빅데이터 처리 프로젝트 발표

3-C 장용수(202144077)

프로젝트 개요

주제 선정 : NHL 선수들의 연봉 대비 퍼포먼스 분석

선정 이유 : NHL은 샐러리캡(모든 팀의 선수 총 연봉이 같음)이 적용되어 있기 때문에 모든 선수들이 받는 돈보다 뛰어난 활약을 보여주는 것이 중요하고, 그런 팀이 우승할 확률이 높습니다.

따라서 선수들의 스탯과 연봉을 분석하여, 각 선수의 연봉 대비 퍼포먼스를 점수로 보여주는 프로젝트를 선정하였습니다.

데이터 수집

* 선수별 스탯(출전경기수, 골, 어시스트, 슈팅횟수 등등) 데이터
  - 수집방법: NHL 공식사이트(https://www.nhl.com/stats/skaters)에서 제공
  - 수집범위: NHL에 샐러리캡이 적용된 2005시즌 이후부터 2022시즌까지
* 선수별 연봉 데이터
  - 수집방법: 'capfriendly'라는 사이트에서 크롤링
  - 수집범위: 2005시즌 ~ 2022시즌
* 시즌별 각 팀의 총 연봉 상한선 데이터
  - 수집방법: 'capfriendly'라는 사이트에서 크롤링
  - 수집범위: 전체

진행과정

1. 스탯데이터와 연봉데이터의 전처리를 진행한다.
(예를 들어, 2005시즌의 연봉 10억은 2022시즌 연봉 21억 1500만원으로 환산된다.)
2. 각 선수의 스탯데이터와 연봉데이터를 병합한다.
3. 스탯데이터를 feature로, 연봉데이터를 target으로 머신러닝(지도학습)을 진행하여 스탯데이터가 입력되면 그에 상응하는 연봉을 출력하는 모델을 생성한다.
4. 생성된 모델을 이용하여 예상연봉을 출력하고, 예상연봉을 실제연봉으로 나누어 각 선수의 가성비를 판단한다.
5. 각 팀별로 가성비가 좋은 선수가 얼마나 있는지, 팀성적과 어떤 연관관계가 있는지 시각화하여 나타낸다.

데이터 수집 - 스탯 데이터

NHL 공식 사이트에서 제공

BY SEASON | BY GAME | ALL-TIME

Legend | Print

SEASONS: 2023-24 - 2023-24 | SUM RESULTS | GAME TYPE: Regular Season | FRANCHISE: All Franchises | POSITION: All Skaters | REPORT: Summary | Search Players

Get Stats | More Filters | Clear Filters

Summary

Export

	Player	Season	Team	S/C	Pos	GP	G↓	A↓	P↓	+/-	PIM	P/GP	EVG	EVP	PPG	PPP	SHG	SHP	OTG	GWG	S	S%	TOI/GP	FOW%
1	Nikita Kucherov	2023-24	TBL	L	R	20	15	20	35	+2	6	1.75	9	17	6	18	0	0	0	1	92	16.3	21:29	0.0
2	J.T. Miller	2023-24	VAN	L	C	22	13	20	33	+10	16	1.50	6	17	6	15	1	1	0	2	50	26.0	20:04	54.9
3	Quinn Hughes	2023-24	VAN	L	D	22	8	25	33	+18	12	1.50	7	18	1	15	0	0	1	2	68	11.8	24:46	--

