붙임3

연구논문/작품 최종보고서

2022학년도 제 2학기

제목 : 기계 학습을 통한 NBA 선수연봉 예측 모델 구현

김호진(2016314786)

2022년 10월 25일

지도교수: 우 홍 욱 서명

계획(10)	주제(20)	개념(20)	상세(30)	보고서(20)	총점(100)

* 지도교수가 평가결과 기재

■ 요약

본 연구에서는 프로스포츠 선수들의 연봉이 선수들의 개인 성적과 팀 승리에 대한 기여 등에 영향을 받는다는 가정하에 NBA(National Basketball Association)에서 뛰고 있는 선수들의 전년도 성적을 기반으로 다음 해 연봉을 예측하는 모델을 만들고자 한다. 관련 데이터의 수집은 NBA의 다양한 통계 자료를 공식적으로 제공하고 있는 Basketball Reference에서 필요한 정보를 크롤링하여 사용한다. 이렇게 얻어진 데이터는 다변량 검정 기법을 이용하여 feature를 분석하고, 이를 기반으로 정제과정을 거친다. 데이터의 변동성을 줄이기 위해서 선수별로 데이터를 가공하고, 연도별 평균값을 이용해 시간에 따른 물가 상승의 영향력을 낮춘다. 데이터 전처리 단계에서는 기계 학습 과정 중 편파성을 일으킬 수 있는 이상치 데이터 값을 파악하여 사전에 제거한다. 이후 탐색적 데이터 분석(Exploratory Data Analysis)기법을 활용하여 가공된 데이터를 분석 및 비교한 뒤 선수들의 연봉을 결정하는 주요 요인을 선택한다. 탐색적 데이터 분석은 회귀 분석(regression analysis)과 히트맵(heat map)을 이용한 시각화(visualization)를 통해 진행한다. 회귀 분석의 경우, feature 간의 독립성을 전제로 하는 분석 방법이기 때문에 예측 성능을 높이기 위해 분산 팽창 요인(VIF) 계수를 활용하여 다중 공선성 문제를 해결한다. 탐색적 데이터 분석을 통해 연구에 적합한 features가 선택되면, 선택된 연봉 결정요인과 기계 학습에 사용되는 다양한 알고리즘을 결합하여 지도학습 기반의 학습을 진행하고 이를 토대로 선수들의 연봉을 예측하는 모델을 만든다. 이때 다양한 학습 모델을 구현하기 위해서 Multiple Linear Regression, Lasso Regression, Ridge Regression과 같이 기계 학습에서 사용되는 기본적인 알고리즘에 더해 Boosting 앙상블을 이용하는 XGBoost, LightGBM를 추가로 활용한다. 모델링이 완료되면 각 모델의 예측값을 확인한다. 인공지능 모델 간의 성능을 비교하기 위해서 RMSE(Root-Mean-Squared Error) Score와 R-Squared score를 평가지표로 사용한다. 또한, Cross Validation을 적용하여 개선된 정확도 측정을 제공한다. 구현 모델 중 가장 높은 예측도를 보인 모델을 선택하여 feature importance를 측정하고, 높은 중요도를 가진 features를 재선정한 뒤 최종 모델을 구현한다. 최종 모델이 예측한 연봉과 실제 연봉 데이터를 비교하는 시각화 자료를 통해 해당 모델이 얼마나 정확한 예측을 하였는지 직관적으로 확인한다. 본 연구의 결과는 선수와 구단 사이에서 합리적인 연봉 계약이 성사될 수 있도록 도움을 주고, 스포츠 팬들에게 새로운 재미를 제공한다. 또한, 저평가 혹은 고평가된 선수들을 쉽게 파악할 수 있어 효율적이고 강력한 팀을 구성하는 데 활용 가치가 충분하다. 더 나아가 본 연구의 성과는 NBA 선수들의 연봉 예측에 국한되지 않고 다양한 분야에서 활용될 수 있다.

■ 서론

가) 제안 배경 및 필요성

오늘날 데이터가 산업 전반의 다양한 분야로 활용 폭을 넓히게 되면서 그동안 직관과 경험에 의존해왔던 스포츠 분야에서도 데이터의 활용이 점차 증가하고 있다. 이러한 추세에 더불어 기술의 발달로 인해 가용할 수 있는 데이터의 양이 폭발적으로 증가하게 되었고 빅데이터를 다루는 기계 학습의 중요성 역시 커지게 되었다. 인공지능 분야의 하나인 기계 학습은 컴퓨터가 데이터를 통해 수행해야 할 작업을 스스로 학습하기 위한 기술을 일컫는다. 기계 학습은 데이터의 규모가 크고 복잡하여 사람이 직접 해결하기 어려운 문제를 해결하는 데 유용하게 사용되고 있으며, 스포츠 산업 외에도 바이오, 생산, 금융, 통신 등 많은 분야에서 활용되고 있다.

다양한 스포츠 분야 중 야구의 경우, 대표적인 데이터 스포츠로써 경기의 모든 과정이 기록원들에 의해 기록되고 있다. 야구에서 각각의 플레이는 독립시행으로 이루어지기 때문에 다른 스포츠에 비해 데이터와 통계를 활용한 의미 있는 분석이 가능하다는 장점을 지닌다. 이러한 장점들을 기반으로 엄청난 양의 데이터가 다양한 측면에서 축적되었으며, 다른 종목에 비해 데이터의 활용이 빠르게 이루어졌다. 특히 영화 머니볼의 실제 주인공인 오클랜드 애슬래틱스의 단장 윌리엄 라마 빈이 경기 데이터를 바탕으로 선수들을 영입하고, 이것이 틀리지 않았음을 성적을 통해 증명해내면서 야구계에서 데이터의 활용은 선택이 아닌 필수가 되었다. 이후 데이터를 활용하기 위한 연구가 꾸준히 이루어졌고 기존의 야구 기록이 갖는 단점을 통계학적으로 보완한 세이버 매트릭스(Saber Metrics)가 등장하게 되었다.

세이버 매트릭스는 MLB에 새로운 혁신을 일으켰고, 이후 다른 스포츠 종목에까지 영향을 미쳤다. 철저하게 통제된 상황에서 투수와 타자 간의 승부로 기본적인 플레이가 진행되는 야구와 다르게 농구는 동적으로 경기가 진행되기 때문에 데이터의 활용이 어려울 거라 생각되었다. 하지만 세이버 매트릭스가 성공적으로 활용되면서 농구계에서도 통계적 요소를 스포츠에 이용하고자 하는 노력이 시작되었다. 다행히 농구는 다른 스포츠에 비해서 기본적으로 기록되는 데이터가 많았기 때문에 기존의 1차 기록을 가공한 PER, WS, BPM 등의 2차 기록이 빠르게 등장할 수 있었다. 오늘날 NBA는 존 홀린저에 의해 개발된 APBR(Association for Professional Basketball Research) 매트릭스를 중점적으로 활용하고 있으며, 우리나라 역시 KBL 레퍼런스에서 경기 기록을 분석한 후 관련 서비스를 제공하고 있다.

이처럼 데이터 활용에 대한 관심이 증가하면서 농구계에서도 기계 학습 기법을 이용해 데이터를 효과적으로 분석하기 위한 연구가 활발하게 진행되고 있다. Fadi

Thabtah는 Naïve Bayes, Artificial Neural Network 그리고 Decision Tree 등의 기계 학습 모델을 사용하여 정확한 NBA 경기 결과를 예측하고자 했으며[1], Jackie B. Yang은 과거의 경기 결과를 토대로 얻은 데이터를 Support Vector Machine (SVM)과 결합하여 NBA 플레이오프 진행 상황을 예측하는 모델을 제안하였다[2]. Santhosh Narayan는 비지도학습 기계 학습 기법을 활용하여 선수들의 포지션이나 감독, 구단의 특징 등에 따라 달라지는 경기 전략이 실제 경기 결과에 어떤 영향을 미치는 지에 대해 알아보았다[3]. NBA뿐만 아니라 국내 프로농구에서도 기계 학습 기법을 활용하려는 움직임은 늘어나고 있다. 김세중 등은 데이터 마이닝 기법을 이용하여 한국 프로농구 경기를 분석하였고[4], KBL에서는 프로농구 데이터를 다양한 각도로 활용하기 위해서 최근 몇 년간 데이터 활용 경진대회를 개최하고 있다[5].

시간이 갈수록 스포츠 시장의 규모는 커지고 있다. 그중에서도 미국 프로스포츠는 전 세계에서 가장 큰 가치를 지닌다. 문화체육관광부가 발간한 2019 스포츠산업백서에 따르면 미국의 스포츠 산업 시장 규모는 약 617조 원에 달한다[6]. 특히 미국프로농구(NBA)는 미국프로미식축구(NFL), 미국프로야구(MLB), 미국프로아이스하키(NHL)와 더불어 미국의 4대 스포츠를 이루고 있으며, 빠른 템포의 경기 속도와 신발, 의류 등을 활용한 공격적인 마케팅을 앞세워 전 세계적인 인기를 이끌고 있다.

이렇게 시장의 규모가 커지면서 선수들의 연봉 역시 천문학적으로 높아졌다. NBA의 대표적인 스타, 스테판 커리의 연봉은 약 480억으로 수년째 리그 내 선수 중 최고의 자리를 차지하고 있다. 이처럼 선수들의 연봉이 빠른 속도로 증가하게 되면서 구단을 운영하는 데 있어 연봉 협상은 매우 중요한 요소가 되었다. 그러면서 선수단 연봉을 줄이고자 하는 구단과 자신의 가치를 인정받으려는 선수들 사이의 갈등이 자연스럽게 발생하였다. 오늘날 연봉과 관련한 선수와 구단 간의 마찰은 흔히 접할 수 있는 뉴스가 되었으며, 2011년에는 수익 배분을 둘러싼 노사 간의 갈등으로 인해 NBA 리그가 잠정 폐쇄되기도 했다.

본 연구에서는 전년도 성적이 기반이 된 합리적인 선수 연봉을 제시하고 있기 때문에 선수와 구단 사이의 갈등을 봉합하고 계약을 성사시키는데 도움을 줄 수 있다. 특히 본 연구의 결과를 하나의 기준으로 삼음으로써 선수와 구단 양측 모두가 만족하는 적절한 합의 지점을 찾아낼 수 있다. 또한, 해당 연구는 NBA 팬들에게도 새로운 흥미를 제공한다. 팬들은 연구 결과를 토대로 구단 운영이 현명하게 되고 있는지 판단할 수 있으며, 어떤 선수가 상대적으로 팀에 도움이 됐는지 쉽게 알아볼 수 있다. 팀을 구성하는 데 있어서도 저평가되거나 고평가된 선수들을 한눈에 파악할 수 있기 때문에 효율적인 비용으로 강력한 전력을 가진 팀을 구성하는 데 도움이 된다. 더 나아가 데이터셋을 주제에 맞춰 적절하게 응용한다면 본 연구가 NBA에만 국한되지 않고 다양한 분야에서 활용될 수 있을 거라고 사료 된다.

나) 연구논문/작품의 목표

스포츠 분야에서 데이터와 기계 학습 간의 결합은 경기력 분석이나 결과 예측 등 스포츠적인 측면에 국한되지 않고 관중 수나 마케팅, 선수들의 연봉과 같은 부가적인 요소의 예측에도 활용되고 있다. 박소현 등은 K-평균 군집화 방법과 K-근접 이웃 방법을 통해 스포츠 경기장을 찾는 관람객의 빅데이터를 분석하고 소비자에게 맞춤형 마케팅 서비스를 제공하는 연구를 진행하였으며[7], 박진욱 등은 인공신경망(Artificial neural network) 모형을 이용하여 한국 프로 야구 관중 수를 예측하고자 했다[8]. 프로스포츠가 오래전부터 정착된 미국의 경우, 선수들의 개인 성적과 연봉 간의 상관 관계에 대한 연구가 여러 차례 있어 왔고, 우리나라에서도 송종우에 의해 한국 프로스포츠 선수들의 연봉에 대한 다변량적 분석이 이루어졌다[9].

다만 이러한 부가적 요소들에 대한 예측은 비교적 최근에 시작되었기 때문에 지금까지 많은 연구가 이루어지지 못했다. 그로 인해 이상적인 데이터셋이나 학습 모델이 불분명한 상태이며, 개선이 필요한 부분도 상당수 존재한다. 그뿐만 아니라 스포츠 분야의 예측 모델의 경우, 많은 노력이 있었음에도 아직까지 최적의 결과를 얻지 못하고 있다. 이처럼 스포츠에서 예측이 어려운 이유는 많은 경기 기록 중 주제에 영향을 미치는 요소를 선별하기가 어렵고, 자료 간의 중복 요인이나 결과에 큰 영향을 미치는 이상치 값으로 인해 예측에 사용된 학습 모델이 좋은 성능을 보이지 못하는 데에 있다.

그러므로 본 연구에서는 선수 연봉에 영향을 주는 요소의 최적화를 위해 득점과 도움 등 기본적인 기록에 더해서 1차 기록을 가공한 PER, WS, PIPM, VORP 등의 2차 기록을 함께 사용하고, 탐색적 데이터 분석을 활용하여 연구에 적합한 데이터 features를 추출한다. 탐색적 데이터 분석은 회귀 분석(regression analysis)과 시각화(visualization)를 통해 진행한다. 또한, 각 feature에 대한 P-Value를 확인하여 모델링을 하는 데 있어 유의미한 feature인지 판단하고, 히트맵(heat map)을 통해 feature 간의 상관관계를 확인하여 연봉을 결정하는 주요 요인들을 찾아낸다. 이 밖에도 VIF(Variance Inflation Factors) 계수를 이용해 다중 공선성(Multicollinearity)을 고려함으로써 예측의 성능을 높인다.

선택된 데이터는 학습 데이터와 테스트 데이터로 나눈 뒤 인공지능 분야의 머신러닝 및 딥러닝 알고리즘인 Linear Regression, Lasso Regression, Ridge Regression, XGBoost, LightGBM 등과 결합하여 NBA 선수들의 연봉 예측을 위한 모델을 구현한다. 이후 각 예측 모델을 비교하여 실제 연봉과 가장 유사한 예측도를 보이는 최적의 모델을 찾아낸다. 다양한 인공지능 모델의 성능을 분석 및 비교하기 위해서 RMSE(Root-Mean-Squared Error) Score와 R-Squared score를 평가지표로 사용하

고, Cross Validation을 적용해 기존보다 정교한 정확도 측정을 제공한다. 가장 정확한 예측도를 보인 모델을 기준으로 feature importance를 측정하고 이를 기반으로 높은 중요도를 가진 features를 재선정하여 최종 모델을 구현한다. 이러한 일련의 과정들을 통해서 데이터 내에 존재하는 다양한 기록 중 연봉 결정에 중요한 역할을 하는 features를 찾아내고, 학습 모델의 성능을 향상시킬 수 있다. 그러므로 본 연구작품은 기존 스포츠 분야의 예측 모델이 가지는 문제점을 어느 정도 해결하고, 사용자에게 개선된 성능을 제공한다.

