Type'], drop_first=True) # drop_first=True
to avoid multicollinearity

# Scale numerical features using MinMaxScaler
numerical_cols = ['Task_Difficulty', 'Task_Duration', 'Experience']
scaler = MinMaxScaler()
```

```
X[numerical_cols] = scaler.fit_transform(X[numerical_cols])

# Convert target labels to numerical values (0, 1, 2, 3, 4) for model training
label_encoder = LabelEncoder()
y_encoded_full = label_encoder.fit_transform(y_original) # Fit and transform on the full original y
# Store the original class names for later use (e.g., classification report)
target_names_ordered = label_encoder.classes_
print(f"Original target classes: {target_names_ordered}")
print(f"Encoded target values: {np.sort(np.unique(y_encoded_full))}")

# Split data into training and testing sets (70% train, 30% test)
# Use y_encoded_full for stratify to ensure consistent distribution in splits
X_train, X_test, y_train_original, y_test_original = train_test_split(X, y_original, test_size=0.3, random_state=42, stratify=y_encoded_full)

# Encode the split target sets for model training
y_train_encoded = label_encoder.transform(y_train_original)
y_test_encoded = label_encoder.transform(y_test_original)

print(f"\nTraining dataset size: {X_train.shape[0]} samples")
print(f"Test dataset size: {X_test.shape[0]} samples")
print(f"Number of features (after encoding): {X_train.shape[1]}")
```

4. Cell 4

```
### Cell 4 ###
# --- 4.2.1 Decision Tree Model Training ---
dt_model = DecisionTreeClassifier(max_depth=5, min_samples_leaf=10, criterion='gini', random_state=42)
dt_model.fit(X_train, y_train_encoded) # Use encoded y_train
print("Decision Tree model trained.")
```

5. Cell 5

```
# --- 4.2.2 Artificial Neural Network (ANN) Model Training ---
# Increased max_iter for better convergence on complex datasets
mlp_model = MLPClassifier(hidden_layer_sizes=(100, 50), activation='relu', solver='adam',
                          learning_rate_init=0.001, max_iter=1000,
```

```
random_state=42, early_stopping=True, n_iter_no_change=50, verbose=False)
mlp_model.fit(X_train, y_train_encoded) # Use encoded y_train
print("ANN model trained.")
```

6. Cell 6

```
# --- 4.2.3 Random Forest Model Training ---
rf_model = RandomForestClassifier(n_estimators=300, max_features='sqrt',
max_depth=10,
                                  min_samples_leaf=5, random_state=42)
rf_model.fit(X_train, y_train_encoded) # Use encoded y_train
print("Random Forest model trained.")
```

7. Cell 7

```
### Cell 7 ###
# --- 4.3 Model Performance Evaluation and Comparison (Tables 4-1, 4-2, 4-3,
4-4, 4-5) ---
models = {
    "결정 트리": dt_model,
    "인공신경망 (ANN)": mlp_model,
    "랜덤 포레스트": rf_model
}

results_summary = []
class_reports_data = {}
confusion_matrices_data = {}
roc_plot_data = {} # Data for ROC curve plotting

for name, model in models.items():
    y_pred_encoded = model.predict(X_test)
```

```
    y_pred_original = label_encoder.inverse_transform(y_pred_encoded) # Convert back to original labels for reporting

    # 4.3.1 Overall Performance Summary (Table 4-1)
    accuracy = accuracy_score(y_test_encoded, y_pred_encoded) # Use encoded for accuracy_score

    # Generate full classification report using original labels for target_names
    report = classification_report(y_test_original, y_pred_original, output_dict=True, zero_division=0, target_names=target_names_ordered)

    # Extract macro-averaged metrics for overall summary table
    macro_precision = report['macro avg']['precision']
    macro_recall = report['macro avg']['recall']
    macro_f1 = report['macro avg']['f1-score']

    results_summary.append({
        "모델": name,
        "정확도 (Accuracy)": accuracy,
        "Macro Precision": macro_precision,
        "Macro Recall": macro_recall,
        "Macro F1-Score": macro_f1
    })

    # Store full classification report for detailed class-wise tables
    class_reports_data[name] = report

    # Calculate confusion matrix using original labels for consistent display
    cm = confusion_matrix(y_test_original, y_pred_original, labels=target_names_ordered)
    confusion_matrices_data[name] = pd.DataFrame(cm, index=[f'실제_{cls}' for
```

```
cls in target_names_ordered],
                                                columns=[f'예측_{cls}' for cls
in target_names_ordered])

    # 4.3.3 ROC Curve and AUC Analysis - Data collection (plotting in next
cell)
    y_pred_proba = model.predict_proba(X_test)

    # Binarize the true labels for multi-class ROC (One-vs-Rest) using
y_test_encoded (numerical)
    lb = LabelBinarizer()
    lb.fit(np.sort(y_test_encoded)) # Fit on sorted unique encoded labels
    y_test_binarized = lb.transform(y_test_encoded)

    # Micro-average ROC
    fpr_micro, tpr_micro, _ = roc_curve(y_test_binarized.ravel(),
y_pred_proba.ravel())
    roc_auc_micro = auc(fpr_micro, tpr_micro)