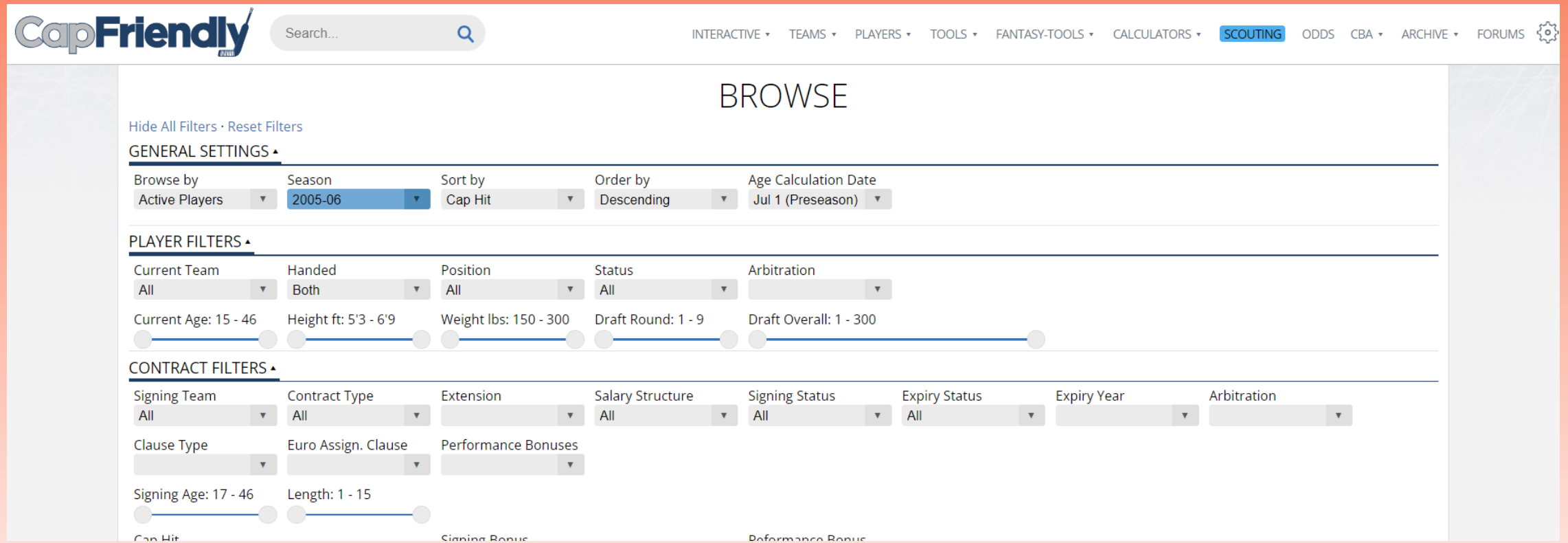
데이터 수집 - 스탯 데이터

엑셀파일로 제공되는 데이터 사진

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Player	Season	Team	S/C	Pos	GP	G	A	P	+/-	PIM	P/GP	EVG	EVP	PPG	PPP	SHG	SHP
2	Nikita Kucherov	20232024	TBL	L	R	20	15	20	35	2	6	1.75	9	17	6	18	0	0
3	J.T. Miller	20232024	VAN	L	C	22	13	20	33	10	16	1.5	6	17	6	15	1	1
4	Quinn Hughes	20232024	VAN	L	D	22	8	25	33	18	12	1.5	7	18	1	15	0	0
5	David Pastrnak	20232024	BOS	R	R	20	13	18	31	7	10	1.55	9	17	4	14	0	0
6	Cale Makar	20232024	COL	R	D	20	5	25	30	18	8	1.5	4	17	1	10	0	3
7	Brayden Point	20232024	TBL	R	C	21	11	18	29	-3	2	1.38	5	16	6	13	0	0
8	Artemi Panarin	20232024	NYR	R	L	19	11	18	29	3	4	1.53	9	17	2	12	0	0
9	Elias Pettersson	20232024	VAN	L	C	22	8	21	29	2	0	1.32	4	15	4	13	0	1
10	Leon Draisaitl	20232024	EDM	L	C	20	9	19	28	-3	24	1.4	3	15	6	12	0	1
11	William Nylander	20232024	TOR	R	R	19	12	15	27	3	4	1.42	6	16	5	10	1	1
12	Mikko Rantanen	20232024	COL	L	R	20	12	14	26	8	16	1.3	9	17	3	9	0	0
13	Nathan MacKinnon	20232024	COL	R	C	20	7	19	26	6	22	1.3	7	17	0	9	0	0
14	Sam Reinhart	20232024	FLA	R	C	20	13	12	25	10	9	1.25	7	17	5	7	1	1
15	Filip Forsberg	20232024	NSH	R	L	20	11	14	25	1	14	1.25	10	16	1	9	0	0
16	Jesper Bratt	20232024	NJD	L	L	19	8	17	25	-3	2	1.32	4	9	4	16	0	0
17	Connor McDavid	20232024	EDM	L	C	18	7	18	25	-4	10	1.39	5	13	2	12	0	0
18	Brock Boeser	20232024	VAN	R	R	22	15	9	24	8	2	1.09	8	15	7	9	0	0
19	Sidney Crosby	20232024	PIT	L	C	20	13	11	24	11	14	1.2	11	21	2	3	0	0
20	Kyle Connor	20232024	WPG	L	L	20	14	9	23	-8	2	1.15	11	14	3	9	0	0

데이터 수집 - 연봉 데이터

크롤링할 사이트 사진 1



데이터 수집 - 연봉 데이터

크롤링할 사이트 사진 2

CapFriendly Search... INTERACTIVE TEAMS PLAYERS TOOLS FANTASY-TOOLS CALCULATORS SCOUTING ODDS CBA ARCHIVE

RESULTS (847)

Stats displayed are season only · 1 GP required when sorting by stats

PLAYER	TEAM	AGE	DATE OF BIRTH	COUNTRY	WEIGHT	HEIGHT	POS	HANDED	CLAUSE	CAP HIT
1. Alex Ovechkin	WSH	23	Sep. 17, 1985	Russia	107	191	LW	Right		$9,538,462
2. Sidney Crosby	PIT	21	Aug. 7, 1987	Canada	91	180	C	Left		$8,700,000
3. Jaromír Jágr	-	36	Feb. 15, 1972	Czech Republic	104	191	RW	Left		$7,920,000
4. Brad Richards	-	28	May 2, 1980	Canada	90	183	C, LW	Left	NTC	$7,800,000
5. Zdeno Chara	-	31	Mar. 18, 1977	Slovakia	113	206	LD	Left	NTC	$7,500,000
6. Dany Heatley	-	27	Jan. 21, 1981	Germany	96	191	LW	Left	NMC	$7,500,000
7. Nicklas Lidström	-	38	Apr. 28, 1970	Sweden	87	185	D	Left	NMC	$7,450,000
8. Marián Hossa	-	29	Jan. 12, 1979	Slovakia	94	185	RW	Left		$7,450,000
9. Scott Gomez	-	28	Dec. 23, 1979	United States	90	180	C	Left		$7,357,143
10. Joe Thornton	-	29	Jul. 2, 1979	Canada	100	193	C, LW	Left	NTC	$7,200,000
11. Brian Campbell	-	29	May 23, 1979	Canada	87	178	LD	Left	M-NTC	$7,142,875
12. Thomas Vanek	-	24	Jan. 19, 1984	Austria	97	188	LW, RW	Right		$7,142,857
13. Chris Drury	-	32	Aug. 20, 1976	United States	86	178	C	Right		$7,050,000
14. Jarome Iginla	-	31	Jul. 1, 1977	Canada	95	185	RW	Right	NMC	$7,000,000
15. Jason Spezza	-	25	Jun. 13, 1983	Canada	98	191	C, RW	Right		$7,000,000
16. Vincent Lecavalier	-	28	Apr. 21, 1980	Canada	98	193	C, LW	Left		$6,875,000
17. Henrik Lundqvist	-	26	Mar. 2, 1982	Sweden	83	185	G	Left		$6,875,000

연봉 데이터 크롤링 - 크롤링 전 준비 코드

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import pandas as pd
```

```
base_url = "https://www.capfriendly.com/browse/active/"
```

데이터 수집 - 연봉 데이터

연봉 데이터 크롤링 - 크롤링 코드

```
#  2006 = 0506시즌
#  22~23시즌까지 크롤링
for year in range(2006, 2024):
  result = []

  url = base_url
  url +=  "{}".format(year)
  url += "?age-calculation-date=october1&display=birthday,country,weightkg,heightcm&hide=expiry-status,salary,skater-stats,goal

  for page in range(1, 35):
    url += ("&pg=" + "{}".format(page))
    html = urlopen(url)
    bs_obj = BeautifulSoup(html, "html.parser")
    tbody_tag = bs_obj.find("tbody")

    for tr in tbody_tag:
      row = []
      td = tr.find_all("td")
      for i in range(11):
        row.append(td[i].text)
      result.append(row)