본 연구작품의 궁극적인 목표는 실제 스포츠 분야에 도움이 될 만한 인공지능 모델을 제공하는 것이다. 실제 스포츠 분야에서 본 연구의 성과를 활용할 수 있게 된다면, 앞서 언급했던 다양한 장점들을 사용자에게 제공할 수 있다. 연봉 예측 결과는 선수들과 구단 사이에서 계약을 성사시키는데 도움이 될 수 있으며, 팬들에게 새로운 흥미를 유발하고 구단이 팀을 구성하는 데 있어 유용한 정보를 제공한다. 더 나아가 주제에 맞게 데이터를 적절하게 응용한다면 NBA 외의 다양한 스포츠 분야에서 본 연구의 성과를 활용할 수 있을 것이다.

실제로 오늘날 많은 스포츠 구단에서 인공지능을 적극적으로 활용하고 있다[10]. 잉글랜드 프리미어리그의 명문 축구클럽인 리버풀 FC는 구글 산하 AI 개발기업 딥마인드와 협력해 전술 분석을 진행하고 있으며, 선수 부상을 막기 위해 AI 기반의 Zone7 분석 프로그램을 도입하였다. 또한, 2022년 베이징 올림픽에 출전한 미국 피겨스케이팅 선수들은 피로를 추적하기 위해서 뉴저지에 위치한 '4D 모션 스포츠(4D Motion Sports)'의 동작 분석 프로그램을 활용하였다. EPL의 경우, 2021/22 시즌부터 오라클을 공식 파트너로 선정하면서 클라우드 기반의 매치 인사이트 솔루션을 도입했고, 해당 기술을 통해 실시간 승리 확률을 예측하여 관련 정보를 시청자들에게 제공하고 있다.

이렇게 실제 스포츠 구단에서 사용되고 있는 인공지능 모델들은 다양하고 전문적인 기계 학습 알고리즘을 활용하여 엄청난 양의 데이터를 학습하기 때문에 성능적인 면에서 본 연구작품의 최종 모델과 차이가 존재한다. 그러므로 구현 모델이 연구의 궁극적인 목표에 도달하기 위해서는 본 연구에 그치지 않고, 추가 연구를 꾸준하게 진행하여 모델을 발전시킬 필요가 있다.

다) 연구작품 전체 overview

연구작품을 진행하기에 앞서 연구의 방향성을 잡기 위해서 주제와 연관된 선행 연구논문을 조사한다. 최근 논문들의 동향을 살펴보며 기계 학습 분야의 전반적인 기술 흐름을 살펴보고, 스포츠 분야와 인공지능 분야가 결합이 된 다양한 사례들을 참고하여 연구의 진행 방식을 결정한다.

연구작품은 총 일곱 단계(데이터 수집, 데이터 정제, 데이터 탐색, 데이터 준비, 모델링, 모델 분석 및 비교, 결과 해석)에 거쳐 진행한다. 데이터 수집 단계에서는 NBA의 다양한 통계 자료를 제공하는 사이트로부터 연구에 필요한 데이터를 가져온다. 본 연구에서는 Basketball Reference에서 선수의 연봉 관련 데이터와 기록 관련 데이터를 크롤링하였다. 이 과정에서 선수 기록 관련 데이터의 feature로 득점, 어시스트, 리바운드, 스틸 등의 1차 기록뿐만 아니라 PER, WS, BPM 등의 2차 기록을 포함하여 해당 선수와 관련된 다양한 통계 자료가 인공지능 예측 모델을 구현하는 데 반영될 수 있도록 했다.

데이터 정제 단계에서는 수집한 데이터의 feature를 분석하고 이를 기반으로 데이터를 정제하는 과정을 가진다. 본 연구의 목표인 2022/23시즌 NBA 선수들의 연봉과 관련된 열을 우선적 가공하고, 시즌별 선수 기록 데이터셋에서 동일 선수에 대한 중복 행을 하나로 통합한다. 또한, 앞으로의 연구 과정에서 사용되지 않는 일부 행과 열은 제거한다. 특히 2020/21시즌이나 2021/22시즌 동안 경기를 전혀 뛰지 않았거나 계약을 맺지 않은 선수들의 경우, 2022/23시즌 연봉을 예측하는 데 있어서 도움이 되지 않기 때문에 결측값으로 판단하여 제거해준다. 데이터셋 내의 포지션은 5개(PG, SG, PF, SF, C)로 간소화한다.

데이터 정제를 마친 이후에는 향후 분석에 활용할 feature를 파악하기 위해서 데이터 탐색 단계를 진행한다. 목표 변수인 연봉과 관련된 주요 통계적 정보들을 요약하여 나타내고, histogram을 통해 NBA 선수들의 연봉과 주요 기록의 분포를 파악한다. 또한, 산포도를 그려 선수들의 주요 기록과 2022/23시즌 연봉 간의 관계를 확인한다. 경기당 득점, 어시스트, 스틸, 리바운드 등은 연봉과의 positive relationship이 존재했지만, eFG%, 나이, 포지션 등에서는 별다른 relationship을 확인할 수 없었다. 탐색적 데이터 분석은 회귀 분석과 시각화를 이용해서 실시하고, 다중 공선성을 고려하여 연봉을 결정하는 주요 요인들을 찾아낸다. 이때 변수 간의 관계에 대한 완벽하고 객관적인 개요를 얻기 위해서 히트맵을 활용하고, 분산 팽창 요인(VIF) 계수를 고려하여 다중 공선성 문제를 가지지 않는 feature를 선정한다.

이후 단계에서는 선정된 feature를 기반으로 하여 인공지능 모델 구현에 필요한 데이터를 준비한다. Feature 간의 단위를 맞추기 위해서 Standard Scaler를 활용하여 Feature Scaling을 진행하고, 범주형 feature에 one-hot encoding을 적용하여 벡터화한다. 모델링 단계에서 사용할 target variable과 features를 각각 지정하고, 데이터셋을 학습 데이터와 테스트 데이터로 분리한다. Target variable인 선수들의 연봉이 우편향 분포(right-skewed distribution)를 따르고 있으므로 인공지능 예측 모델을 구현하기 위한 정규화를 진행한다.

데이터가 준비되고 나면 기계 학습에서 사용되는 지도학습 방법과 결합하여 NBA 선수들의 연봉을 예측하는 인공지능 모델을 구현한다. 본 연구에서는 Multiple Linear Regression, Lasso Regression, Ridge Regression, XGBoost, LightGBM 등을 사용하여 다양한 예측 모델을 구현했으며, 사용되는 기계 학습 알고리즘의 특징과 차이점을 파악하여 향상된 예측 성능을 가지는 모델을 만들고자 했다.

모델링이 완료되면 모델 분석 및 비교 단계를 통해 각 모델의 성능을 확인한다. 다양한 인공지능 모델 간 비교를 위해서 RMSE(Root-Mean-Squared Error) Score와 R-Squared score를 평가지표로 사용한다. 또한, Cross Validation을 사용하여 정교한 정확도 측정을 제공한다.

결과 해석 단계에서는 가장 정확한 예측도를 보인 모델을 선택하여 Feature importance를 측정한다. 이렇게 측정한 Feature importance를 기반으로 다른 feature와 유사하거나 중복되는 일부 feature를 제외하고 높은 중요도를 가진 feature를 선정한다. 본 연구에서는 XGB-Regressor를 사용한 모델이 가장 높은 정확도 점수를 얻었기 때문에 이를 기준으로 feature를 재선정하여 최종 모델을 구현했다. 테스트 데이터에 대한 최종 모델의 성능은 Root Mean Squared Error: 3.4341, R-squared: 0.7604로 기존보다 향상되었음을 알 수 있다. 최종적으로 시각화 자료를 통해 예측 연봉과 실제 연봉의 차이를 직관적으로 확인하고, NBA 선수의 연봉 예측에 이상적인 데이터셋과 학습 모델을 파악한다.

■ 관련 연구

1. 데이터 마이닝 기법을 이용한 한국 프로농구 경기 분석[4] (김세중, 2010)

해당 논문은 농구 경기의 승패와 관련된 요인과 패턴을 조사하기 위한 목적으로 데이터 마이닝 기법을 이용해 진행되었다. 연구를 위해서 한국프로농구연맹 사이트(www.kbl.or.kr)를 통해 2005~2009년 동안의 KBL(Korea Basketball League) 경기 기록을 수집하였고, 이를 모델개발 및 판별에 활용했다. 선택된 데이터를 분석하기 위해서 CART 알고리즘이 사용되었으며, 10개의 인자와 15개의 패턴으로 구성된 CART 알고리즘에 의해 의사결정 나무 모델(Decision Tree Model)이 생성되었다. 의사결정나무의 분석에는 SPSS Clementine이 활용되었고, 변수 선택은 결측치가 존재하는 변수와 변화량이 지나치게 작거나 큰 변수를 제거하는 동시에 변수들을 중요도에 따라 순위화하여 진행되었다. 연구 결과, 승리에 대한 적중률은 70.5%, 패배에 대한 적중률은 82.5%로 나타났다. 승리와 관련된 주요 요인과 패턴에는 수비 리바운드, 2점슛, 3점슛, 2점슛 시도 횟수, 3점슛 시도 횟수 요인에 의한 패턴, 수비 리바운드, 2점슛 요인에 의한 패턴 그리고 수비 리바운드, 어시스트, 2점슛, 블록슛 요인에 의한 패턴이 있었으며, 패배와 관련된 주요 요인과 패턴에는 수비 리바운드, 어시스트, 스틸, 턴오버 요인과 관련된 패턴 그리고 수비 리바운드, 어시스트, 스틸, 턴오버, 자유투 요인과 관련된 패턴이 있었다.

해당 논문의 연구 결과는 농구 경기와 관련된 요인과 패턴을 구체적으로 파악할 수 있고, 이러한 요소가 경기 분석에 도움이 된다는 것을 나타낸다. 앞서 언급한 바와 같이 프로스포츠 선수들의 연봉은 선수들의 개인 성적과 팀 승리에 대한 기여 등에 영향을 받기 때문에 경기의 승패 여부는 선수들의 연봉을 결정하는 데 있어서 중요한 요소 중 하나로 판단할 수 있다. 그러므로 해당 논문에서 발견한 주요 요인과 패턴들을 본 연구에서도 적극적으로 활용했다. NBA 선수의 시즌별 기록 요인들을 살펴보는 과정에서 수비 리바운드, 2점슛, 3점슛, 어시스트, 스틸, 턴오버, 자유투 등의 요인을 우선적으로 확인하였으며, 해당 요인에 대해 histogram을 출력하여 각각의 분포를 파악하고 산포도를 통해 목표 예측값인 2022/23시즌 연봉과의 관계를 알아보았다. 또한, 본 연구의 주요 요인을 선택하는 과정에서 논문의 CART algorithm을 통해 생성된 최종 의사결정 나무 모형을 활용한 방식과 기준을 참고해 진행하였다.

2. 한국 프로스포츠 선수들의 연봉에 대한 다변량적 분석[9] (송종우, 2008)

해당 논문에서는 프로농구와 프로야구 선수들의 개인 성적을 기반으로 연봉을 예측 분석하였다. Data visualization 기법을 통해 변수 사이의 관계, 이상점 발견, 모형진단 등을 수행하고, 다중선형회귀 모형(Multiple Linear Regression Model)과 트리 모형(Regression Tree Model)을 이용하여 자료를 분석하고 모델 간 비교를 진행했다. 이후 최적 모델을 선택하기 위해 Cross-Validation 기법을 이용했다. 특히 자동으로 변수를 선택해주는 stepwise regression 방법을 단순하게 사용하지 않고 설명 변수들 사이의 관계, 설명 변수와 반응 변수 사이의 관계를 먼저 조사하고, 이를 기반으로 선택된 변수를 가지고 stepwise regression과 regression tree 방법론을 이용하여 적절한 변수 및 최종 모형을 선택했다. 분석 결과를 통해 프로농구에서는 경기당 득점, 어시스트, 자유투 성공 수, 경력 등이 중요한 변수였음을 알아냈고, 프로야구 투수의 경우에는 경력, 9이닝 당 삼진 수, 방어율, 피홈런 수 등이, 타자의 경우에는 경력, 안타 수, FA 유무 여부 등이 중요한 변수였음을 알아냈다.

본 연구와 해당 논문 모두 프로스포츠 선수들의 연봉이 선수들의 개인 성적과 팀에 대한 기여로부터 결정된다는 전제에서 진행되었기 때문에 연구 과정의 많은 부분이 본 연구에서 활용되었다. 해당 연구에서 진행된 바와 같이 본 연구에서도 변수들 사이의 산포도를 그려서 변수 사이의 관계를 조사했으며, 모델 학습에 Cross-Validation 방법을 활용하였다. 데이터를 training set과 test set으로 나누고 난 뒤 모형 적합에 training set만을, 평가에 test set만을 사용하는 대신 Cross-Validation 방법을 이용하면서 구현 모델이 test set에 과적합 되는 문제를 방지하고 예측 정확도를 높이는 효과를 얻을 수 있었다. 또한, 해당 논문에서 프로농구 선수들의 연봉을 예측하는데 시즌 동안 한 번도 출장하지 않은 선수와 시즌 도중 군대에 가거나 은퇴한 선수를 모두 제외한 점을 참조하여 본 연구에서도 2020/21시즌이나 2021/22시즌 동안 경기를 전혀 뛰지 않았거나 계약을 맺지 않은 선수들을 결측값으로 판단하고 제거해주었다. 이 밖에도 해당 논문에서 연봉을 결정하는 중요 변수들을 선정하는 과정을 참고하여 본 연구에서도 유사하게 진행하였다. 다만, 해당 논문에서는 다중선형회귀 모형과 트리 모형만을 이용하여 모델을 형성하였으나 본 연구에서는 Lasso Regression, Ridge Regression, XGBoost, LightGBM 등을 추가로 활용하면서 다양한 모델을 제공했고, 히트맵과 그래프 등의 시각 자료를 적극적으로 활용하여 결과를 직관적으로 확인할 수 있도록 했다.

