

블랙잭 카드카운팅의 수익률 상승 분석

10729 정준영

내용

I. 탐구 동기.....	2
II. 블랙잭의 규칙	3
III. 카드카운팅	4
IV. 블랙잭 게임 제작	5
1. 파이썬 코드	5
2. 게임 플레이	6
V. 데이터 분석	8
1. 데이터 입력	9
2. 데이터 필터링	10
1) 카운팅 여부 필터	10
2) 수익 발생 필터링	11
3. 결과값	12
4. 결과 분석	13
VI. 결론 및 향후계획	14

I. 탐구 동기

수학 시간에 선생님께서 '경우의 수' 단원에 들어가기 앞서 영화 <21>의 리뷰 영상을 보여주셨다. 영화 <21>¹은 2008년에 개봉한 영화로 실화 기반 소설 벤 메즈리치의 소설 <MIT 수학 천재들의 카지노 무너뜨리기(Bringing Down the House)>을 기반으로 만들어졌다. 주요 내용은 MIT의 학생들과 교수 '미키'가 블랙잭 팀을 만들어 '카드 카운팅'기술을 배운 뒤 이를 이용하여 라스베가스에서 큰 돈을 벌었다.

이 영화의 리뷰를 본 후 나는 블랙잭과 카드카운팅에 관심을 가지게 되었고, 진로이자 관심분야인 코딩을 결합하여 직접 블랙잭 게임을 만들어보고 카드카운팅을 사용함으로써 상승하는 수익률을 분석하고 싶어졌다.

¹ [<21>, 네이버 영화](#)

II. 블랙잭의 규칙

블랙잭(Black Jack)은 카드의 합이 21 또는 21에 가까운 사람이 이기는 게임이다. 카드는 조커를 제외한 52장을 사용한다.²

우선 플레이어는 베팅을 하며 이후 딜러가 카드를 각각 2장씩 돌린다 (이때 딜러의 두번째 카드는 뒤집어서 보이지 않게 둔다). 이후 플레이어는 카드를 한 장 더 받는 '히트(HIT)', 더 이상 카드를 받지 않고 턴을 넘기는 '스텐드(Stand)', 똑 같은 카드 두장을 분리하여 따로 플레이하는 '스플릿(Split)', 베팅액을 2배로 높이는 동시에 히트를 하는 '더블 다운(Double Down)', 게임을 포기하는 '서렌더(Surrender)' 등의 동작 중 하나를 택하여 게임을 진행한다. 스텐드를 택할 시 다음 플레이어 또는 딜러에게 턴이 넘어가며, 히트와 스텐드를 제외한 동작은 처음 카드 두장을 받고 난 직후(어떠한 액션도 취하지 않았을 때)만 가능하다.

카드의 숫자는 J, Q, K는 10으로, A는 1 또는 11로 그 외의 카드는 그 카드의 숫자로 계산된다. A, J, Q, K를 그림카드라 하며 처음 받은 두장이 그림카드로만 구성되어 21을 만들었다면 '블랙잭'으로 블랙잭이 아닌 상황의 21보다 더 높은 가치를 지닌다.

만약 카드합이 21을 초과할 경우 '버스트(Bust)'로 그 즉시 게임에서 패배한다. 딜러와 플레이어가 동점일 경우 '푸시(Push)'로 무승부가 되며 베팅액 100%를 돌려받는다. 서렌더를 한 상황에서는 베팅액의 50%만 받

² [블랙 잭, 두산백과](#)

고 게임을 끝낸다.

돈은 승리시 베팅액의 200%를, 블랙잭으로 승리시 보편적으로 베팅액의 250%를 받는다.

III.카드카운팅

카드카운팅은 블랙잭에서 사용되는 카드를 모두 카운팅하여 다음에 나올 카드의 숫자를 예측하는 기술이다. 사용되지 않은 덱에 그림카드와 10이 많을수록 플레이어는 21과 블랙잭을 메이드하기 더 쉬워지므로 승률이 올라간다. 이러한 승률이 오르는 상황인 그림카드가 많은가 적은가를 카드카운팅으로 계산하여 베팅을하고 히트/스텐드를 결정한다.