  df = pd.DataFrame(result)
  df.to_csv("./{y}sal.csv".format(y=year-1), header=['PLAYER', 'TEAM', 'AGE', 'DATE OF BIRTH', 'COUNTRY', 'WEIGHT', 'HEIGHT', '
```

데이터 수집 - 연봉 데이터

크롤링 된 csv파일

A1 PLAYER

	A	B	C	D	E	F	G	H	I	J	K	L
1	PLAYER	TEAM	AGE	DATE OF BIRTH	COUNTRY	WEIGHT	HEIGHT	POS	HANDED	CLAUSE	CAP HIT	
2	1. Alex Ovechkin	WSH	23	Sep. 17, 1985	Russia	107	191	LW	Right		$9,538,462	
3	2. Sidney Crosby	PIT	21	Aug. 7, 1987	Canada	91	180	C	Left		$8,700,000	
4	3. Jarom챠r J챴gr	-	36	Feb. 15, 1972	Czech Republic	104	191	RW	Left		$7,920,000	
5	4. Brad Richards	-	28	02-May-80	Canada	90	183	C, LW	Left	NTC	$7,800,000	
6	5. Zdeno Chara	-	31	Mar. 18, 1977	Slovakia	113	206	LD	Left	NTC	$7,500,000	
7	6. Dany Heatley	-	27	Jan. 21, 1981	Germany	96	191	LW	Left	NMC	$7,500,000	
8	7. Nicklas Lidstr철m	-	38	Apr. 28, 1970	Sweden	87	185	D	Left	NMC	$7,450,000	
9	8. Mari챴n Hossa	-	29	Jan. 12, 1979	Slovakia	94	185	RW	Left		$7,450,000	
10	9. Scott Gomez	-	28	Dec. 23, 1979	United States	90	180	C	Left		$7,357,143	
11	10. Joe Thornton	-	29	Jul. 2, 1979	Canada	100	193	C, LW	Left	NTC	$7,200,000	
12	11. Brian Campbell	-	29	23-May-79	Canada	87	178	LD	Left	M-NTC	$7,142,875	
13	12. Thomas Vanek	-	24	Jan. 19, 1984	Austria	97	188	LW, RW	Right		$7,142,857	
14	13. Chris Drury	-	32	Aug. 20, 1976	United States	86	178	C	Right		$7,050,000	
15	14. Jarome Iginla	-	31	Jul. 1, 1977	Canada	95	185	RW	Right	NMC	$7,000,000	
16	15. Jason Spezza	-	25	Jun. 13, 1983	Canada	98	191	C, RW	Right		$7,000,000	
17	16. Vincent Lecavalier	-	28	Apr. 21, 1980	Canada	98	193	C, LW	Left		$6,875,000	
18	17. Henrik Lundqvist	-	26	Mar. 2, 1982	Sweden	83	185	G	Left		$6,875,000	
19	18. Nikolai Khabibulin	-	35	Jan. 13, 1973	Russia	96	185	G	Left		$6,750,000	

데이터 수집 - 시즌별 샐러리캡 데이터

크롤링할 사이트 사진

SALARY CAP

Last Updated: Jun. 25, 2023

CBA FAQ | BUYOUT FAQ | LTIR FAQ | WAIVERS FAQ | RESERVE LIST FAQ | OFFER SHEET FAQ | ELC COMPENSATION

SEASON	CONFIRMED	% CHANGE	UPPER LIMIT	LOWER LIMIT	MIN. SALARY
2025-26	NHL Estimate	5.14%	$92,000,000	$68,000,000	$775,000
2024-25	NHL Estimate	4.79%	$87,500,000	$64,700,000	$775,000
2023-24	**Jun. 23, 2023**	**1.21%**	**$83,500,000**	**$61,700,000**	**$775,000**
2022-23	Mar. 29, 2022	1.23%	$82,500,000	$61,000,000	$750,000
2021-22	Jul. 1, 2021	0.00%	$81,500,000	$60,200,000	$750,000
2020-21	Jul. 10, 2020	0.00%	$81,500,000	$60,200,000	$700,000
2019-20	Jun. 22, 2019	2.52%	$81,500,000	$60,200,000	$700,000
2018-19	Jun. 21, 2018	6.00%	$79,500,000	$58,800,000	$650,000
2017-18	Jun. 18, 2017	2.74%	$75,000,000	$55,400,000	$650,000
2016-17	Jun. 21, 2016	2.24%	$73,000,000	$54,000,000	$575,000
2015-16	Jun. 23, 2015	3.48%	$71,400,000	$52,800,000	$575,000
2014-15	Jun. 27, 2014	7.31%	$69,000,000	$51,000,000	$525,000
2013-14	Jan. 6, 2013	7.17%	$64,300,000	$47,500,000	$525,000
2012-13	Jun. 28, 2012	-6.69%	$60,000,000	$44,000,000	$525,000
2011-12	Jun. 23, 2011	8.25%	$64,300,000	$48,300,000	$525,000
2010-11	Jun. 24, 2010	4.58%	$59,400,000	$43,400,000	$500,000
2009-10	Jun. 26, 2009	0.18%	$56,800,000	$40,800,000	$500,000

데이터 수집 - 시즌별 샐러리캡 데이터

시즌별 샐러리캡 데이터 크롤링 - 전체 코드

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import pandas as pd

url = "https://www.capfriendly.com/salary-cap"

html = urlopen(url)
bs_obj = BeautifulSoup(html, "html.parser")
result = []

thead_tag = bs_obj.find("thead")
th = thead_tag.find_all("th")

header = []
for i in range(6):
  header.append(th[i].text)

tbody_tag = bs_obj.find("tbody")
for tr in tbody_tag:
  row = []
  td = tr.find_all("td")
  for i in range(6):
    row.append(td[i].text)
  result.append(row)

df = pd.DataFrame(result)
df.to_csv('season_salarycap_data.csv', header=header, index=False)
```

데이터 수집 - 시즌별 샐러리캡 데이터

크롤링 된 csv 파일

	A	B	C	D	E	F
1	SEASON	CONFIRMED	% CHANGE	UPPER LIMIT	LOWER LIMIT	MIN. SALARY
2	2025-26	NHL Estimate	5.14%	$92,000,000	$68,000,000	$775,000
3	2024-25	NHL Estimate	4.79%	$87,500,000	$64,700,000	$775,000
4	2023-24	Jun. 23, 2023	1.21%	$83,500,000	$61,700,000	$775,000
5	2022-23	Mar. 29, 2022	1.23%	$82,500,000	$61,000,000	$750,000
6	2021-22	Jul. 1, 2021	0.00%	$81,500,000	$60,200,000	$750,000
7	2020-21	Jul. 10, 2020	0.00%	$81,500,000	$60,200,000	$700,000
8	2019-20	Jun. 22, 2019	2.52%	$81,500,000	$60,200,000	$700,000
9	2018-19	Jun. 21, 2018	6.00%	$79,500,000	$58,800,000	$650,000
10	2017-18	Jun. 18, 2017	2.74%	$75,000,000	$55,400,000	$650,000
11	2016-17	Jun. 21, 2016	2.24%	$73,000,000	$54,000,000	$575,000
12	2015-16	Jun. 23, 2015	3.48%	$71,400,000	$52,800,000	$575,000
13	2014-15	Jun. 27, 2014	7.31%	$69,000,000	$51,000,000	$525,000
14	2013-14	Jan. 6, 2013	7.17%	$64,300,000	$47,500,000	$525,000
15	2012-13	Jun. 28, 2012	-6.69%	$60,000,000	$44,000,000	$525,000
16	Dec-11	Jun. 23, 2011	8.25%	$64,300,000	$48,300,000	$525,000
17	Nov-10	Jun. 24, 2010	4.58%	$59,400,000	$43,400,000	$500,000
18	Oct-09	Jun. 26, 2009	0.18%	$56,800,000	$40,800,000	$500,000
19	Sep-08	Jun. 26, 2008	12.72%	$56,700,000	$40,700,000	$475,000
20	Aug-07	Jun. 29, 2007	14.32%	$50,300,000	$34,300,000	$475,000
21	Jul-06	Jun. 27, 2006	12.82%	$44,000,000	$28,000,000	$450,000

스탯데이터와 연봉데이터 병합

1. 시즌별 샐러리캡 데이터 전처리

1-1. 파일 읽고, 쓸모없는 행 삭제

```
import pandas as pd

filename1 = '/content/season_salarycap_data.csv'

cap_df = pd.read_csv(filename1, header=0)

cap_df.drop(['CONFIRMED', '% CHANGE', 'LOWER LIMIT '], axis=1, inplace=True)
```

1-2. SEASON 행 슬라이싱, SALARY 행 '$' 기호 빼기

```
for i in range(0, len(cap_df)):
# '2012-13'식으로 되어있는 SEASON 행의데이터를 '2012'로 슬라이싱
  cap_df.loc[i, 'SEASON'] = cap_df.loc[i, 'SEASON'][:4]
  temp = []
  for n in cap_df.loc[i, 'UPPER LIMIT ']:
# isdigit 함수를 이용하여 '$' 기호를 제거
    if n.isdigit():
      temp.append(n)
  cap_df.loc[i, 'UPPER LIMIT '] = ''.join(temp)
```