3. 빅데이터를 활용한 인공지능 주식 예측 분석[11] (최훈, 2021)

과거 주식 시장에서는 사람들의 기업 분석 및 투자기법을 통한 노동 집약적인 투자가 이루어졌다면 최근 들어서는 인공지능 및 데이터를 활용한 주식 투자가 널리 이용되고 있다. 그러나 현재까지 인공지능을 통한 주식 예측 성공률은 그리 높지 않아 다양한 인공지능 모델을 활용하여 주식 예측률을 높이려는 시도가 계속되고 있다. 해당 논문에서는 인공지능을 활용한 다양한 주가 예측 모델에 대해 살펴보고 각 모델의 장단점 및 예측률을 파악하고 있다. 이를 위해, 주가 예측 인공지능 프로그램으로 인공신경망(Artificial Neural Network, ANN), 심층 학습 또는 딥 러닝(Deep Structured Learning, DNN), K-근접 이웃 알고리즘(K-Nearest Neighbor, K-NN), 합성곱 신경망(Convolutional Neural Network, CNN), 순환 신경망(Recurrent Neural Network, RNN), LSTM(Long Short Term Memory) 등이 연구에 사용되었다. 이렇게 선택된 인공지능 알고리즘 모델에 대해 예측에 사용되는 주가 속성과 모델의 장단점 및 예측 정확도를 측정하였고, 해당 정보를 사용자에게 제공하였다. 가장 높은 주식 예측률을 보인 인공지능 모형은 LSTM으로 84.9%의 정확도를 보였으며, 단방향 LSTM 순환신경망을 이용했을 때보다 양방향 LSTM 순환신경망을 이용했을 때 더 좋은 성능을 나타냈다.

이처럼 해당 논문에서는 주식 시장을 예측하고 분석하는 데 있어서 하나의 인공지능 모형만을 사용하지 않고 인공신경망(ANN), 딥 러닝(DNN), k-최근접 이웃 알고리즘(k-NN), 합성곱 신경망(CNN), 순환 신경망(RNN), LSTM 등 다양한 인공지능 알고리즘을 사용하여 모델을 구현했다. 그러면서 각 모델이 가지는 장단점을 구체적으로 제시하였고 사용자에게 더 넓은 선택의 폭을 제공했다. 이러한 점을 참고하여 본 연구에서도 다양한 인공지능 알고리즘을 활용해서 예측 모델을 구현하고자 했다. 본 연구에서는 Multiple Linear Regression, Lasso Regression, Ridge Regression과 같이 기계 학습에서 사용되는 기본적인 알고리즘에 더하여 boosting 앙상블을 이용하는 XGBoost, LightGBM를 추가로 사용하였고, 연구에서 사용되는 모든 기계 학습 알고리즘의 특징과 차이점을 정확히 파악한 뒤 이를 기반으로 모델이 좋은 성능을 낼 수 있도록 데이터를 가공하고 준비하였다. 또한 구현 모델 중 가장 정확한 예측 성능을 보인 모델을 선택한 뒤 Feature importance를 측정하고, 이를 기반으로 feature를 재선정하여 최종 모델을 구현하면서 예측의 정확도를 높이고자 했다.

4. 인공지능 기반 수요예측 기법의 리뷰[12] (정혜린, 2019)

최근 다양한 분야에서 빅데이터가 생성됨에 따라, 많은 기업에서 빅데이터 분석이 가능한 인공지능(AI) 기반의 시스템을 구축하여 이익 창출을 시도하고 있다. 특히 재무, 조달, 생산 및 마케팅과 같은 분야의 국가 및 기업 경영 관리에 있어서 최소의 오차와 최대의 정확도를 갖춘 수요예측은 절대적으로 중요한 요소이다. 이러한 요소를 이루기 위해서는 각 분야의 수요패턴을 고려한 적절한 모델을 필수적으로 적용해야 한다. 전통적으로 사용되는 autoregressive(AR), moving average(MA), auto regressive integrated moving average(ARIMA), exponential smoothing 등의 시계열모델과 회귀모델로도 실제 데이터의 복잡하고 비선형적인 패턴을 분석할 수 있기는 하지만, 다양한 비선형 모델 중 적절한 모델을 선택하는 것은 사전 지식 없이는 너무나도 어려운 일이다. 머신러닝과 딥러닝 기법은 이러한 문제를 어느 정도 해결하고, 정형 데이터와 이미지나 텍스트의 비정형 데이터 분석을 통한 수요예측에서 높은 정확도를 보인다. 따라서 해당 논문에서는 수요예측이 비교적 활발하게 일어나는 중요한 분야를 수/전력, 금융, Ride-services, 관광, 단기수요, 예비부품 및 신제품 분야로 나누어 설명하고, 각 분야의 특징적인 성격을 고려하여 인공지능 기반의 수요예측에 사용된 머신러닝과 딥러닝 기법(SVM, Boosting, Gaussian process, RBM, RNN)을 소개하고 있다.

본 연구는 NBA 선수의 연봉 예측에 초점을 두고 연구를 진행했지만, 해당 논문에서는 수요예측이 이루어지고 있는 전반적인 분야를 소개하고 인공지능 기법에 기반한 다양한 기법을 제시하고 있다. 또한, 인공지능이 활용된 분야와 사용된 인공지능 기법을 단순하게 정리하는 것에 그치지 않고 이론적인 설명을 첨가하여 논문 저자의 이해를 돕고 있다. 해당 논문의 연구 결과는 예측 모델을 구현하는 데 있어서 인공지능을 사용하는 것이 기존의 시계열모델과 회귀모델이 가지는 한계를 극복하고 개선된 성능을 제공한다는 것을 증명한다. 그러므로 본 연구에서도 모델의 정확한 예측 성능을 위해서 다양한 인공지능 기법을 활용하여 모델링을 진행했으며, 연구에 사용된 기계 학습 기법(Multiple Linear Regression, Ridge Regression, Lasso Regression, XGBoost, LightGBM)을 단순하게 나열하고 사용하는 대신 각각의 특징과 장단점 및 운영상의 차이점을 추가로 기재하면서 본 연구의 독자가 연구 내용을 이해할 수 있도록 돕고, 보고서의 가독성을 향상시켰다.

5. NBA Game Result Prediction Using Feature Analysis and Machine Learning[1] (Fadi Thabtah, 2019)

해당 논문은 NBA 경기 결과에 영향을 미치는 요인들을 발견하는 것을 목표로 NBA 경기 결과를 예측하는 새로운 기계 학습 프레임워크를 제안하고 있다. 이를 위해서 과거의 데이터를 통한 기계 학습 방법이 NBA 경기 결과를 예측하는 데 활용될 수 있는지 알아보고, 경기에 영향을 미치는 주요 요인이 무엇인지 확인하였다. 목표 달성을 위한 모델을 구현하기 위해서 나이브 베이즈(Naïve Bayes), 인공신경망(Artificial Neural Network) 및 의사결정나무(Decision Tree) 등이 사용되었고, 농구의 다양한 요인들에 따른 모델의 성능을 비교하는 과정을 통해 예측 모델의 정확성이나 효율성과 같은 성능 향상에 기여하는 주요 요인들을 발견했다. 연구 결과, NBA 경기 결과에 영향을 미치는 가장 중요한 요인은 DRB(Defensive Rebound)였으며, TPP(Three-Point-Percentage), FT(Free Throws made), TRB(Total Rebound) 등도 주요 요인으로 선택되었다. 해당 요인들로 만들어진 모델은 기존보다 2~4% 증가한 예측 정확도를 보였다.

본 연구를 진행하는 데 있어서 모델별로 데이터셋에 따라 Accuracy, Precision, Recall, F1 등을 기준으로 성능을 평가하는 점을 참고했다. 다만, 해당 논문에서는 주요 요인들을 결정하는 과정에서 각 요인의 t-value를 측정하여 사용했지만, 본 연구에서는 p-value를 사용하였다. 본 연구에서는 회귀식이 활용되며, 회귀식의 유의성 검정을 위해서는 F 통계량(F-statistic)과 P-value인 Prob(F-statistic) 수치를 함께 살펴봐야 하기 때문에 이러한 변화를 선택하게 되었다.

6. Predicting NBA Games Using Neural Networks[13] (Bernard Loeffelholz, 2009)

해당 논문에서는 신경망(Neural Networks)을 사용하여 NBA 농구팀의 성적을 예측했다. 이를 위해 NBA의 620경기에 대한 통계가 수집되었고, feed forward, radial basis, probabilistic, generalized regression neural networks 등이 신경망을 훈련시키기 위해 사용되었다. 신경망 융합에는 Bayes belief networks와 Probabilistic neural network fusion을 활용하였다. 또한, 가장 효과적인 예측을 보여주는 요인들을 신경망에 입력하기 위해서 Signal-to-noise 비율로부터 얻은 집합과 전문가의 의견을 참고해 요인들을 식별했다. 네트워크로부터 얻은 결과를 농구 분야 전문가들의 예측과 비교하였고, 최고 성능을 가지는 네트워크는 74.33%의 정확도를 보이며 68.67%의 정확성을 가진 전문가 예측에 비해 우승팀을 정확하게 예측하였다.

해당 논문의 NBA 경기와 관련된 통계 자료를 수집하고, 특징들을 분류하는 방식을 참고해서 본 연구의 데이터 준비 단계에 활용하였다. 다만, NBA 농구팀의 성적을 예측하는 해당 논문과 다르게 본 연구의 목표는 NBA 선수의 연봉을 정확하게 예측하는 것이기 때문에 팀 기록이 아닌 선수의 기록과 관련된 통계 자료를 수집하였으며, 데이터 내에 더 많은 feature가 포함되도록 했다. 또한 이렇게 준비된 데이터의 feature 간 단위를 맞추기 위해서 standard scaler를 활용했고, 해당 논문과 마찬가지로 범주형 feature를 다루기 위한 one-hot encoding을 적용하였다.

7. Current Approaches to the Use of Artificial Intelligence for Injury Risk Assessment and Performance Prediction in Team Sports[14] (João Gustavo Claudino, 2019)

해당 논문은 스포츠 성능과 부상 위험을 조사하기 위해 어떤 인공지능 기법이 적용되었는지 파악하고, 각 스포츠 분야에서 사용되어 온 인공지능 기법을 알아보기 위해서 진행되었다. 이를 위해, 가장 먼저 PubMed, Scopus, Web of Science 등의 온라인 데이터베이스를 통해 팀 스포츠 선수들을 대상으로 적용된 인공지능 기술과 방법을 찾아보는 과정을 가졌다. 검토 대상에는 58개의 연구가 포함되었으며, 연구 결과 12가지 팀 스포츠 분야에서 11개의 인공지능 기술과 방법이 적용되었다는 사실이 확인되었다. 표본 추출은 6,456명의 참가자로부터 구성되었고, 이들 중 76%가 프로 선수였다. 가장 많이 사용된 인공지능 기술은 인공신경망(Artificial Neural Network), 의사결정나무(Decision Tree Classifier), 서포트 벡터 머신(Support Vector Machine) 이었으며 마르코프 프로세스(Markov Process)의 성능이 가장 우수한 것으로 나타났다. 또한, 주로 축구, 농구, 핸드볼, 배구 분야에서 인공지능이 활발하게 적용되고 있음이 확인되었다.

해당 논문의 연구 결과는 팀 스포츠에서 인공지능이 널리 사용되고 있다는 것을 시사한다. 이러한 점에 근거하여 본 연구의 인공지능 모델 역시 일정 수준 이상의 예측 정확도를 가지게 된다면 실제 스포츠 분야에서 충분히 활용될 수 있을 거라고 판단했다. 이 밖에도 해당 논문을 통해 스포츠 분야의 인공지능 활용이 경기력 분석이나 결과 예측 등 스포츠적인 측면에 국한되지 않고 선수들의 부상 예측 등 다양한 측면에서 활용되고 있음을 알게 되었고, 연구 주제를 선정하는 과정에서 더 넓은 선택의 폭을 가질 수 있었다.

■ 제안 작품 소개

1. 데이터 수집

NBA의 다양한 통계 정보를 제공하고 있는 Basketball Reference로부터 웹 크롤링을 하여 필요한 데이터들을 얻는다. 데이터셋은 연봉 관련 데이터와 기록 관련 데이터로 분리하여 가져온 뒤 추후 병합한다. 선수의 연봉 관련 데이터는 2022/23시즌 이후의 계약 정보를 선수 이름, 팀 이름, 연봉 등의 feature가 포함되도록 하여 가져온다. 만약 미계약 시즌이 있는 선수가 존재할 경우, 해당 부분을 공란으로 처리한다. 선수의 기록 관련 데이터는 2019/20, 2020/21, 2021/22시즌에 관련된 정보를 차례로 가져온다. 이때 feature에는 득점, 어시스트, 리바운드, 스틸 등의 1차 기록에 더해, 이를 가공한 PER, WS, BPM 등의 2차 기록을 포함하여 해당 선수와 관련된 다양한 통계 자료가 모델에 반영될 수 있도록 한다.

2. 데이터 정제

탐색적 데이터 분석(Exploratory Data Analysis)을 진행하기에 앞서 수집한 데이터의 feature를 분석하고 이를 기반으로 데이터를 정제하는 과정을 가진다. 제일 먼저 데이터셋으로 지정된 파일들(20/21시즌 선수별 기록 데이터셋, 21/22시즌 선수별 기록 데이터셋, 22/23시즌 선수별 연봉 데이터셋)의 최초 다섯 행을 차례로 살펴보면서 데이터 내에 존재하는 feature의 종류를 확인하고, 가공할 열을 선택한다.

본 연구의 목표는 2022/23시즌 NBA 선수들의 연봉을 예측하는 것이므로 해당 시즌의 연봉과 관련된 열을 우선적 가공한다. 해당 열의 이름을 '2022-23: Salary 22/23'로 수정하고, 연봉을 천의 단위로 나타낸다. 또한, 해당 목표를 구하는 데 있어서 선수가 어느 팀에서 뛰었는지는 중요한 요소가 아니기 때문에 시즌별 선수 기록 데이터셋에서 동일 선수에 대한 중복 행을 하나로 통합한다. 소속 팀, 2023년 이후의 연봉 정보, 계약 방식 등 앞으로 있을 연구 과정에서 사용되지 않는 일부 열들은 제거한다. 각 열 마다 존재하는 결측값을 확인하고, 2020/21시즌과 2021/22시즌에 대하여 선수 기록이나 연봉 정보가 전혀 없는 행들을 제거한다. 2020/21시즌이나 2021/22시즌 동안 경기를 전혀 뛰지 않았거나 계약을 맺지 않은 선수들은 목표치인 2022/23시즌 연봉을 예측하는 데 있어서 도움이 되지 않기 때문에 결측값으로 판단하여 제거하는 편이 합리적이다. 데이터셋 내에 존재하는 포지션은 5개의 포지션(PG, SG, PF, SF, C)으로 간소화하여 나타낸다.