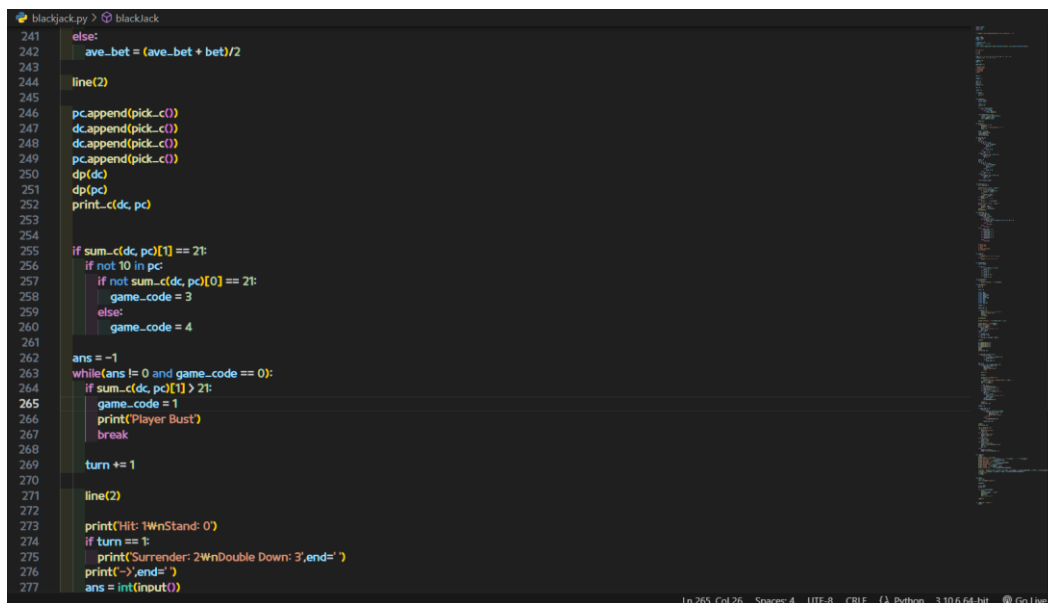
카드카운팅은 기본적으로 2~6이 나오면 +1, 10~ 와 A가 나오면 -1을 한다. 카드카운트가 더 높을수록 다음카드가 그림카드일 확률도 올라간다.

전체 카드 52장 중 그림카드는 20장이다. 이때 그림카드가 나올 확률은 $20/52 \times 100 = 38.46\%$ 이다. 만약 그림카드 4장 그 외 카드 8장이 사용되어 카운트가 4인 상황에서는 전체 카드 40장 중 그림카드가 16장 있으므로 이때 그림카드가 나올 확률은 40%로 약 1.5%p 증가한다. 극단적으로 그림카드 2장 그 외 20장을 사용하여 카운트가 18이 나온 상황에선 그림카드가 $16/30 \times 100 = 53.3\%$ 의 확률로 나온다.

IV.블랙잭 게임 제작

1. 파이썬 코드

블랙잭 게임은 파이썬(Python)언어를 사용하여 만들었다. 스플릿 등의 세부적인 추가물은 제외하고 히트, 스텐드, 더블다운, 서렌더의 4가지 동작만 구현하였다.

A screenshot of a code editor window titled 'blackjack.py' and 'blackjack'. The code is written in Python and implements a blackjack game. It includes logic for calculating average bets, dealing cards, checking for busts, and handling player actions like hit, stand, double down, and surrender. The code is color-coded and includes line numbers from 241 to 277.

```
241 else:
242     ave_bet = (ave_bet + bet)/2
243
244 line(2)
245
246 pc.append(pick_c())
247 dc.append(pick_c())
248 dc.append(pick_c())
249 pc.append(pick_c())
250 dp(dc)
251 dp(pc)
252 print_c(dc, pc)
253
254
255 if sum_c(dc, pc)[1] == 21:
256     if not 10 in pc:
257         if not sum_c(dc, pc)[0] == 21:
258             game_code = 3
259         else:
260             game_code = 4
261
262 ans = -1
263 while(ans != 0 and game_code == 0):
264     if sum_c(dc, pc)[1] > 21:
265         game_code = 1
266         print("Player Bust")
267         break
268     turn += 1
269
270 line(2)
271
272 print("Hit: 1\nStand: 0")
273 if turn == 1:
274     print("Surrender: 2\nDouble Down: 3,end=")
275     print("-> ",end=" ")
276     ans = int(input())
277
```

전체 코드는 [깃허브\(Git Hub\)](#)에 업로드 해두었다.