1. 시즌별 샐러리캡 데이터 전처리

1-3. 각 시즌의 샐러리캡을 2022년 기준으로 조정하는 'exchange rate' 칼럼 추가

```
for i in range(0, len(cap_df)):
  # cap_df.loc[3, ] -> 2022시즌
  cap_df.loc[i, 'exchange rate'] = int(cap_df.loc[3, 'UPPER LIMIT ']) / int(cap_df.loc[i, 'UPPER LIMIT '])
```

1-4. 1-2에서 슬라이싱한 SEASON 행을 인덱스로 설정

```
cap_df.set_index('SEASON', inplace=True)
```

2. 스탯 데이터 전처리

2-1. 병합의 기준이 되는 'PLAYER'행의 이름 변경. 그 외 행 이름 변경

```
stat_df.rename(columns={'Player':'PLAYER', 'PP':'PPG', 'PP.1':'PPA', 'SH':'SHG', 'SH.1': 'SHA'}, inplace=True)
```

2-2. 불필요한 행 제거

```
stat_df.drop(['Rk', 'Age', 'Tm', 'PIM', '+/-', 'FO%', '-9999'], axis=1, inplace=True)
```

3. 연봉 데이터 전처리

3-1. 병합의 기준이 되는 'PLAYER'행 데이터 처리.

```
# Salary 데이터 Player 이름의 번호 제거
#예시) 121. Steve Jobs    ->    Steve Jobs
slice_name = sal_df.loc[i, 'PLAYER'].split(' ')
sal_df.loc[i, 'PLAYER'] = slice_name[1] + ' ' + slice_name[2]
```

3-2. 연봉값의 '$' 기호 제거

```
cap  = []
    for n in sal_df.loc[i, 'CAP HIT']:
      if n.isdigit():
        cap.append(n)
    sal_df.loc[i, 'CAP HIT'] = ''.join(cap)
```

3-3. 불필요한 행 제거

```
sal_df.drop(['TEAM', 'AGE', 'DATE OF BIRTH', 'COUNTRY', 'WEIGHT', 'HEIGHT', 'POS', 'HANDED', 'CLAUSE'], axis=1, inplace=True)
```

3. 연봉 데이터 전처리

3-1. 병합의 기준이 되는 'PLAYER'행 데이터 처리.

```
# Salary 데이터 Player 이름의 번호 제거
#예시) 121. Steve Jobs    ->    Steve Jobs
slice_name = sal_df.loc[i, 'PLAYER'].split(' ')
sal_df.loc[i, 'PLAYER'] = slice_name[1] + ' ' + slice_name[2]
```

3-2. 연봉값의 '$' 기호 제거

```
cap  = []
    for n in sal_df.loc[i, 'CAP HIT']:
      if n.isdigit():
        cap.append(n)
    sal_df.loc[i, 'CAP HIT'] = ''.join(cap)
```

3-3. 불필요한 행 제거

```
sal_df.drop(['TEAM', 'AGE', 'DATE OF BIRTH', 'COUNTRY', 'WEIGHT', 'HEIGHT', 'POS', 'HANDED', 'CLAUSE'], axis=1, inplace=True)
```

4. 스탯데이터 연봉데이터 병합

4-1. 반복문을 돌며 시즌별 데이터 읽기

```
for year in range (2005, 2023):
  filename2 = '/content/' + str(year) + 'stat.csv'
  filename3 = '/content/' + str(year) + 'sal.csv'

  stat_df = pd.read_csv(filename2, header=0)
  sal_df = pd.read_csv(filename3, header=0)
```

4-2. 데이터 병합

```
# 유일하게 일치하는 PLAYER 행을 기준으로 병합
merge_df = pd.merge(stat_df, sal_df, how='inner')
```

4. 스탯데이터 연봉데이터 병합

4-3. 1-3에서 추가한 'exchange rate' 행의 데이터를 이용하여 선수들의 연봉을 2022 시즌에 맞게 조정

```
for i in range(0, len(merge_df)):
    merge_df.loc[i, 'exchange cap'] = int((int(merge_df.loc[i, 'CAP HIT']) - int(cap_df.loc[str(year), 'MIN. SALARY ])) * cap_
                                                                                                    df.loc[str(year), 'exchange rate'])
```

4-4. 시즌별로 csv파일로 저장

```
merge_df.to_csv(str(year) + "_merge.csv")
```

1. 라이브러리 추가

```
import pandas as pd
import tensorflow as tf
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.model_selection import train_test_split
```

2. feature로 사용할 행 선택. 05~22시즌 데이터를 통합 저장할 X_train_pre, y 선언

```
cols_train = ['Pos', 'GP', 'G', 'A', 'PTS', 'PS', 'EV', 'PPG', 'SHG', 'GW', 'EVA', 'PPA', 'SHA', 'S', 'TOI']
X_train_pre = pd.DataFrame()
y = []
```

3. 05~22시즌 데이터 읽기 및 처리

3-1. 읽기

```
for year in range(2005, 2023):
  filename = '/content/' + str(year) + '_merge.csv'

  df = pd.read_csv(filename, header=0)
```

3-2. 처리

```
  for i in range(0, len(df)):
    # C(Center), RW(Right Wing), LW(Left Wing)은 모두 공격수이므로 0으로 변경
    if df.loc[i, 'Pos'] in ['C', 'RW', 'LW']:
      df.loc[i, 'Pos'] = 0
    else:
    # 수비수의 경우 1로 변경
      df.loc[i, 'Pos'] = 1
    # 출전경기수가 20경기 미만인 선수 데이터는 학습에서 제외시킴
    if df.loc[i, 'exchange cap'] < 0 or df.loc[i, 'GP'] < 20:
      df.drop(i, inplace=True)
```

4. 머신러닝 진행

4-1. 학습셋과 테스트셋 8:2로 설정

```
X_train, X_test, y_train, y_test = train_test_split(X_train_pre, y, test_size=0.2)
```

4-2. 신경망 모델 생성

```
model = Sequential();
model.add(Dense(15, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(60, activation='relu'))
model.add(Dense(120, activation='relu'))
model.add(Dense(190, activation='relu'))
model.add(Dense(40, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

4-3. 얼리스탑핑, 검증셋 설정 후 학습 진행

```
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=20)
modelpath = "./data/model/Salary.hdf5"
checkpointer = ModelCheckpoint(filepath=modelpath, monitor="val_loss", verbose=0, save_best_only=True)
history = model.fit(X_train, y_train, validation_split=0.25, epochs=2000, batch_size=32, callbacks=[early_stopping_callback, ch
```