3. 데이터 탐색

데이터 정제를 마친 이후에는 향후 분석에 활용할 features를 파악하기 위해서 데이터 탐색을 진행한다. 제일 먼저 목표 변수인 연봉과 관련된 주요 통계적 정보들(count, mean, standard variation, min, max)을 요약하여 나타내고, histogram을 통해 연봉의 분포를 확인한다. 득점, 어시스트, 스틸과 같은 NBA 선수 기록의 주요 features 역시 histogram을 출력하여 분포를 파악하고, 산포도를 그려 2022/23시즌 연봉과의 관계를 확인한다.

탐색적 데이터 분석은 회귀 분석(regression analysis)과 시각화(visualization)를 통해 진행한다. 회귀 분석을 통해 각 feature에 대한 P-Value를 확인하고 모델링 하는 데 있어 유의미한 feature인지 판단한다. 일반적으로 P-value가 0.05 이하면 'F 통계량이 유의한 의미를 가진다'라는 결론을 내릴 수 있으며, 이는 회귀 분석이 유의미한 결과를 가진다는 뜻과 같다. 또한, 히트맵(heat map) 등을 활용하여 시각화를 진행하고, feature 간의 상관관계를 확인하여 연봉을 결정하는 주요 요인들을 찾아낸다. 이러한 과정을 통해 객관적인 분석을 진행하고 변수 간의 관계에 대해 보다 완벽한 개요를 얻을 수 있다.

회귀 분석은 feature 간의 독립성을 전제로 하는 분석 방법이기 때문에 예측의 성능을 높이기 위해서 다중 공선성을 추가로 고려한다. 다중 공선성은 분산 팽창 요인(VIF) 계수로 평가할 수 있으며, 보편적으로 VIF 계수가 10~15 정도를 넘으면 다중 공선성 문제가 발생했다고 판단한다. 해당 기준을 기반으로 다음의 선정 과정을 거쳐 다중 공선성 문제가 나타나지 않는 적절한 feature를 선정한다.

[선정 과정]

1. VIF 계수가 높은 feature를 제거한다. 단, (FG, FG%)와 같이 유사한 features 중에서는 하나만을 제거한다.
2. 다중 공선성을 재검증한다. VIF 계수가 비정상적으로 높은 feature를 제거했을 때, 다른 features의 다중 공선성이 감소하는 것을 확인할 수 있다.
3. 여전히 VIF 계수가 높은 features를 제거한다.

데이터 탐색 단계를 통해서 features 간의 연관성을 고려하고 모델 학습에 사용할 features의 선택 기준을 선정한다.

4. 데이터 준비

인공지능 모델의 구현에 필요한 데이터를 준비한다. Feature 사이의 단위를 맞추기 위해서 Standard Scaler를 활용하여 Feature Scaling을 진행하고, 연속형이 아닌 범주형 feature에 대해서 one-hot encoding을 적용하여 벡터화한다. 이후 모델링 단계에서 사용할 target variable과 features를 각각 지정하고, 데이터셋을 학습 데이터와 테스트 데이터로 분리한다. Target variable인 선수들의 연봉이 정규 분포(normal distribution)가 아닌 우편향 분포(right-skewed distribution) 모양을 따르기 때문에 인공지능 모델을 구현하기 위한 정규화를 진행한다.

5. 모델링

탐색적 데이터 분석을 통해 연구에 적합한 features가 선택되면, 기계 학습에서 사용되는 지도학습 방법과 결합하여 NBA 선수들의 2022/23시즌 연봉을 예측하는 인공지능 모델을 구현한다. 이때 다양한 모델을 구현하기 위해서 Linear Regression, Lasso Regression, Ridge Regression과 같이 기계 학습에서 사용되는 기본적인 알고리즘에 더하여 boosting 앙상블을 이용하는 XGBoost, LightGBM 등을 추가로 활용해 모델링을 진행한다.

다중 선형 회귀(Multiple Linear Regression)는 여러 개의 독립 변수와 하나의 종속 변수 간의 선형 관계를 모델링 하는 것으로 각각의 독립 변수가 종속 변수와 선형 관계를 이루고, 독립 변수 간에 높은 수준의 상관관계가 존재하지 않으면서, 잔차(residual)가 정규 분포를 이룰 때 좋은 성능을 보인다. 그러나 다중 선형 회귀 모델은 다중 공선성 문제를 가질 수 있다. Ridge Regression은 람다 값을 조절해 추정에 개선된 효율성을 제공하고 다중 공선성 문제를 완화한다. Lasso Regression은 Ridge Regression과 개념적으로 매우 유사한 방법이지만, 0이 아닌 계수에 대해 penalty를 추가로 부여하고 계수의 제곱합이 아닌 계수의 절대값으로 제한하는 점에서 차이가 있다.

XGBoost는 여러 개의 의사 결정 나무(Decision Tree)를 boosting 앙상블로 구현한 모델이다. Boosting 방식은 한 개의 예측 모델에 대한 error를 줄이는 방식의 앙상블 기법으로, XGBoost는 병렬 학습이 지원되기 때문에 빠른 학습이 가능하다는 장점이 있다. LightGBM 역시 XGBoost와 마찬가지로 GBM 모델을 기반으로 만들어진 모델이다. 다만, XGBoost와 다르게 학습 방법에서 트리를 생성할 때 Level-wise

방법으로 생성하지 않고 Leaf-wise 방법으로 트리를 만들어 나가면서 메모리를 적게 사용하고, 모델이 빠르게 생성되며, 다른 부스팅 방법의 알고리즘에 비해 높은 성능을 보인다. 그렇지만 데이터 수가 적으면, 과적합이 발생할 가능성이 존재한다.

연구에 사용되는 모든 알고리즘의 특징과 차이점을 정확히 파악해야지만 좋은 예측 성능을 가지는 모델을 구현할 수 있다. 그러므로 본 연구에서는 사용되는 기계학습 알고리즘(Multiple Linear Regression, Ridge Regression, Lasso Regression, XGBoost, LightGBM)에 대한 정확한 이해를 기반으로 모델이 좋은 성능을 낼 수 있도록 데이터를 가공하고 준비했다. 이렇게 준비된 데이터를 기반으로 모델링이 완료되면 각 모델의 예측값을 확인한다.

6. 모델 분석 및 비교

다양한 인공지능 모델에 대한 분석 및 비교를 위해 RMSE(Root-Mean-Squared Error) Score와 R-Squared score를 평가지표로 사용한다. 실제값과 예측값의 차이를 절대적인 수치로 나타낸 RSME Score의 경우, 그 값이 클수록 모델의 예측이 부정확하다는 것을 의미한다. R-Squared score는 회귀 분석으로 추정한 모델이 주어진 데이터를 얼마나 잘 설명하는지 나타내는 점수로, 데이터를 잘 설명하는 모델일수록 1에 가까운 점수를 가진다. 따라서 해당 지표들을 이용하여 구현한 모델에 대한 점수를 각각 확인하면, 다양한 모델의 성능을 비교해 볼 수 있다. 또한, Cross Validation을 통해 기존보다 정교한 정확도 측정을 제공한다.

7. 결과 해석

구현한 모델 중 가장 정확한 예측도를 보인 모델을 선택한 뒤, Feature importance를 측정한다. 이렇게 측정한 Feature importance를 기반으로 다른 feature와 유사하거나 중복되는 일부 feature를 제외하고 높은 중요도를 가진 features를 선정한다. 그리고 선택된 features를 사용하여 최종 모델을 구현한다. 최종 모델이 테스트 데이터에 대해 어떻게 동작하는지 확인한 이후 예측 연봉과 실제 연봉 데이터를 비교하는 시각화 자료를 만든다. 해당 자료를 통해서 선택된 학습 모델이 얼마나 정확한 예측을 하였는지 직관적으로 확인할 수 있으며, NBA 선수의 연봉 예측에 있어 이상적인 데이터셋과 학습 모델에 대한 파악이 가능하다.

■ 구현 및 결과분석

Basketball Reference에서 수집한 NBA 선수의 연봉 관련 데이터와 기록 관련 데이터를 대상으로 feature를 분석하고 이를 기반으로 데이터를 정제한다. 이를 위해서 데이터셋으로 지정된 파일들의 최초 다섯 행을 살펴보면서 데이터 내에 존재하는 feature의 종류를 확인하고, 가공할 열을 선택한다.

	Rk	Player	Tm	Salary 22/23	2023-24	2024-25	2025-26	2026-27	2027-28	Signed Using	Guaranteed
0	1	Stephen Curry	GSW	48070.014	51915615.0	55761216.0	59606817.0	NaN	NaN	Bird	$215,353,662
1	2	Russell Westbrook	LAL	47063.478	NaN	NaN	NaN	NaN	NaN	Bird Rights	$47,063,478
2	3	LeBron James	LAL	44474.988	NaN	NaN	NaN	NaN	NaN	Bird	$44,474,988
3	4	Kevin Durant	BRK	44119.845	47649433.0	51179021.0	54708609.0	NaN	NaN	Bird	$197,656,908
4	5	Bradley Beal	WAS	43279.250	46741590.0	50203930.0	53666270.0	57128610.0	NaN	Bird	$193,891,040

그림 1: NBA 선수의 연봉 관련 데이터셋

	Rk	Player	Pos	Age 21/22	Tm 21/22	G 21/22	GS 21/22	MP 21/22	FG 21/22	...	USG% 21/22	OWS 21/22	DWS 21/22	WS 21/22	WS/48 21/22	OBPM 21/22	DBPM 21/22	BPM 21/22	VORP 21/22
0	1	Precious Achiuwa	C	22	TOR	73	28	1725	265	...	18.5	0.4	2.1	2.5	0.070	-2.0	-0.6	-2.6	-0.2
1	2	Steven Adams	C	28	MEM	76	75	1999	210	...	12.0	3.8	3.0	6.8	0.163	1.0	1.0	2.0	2.0
2	3	Bam Adebayo	C	24	MIA	56	56	1825	406	...	25.0	3.6	3.5	7.2	0.188	1.7	2.1	3.8	2.7
3	4	Santi Aldama	PF	21	MEM	32	0	360	53	...	18.4	-0.1	0.4	0.3	0.044	-4.2	-1.5	-5.7	-0.3
4	5	LaMarcus Aldridge	C	36	BRK	47	12	1050	252	...	22.4	2.1	1.0	3.1	0.141	1.3	-0.6	0.7	0.7

그림 2: NBA 선수의 기록 관련 데이터셋

우선 본 연구의 목표 변수인 2022/23시즌 NBA 선수들의 연봉과 관련된 열을 가공하고, 시즌별 선수 기록 데이터셋에서 동일 선수에 대한 중복 행을 하나로 통합한다. 향후 연구 과정에서 사용되지 않는 일부 열들은 제거한다. 또한, 2020/21시즌이나 2021/22시즌 동안 경기를 전혀 뛰지 않았거나 계약을 맺지 않은 선수들은 결측값으로 판단하여 해당 행을 모두 제거해준다. 데이터셋 내에 존재하는 포지션은 5개의 포지션(PG, SG, PF, SF, C)으로 간소화하여 나타낸다.

	level_0	index	Player	Pos	Age 21/22	Tm 21/22	G 21/22	GS 21/22	MP 21/22	...	OWS 21/22	DWS 21/22	WS 21/22	WS/48 21/22	OBPM 21/22	DBPM 21/22	BPM 21/22	VORP 21/22	Salary 22/23
0	0	0	Precious Achiuwa	C	-0.858388	TOR	0.921005	-0.039430	0.547006	...	-0.637248	0.747301	-0.165432	-0.446067	-0.597195	-0.424855	-0.666235	-0.750544	2840.160
1	1	2	Steven Adams	C	0.569737	MEM	1.073615	1.699017	0.921009	...	1.240203	1.665465	1.518277	1.032027	0.595834	0.728297	0.796885	0.963516	17926.829
2	2	3	Bam Adebayo	C	-0.382346	MIA	0.056218	0.996241	0.683503	...	1.129765	2.175555	1.674901	1.429365	0.874207	1.521090	1.369410	1.508899	30351.780
3	3	6	Nickeil Alexander-Walker	SG	-0.620367	TOT	0.514047	-0.298348	0.193476	...	-1.465535	-0.272880	-1.105176	-1.510930	-0.517660	-0.785215	-0.761656	-0.828456	5009.633
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
405	405	696	Lindell Wigginton	PG	-0.620367	MIL	-1.825966	-1.075101	-1.535950	...	-0.858124	-1.293061	-1.066020	-0.859297	-1.074406	-1.361791	-1.429602	-0.750544	1637.966
406	406	698	Zlaire Williams	SF	-1.334429	MEM	0.361437	0.071535	0.029679	...	-0.250714	-0.272880	-0.282900	-0.287132	-0.756265	-0.424855	-0.793463	-0.828456	4591.680
407	407	699	McKinley Wright IV	PG	-0.620367	MIN	-2.538144	-1.075101	-1.781646	...	-0.858124	-1.395079	-1.144332	-0.096410	-0.358589	-2.010439	-1.175146	-0.594721	1637.966
408	408	702	Omer Yurtseven	C	-0.620367	MIA	0.056218	-0.631242	-0.843907	...	-0.416371	0.033175	-0.322056	0.745945	-0.358589	0.295865	-0.157324	-0.438897	1752.638

그림 3: 데이터 가공 이후 데이터셋

데이터 정제 단계 이후, 분석에 사용할 요인을 선택하기 위한 데이터 탐색을 진행한다. 제일 먼저 본 연구의 목표 변수인 2022/23 시즌 NBA 선수들의 연봉과 관련된 주요 통계적 정보들을 천의 단위로 살펴보았고, 다음의 결과를 얻었다.

표 1: NBA 선수연봉 관련 주요 통계 정보

Count	409.000000
Mean	9993.430090
Std	10628.437409
Min	50.000000
25%	2000.000000
50%	5739.840000
75%	13906.976000
Max	48070.014000

이러한 통계적 정보들을 기반으로 히스토그램을 그려 연봉의 분포를 확인한다.

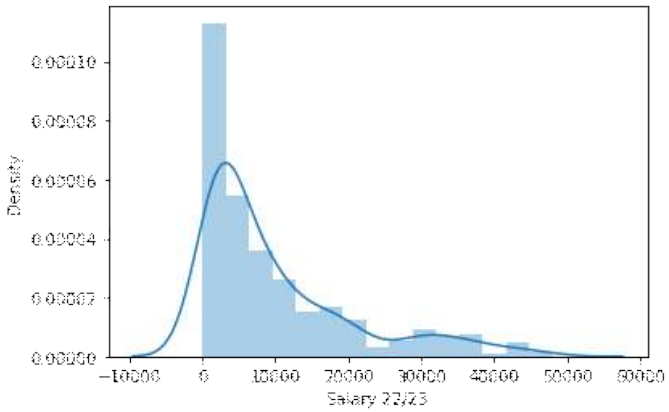

그림 4: 2022/23 시즌 NBA 선수연봉 히스토그램

해당 히스토그램을 통해서 다음을 확인할 수 있다.