    # Store individual class ROC data for potential plotting (and for macro AUC
calculation)
    per_class_roc = []
    per_class_aucs = []
    for i, class_label_encoded in enumerate(lb.classes_): # Iterate through ordered
encoded classes
        fpr, tpr, _ = roc_curve(y_test_binarized[:, i], y_pred_proba[:, i])
        per_class_roc.append({'fpr': fpr, 'tpr': tpr})
        per_class_aucs.append(auc(fpr, tpr))

    # Macro-average AUC is the average of individual class AUCs
    roc_auc_macro = np.mean(per_class_aucs)
```

```
    roc_plot_data[name] = {
        'micro_fpr': fpr_micro, 'micro_tpr': tpr_micro, 'micro_auc': roc_auc_micro,
        'macro_auc': roc_auc_macro,
        'classes': target_names_ordered, # Use original class names for legend
        'per_class_roc': per_class_roc,
        'per_class_aucs': per_class_aucs
    }

# Print 4.3.1 Overall Performance Summary (Table 4-1)
print("\n--- 4.3.1 전체 성능 요약 (표 4-1) ---")
summary_df = pd.DataFrame(results_summary)
print(summary_df.round(3)) # Round to 3 decimal places for presentation

# Print 4.3.2 Confusion Matrices and Class-wise Performance Reports (Tables 4-2, 4-3, 4-4, 4-5)
print("\n--- 4.3.2 혼동 행렬 및 클래스별 성능 분석 ---")
for name, cm_df in confusion_matrices_data.items():
    print(f"\n--- {name} 혼동 행렬 ---")
    print(cm_df.to_string()) # .to_string() for full table display in console

    print(f"\n--- {name} 클래스별 분류 보고서 ---")
    # Convert report dictionary to DataFrame for better readability
    report_df = pd.DataFrame(class_reports_data[name]).transpose()
    report_df['support'] = report_df['support'].astype(int) # Ensure support is integer
    print(report_df.round(3).to_string()) # .to_string() for full table display
```

8. Cell 8

```
### Cell 8 ###
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# --- Remove Korean font settings ---
# No special font settings for English

# Assuming 'models' and 'roc_plot_data' are available from previous cells.
# If running this cell independently, ensure the necessary variables are defined or loaded.
# Example definition if running standalone (replace with your actual data/models):
# models = {
#     "Decision Tree": None,
#     "Artificial Neural Network (ANN)": None,
#     "Random Forest": None
# }
# roc_plot_data = {
#         "Decision Tree": {'micro_fpr': np.array([0, 0.2, 0.8, 1]), 'micro_tpr': np.array([0, 0.4, 0.9, 1]), 'micro_auc': 0.78, 'classes': ['A', 'B', 'C', 'D', 'S']},
#     "Artificial Neural Network (ANN)": {'micro_fpr': np.array([0, 0.01, 0.02, 1]), 'micro_tpr': np.array([0, 0.98, 0.99, 1]), 'micro_auc': 1.00, 'classes': ['A', 'B', 'C', 'D', 'S']},
#         "Random Forest": {'micro_fpr': np.array([0, 0.1, 0.5, 1]), 'micro_tpr': np.array([0, 0.7, 0.95, 1]), 'micro_auc': 0.88, 'classes': ['A', 'B', 'C', 'D', 'S']}
# }


# --- Plotting ROC Curves (Figure 4-1) ---
```

```
plt.figure(figsize=(12, 8))
colors = sns.color_palette('tab10', n_colors=len(models)) # Using a standard color palette

for i, (name, data) in enumerate(roc_plot_data.items()):
    plt.plot(data['micro_fpr'], data['micro_tpr'], color=colors[i], lw=2,
             label=f'{name} (Micro-Avg AUC = {data["micro_auc"]:.2f})')

plt.plot([0, 1], [0, 1], color='gray', lw=1, linestyle='--', label='Random Guess (AUC = 0.5)')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('Receiver Operating Characteristic (ROC) Curve Comparison')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

9. Cell 9

```
### Cell 9 ###
# --- 4.5.2 Feature Importance Analysis (Random Forest) (Table 4-7 & Figure 4-2) ---
print("\n--- 4.5.2 주요 변수 영향도 분석 (랜덤 포레스트) ---")

# Extract feature importances from the Random Forest model
feature_importances = rf_model.feature_importances_
feature_names = X_train.columns
```

```
# Create a DataFrame for better presentation
importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importances})

# Sort features by importance in descending order
importance_df = importance_df.sort_values(by='Importance', ascending=False)

# Calculate importance percentage
importance_df['Importance_Percentage'] = importance_df['Importance'] * 100
print(importance_df[['Feature', 'Importance_Percentage']].round(1).to_string())  # .to_string() for full table display

# Plotting Feature Importance (Figure 4-2 Concept)
plt.figure(figsize=(12, 7))
sns.barplot(x='Importance_Percentage', y='Feature', data=importance_df, palette='viridis') # Using a vibrant color palette
plt.title('Feature Importance from Random Forest Model')
plt.xlabel('Importance (%)')
plt.ylabel('Features')
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```