```
Epoch 488/2000
179/179 [==============================] - 1s 4ms/step - loss: 3363939811328.0000 - val_loss: 3465402646528.0000
Epoch 489/2000
179/179 [==============================] - 1s 4ms/step - loss: 3379070500864.0000 - val_loss: 3437839515648.0000
Epoch 490/2000
179/179 [==============================] - 1s 4ms/step - loss: 3430201688064.0000 - val_loss: 3512625004544.0000
Epoch 491/2000
179/179 [==============================] - 1s 3ms/step - loss: 3395020914688.0000 - val_loss: 3552404832256.0000
Epoch 492/2000
179/179 [==============================] - 1s 4ms/step - loss: 3376854597632.0000 - val_loss: 3490428747776.0000
Epoch 493/2000
179/179 [==============================] - 1s 4ms/step - loss: 3371213520896.0000 - val_loss: 3441027186688.0000
```

5. 생성된 모델 테스트

5-1. 테스트 코드

```
from keras.models import load_model
model = load_model('/content/data/model/Salary.hdf5')

real_sal = []
pred_sal = []
Y_prediction = model.predict(X_test).flatten()
for i in range(25):
  real = y_test[i]
  prediction = Y_prediction[i]
  print("실제 연봉: {:.2f}, 예상연봉: {:.2f}".format(real, prediction))
  real_sal.append(real)
  pred_sal.append(prediction)
```

5-2. 테스트 결과

```
60/60 [==============================] - 0s 3ms/step
실제 연봉: 183051.00, 예상연봉: 3866395.25
실제 연봉: 3668592.00, 예상연봉: 2737636.25
실제 연봉: 5446742.00, 예상연봉: 4629492.50
실제 연봉: 10312499.00, 예상연봉: 5005505.00
실제 연봉: 342146.00, 예상연봉: 679914.25
실제 연봉: 338052.00, 예상연봉: 473801.62
실제 연봉: 4479166.00, 예상연봉: 2980930.50
실제 연봉: 180555.00, 예상연봉: 3244295.25
실제 연봉: 683282.00, 예상연봉: 398225.09
실제 연봉: 3520000.00, 예상연봉: 1248546.12
실제 연봉: 3163343.00, 예상연봉: 2041945.12
실제 연봉: 395547.00, 예상연봉: 1089763.88
실제 연봉: 544674.00, 예상연봉: 654013.62
실제 연봉: 302500.00, 예상연봉: 800722.38
실제 연봉: 9920245.00, 예상연봉: 5006274.50
실제 연봉: 4779354.00, 예상연봉: 4402763.50
실제 연봉: 14733865.00, 예상연봉: 5574693.50
실제 연봉: 348591.00, 예상연봉: 294595.59
실제 연봉: 156750.00, 예상연봉: 342213.03
실제 연봉: 3592534.00, 예상연봉: 4767589.50
실제 연봉: 404411.00, 예상연봉: 5553312.50
실제 연봉: 50613.00, 예상연봉: 354191.66
실제 연봉: 2350000.00, 예상연봉: 308396.31
실제 연봉: 75000.00, 예상연봉: 2444618.50
실제 연봉: 3957457.00, 예상연봉: 4324328.50
```

퍼포먼스 점수 계산

1. 사전 준비

1-1. 라이브러리 추가

```
from keras.models import load_model
import pandas as pd
import numpy as np
```

1-2. 생성된 모델 불러오기

```
model = load_model('/content/Salary2.hdf5')
```

1-3. 학습에 사용되었던 행 저장

```
cols_train = ['Pos', 'GP', 'G', 'A', 'PTS', 'PS', 'EV', 'PPG', 'SHG', 'GW', 'EVA', 'PPA', 'SHA', 'S', 'TOI']
```

1-4. 퍼포먼스 점수를 계산해볼 22시즌 병합 데이터 불러오기

```
df = pd.read_csv('/content/2022_merge.csv')
```

1-5. 불러온 병합데이터 모델에 넣고 돌리기 전 전처리

```
for i in range(0, len(df)):
    if df.loc[i, 'Pos'] in ['C', 'RW', 'LW']:
      df.loc[i, 'Pos'] = 0
    else:
      df.loc[i, 'Pos'] = 1
    if df.loc[i, 'exchange cap'] <= 0:
      df.drop(i, inplace=True)

df['Pos'] = np.asarray(df['Pos']).astype(np.int64)
df = df.reset_index()
```

2. 퍼포먼스 점수 계산

2-1. 선수 스탯을 모델에 입력하여 출력된 예상연봉을 'predict salary' 행에 추가

```
ndf = df.loc[:, cols_train]
df['predict salary'] = model.predict(ndf)
```

2-2. 예상연봉을 실제연봉으로 나누어 'performance score' 행에 추가

(performance score가 1이면 정확히 받은만큼 하는 선수, 1보다 작으면 돈 값 못하는 선수, 1보다 크면 가성비 좋은 선수)

```
df['performance score'] = df['predict salary'] / df['exchange cap']
```

2-3. 저장 및 계산 결과

```
24/24 [==============================] - 0s 2ms/step
   index  Unnamed: 0           PLAYER  Pos  GP   G   A  PTS    PS  EV  ...  ₩
0      0           0   Connor McDavid    0  82  64  89  153  18.2  39  ...
1      1           1    Leon Draisaitl    0  80  52  76  128  14.2  19  ...
2      2           2   Nikita Kucherov    0  82  30  83  113  10.7  22  ...
3      3           3  Nathan MacKinnon    0  71  42  69  111  13.1  30  ...
4      4           4   Jason Robertson    0  82  46  63  109  13.9  33  ...

    ATOI  BLK  HIT  FOW  FOL    FO%   CAP HIT  exchange cap  predict salary  ₩
0  22:23   40   89  525  486   51.9  12500000    11750000.0      11561605.0
1  21:44   40   66  807  663   54.9   8500000     7750000.0      10208779.0
2  20:08   28   61    2    0  100.0   9500000     8750000.0       9927170.0
3  22:19   40   53  519  649   44.4   6300000     5550000.0      11472997.0
4  18:50   19   57    0    1    0.0   7750000     7000000.0       7366381.5

  performance score
0          0.983966
1          1.317262
2          1.134534
3          2.067207
4          1.052340

[5 rows x 29 columns]
```

Top 10 Overpaid Centers in the NHL in 2022-23

March 13, 2023 by Rob Couch

출처: https://thehockeywriters.com/nhl-top-10-overpaid-centers-2022-23/

가장 과분한 연봉을 받는 10명의 선수의 퍼포먼스 점수는 어떻게 나올까?

퍼포먼스 점수 계산

1. Jonathan Toews ($10.5 million AAV)

Jonathan Toews is in a very similar position to Backstrom, except the Chicago Blackhawks are in full rebuild mode, and his contract is over at the end of the season. Fortunately, the Blackhawks aren't competing for anything except the first overall pick, as this is a horrible contract at this point, and Toews' health issues prevented the team fom trading him for assets before the deadline (from "Chicago Blackhawks Star Jonathan Toews Is Stepping Away From Hockey, Says He's Dealing With Long Covid", *Forbes*, Feb. 19, 2023).