1. 큰 표준편차는 연봉이 분산되어 있음을 나타낸다.
2. 선수의 연봉 히스토그램이 정규 분포(normal distribution)가 아닌 우측으로 치우친 분포(right-skewed distribution)인 점은 소수의 선수만이 고액 연봉을 받고 있음을 의미한다.

이 밖에도 NBA 선수들의 주요 기록(경기당 득점, 어시스트, 스틸, 리바운드)에 대해서도 히스토그램을 그려 각 지표의 분포를 알아보았고, 대부분 연봉과 마찬가지로 우편향 분포(right-skewed distribution)를 따르는 것을 확인했다.

히스토그램을 통해 연봉 및 주요 지표에 대한 분산을 확인한 이후에는 2022/23시즌 선수들의 연봉과 2021/22시즌 선수들의 주요 기록 간의 관계를 살펴보기 위해 산포도를 활용하여 연구를 진행한다.

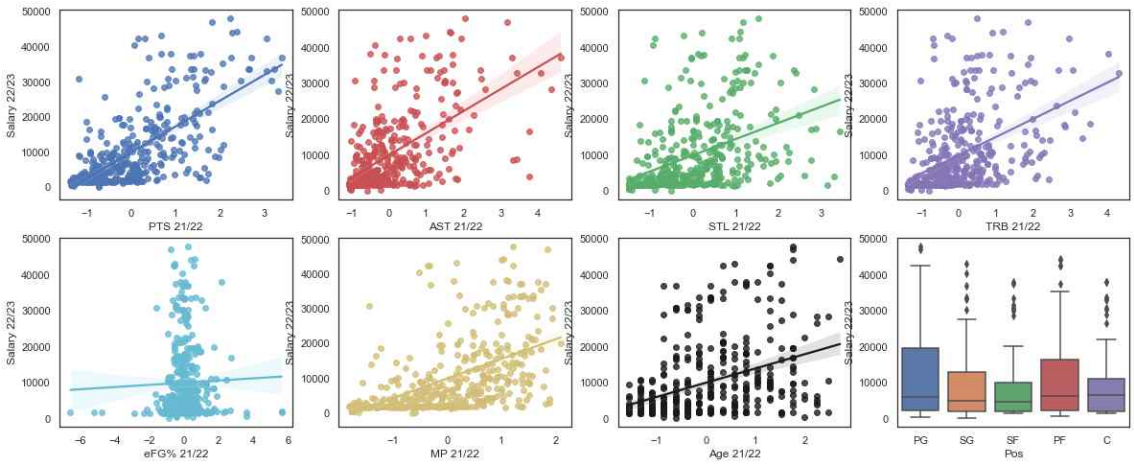

그림 5: 2022/23시즌 NBA 선수들의 연봉과 2021/22시즌 NBA 선수들의 주요 기록 간의 산포도 (득점, 어시스트, 스틸, 리바운드, eFG%, 경기당 플레이 시간, 나이, 포지션 순)

위의 산포도 그림을 통해서 경기당 득점(PTS), 어시스트(AST), 스틸(STL), 리바운드(TRB)와 연봉 사이에 positive relationship이 존재하는 것을 확인할 수 있다. 이 밖에도 농구에서는 2점 슛과 3점 슛 간의 차이가 존재하기 때문에 Normal Field Goal Percentage(FG%) 대신 Effective Field Goal Percentage(eFG%)를 선택하여 연봉과 비교해보았으나 별다른 relationship을 발견할 수는 없었다. 그림 2처럼 eFG%가 높으면서 연봉이 낮은 선수들은 있었지만, eFG%가 낮으면서 높은 연봉을 받는 선수는 존재하지 않았다. 경기당 플레이 시간(MP)과 연봉 사이에는 exponential relationship으로 보이는 positive relationship이 존재했다. 그러나 좋은 기록을 가지는 선수의 경우, 다음 시즌에 높은 연봉과 많은 플레이 시간을 동시에 보장받을 가능성이 높기 때문에 이러한 positive relationship이 좋은 기록으로부터 기인한 것으로 판단했다. 선수의 나이와 포지션은 연봉과 별다른 관계를 보이지 않았다.

지금까지의 분석은 일반적인 상식과 직관에 따라 연봉을 결정하는 데 있어 중요하다고 생각되는 요소에 기반을 두고 진행되었기 때문에, 좀 더 객관적인 분석을 진행하고 변수 간의 관계에 대해 완벽한 개요를 얻기 위해서 히트맵(heat map)을 활용하였다. 히트맵을 통해 시각적인 자료를 제시하고, feature 간의 상관관계를 확인하여 연봉을 결정하는 주요 요인들을 찾아냈다.

본 연구에 사용된 데이터 내 모든 feature에 대한 히트맵은 다음과 같다.

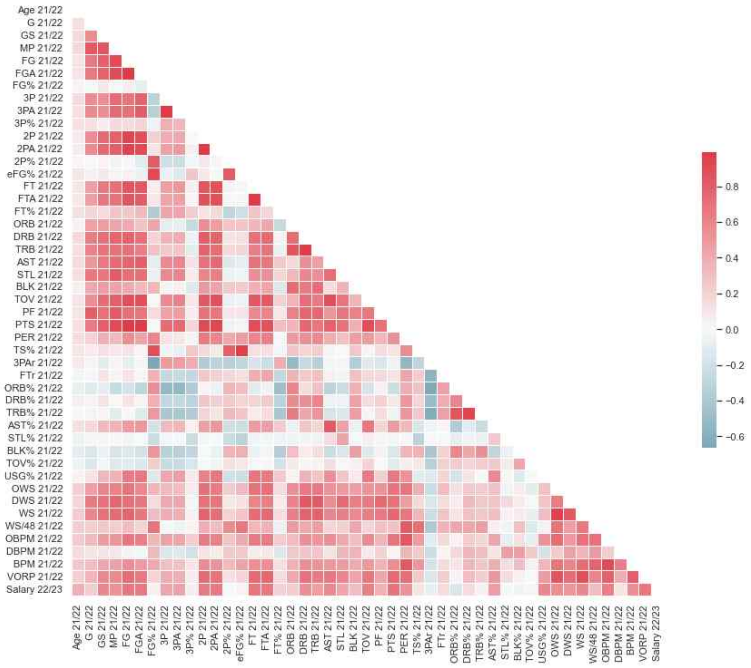

그림 6: 변수 간의 상관관계를 나타내는 히트맵

히트맵을 관찰했을 때 정사각형의 색깔이 붉을수록 높은 상관관계를 갖는 변수들로, 이는 예측 모델에 거의 동일한 정보를 제공하고 추정에 대한 분산을 증가시킨다. 회귀 분석은 특징 간의 독립성을 전제하므로 올바른 회귀 분석을 위해서는 이러한 쌍들을 제거해야 한다. 또한, 히트맵을 통해서 앞서 분석한 주요 통계 요소들 이외에 추가로 고려해야 하는 변수가 존재하는 것을 확인할 수 있다.

앞서 설명했듯이 다중 공선성은 분산팽창요인(VIF, Variance Inflation Factor)으로 평가할 수 있으며, 일반적으로 VIF 계수가 10~15 정도를 넘으면 다중 공선성 문제가 발생한 것으로 판단한다. 그러므로 본 연구에서는 해당 기준에 맞춰서 다중 공선성 문제가 나타나지 않는 적절한 features를 선정했다.

다중 공선성 문제가 나타나지 않도록 선정된 features는 다음과 같다.
['Age 21/22', 'G 21/22', 'GS 21/22', 'MP 21/22', 'FGA 21/22','3P 21/22', '2PA 21/22', 'FT 21/22', 'DRB 21/22', 'AST 21/22', 'STL 21/22', 'BLK 21/22', 'TOV 21/22', 'PF 21/22', 'PTS 21/22', 'PER 21/22', 'TS% 21/22', '3PAr 21/22', 'FTr 21/22', 'AST% 21/22', 'STL% 21/22', 'BLK% 21/22', 'TOV% 21/22', 'USG% 21/22','WS/48 21/22' ,'OBPM 21/22', 'VORP 21/22']

다만, feature 간의 관계를 분석할 때는 많은 주의가 필요하다. 예를 들어 연봉과 경기당 플레이 시간(MP, minutes played)이나 선발 출전 횟수(GS, games started)의 상관관계를 분석할 때는 단순하게 선발 출전을 많이 하는 선수가 더 많은 돈을 번다고 단정 지을 수 없다. 코치는 공격과 수비에 대한 통계적 지표를 참고하여 선수의 플레이 시간을 결정하고, 이에 따라 다음 시즌의 연봉이 결정되므로 둘 사이가 확실한 인과관계라고 단정 지을 수 없기 때문이다. 또한, 높은 상관관계만을 근거로 TOV(Turnover)가 많을수록 연봉이 높아진다고 말할 수 없다. 공격력과 수비력이 좋은 선수는 더 많은 경기를 뛰고, 더 높은 연봉을 보장받게 되므로 연봉이 높은 선수는 자연스럽게 플레이 시간이 많아지게 되고 평균적으로 턴오버가 많아지게 되기 때문이다. 이처럼 예측 모델 구현에 사용될 feature를 단순하게 선정하지 않고 다중공선성 문제, 변수 간의 관계 등을 다방면으로 고려하면서 예측의 정확성을 높이는 요소를 선별하고자 했다.

데이터 탐색 단계를 마친 이후에는 feature의 단위를 맞추기 위한 Feature Scaling, 범주형 feature의 벡터화를 위한 One hot encoding, Target variable의 정규화를 차례로 진행하여 인공지능 예측 모델의 구현에 필요한 데이터를 준비한다.

	Age 21/22	G 21/22	GS 21/22	MP 21/22	FGA 21/22	3P 21/22	2PA 21/22	FT 21/22	DRB 21/22	...	USG% 21/22	WS/48 21/22	OBPM 21/22	VORP 21/22	Pos_PG	Pos_SG	Pos_SF	Pos_PF	Pos_C
0	-0.858388	0.921005	-0.039430	0.547006	0.326680	-0.215207	0.645045	-0.177885	1.006198	...	-0.094090	-0.446067	-0.597195	-0.750544	0.0	0.0	0.0	0.0	1.0
1	0.569737	1.073615	1.699017	0.921009	-0.299895	-1.107834	0.375175	0.139407	1.620256	...	-1.279017	1.032027	0.595834	0.963516	0.0	0.0	0.0	0.0	1.0
2	-0.382346	0.056218	0.996241	0.683503	0.687176	-1.107834	1.808859	1.704711	1.737219	...	1.090837	1.429365	0.874207	1.508899	0.0	0.0	0.0	0.0	1.0
3	-0.620367	0.514047	-0.298348	0.193476	0.546983	0.565843	0.202289	-0.146156	-0.287707	...	0.926771	-1.510930	-0.517660	-0.828456	0.0	1.0	0.0	0.0	0.0
4	0.093695	0.564916	1.181182	0.656204	0.229404	1.426591	-0.480819	-0.325954	0.004701	...	-0.695669	0.189673	0.436763	0.262310	0.0	1.0	0.0	0.0	0.0

그림 7: 최종적으로 준비된 데이터셋

데이터셋은 학습 데이터와 테스트 데이터로 분리하여 NBA 선수들의 연봉을 예측하는 인공지능 모델 구현에 사용한다. Linear Regression, Lasso Regression, Ridge Regression, XGBoost, LightGBM 등 다양한 기계 학습 알고리즘 이용해서 모델링을 진행하였으며, cross-validation을 통해 예측 정확도를 개선하였다. 평가 기준으로 RMSE(Root-Mean-Squared Error)와 R-Squared를 사용하여 모델 간 성능을 비교했다. 각 모델의 평가 수치는 다음의 표와 같다.

표 2: 인공지능 모델 별 RMSE / R-Squared 수치

	RMSE	R-Squared
Linear Regression	4.0602	0.6549
Ridge Regression	3.9691	0.6681
Lasso Regression	4.349	0.6035
LightGBM	4.0509	0.6556
XGBoost	3.8072	0.6957

XGB-Regressor를 사용한 모델이 가장 높은 정확도 점수를 얻었기 때문에 이를 기준으로 Feature importance를 측정하고, 높은 중요도를 가진 features를 재선정하여 최종 모델을 구현한다. 일부 feature의 경우, 서로 중복되거나 다른 feature와 유사하여 선정에서 제외되었다. 테스트 데이터에 대한 최종 모델의 성능은 Root Mean Squared Error: 3.4604, R-squared: 0.7568로 기존보다 향상되었다.

마지막으로 예측 연봉과 실제 연봉 데이터를 비교하는 시각화 자료를 만들어 학습 모델이 얼마나 정확한 예측을 하였는지 직관적으로 확인하였다.

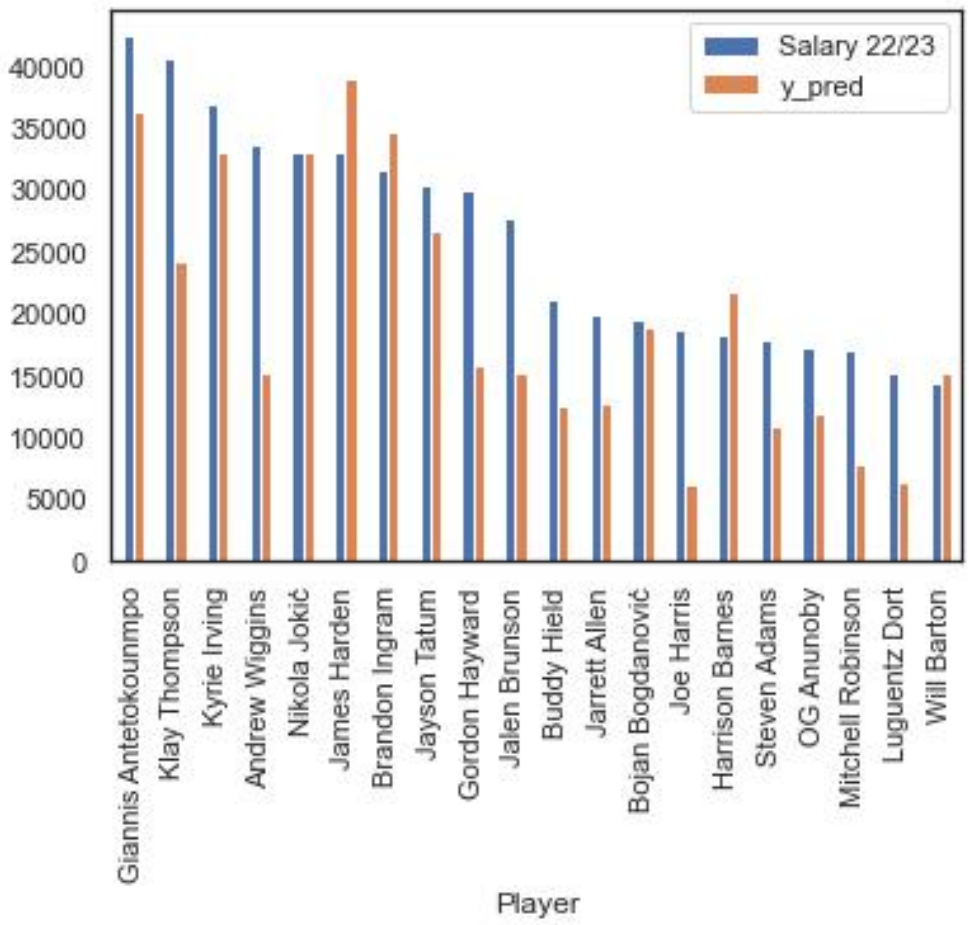

그림 8: 최종 모델의 예측 연봉과 실제 연봉 데이터를 비교하는 시각화 자료

■ 결론 및 소감

연구작품의 주제로 기계 학습을 통한 NBA 선수 연봉 예측 모델의 구현을 선정하게 된 이유는 평소 관심이 많았던 스포츠 분야와 인공지능 분야가 서로 결합이 된 프로젝트를 진행해보고 싶었기 때문이었다. 그러면서도 연구를 통해 생성된 결과물이 실생활에서 활용될 만큼의 성능을 가지도록 구현하고 싶었다. 실제로 많은 스포츠 구단이 인공지능을 적극적으로 활용하고 있고, 농구계에서도 기계 학습 기법을 이용하여 데이터를 효과적으로 분석하기 위한 연구가 활발하게 진행되고 있는 만큼 본 연구의 주제가 이러한 목적을 이루기 위해 적합하다고 판단했다.