2. 게임 플레이

```
1 game
counting: 0
leftCard: 41
Money: 5000
Bet Money -> 100
-----
Dealer's Card: 7 ? = ?
Player's Card: K 2 = 12
counting: -0.5
-----
Hit: 1
Stand: 0
Surrender: 2
Double Down: 3 -> 1
-----
Dealer's Card: 7 ? = ?
Player's Card: K 2 9 = 21
counting: 0.5
```

카드덱을 랜덤하게 뽑고 딜러와 플레이어가 1v1으로 플레이한다. 베팅 액을 직접 정할 수 있으며 0~3의 숫자를 입력하여 액션 중 하나를 취할 수 있다.

```

Hit: 1
Stand: 0
-> 0
-----
Dealer's Card: 7 ? = ?
Player's Card: K 2 9 = 21
counting: 0.5
Dealer's Card: 7 8 7 = 22
Player's Card: K 2 9 = 21
counting: 0.0
Dealer Bust
-----
Dealer's Card: 7 8 7 = 22
Player's Card: K 2 9 = 21
counting: 0.0
Player Win

```

게임이 끝나면 누가 승리하였는지 표시가 되며 그에 따라 돈을 받는다.

```

Dealer Win
-----

count: 1
win/lose/all: 5 / 7 / 15
win rate: 33.33333333333333
blackJack: 1
blackjack rate: 6.666666666666667
average bet: 469.140625
money: 5650
income: 650
income rate: 13.0

```

```

1 count,win,lose,all,winrate,blackjack,blackjackrate,averagebet,money,income,incomerate
2 1,11,12,25,44,0,1,4,0.32500,2259016037,127500,122500,2450.0
3 0,11,13,25,44,0,1,4,0.867,1875,1000,-4000,-80.0
4 0,11,14,25,44,0,0,0,0.275,0007152557373,200,-4800,-96.0
5 0,1,6,25,4,0,0,0,0.1000,0,0,-5000,-100.0
6 0,11,12,25,44,0,0,0,0.656,7578315734863,4750,-250,-5.0
7 0,12,11,25,48,0,1,4,0.875,0002682209015,11250,6250,125.0
8 0,13,11,25,52,0,2,8,0.1000,0,8000,3000,60.0
9 0,11,13,25,44,0,0,0,0.500,3971517086029,3400,-1600,-32.0
10 1,11,12,25,44,0,0,0,0.499,9999108314514,3501,-1499,-29.98
11 1,13,10,25,52,0,1,4,0.500,0,6750,1750,35.0
12 1,9,13,25,36,0,0,0,0.576,995849609375,1500,-3500,-70.0
13 0,11,12,25,44,0,1,4,0.500,0,4500,-500,-10.0
14 0,8,8,25,32,0,1,4,0.514,713317155838,3000,-2000,-40.0
15 1,12,13,25,48,0,1,4,0.500,0014305114746,4750,-250,-5.0
16 1,13,5,25,52,0,1,4,0.750,0,6500,1500,30.0
17 1,12,6,25,48,0,0,0,0.1409,9221229553223,16000,11000,220.0
18 1,13,9,25,52,0,1,4,0.500,0,5750,750,15.0
19 1,2,0,3,66,666666666666666,0,0,0.500,0,6000,1000,20.0
20 0,11,12,25,44,0,0,0,0.500,74148178100586,4500,-500,-10.0
21 1,8,9,25,32,0,0,0,0.373,0583190917969,0,-5000,-100.0
22 1,3,9,25,12,0,0,0,0.1000,0,0,-5000,-100.0
23 0,10,11,25,40,0,0,0,0.755,950927734375,2250,-2750,-55.00000000000001
24 1,7,3,15,46,666666666666666,0,0,0.549,1943359375,10250,5250,105.0
25 0,7,6,15,46,666666666666666,3,20,0.549,8046875,5500,500,10.0
26 1,6,3,15,40,0,1,6,666666666666667,208,911328125,11650,6650,133.0
27 1,1,3,15,6,666666666666667,0,0,0,1875,0,0,-5000,-100.0
28 0,7,7,15,46,666666666666664,1,6,666666666666667,625,9765625,5000,0,0.0

```

미리 정해 둔 게임만큼 플레이를 하였다면 승률과 수익률 등 게임 리포트가 작성되며 이는 정보시간 때 배운 파일입출력을 이용하여 텍스트 파일로 저장된다(추후 해당 텍스트 파일로 데이터 분석을 함).

V. 데이터 분석



데이터 분석은 2학기 방과후에서 배운 삼성의 '브라이틱스 스튜디오 (Brightics Studio)'를 사용하였다.