```
calc_df.set_index('PLAYER', inplace=True)
print(calc_df.loc['Jonathan Toews', 'performance score'])
```

```
0.3617612564102564
```

2. Nicklas Backstrom ($9.2 million AAV)

Nicklas Backstrom has had a tough road battling through potential career-ending injuries to return to action this season. He has played only 24 games but has struggled to produce as he has throughout his career playing in the Capitals' top-six. His four goals and 12 points aren't a big enough contribution, as he has averaged over a point-per-game five times in his career and has 1023 points in 1084 games.

As his health and age have caught up to the 35-year-old, his $9.2 million AAV cap hit for two more years doesn't look great, especially when the Capitals are trying to stay competitive. If he can't return to averaging a point per game, those final couple of seasons will be rough. Backstrom is a respected veteran who has done a ton for the Capitals, so a contract of this size for a player past the age of 30 wasn't going to end well. But his struggles over the past two seasons shouldn't damage his legacy.

```
print(calc_df.loc['Nicklas Bäckström', 'performance score'])
```

```
0.4388629881656805
```

3. Jonathan Drouin ($5.5 million AAV)

Jonathan Drouin is a new addition to this list and lands right near the top, as his offense has been horrible this season. Fortunately, he is a free agent after the season when the Montreal Canadiens can move on from him. He is the result of one of the worst trades in franchise history when the Canadiens were hoping to trade for a top-line center, but Drouin has battled injuries throughout his tenure with the team, and his play has majorly suffered because of it. In return for Drouin, the Tampa Bay Lightning got the now two-time Stanley Cup champion and elite defenceman Mikhail Sergachev, who could have changed the script in Montreal.

```
print(calc_df.loc['Jonathan Drouin', 'performance score'])
```

```
0.4165814473684211
```

1. 사전 준비

1-1. 라이브러리 추가

```
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import numpy as np
```

1-2. 시각화할 데이터 불러오기

```
calc_df = pd.read_csv('/content/2022_calc.csv')
# stat 데이터에 있는 '팀이름' 칼럼을 사용하기 위해 불러옴.
stat_df = pd.read_csv('/content/2022stat.csv')
```

2. 데이터 시각화1 - Performance Score를 다섯 구간으로 분류한 뒤 카운트

2-1. 퍼포먼스 점수를 이용하여 선수들을 평가한 'judge'행 추가.

```
# 2 이상: Very Good
# 1.2 ~ 2.0 : Good
# 0.8 ~ 1.2 : So So
# 0.5 ~ 0.8 : Bad
# 0.5 미만 : Very Bad
for i in range(0, len(merge_df)):
  if not np.isfinite(merge_df.loc[i, 'performance score']):
    merge_df.drop(i, inplace=True)
  elif merge_df.loc[i, 'performance score'] >= 2:
    merge_df.loc[i, 'judge'] = 'Very Good'
  elif merge_df.loc[i, 'performance score'] >= 1.2:
    merge_df.loc[i, 'judge'] = 'Good'
  elif merge_df.loc[i, 'performance score'] >= 0.8:
    merge_df.loc[i, 'judge'] = 'So So'
  elif merge_df.loc[i, 'performance score'] >= 0.5:
    merge_df.loc[i, 'judge'] = 'Bad'
  else:
    merge_df.loc[i, 'judge'] = 'Very Bad'
```

2-2. 시각화용 데이터프레임 생성

```
# '팀이름'을 인덱스로하고, judge 행의 domain 을 행으로 하는 시각화용 데이터프레임 생성
visual_df1 = pd.DataFrame(0,
                          index=merge_df['Tm'].unique(),
                          columns=['Very Bad', 'Bad', 'So So', 'Good', 'Very Good'])
```

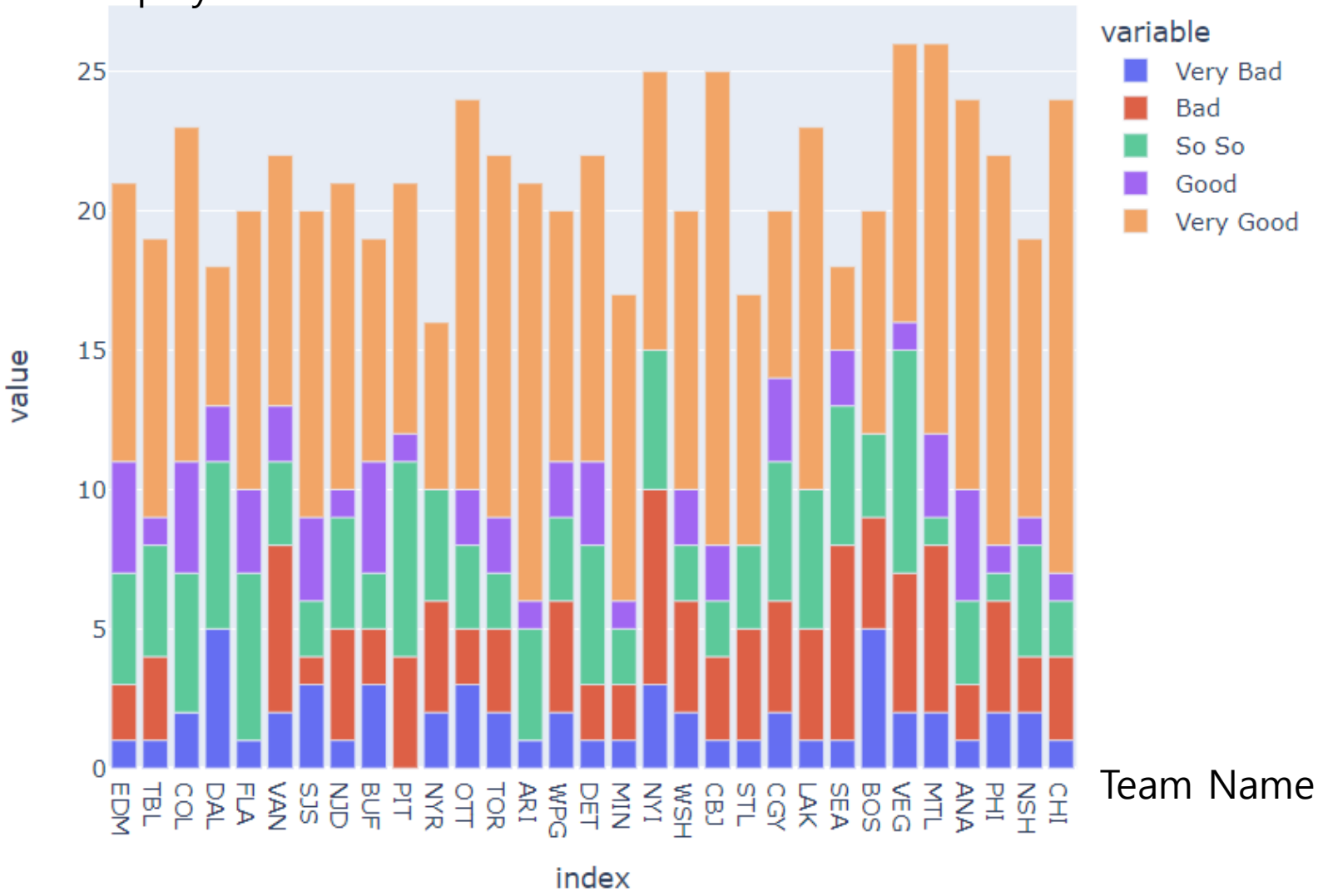
2-3. 팀이름을 기준으로 그룹핑하여 팀별로 ['Very Bad', 'Bad', 'So So', 'Good', 'Very Good']이 각각 몇 명인지 계산