그러나 주제를 정하고 연구작품을 진행하면서 많은 시행착오와 어려움을 겪게 되었다. 그동안 소프트웨어 프로젝트를 진행해 본 경험이 많지 않았고, 제안서나 보고서 등의 관련 문서를 작성해본 적은 더욱 없었기 때문에 연구의 방향을 잡는 데 오랜 시간이 걸렸고 진행도 빠르게 이루어지지 못했다. 그래서 주제와 관련된 프로젝트 자료와 논문을 찾아보면서 어려움을 해결하고자 했다. 인공지능 예측 모델을 주제로 한 다양한 논문들을 읽어보면서 기계 학습 분야의 최신 동향을 파악할 수 있었고, 이전까지 알지 못했던 지식을 배울 수 있었다. 또한, 본 연구를 진행하는 데 논문에서 제시된 기준과 연구 방법 등을 참고하여 활용하기도 했다. 이처럼 연구 진행에서 많은 어려움을 겪기는 했지만, 해당 문제들을 해결하는 과정에서 주제와 관련된 다양한 지식을 배울 수 있었다.

이렇게 연구된 내용과 결과는 기존 스포츠 분야의 예측 모델이 가지는 문제점을 해결하고, 사용자에게 개선된 성능을 제공한다. 스포츠 분야에서 데이터와 기계 학습 간의 결합이 비교적 최근 들어 활용되면서 아직까지 많은 연구가 이뤄지지 못했고, 그로 인해 이상적인 데이터셋과 학습 모델이 불분명한 상황이다. 그러나 데이터의 양이 급격하게 증가하고 다양한 분야로 활용 폭을 넓히면서 스포츠 분야에서 빅데이터를 다루는 기계 학습의 중요성은 커지고 있으며, 실제로 많은 스포츠 구단과 관련 기업에서 인공지능을 적극적으로 활용하고 있다.

본 연구에서는 득점과 도움 등 기존에 사용되었던 기록에 더해 1차 기록을 가공한 PER, WS, VORP 등의 2차 기록을 함께 사용하여 NBA 선수의 연봉에 영향을 주는 요소를 최적화했고, 회귀 분석과 시각화를 통해 탐색적 데이터 분석을 진행했다. 또한 히트맵을 활용하여 feature 간의 상관관계를 확인한 뒤 VIF 계수로 다중 공선성을 고려하면서 예측 정확도를 높이는 요인을 선정하고자 했다. 사용자에게 더 많은 선택지를 제공하기 위해서 다양한 인공지능 기법을 활용하여 모델링을 진행했고, 연구에 사용된 기계 학습 기법의 특징과 장단점 및 모델 운영에서의 차이점을 추가로 기재하여 본 연구에 대한 이해를 돕고 가독성을 높였다.

한 가지 아쉬웠던 부분은 구현 모델의 예측 정확도가 생각보다 낮게 나온 점이었다. 농구의 경우, 선수의 스탯에 관심을 가지고 관련된 기록을 수집한 기간이 오래되지 않다 보니 아직까지도 세부적인 데이터가 부족한 상황이다. 그러므로 기록이 지금보다 활성화되어 기계 학습에 더 많은 데이터 features를 사용할 수 있게 된다면 더욱 정확한 예측이 가능할 수 있을 거라 여겨진다. 또한, 농구라는 스포츠 특성상 포지션마다 선수가 맡아야 하는 역할이 매우 다르다는 문제가 존재한다. 포지션 외에도 선수의 특성이나 감독의 전술적 판단에 따라, 혹은 상대방의 전략에 따라 경기 상황이 무궁무진하게 바뀌기 때문에 정확한 예측을 위해서는 이러한 요인으로부터 최소한의 영향을 받는 features를 찾아야 한다. 그렇지만, 본 연구에서는 이에 대한 뾰족한 해결책을 찾지 못했다. 이 밖에도 현재 데이터로는 연봉을 산정하는데 있어서 선수들의 부상 여부나 중요 순간 활약도, 기록되지 못하는 공헌도 등을 반영하기 어렵다. 특히 FA 계약과 같은 외부적 요인에 의한 이상치 값은 정확한 연봉 예측에 있어서 큰 혼란을 유발했다. 그러므로 추후 진행되는 연구에서 해당 요소들을 적절하게 조정하여 예측에 반영한다면, 본 연구보다 개선된 결과를 얻을 수 있을 거라 기대한다.

이렇게 프로젝트를 진행하고 관련 보고서를 작성하는 과정을 통해 지금뿐만 아니라 미래에도 많은 도움이 될 경험을 할 수 있었다. 컴퓨터공학 복수전공을 시작하고 관련 전공과목을 수강하면서 학교에서 배운 내용으로 어떤 일을 할 수 있을지에 대한 의문을 줄곧 가지고 있었다. 그러나 졸업 프로젝트를 진행하면서 지금까지 배운 지식을 기반으로 실제 세상에서 도움이 될만한 소프트웨어 서비스를 개발할 수 있다는 것을 알게 되었다. 또한 많은 시행착오를 거치면서 개발자로서 부족한 점을 파악하게 되었고, 이를 해결하는 과정에서 많은 성장을 이루기도 했다. 제안서, 중간보고서, 최종보고서, 발표 자료 등 연구와 관련된 문서 작업을 하는 일 역시 처음에는 귀찮고 어렵게만 느껴졌지만, 문서를 작성하는 과정을 통해 연구를 체계적으로 구조화하여 진행할 수 있었고 이를 통해 소프트웨어 작업에서 문서화의 중요성과 필요성을 체감할 수 있었다. 기본적인 실력이 부족하고 프로젝트 경험이 많지 않아 연구 결과가 기대한 바에 미치지는 못했지만, 졸업 연구작품을 진행하면서 추후 유능한 개발자가 되기 위해 어떤 노력이 필요하고 어떤 부분을 보완해야 하는지 알 수 있게 되어 상당히 유익한 경험을 했다고 생각한다.

■ 참고문헌

[1] F. Thabtah, L. Zhang and N. Abdelhamid, "NBA Game Result Prediction Using Feature Analysis and Machine Learning.", Ann. Data. Sci. 6, 103–116, 2019.

[2] Jackie B. Yang and Ching-Heng Lu, "PREDICTING NBA CHAMPIONSHIP BY LEARNING FROM HISTORY DATA", AI & ML for ED, 2012.

[3] S. Narayan, "Applications of Machine Learning: Basketball Strategy", Massachusetts Institute of Technology, 2015.

[4] S. J. Kim, C. K. Heo and J. H. Do, "Analysis on the Korean basketball league using data mining", Korean Journal of Sports Science, 19(2), 1413-1420, 2010.

[5] 맹봉주, "KBL-국민체육진흥공단, 프로농구 데이터 활용 경진대회 개최", SPOTV, 2019년 10월 23일자.

[6] 문화체육관광부(2021), 2019 스포츠 산업백서

[7] S. H. Park, S. Y. Lim and Y. H. Park, "A Study on Application of Machine Learning Algorithms to Visitor Marketing in Sports Stadium", Journal of Digital Contents Society, vol. 19, no. 1, pp. 27-33, 2018.

[8] J. Park and S. Park, "A Study on Prediction of Attendance in Korean Baseball League Using Artificial Neural Network", KIPS Transactions on Software and Data Engineering, vol. 6, no. 12, pp. 565–572, Dec. 2017.

[9] J. W. Song, "A Multivariate Analysis of Korean Professional Players Salary," The Korean Journal of Applied Statistics, vol. 21, no. 3, pp. 441–453, Jun. 2008.

[10] 김정은, "인공지능(AI)은 스포츠 분야에서 어떤 역할을 할 수 있을까?", THE DAILYPOST, 2022년 6월 21일자.

[11] H. Choi, "Stock prediction analysis through artificial intelligence using big data", Journal of the Korea Institute of Information and Communication Engineering vol. 25, No. 10, pp.1435~1440, Oct. 2021.

[12] H. R. Jeong and C. W. Lim, "A review of artificial intelligence based demand forecasting techniques", The Korean Journal of Apllied Statistics, vol. 32, no. 6, pp. 795-835, Aug. 2019.

[13] Bernard Loeffel holz, Earl Bednar and Kenneth W Bauer, "Predicting NBA Games Using Neural Networks", Journal of Quantitative Analysis in Sports, Vol. 5, No. 1, 2009.

[14] João Gustavo Claudino, Daniel de Oliveira Capanema and Thiago Vieira de Souza, "Current Approaches to the Use of Artificial Intelligence for Injury Risk Assessment and Performance Prediction in Team Sports: a Systematic Review", Sports Medicine-Open, 2019.