1. 데이터 입력

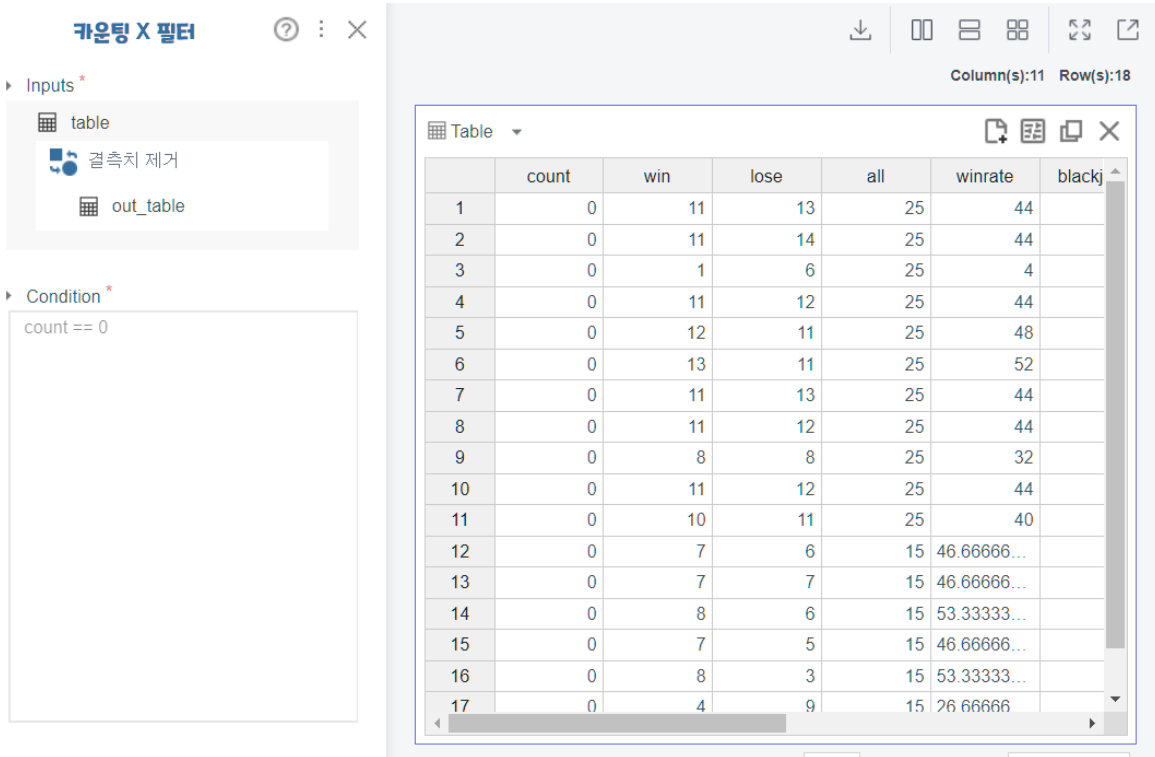
	count	win	lose	all	winrate	blackjack	blackjackrate	averagebet	money	income	incomerate
5	0	11	12	25	44	0	0	656.7578...	4750	-250	-5
6	0	12	11	25	48	1	4	875.0002...	11250	6250	125
7	0	13	11	25	52	2	8	1000	8000	3000	60
8	0	11	13	25	44	0	0	500.3971...	3400	-1600	-32
9	1	11	12	25	44	0	0	499.9999...	3501	-1499	-29.98
10	1	13	10	25	52	1	4	500	6750	1750	35
11	1	9	13	25	36	0	0	576.9958...	1500	-3500	-70
12	0	11	12	25	44	1	4	500	4500	-500	-10
13	0	8	8	25	32	1	4	514.7133...	3000	-2000	-40
14	1	12	13	25	48	1	4	500.0014...	4750	-250	-5
15	1	13	5	25	52	1	4	750	6500	1500	30
16	1	12	6	25	48	0	0	1409.922...	16000	11000	220
17	1	13	9	25	52	1	4	500	5750	750	15
18	1	2	0	3	66.66666...	0	0	500	6000	1000	20
19	0	11	12	25	44	0	0	500.7414...	4500	-500	-10
20	1	8	9	25	32	0	0	373.0583...	0	-5000	-100
21	1	3	9	25	12	0	0	1000	0	-5000	-100
22	0	10	11	25	40	0	0	755.9509...	2250	-2750	-55.0000...