```
grouped1 = merge_df.groupby('Tm')
for key, group in grouped1:
  for s in group['judge']:
    visual_df1.loc[key, s] += 1
```

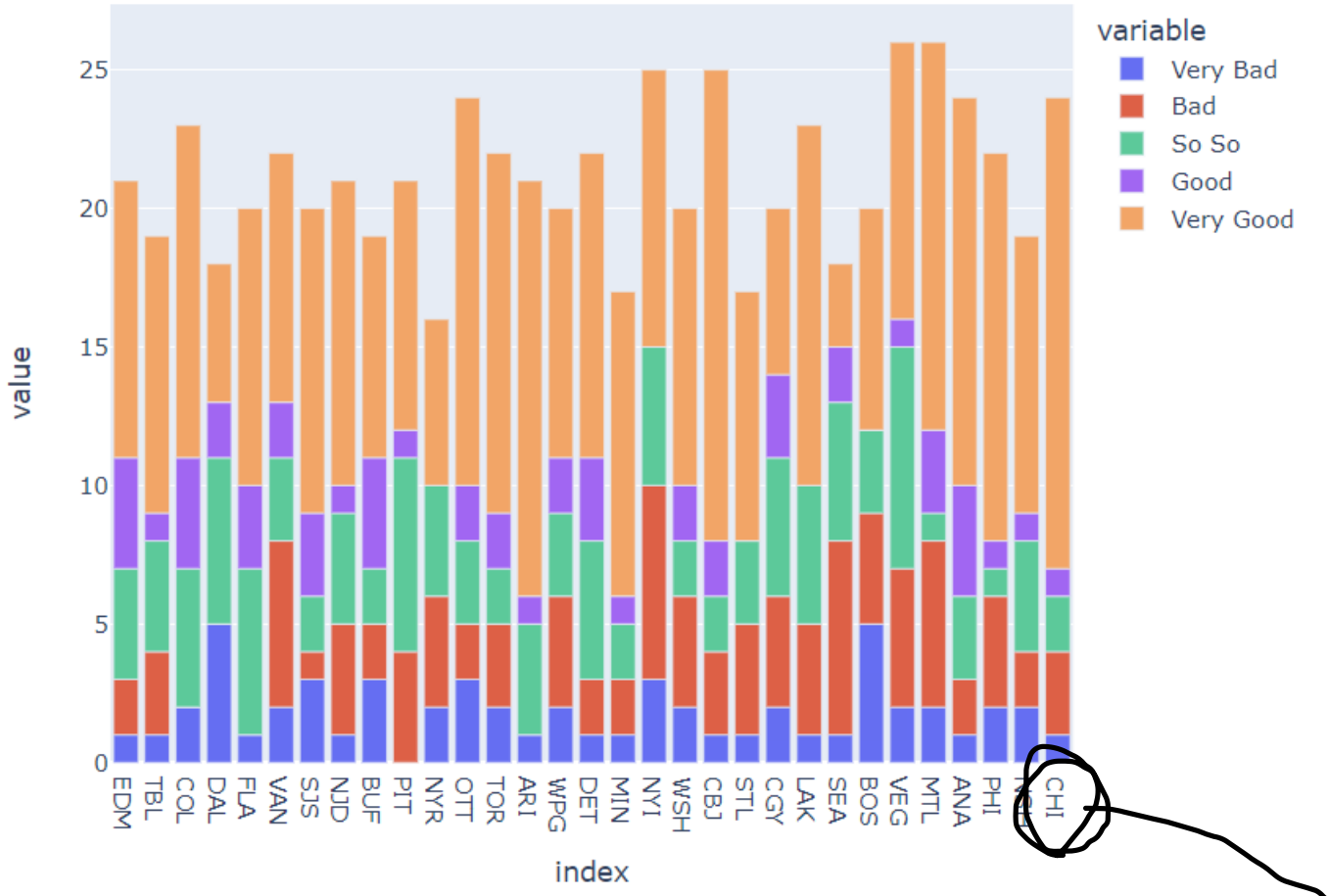
2-4. 시각화 결과

```
fig = px.bar(visual_df1, x=visual_df1.index, y=visual_df1.columns)
fig.update_layout(width=700)
```

Number of players



Team Name



Western

Central

Rank		Team	GP	W	L	OT	PTS
1	y	Colorado Avalanche	82	51	24	7	109
2	x	Dallas Stars	82	47	21	14	108
3	x	Minnesota Wild	82	46	25	11	103
4	x	Winnipeg Jets	82	46	33	3	95
5		Nashville Predators	82	42	32	8	92
6		St. Louis Blues	82	37	38	7	81
7		Arizona Coyotes	82	28	40	14	70
8		Chicago Blackhawks	82	26	49	7	59

3. 데이터 시각화2 - 각 팀 총 연봉내 비중을 반영

3-1. 선수별로 본인의 연봉이 팀내 총 연봉의 몇 %인지 나타내는'weight' 행 추가

```
cap_sum_df = merge_df.groupby('Tm')['exchange cap'].sum()
for i in range(0, len(merge_df)):
  merge_df.loc[i, 'weight'] = merge_df.loc[i, 'exchange cap'] / cap_sum_df[merge_df.loc[i, 'Tm']]
```

3-2. 시각화용 데이터프레임 생성