■ 부록 (원천코드)

```
#!/usr/bin/env python
# coding: utf-8
# 이전 두 시즌의 통계적 수치를 입력 변수로 사용하여 NBA 선수들의 연봉을 예측하는 인공지능 모델을 구축하는 프로젝트
# ## Import packages and data
# 해당 프로젝트에서는 2020/21 시즌 및 2021/22 시즌 NBA 선수별 통계 기록과 2022/23 시즌 NBA 선수 연봉을 데이터로 활용한다.
# In[1]:


# Data Manipulation
import pandas as pd
pd.options.mode.chained_assignmentNone
importnumpy as np
importmissingno as msno


# In[2]:


# Visualization
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
importseaborn as sns


# In[3]:


# Machine Learning
fromsklearn.model_selection importtrain_test_split, KFold, cross_val_score
fromsklearn importmodel_selection, metrics
fromsklearn.preprocessing       importOneHotEncoder,       StandardScaler,
RobustScaler, MaxAbsScaler
fromsklearn.decomposition importPCA
fromsklearn.linear_model importLinearRegression, Ridge, Lasso
```

```
fromsklearn.tree importDecisionTreeRegressor
fromsklearn.ensemble                        importGradientBoostingRegressor,
RandomForestRegressor, AdaBoostRegressor
fromxgboost importXGBRegressor
fromlightgbm importLGBMRegressor


# In[4]:


# Other packages
fromscipy.stats importnorm
fromscipy importstats


# In[5]:


# Import data
df_salarypd.read_csv('./file/NBA_Player_Salary.csv')
df_stats2021pd.read_csv('./file/NBA_Player_Stats(2021).csv')
df_stats2122pd.read_csv('./file/NBA_Player_Stats(2122).csv')
df_list= [df_stats2021, df_stats2122, df_salary]


# ## 1. Data Cleaning
# 데이터셋으로 지정된 파일 각각의 처음 다섯 행을 살펴본다.
# In[6]:


# Salary
df_salary.head()


# In[7]:


# Season Stats 20/21
df_stats2021.head()
```

```
# In[8]:


# Season Stats 21/22
df_stats2122.head()


# 2022/23 시즌 선수들의 연봉을 예측하는 것이 프로젝트의 목표이므로 해당 시즌의 연봉에 관한 열을 가공한다.
# In[9]:


# Rename salary column
df_salarydf_salary.rename(columns= {'2022-23': 'Salary 22/23'})
# Transform salary to 1000
df_salary['Salary 22/23'] = df_salary['Salary 22/23']/1000


# 시즌별 선수 통계 데이터셋에서 동일 플레이어에 대한 중복 행을 삭제한다.
# * 프로젝트의 목표인 2022/23 시즌 연봉을 구하는 데 있어서 선수가 한 시즌 동안 어느 팀에서 뛰었는 지는 중요한 요소가 아니다.<br>그러므로 해당 시즌 동안 여러 팀에서 뛴 선수들의 행을 하나로 통합한다.
# In[10]:


# As total stats always is in the top row we can simply use the drop_duplicates function
df_stats2021df_stats2021.drop_duplicates(['Player'])
df_stats2122df_stats2122.drop_duplicates(['Player'])


# ### Merge Datasets
# 각 시즌별 선수 통계 데이터셋의 모든 열에 해당하는 연도를 할당한다.
# In[11]:


# Add season year to corresponding columns
```

```
columns_renamed= [s' 20/21'for s in list(df_stats2021.columns)]
df_stats2021.columnslist(df_stats2021.columns)[:3] + columns_renamed[3:]
columns_renamed= [s' 21/22'for s in list(df_stats2122.columns)]
df_stats2122.columnslist(df_stats2122.columns)[:3] + columns_renamed[3:]
# Delete Pos column from 17/18 df; we need it only once
df_stats2021df_stats2021.drop('Pos', axis1)


# In[12]:


# Merge datasets
df_statsdf_stats2021.merge(df_stats2122,              how'outer',left_on=
['Player'],right_on= ['Player'])
dfdf_stats.merge(df_salary,   how'outer',   left_on=   ['Player'],right_on=
['Player'])
df.head()


# 데이터셋의 각각의 열에 해당하는 데이터 타입을 확인한다.
# In[13]:


df.dtypes


# ### Drop unnecessary columns
# 앞으로의 프로젝트 과정에서 사용되지 않는 일부 열을 제거한다.
# In[14]:


# Columns of dataset
df.columns


# In[15]:


# Drop unnecessary columns
```

```
dfdf.drop(['Rk_x', 'Rk_y', 'Rk', 'Tm', '2023-24', '2024-25', '2025-26',
'2026-27', '2027-28',
             'Signed Using', 'Guaranteed'], axis1)


# ### Missing values
# 각 열에 존재하는 결측값을 확인한다.
# In[16]:


# Number of missing values for each column
df.isnull().sum()


# 20/21 시즌과 21/22 시즌에 대하여 Salary나 Stat 정보가 없는 행은 제거한다.
#
# * 선수들 중 일부는 20/21 시즌이나 21/22 시즌에 출전하지 않았으며, 몇몇 선수들
은 21/22 시즌에 대한 계약을 맺지 못했다.<br> 이러한 경우는 연봉을 예측하는 데 있
어서 도움이 되지 못하기 때문에 결측값으로 판단하여 제거하는 것이 합리적이다.
# In[17]:


# Drop rows with NaN
# Create Dataframe without stats for season 20/21
df1df.dropna(subset= ['Salary 22/23', 'PTS 21/22', 'eFG% 21/22'])
df1df1.reset_index()
columnslist(df1.columns)
for i in columns:
    if '20/21'in i:
        df1df1.drop([i], axis1)
df1df1.reset_index()
# Create Dataframe with stats for season 20/21
df2df.dropna(subset= ['Salary 22/23', 'PTS 21/22', 'eFG% 21/22', 'PTS
20/21', 'eFG% 20/21'])
df2df2.reset_index()


# 데이터셋 내에 존재하는 포지션을 확인한다.
# In[18]:
```

```
# Get unique positions
print(df1.Pos.unique())
print(df2.Pos.unique())


# 5개의 포지션: PG, SG , PF, SF, C으로 간소화하여 분류한다.
# In[19]:


# Replace duplicate positions with first position.
df1df1.replace({'SG-PG': 'SG', 'PG-SG': 'PG', 'SF-SG': 'SF', 'SG-SF':'SG',
'C-PF': 'C', 'PF-C':'PF'})
df2df2.replace({'SG-PG': 'SG', 'PG-SG': 'PG', 'SF-SG': 'SF', 'SG-SF':'SG',
'C-PF': 'C', 'PF-C':'PF'})


# In[20]:


df1.head()


# 주요 기록들에 대한 absolute growth를 얻기 위해서 데이터 프레임(df2)에 대한
20/21 시즌의 통계를 사용한다.
# In[21]:


# Get absolute growth
# List of stats of which we want the growth
list_growth= ['eFG%','TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS']
# Add absolute growth columns
for i in list_growth:
    df2[i' +-'] = df2[i' 21/22'] - df2[i' 20/21']

# Drop 17/18 columns
columnslist(df2.columns)
for i in columns:
```

```
    if '20/21'in i:
        df2df2.drop([i], axis1)


# In[22]:


print(df1.shape)
print(df2.shape)


# 20/21 시즌과 21/22 시즌에 대한 선수별 통계 기록과 22/23 시즌의 연봉 정보를 가
지는 행은 총 340개가 존재한다.<br>
# 21/22 시즌의 선수별 통계 기록과 22/23 시즌의 연봉 정보를 가지고 있는 행은 409
개로, 69개의 추가 행이 있다.
# ## 2. Exploring our dataset
# ### Target variable인 연봉 분석 (1000 단위)
# In[23]:


# Setup dataframe
df_saldf1[['Player', 'Salary 22/23']]
df_sal.sort_values(by'Salary 22/23', ascendingFalse, inplaceTrue)
# Create barchart
sns.catplot(x'Player',         y'Salary         22/23',         kind'bar',
datadf_sal.head()).set(xlabelNone)
plt.title('Players with highest salary (in 1000)')
plt.ylim([42000, 48000])
plt.xticks(rotation90)


# 연봉 관련 주요 통계적 요소에 대한 정보 요약
# In[24]:


# Statistics summary
df1['Salary 22/23'].describe()
```

```
# 연봉 분포(salary distribution)에 대한 히스토그램 모양 확인
# In[25]:


# Histogram
sns.distplot(df1['Salary 22/23'])


# 히스토그램을 통해 다음을 확인할 수 있다.
#
# 1. 연봉이 분산되어 있음을 나타내는 큰 표준 편차
# 2. 선수들의 연봉이 정규 분포(normal distribution)를 따르지 않음
# 3. 우편향 분포 (right-skewed distribution)
# ### Analysis of most important player stats
# NBA 선수들의 주요 기록(경기당 득점, 어시스트, 스틸, 리바운드)에 대해 살펴본다.
# 우선 각 지표의 리더를 확인한다.
# In[26]:


# Setup dataframes
df_ptsdf1[['Player', 'PTS 21/22']]
df_pts.sort_values(by'PTS 21/22', ascendingFalse, inplaceTrue)
df_astdf1[['Player', 'AST 21/22']]
df_ast.sort_values(by'AST 21/22', ascendingFalse, inplaceTrue)
df_stldf1[['Player', 'STL 21/22']]
df_stl.sort_values(by'STL 21/22', ascendingFalse, inplaceTrue)
df_trbdf1[['Player', 'TRB 21/22']]
df_trb.sort_values(by'TRB 21/22', ascendingFalse, inplaceTrue)
# Set up figure
f, axesplt.subplots(2, 2, figsize=(20, 15))
sns.despine(left=True)
# Create barcharts
sns.barplot(x'PTS   21/22',   y'Player',   datadf_pts.head(),   color"b",
axaxes[0, 0]).set(ylabelNone)
sns.barplot(x'AST   21/22',   y'Player',   datadf_ast.head(),   color"r",
axaxes[0, 1]).set(ylabelNone)
sns.barplot(x'STL   21/22',   y'Player',   datadf_stl.head(),   color"g",
axaxes[1, 0]).set(ylabelNone)
sns.barplot(x'TRB   21/22',   y'Player',   datadf_trb.head(),   color"m",
```

```
axaxes[1, 1]).set(ylabelNone)


# 히스토그램을 이용하여 각 지표의 분산을 확인한다.
# In[27]:


# Set up figure
f, axesplt.subplots(2, 2, figsize=(20, 15))
sns.despine(left=True)
# Histograms
sns.distplot(df1['PTS 21/22'], color"b", axaxes[0, 0])
sns.distplot(df1['AST 21/22'], color"r", axaxes[0, 1])
sns.distplot(df1['STL 21/22'], color"g", axaxes[1, 0])
sns.distplot(df1['TRB 21/22'], color"m", axaxes[1, 1])


# 농구의 중요 통계적 수치 역시 대부분 우편향 분포(right-skewed distribution)임
을 확인할 수 있다.
# ### Relationship with possible features
# 22/23 시즌 연봉과 21/22 시즌 경기당 득점, 어시스트, 스틸, 리바운드 사이의 관
계를 각각 살펴본다.
# In[28]:


# Set up figure
f, axesplt.subplots(2, 2, figsize=(20, 15))
# Regressionplot
sns.regplot(xdf1['PTS 21/22'], ydf1['Salary 22/23'], color="b", ax=axes[0,
0])
sns.regplot(xdf1['AST 21/22'], ydf1['Salary 22/23'], color="r", ax=axes[0,
1])
sns.regplot(xdf1['STL 21/22'], ydf1['Salary 22/23'], color="g", ax=axes[1,
0])
sns.regplot(xdf1['TRB 21/22'], ydf1['Salary 22/23'], color="m", ax=axes[1,
1])


# 선택된 모든 통계적 수치들과 연봉 사이에는 positive한 선형 관계가 있다.
```

```
# In[29]:


# Relationship with effecitve field goal percentage
sns.regplot(xdf1['eFG% 21/22'], ydf1['Salary 22/23'])


# 농구에서는 2점슛과 3점슛 간의 차이가 있기 때문에 normal field goal percentage 대신 effective field goal percentage를 선택했다.<br>
# 그러나 위 그래프에서 확인할 수 있듯이 postive한 선형 관계를 관찰하기 어렵다.<br>
# (eFG%가 높으면서 연봉이 낮은 선수들은 있지만, eFG%가 낮으면서 높은 연봉을 받는 선수는 존재하지 않는다.)
# In[30]:


# Relationship with minutes played per game
sns.regplot(xdf1['MP 21/22'], ydf1['Salary 22/23'])


# 경기당 경기 시간과 연봉 사이에는 exponential relationship으로 보이는 positive relationship이 존재한다.<br>
# 그러나 좋은 기록을 가지는 선수는 다음 시즌에 높은 연봉과 많은 경기시간을 보장받을 가능성이 높기 때문에 해당 관계의 분석에 신중할 필요가 있다.<br>
# 즉, 이러한 postive relationship은 기록에서 기반한 것으로 판단이 가능하다.
# In[31]:


# Relationship with age
sns.regplot(xdf1['Age 21/22'], ydf1['Salary 22/23'])


# 선수의 나이와 연봉 사이에는 선형적인 관계가 없다.
# In[32]:


# Relationship with Position
sns.boxplot(x'Pos', y'Salary 22/23', datadf1, order= ['PG', 'SG', 'SF', 'PF', 'C'])
```

```
# ### Correlation matrix
# 지금까지의 분석은 직관에 따라 연봉을 결정하는 데 있어 중요하다고 생각되는 요소들에 기반을 두고 진행했다.<br>
# 좀 더 객관적인 분석을 진행하고 변수 간의 관계에 대하여 완벽한 개요를 얻기 위해서 히트맵을 활용한다.
# In[33]:


sns.set(style"white")
cor_matrixdf1.loc[:, 'Age 21/22': 'Salary 22/23'].corr()
# Generate a mask for the upper triangle
masknp.triu(np.ones_like(cor_matrix, dtypenp.bool))
plt.figure(figsize= (15, 12))
# Generate a custom diverging colormap
cmapsns.diverging_palette(220, 10, as_cmapTrue)
sns.heatmap(cor_matrix, maskmask, cmapcmap, center0,
            squareTrue, linewidths.5, cbar_kws= {"shrink": .5})


# 히트맵을 통한 분석
#
# 1. 다중 공선성: 정사각형의 색깔이 붉을수록 높은 상관 관계를 가지는 변수들로, 예측 모델에 거의 동일한 정보를 제공하고 추정에 대한 분산을 증가시킨다. 회귀 분석은 특징들 간의 독립성을 전제로 하기 때문에 올바른 회귀 분석을 위해서는 이러한 쌍을 제거해야한다. 다중 공선성은 분산팽창요인(VIF, Variance Inflation Factor) 이라는 계수로 평가할 수 있으며, 일반적으로 VIF 계수가 10~15 정도를 넘으면 다중 공선성 문제가 발생했다고 판단한다.
#
#
# 2. 22/23 시즌 연봉과의 상관관계: 앞서 분석했듯이, 연봉과 네 가지의 주요 통계 요소들 사이에는 선형적인 관계가 존재한다. 그러나 이 밖에도 추가적으로 고려해야 할 다른 변수들도 있음을 확인할 수 있다.
#
# 연봉과 경기당 플레이 시간(MP, minutes played)이나 선발 출전 횟수(GS, games started)의 상관관계를 분석할 때는 신중해야 하는데, 단순하게 선발 출전을 많이 하는 선수가 더 많은 돈을 번다고 할 수 없기 때문이다. 공격과 수비에 대한 통계를 참고하여 코치는 선수가 경기에 얼마나 오래 출전할 지를 결정하고, 이에 따라 다음 시즌의
```

```
연봉 역시 결정되므로 인과관계라고 하기는 어렵다. 다만 총 게임 수(G)는 모델에 활용이 가능하다. 이 통계는 선수의 취약성과 연봉의 관계를 측정할 수 있다. 또한 높은 상관관계로 인해 TOV(Turnover)가 많을수록 연봉이 높아진다고 말하기는 어렵다.  공격력과 수비력이 좋은 선수는 더 많은 경기 시간과 더 높은 연봉을 보장 받는다. 그렇기 때문에 연봉이 높은 좋은 선수는 경기 시간이 많아지고, 평균적으로 턴오버 역시 더 많을 수 밖에 없다.
# In[34]:


fromstatsmodels.stats.outliers_influence importvariance_inflation_factor
Vdf1[['Age 21/22', 'G 21/22', 'GS 21/22', 'MP 21/22', 'FG 21/22', 'FGA 21/22',
         'FG% 21/22', '3P 21/22', '3PA 21/22', '2P 21/22', '2PA 21/22', '2P% 21/22',
         'eFG% 21/22', 'FT 21/22', 'FTA 21/22', 'ORB 21/22', 'DRB 21/22', 'TRB 21/22',
         'AST 21/22', 'STL 21/22', 'BLK 21/22', 'TOV 21/22', 'PF 21/22', 'PTS 21/22',
          'PER 21/22', 'TS% 21/22', '3PAr 21/22', 'FTr 21/22', 'ORB% 21/22', 'DRB% 21/22',
          'TRB% 21/22', 'AST% 21/22', 'STL% 21/22', 'BLK% 21/22', 'TOV% 21/22', 'USG% 21/22',
         'OWS 21/22', 'DWS 21/22', 'WS 21/22', 'WS/48 21/22','OBPM 21/22', 'DBPM 21/22',
        'BPM 21/22', 'VORP 21/22']]
# 특징마다의 VIF 계수를 출력합니다.
vifpd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(V.values, i) for i in range(V.shape[1])]
vif["features"] = V.columns
vif.round(1)


# 다중 공선성 문제가 없는 적절한 특징들을 선정한다.
#
# * 선정 과정
# 1. VIF 계수가 높은 특징을 우선적으로 제거한다. 단, (FG, FG%)와 같이 유사한 특징 중에서는 하나만을 제거한다.
#
```

```
# 2. 다시 다중 공선성을 검증한다.<br>
# (VIF 계수가 비정상적으로 높은 특징을 제거해주면, 다른 특징들의 공선성이 감소하는 것을 확인할 수 있다.)
#
# 3. 여전히 VIF 계수가 높은 특징들을 제거한다.
# In[35]:


Vdf1[['Age 21/22', 'G 21/22', 'GS 21/22', 'MP 21/22', 'FGA 21/22','3P 21/22', '2PA 21/22',