데이터 입력칸에서 앞서 만든 리포트 텍스트 파일을 업로드 하여 이를 사용 가능 하도록 ','를 통해 구분되어있는 데이터를 표로 분리하였다.

데이터는 총 39개이다. 한 세트당 총 15 ~ 25판의 게임을 진행했으며 데이터 39개의 총 플레이 게임 판 수는 783판이다.

2. 데이터 필터링

1) 카운팅 여부 필터



카운팅 X 필터

Inputs *

table

결측치 제거

out_table

Condition *

count == 0

Column(s):11 Row(s):18

	count	win	lose	all	winrate	blackj
1	0	11	13	25	44	
2	0	11	14	25	44	
3	0	1	6	25	4	
4	0	11	12	25	44	
5	0	12	11	25	48	
6	0	13	11	25	52	
7	0	11	13	25	44	
8	0	11	12	25	44	
9	0	8	8	25	32	
10	0	11	12	25	44	
11	0	10	11	25	40	
12	0	7	6	15	46.66666...	
13	0	7	7	15	46.66666...	
14	0	8	6	15	53.33333...	
15	0	7	5	15	46.66666...	
16	0	8	3	15	53.33333...	
17	0	4	9	15	26.66666...	

카운팅 여부를 확인하는 count의 값에 따라 0 과 1 인 경우를 따로 분리하였다.

2) 수익 발생 필터링

유수익 게임 필터 ⓘ ⋮ ✕

▶ Inputs *

table
결측치 제거
out_table

▶ Condition *

incomerate > 0

Column(s):11 Row(s):19

Table

	count	win	lose	all	winrate	blackj
1	1	11	12	25	44	
2	0	12	11	25	48	
3	0	13	11	25	52	
4	1	13	10	25	52	
5	1	13	5	25	52	
6	1	12	6	25	48	
7	1	13	9	25	52	
8	1	2	0	3	66.66666...	
9	1	7	3	15	46.66666...	
10	0	7	6	15	46.66666...	
11	1	6	3	15	40	
12	1	6	4	15	40	
13	0	8	6	15	53.33333...	
14	0	7	5	15	46.66666...	
15	1	6	6	15	40	
16	1	5	7	15	33.33333...	
17	1	7	4	15	46.66666	

수익이 발생한 데이터만 따로 필터링하였다.

3. 결과값

- 카운팅 X

	column_nam	max	min	range	avg	stddev	median
1	winrate	53.3333...	4	49.3333...	41.8518...	11.5527...	44
2	blackjac...	20	0	20	3.18518...	5.05424...	0
3	incomerate	125	-100	225	-13.7777...	57.2523...	-10
4	averagebet	1000	275.000...	724.999...	646.603...	195.553...	625.854...

- 카운팅 O

	column_nam	max	min	range	avg	stddev	median
1	winrate	66.6666...	6.66666...	59.9999...	39.6825...	13.5574...	40
2	blackjac...	13.3333...	0	13.3333...	3.17460...	3.66305...	4
3	incomerate	2450	-100	2550	133.762...	536.950...	15
4	averagebet	32500.2...	100	32400.2...	2178.24...	6960.69...	500.244...

- 유수익

	column_nam	avg
1	count	0.684210...

4. 결과 분석

카운팅을 하였을 때의 평균 승률은 약 40%, 평균 수익률은 약 134%이다.

카운팅을 하지 않았을 때의 평균 승률은 약 42%, 평균 수익률은 약 -14% 이다.

카운팅을 할 시 승률은 2%p 떨어졌지만 수익률은 반대로 148%p나 올랐다.

카운팅 유무와 관계없이 블랙잭은 약 3.18%의 확률로 나왔다.

평균 베팅액은 카운팅을 하였을 때 2100, 안했을 때 600으로 카운팅을 할 때 약 3.5배 더 높다. 이는 카운팅을 할 때 카운트가 높을 때 베팅을 더 높게(카운트가 매우 높을때는 올인) 하였기 때문으로 보인다.

수익이 발생한 케이스 중 카드카운팅을 사용한 비율은 68%이다.

VI.결론 및 향후계획

카드카운팅을 사용할 시 수익이 148%p 상승하는 효과가 있었다. 이를 통해 카드카운팅을 통하여 더 많은 수익을 얻을 수 있다는 것을 알 수 있다.

원래 처음에 계획했던 블랙잭 AI를 완성하지 못했다. AI를 만들기 위해 더 공부하여 내년에는 원래 최종목표였던 블랙잭 AI를 완성시킬 계획이다.