```
visual_df2 = pd.DataFrame(0,
                          index=merge_df['Tm'].unique(),
                          columns=['Very Bad', 'Bad', 'So So', 'Good', 'Very Good'])
```

3. 데이터 시각화2 - 각 팀 총 연봉내 비중을 반영

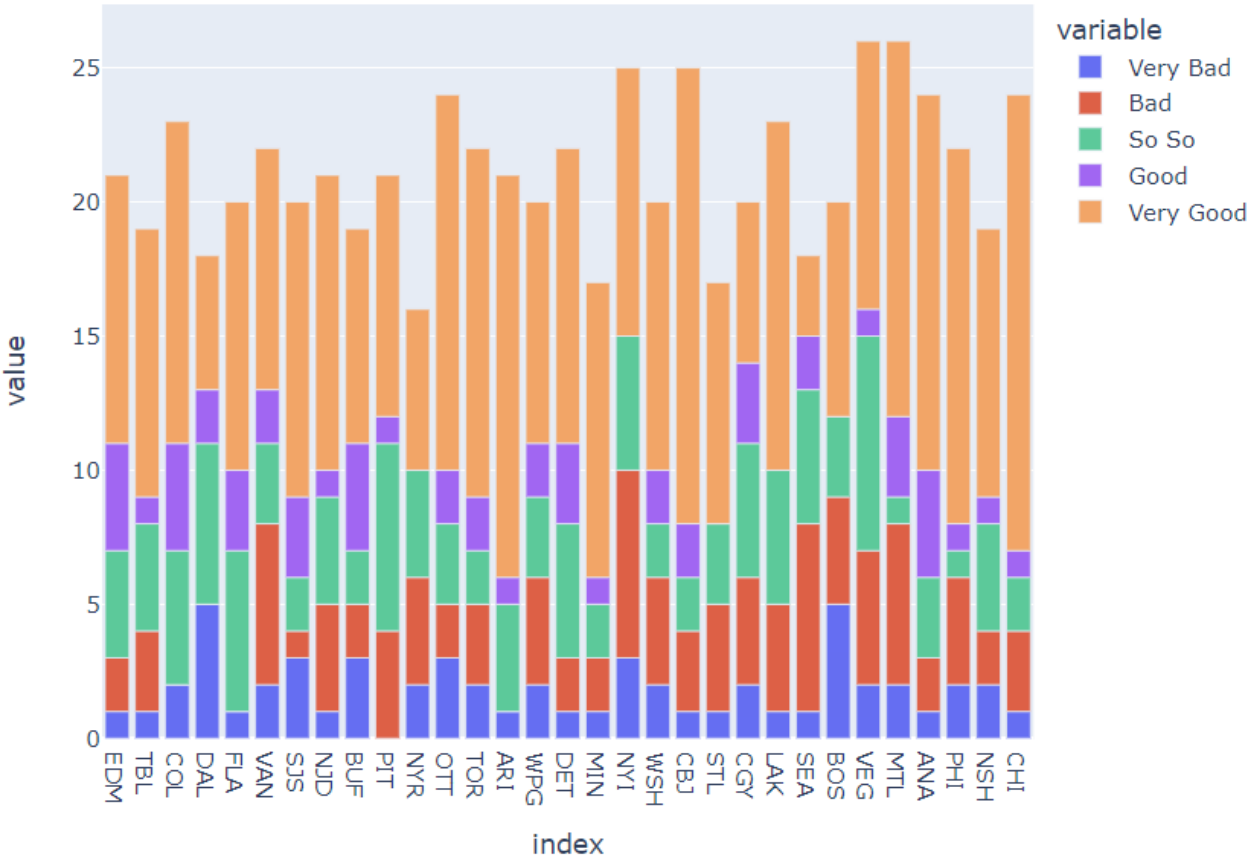
3-3. 팀이름과 judge를 기준으로 그룹핑하고, weight를 더한다.

```
weight_sum_df = merge_df.groupby(['Tm', 'judge'])['weight'].sum()
for t in visual_df2.index:
  for w in visual_df2.columns:
    try:
      visual_df2.loc[t, w] = weight_sum_df[t, w]
    except:
      continue
```

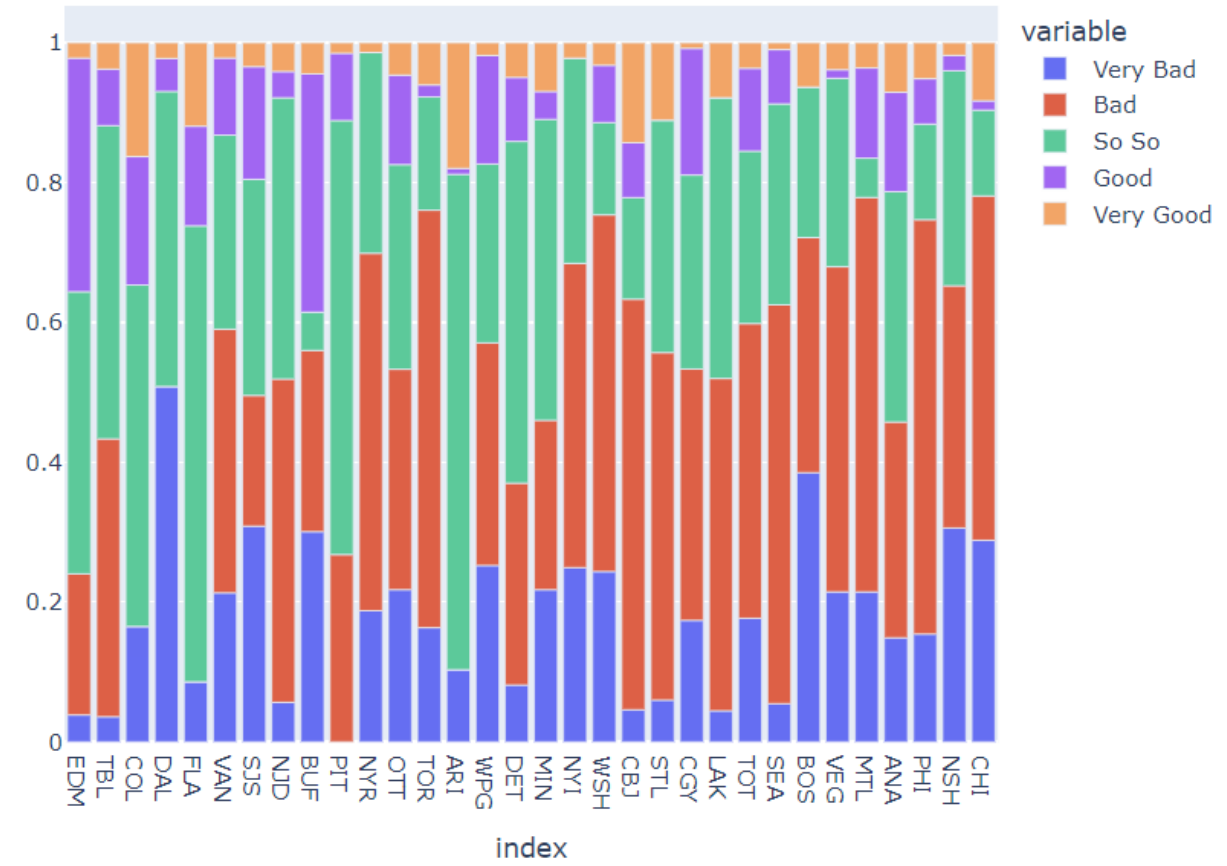
3-4. 시각화 결과

```
fig2 = px.bar(visual_df2, x=visual_df2.index, y=visual_df2.columns)
fig2.update_layout(width=700)
```

단순히 퍼포먼스 구간 별
선수들의 숫자만 센 것



퍼포먼스 구간 별 연봉이 팀 내 총 연봉에서
차지하는 비율을 나타낸 것



연봉 1억인 선수 10명의
performance score가 2.0
-> +10억

연봉 100억인 선수 1명의
performance score가 0.5
-> -50억

프로젝트 기획 단계에서 예상한 결과는

1. 팬들 사이에서 평가가 좋지 못한 선수는 퍼포먼스 점수가 낮게 나올 것이다.
2. 성적이 좋은 팀에 가성비가 좋은 선수가 많을 것이다.

이 2가지였다.

첫 번째는 예상대로였지만 두 번째는 그렇지 않았다.

생각해보니 단순히 선수의 숫자가 중요한 것이 아니라,
팀의 한정된 샐러리 자원 중에서 가성비 좋게 쓰인 자원의 비율이 중요한 것이었다.

두 번째 예상은 성적이 좋은 팀은 가성비가 좋은 선수에게 쓴 자원이 많다는 결과로 바뀌었다.

감사합니다.