         'FT 21/22', 'DRB 21/22', 'AST 21/22', 'STL 21/22', 'BLK 21/22', 'TOV 21/22', 'PF 21/22', 'PTS 21/22',
          'PER 21/22', 'TS% 21/22', '3PAr 21/22', 'FTr 21/22', 'AST% 21/22', 'STL% 21/22',
          'BLK% 21/22', 'TOV% 21/22', 'USG% 21/22','WS/48 21/22','OBPM 21/22', 'VORP 21/22']]
# 특징마다의 VIF 계수를 출력합니다.
vifpd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(V.values, i) for i in range(V.shape[1])]
vif["features"] = V.columns
vif.round(1)


# ### What about the relationship between the absolute changes and salary?
# 20/21 시즌과 21/22 시즌의 성적 변화가 선수들의 연봉을 결정하는 데 영향을 미치는 지 확인한다.
# In[36]:


cor_matrixdf2.loc[:, ['eFG% +-','TRB +-', 'AST +-', 'STL +-', 'BLK +-', 'TOV +-', 'PF +-', 'PTS +-',
                      'Salary 22/23']].corr()
# Generate a mask for the upper triangle
masknp.triu(np.ones_like(cor_matrix, dtypenp.bool))
plt.figure(figsize= (10, 8))
# Generate a custom diverging colormap
cmapsns.diverging_palette(220, 10, as_cmapTrue)
sns.heatmap(cor_matrix, maskmask, cmapcmap, center0,
```

```
            squareTrue, linewidths.5, cbar_kws= {"shrink": .5})


# Absolute change와 연봉 간에는 선형적인 관계는 없어 보인다.<br> 그러므로 예측 모델을 구현 하는 데 있어서 absolute changes와 20/21 시즌의 통계 수치를 제외한다.
# ## 3. Data Preparation
# ### Feature Scaling
# 특징들 간의 단위를 맞추기 위해서 StandardScaler를 이용한 Scaling을 진행한다.
# In[37]:


# Define the function scaling for each feature
def standard_scaling(df, scale_columns):
    for col in scale_columns:
        series_meandf[col].mean()
        series_stddf[col].std()
        df[col] = df[col].apply(lambda x: (x-series_mean)/series_std)
    return df


# In[38]:


# Scaling for each features
scale_columns= ['Age 21/22', 'G 21/22', 'GS 21/22', 'MP 21/22', 'FG 21/22', 'FGA 21/22',
         'FG% 21/22', '3P 21/22', '3PA 21/22', '2P 21/22', '2PA 21/22', '2P% 21/22',
         'eFG% 21/22', 'FT 21/22', 'FTA 21/22', 'ORB 21/22', 'DRB 21/22', 'TRB 21/22',
         'AST 21/22', 'STL 21/22', 'BLK 21/22', 'TOV 21/22', 'PF 21/22', 'PTS 21/22',
          'PER 21/22', 'TS% 21/22', '3PAr 21/22', 'FTr 21/22', 'ORB% 21/22', 'DRB% 21/22',
          'TRB% 21/22', 'AST% 21/22', 'STL% 21/22', 'BLK% 21/22', 'TOV% 21/22', 'USG% 21/22',
         'OWS 21/22', 'DWS 21/22', 'WS 21/22', 'WS/48 21/22','OBPM 21/22', 'DBPM 21/22',
```

```
        'BPM 21/22', 'VORP 21/22']
df1standard_scaling(df1, scale_columns)
df1.head(5)


# ### Define target variable and features
# In[39]:


ydf1.loc[:, 'Salary 22/23']
xdf1.loc[:, ['Pos', 'Age 21/22', 'G 21/22', 'GS 21/22', 'MP 21/22', 'FGA
21/22','3P 21/22', '2PA 21/22',
         'FT 21/22', 'DRB 21/22', 'AST 21/22', 'STL 21/22', 'BLK 21/22',
'TOV 21/22', 'PF 21/22', 'PTS 21/22',
          'PER 21/22', 'TS% 21/22', '3PAr 21/22', 'FTr 21/22', 'AST%
21/22', 'STL% 21/22',
          'BLK% 21/22', 'TOV% 21/22', 'USG% 21/22','WS/48 21/22','OBPM
21/22', 'VORP 21/22']]
print(x.shape)
print(y.shape)


# ### One-Hot encoding for feature position
# 범주형 변수를 다루기 위해서 One-Hot encoder를 사용한다.
# In[40]:


# Instantiate OneHotEncoder
ohe= OneHotEncoder(categories= [['PG', 'SG', 'SF', 'PF', 'C']])
# Apply one-hot encoder
x_ohepd.DataFrame(ohe.fit_transform(x['Pos'].to_frame()).toarray())
# Get feature names
x_ohe.columnsohe.get_feature_names(['Pos'])
# One-hot encoding removed index; put it back
x_ohe.indexx.index
# Add one-hot encoded columns to numerical features and remove categorical
column
xpd.concat([x, x_ohe], axis=1).drop(['Pos'], axis=1)
# How does it look like?
```

```
x.head()


# ### Split Data into train and test
# 회귀 분석을 위한 학습, 테스트 데이터셋 분리한다.
# In[41]:


# Split data using train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size0.2,
random_state0)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)


# ### Normalise y
# 앞서 종속 변수(dependent variable)가 정규 분포를 근사적으로 따르지 않는 것을
확인했다.<br>
# 그러므로 머신러닝 모델을 구현하기 위해서 정규화를 해준다.
# In[42]:


#Apply cube-root transformation
y_trainpd.DataFrame(np.cbrt([y_train])).T
y_testpd.DataFrame(np.cbrt([y_test])).T
ypd.DataFrame(np.cbrt([y])).T
#transformed histogram and normal probability plot
f, axesplt.subplots(1, 2, figsize= (10, 5), sharexTrue)
sns.distplot(y_train, color"skyblue", fit= norm, axaxes[0],
axlabel"y_train")
sns.distplot(y_test, color"olive",fit= norm, axaxes[1], axlabel"y_test")
#sns.distplot(y, color = "olive",fit = norm, axlabel = "y")


# ## 4. Building Machine Learning Models
# ### Basic machine learning algorithms
# Model 평가 기준으로 RMS(Root-Mean-Squared Error)와 R-Squared를 활용한다.
```

```
#
# 1. RMSE score: 실제값과 예측값의 차이를 절대값으로 나타낸 것으로, 해당 값이 높을수록 예측이 부정확하다는 뜻이다.
#
#
# 2. R-Squared score: 회귀 분석으로 추정한 모델이 주어진 데이터를 얼마나 잘 설명하는지 나타내는 점수로, 데이터를 잘 설명하는 모델일수록 1에 가깝다.
# In[43]:


# Function which uses an algorithm as input and returns the desired accuracy metrics and some predictions
def alg_fit(alg, x_train, y_train, x_test, name, y_true, df, mse, r2):

    # Model selection
    modalg.fit(x_train, y_train)

    # Prediction
    y_predmod.predict(x_test)

    # Accuracy
    acc1round(mse(y_test, y_pred), 4)
    acc2round(r2(y_test, y_pred), 4)

    # Accuracy table
    x_test['y_pred'] = mod.predict(x_test)
    df_accpd.merge(df, x_test, how'right')
    x_test.drop(['y_pred'], axis1, inplaceTrue)
    df_accdf_acc[[name, y_true, 'y_pred']]
    df_acc.sort_values(byy_true, ascendingFalse, inplaceTrue)
    df_acc['y_pred'] = df_acc['y_pred']**3

    return y_pred, acc1, acc2, df_acc


# ### Linear Regression
# In[44]:
```

```
# Linear Regression
y_pred_lin,    mse_lin,    r2_lin,    df_acc_linalg_fit(LinearRegression(),
x_train, y_train, x_test, 'Player', 'Salary 22/23',
                                                                    df1,
metrics.mean_squared_error, metrics.r2_score)
print("Root Mean Squared Error: %s"round(np.sqrt(mse_lin), 4))
print("R-squared: %s"r2_lin)
df_acc_lin.head(10)


# ### Ridge Regression
# 선형 회귀(Linear Regression) 모델은 다중 공선성 문제를 가지기 때문에 이에 대한 적합한 해결책을 찾아야 한다.<br>
# 다중 공선성 문제는 부정확한 추정을 유발하고 큰 표준 오차를 가지게 한다.<br>
# Ridge regression은 람다 값을 조절해 추정에 개선된 효율성을 제공하여 해당 문제를 완화한다.
# In[45]:


# Ridge Regression
y_pred_rid,  mse_rid,  r2_rid,  df_acc_ridalg_fit(Ridge(alpha1),  x_train,
y_train, x_test, 'Player', 'Salary 22/23',
                                                                    df1,
metrics.mean_squared_error, metrics.r2_score)
print("Root Mean Squared Error: %s"round(np.sqrt(mse_rid), 4))
print("R-squared: %s"r2_rid)
df_acc_rid.head(10)


# ### Lasso Regression
# Lasso Regression는 Ridge Regression과 개념적으로 매우 유사하다. 0이 아닌 계수에 대해 패널티를 추가로 부여하며, Ridge Regression과 다르게 계수의 제곱합이 아닌 계수의 절대값으로 제한한다.
# In[46]:


# Lasso Regression
y_pred_las, mse_las, r2_las, df_acc_lasalg_fit(Lasso(alpha0.001), x_train,
y_train, x_test, 'Player', 'Salary 22/23',
```

```
                                                                df1,
metrics.mean_squared_error, metrics.r2_score)
print("Root Mean Squared Error: %s"round(np.sqrt(mse_las), 4))
print("R-squared: %s"r2_las)
df_acc_las.head(10)


# ### Cross Validation
# Cross Validation을 통해 보다 정교한 정확도 측정을 제공한다.
# In[47]:


def alg_fit_cv(alg, x, y, mse, r2):

    # Cross validation
    cv= KFold(shuffleTrue, random_state0, n_splits5)

    # Accuracy
    scores1= cross_val_score(alg, x, y, cvcv, scoringmse)
    scores2= cross_val_score(alg, x, y, cvcv, scoringr2)
    acc1_cvround(scores1.mean(), 4)
    acc2_cvround(scores2.mean(), 4)

    return acc1_cv, acc2_cv


# In[48]:


# Linear Regression
mse_cv_lin,       r2_cv_linalg_fit_cv(LinearRegression(),       x,       y,
'neg_mean_squared_error', 'r2')
print("Root Mean Squared Error: %s"round(np.sqrt(mse_cv_lin*-1), 4))
print("R-squared: %s"r2_cv_lin)


# In[49]:
```

```
# Ridge Regression
mse_cv_rid,        r2_cv_ridalg_fit_cv(Ridge(alpha23),        x,        y,
'neg_mean_squared_error', 'r2')
print("Root Mean Squared Error: %s"round(np.sqrt(mse_cv_rid*-1), 4))
print("R-squared: %s"r2_cv_rid)


# In[50]:


# Lasso Regression
mse_cv_las, r2_cv_lasalg_fit_cv(Lasso(), x, y, 'neg_mean_squared_error',
'r2')
print("Root Mean Squared Error: %s"round(np.sqrt(mse_cv_las*-1), 4))
print("R-squared: %s"r2_cv_las)


# ### Advanced models
#
# ### LightGBM Regressor
# In[51]:


# LightGBM Regressor (after some parameter tuning)
lgbm= LGBMRegressor(objective'regression',
                    num_leaves20,
                    learning_rate0.03,
                    n_estimators200,
                    max_bin50,
                    bagging_fraction0.85,
                    bagging_freq4,
                    bagging_seed6,
                    feature_fraction0.2,
                    feature_fraction_seed7,
                    verbose= -1)
mse_cv_lgbm, r2_cv_lgbmalg_fit_cv(lgbm, x, y, 'neg_mean_squared_error',
'r2')
print("Root Mean Squared Error: %s"round(np.sqrt(mse_cv_lgbm*-1), 4))
print("R-squared: %s"r2_cv_lgbm)
```

```
# ### XGB-Regressor
# In[52]:


# XGB-Regressor (after some parameter tuning)
xgb= XGBRegressor(n_estimators300,
                  max_depth2,
                  min_child_weight0,
                  gamma8,
                  subsample0.6,
                  colsample_bytree0.9,
                  objective'reg:squarederror',
                  nthread= -1,
                  scale_pos_weight1,
                  seed27,
                  learning_rate0.02,
                  reg_alpha0.006)
mse_cv_xgb, r2_cv_xgbalg_fit_cv(xgb, x, y, 'neg_mean_squared_error', 'r2')
print("Root Mean Squared Error: %s"round(np.sqrt(mse_cv_xgb*-1), 4))
print("R-squared: %s"r2_cv_xgb)


# ### Feature importance
# XGB-Regressor를 사용하여 가장 높은 정확도 점수를 얻었기 때문에 이를 이용한 모
델로 Feature importance를 측정한다.
# In[53]:


# Model
modxgb.fit(x, y)
# Feature importance
df_feature_importancepd.DataFrame(xgb.feature_importances_,
indexx.columns,
                                                  columns=  ['feature
importance']).sort_values('feature importance', ascendingFalse)
df_feature_importance
```

```
# ### New features
# 높은 중요도를 가진 특징을 사용한다. 다만 일부 특징의 경우, 중복되거나 다른 특
징에 가깝기 때문에 사용할 수 없다. 예를 들어, 경기당 필드골(FG)은 이미 경기당 포
인트(PTS)에 포함되기 때문에 필요하지 않다.
# In[54]:


# Drop out features with low importance or which are redundant
x_newx.loc[:, ['Age  21/22', 'GS  21/22', 'FGA  21/22','3P  21/22', '2PA
21/22', 'FT 21/22',
    'DRB 21/22', 'AST 21/22', 'PTS 21/22', 'PER 21/22', 'USG% 21/22','OBPM
21/22', 'VORP 21/22']]


# ### Final Model
#
# In[55]:


# XGB-Regressor (after some parameter tuning)
xgb_new= XGBRegressor(n_estimators270,
                      max_depth2,
                      min_child_weight0,
                      gamma18,
                      subsample0.7,
                      colsample_bytree0.9,
                      objective'reg:squarederror',
                      nthread= -1,
                      scale_pos_weight1,
                      seed27,
                      learning_rate0.023,
                      reg_alpha0.02)
mse_cv_xgb,         r2_cv_xgbalg_fit_cv(xgb_new,         x_new,         y,
'neg_mean_squared_error', 'r2')
print("Root Mean Squared Error: %s"round(np.sqrt(mse_cv_xgb*-1), 4))
print("R-squared: %s"r2_cv_xgb)
```

```
# ### Let's see how it performs on some test data
# In[56]:


# Split data now with x_new
x_train, x_test, y_train, y_test= train_test_split(x_new, y, test_size0.2,
random_state0)
# Use function to fit algorithm
y_pred_xgb, mse_xgb, r2_xgb, df_acc_xgbalg_fit(xgb_new, x_train, y_train,
x_test, 'Player', 'Salary 22/23',
                                                                    df1,
metrics.mean_squared_error, metrics.r2_score)
print("Root Mean Squared Error: %s"round(np.sqrt(mse_xgb), 4))
print("R-squared: %s"r2_xgb)
df_acc_xgb.head(10)


# ### Visualization
# 예상 연봉과 실제 연봉 비교한다.
# In[57]:


df_acc_xgb.head(20).plot(x='Player',    y=['Salary    22/23',    'y_pred'],
kind="bar")
```