Analysis of Diamond Stars game from Pokerstars

Diamond Stars is one of the most popular slots game on Pokerstars. In this notebook, I'll analyse the payout table and attempt to recreate the reel details of Diamond Stars. The RTP as stated is 91.97% plus 1.16% for jackpot.

The following shows the payout tables seen in game.

Let's start with 3 reels. I put a 2x and 3x wild on the first reel, a 4x wild on the second reel, and a 5x reel on the third reel. Then I fill the remaining symbols table with estimation. There are 100 symbols on each reel, which is a common setup.

```
# define the reels using a list of 3 dictionaries, each dictionary representing a
reel
reels = [
    {
        'wild-5': 0,
        'wild-4': 0,
        'wild-3': 1,
        'wild-2': 1,
        'logo-red': 3,
        'logo-blue': 5,
        'bar-3': 8,
        'bar-2': 16,
        'bar-1': 24,
        'cherry': 42
    },
    {
        'wild-5': 0,
        'wild-4': 1,
        'wild-3': 0,
        'wild-2': 0,
        'logo-red': 3,
        'logo-blue': 5,
        'bar-3': 8,
        'bar-2': 16,
        'bar-1': 24,
        'cherry': 43
    },
    {
        'wild-5': 1,
        'wild-4': 0,
        'wild-3': 0,
        'wild-2': 0,
        'logo-red': 3,
        'logo-blue': 5,
        'bar-3': 8,
        'bar-2': 16,
        'bar-1': 24,
        'cherry': 43
    }
]
```

```
# assert every reel has exactly 100 possible outcomes
for reel in reels:
    assert sum(reel[logo] for logo in reel) == 100
```

We will make a few helper functions to help us to calculate the payouts. The first helper function calculates the amount of payout with all three symbols being wilds on each payline. The event of hitting a wild on each reel is an independent event and we could take advantage of this.

```
# helper function to calculate adjusted payout for three wilds
def calculate_payout_three_wilds(wild_payout=300):
    current_multiplier = 1
    # each reel can be considered independently
    for reel in reels:
        # add the sum of products of each wild multiplier by its frequencies
        current_multiplier *= sum(i * reel[f'wild-{i}'] for i in (2, 3, 4, 5))
    return current_multiplier * wild_payout
```

```
calculate_payout_three_wilds()
```

```
30000
```

Then we would write the payout function for each symbol. A payline including either the desirable symbol or the wild symbols would work, but at least one symbol on the payline should not be wild. A single function would do the job, we just need to call the function with different symbols and payouts as parameters.

```
# helper function to calculate adjusted payout for a particular symbol, including
possible wilds
def calculate_payout_by_symbol_or_wild(symbol, symbol_payout):
    hits = (symbol, 'wild-2', 'wild-3', 'wild-4', 'wild-5')
    payout_multiplier_sum = 0
    for reel0_symbol in hits:
        for reel1_symbol in hits:
            for reel2_symbol in hits:
                # must not be all wilds
                reel_symbols = (reel0_symbol, reel1_symbol, reel2_symbol)
                if symbol not in reel_symbols:
                    continue

                current_multiplier = 1
                if reel0_symbol[:4] == 'wild':
                    current_multiplier *= int(reel0_symbol[5])
                if reel1_symbol[:4] == 'wild':
                    current_multiplier *= int(reel1_symbol[5])
                if reel2_symbol[:4] == 'wild':
                    current_multiplier *= int(reel2_symbol[5])

                # calculate the number of possibilites of
                # hitting this particular combination
                num_possibilities = reels[0][reel0_symbol] *\
```

```
                                    reels[1][reel1_symbol] *\
                                    reels[2][reel2_symbol]
                payout_multiplier_sum += num_possibilities * current_multiplier


    return payout_multiplier_sum * symbol_payout
```

```
print(f"{calculate_payout_by_symbol_or_wild('logo-red', 100)=}")
print(f"{calculate_payout_by_symbol_or_wild('logo-blue', 50)=}")
print(f"{calculate_payout_by_symbol_or_wild('bar-3', 40)=}")
print(f"{calculate_payout_by_symbol_or_wild('bar-2', 20)=}")
print(f"{calculate_payout_by_symbol_or_wild('bar-1', 10)=}")
print(f"{calculate_payout_by_symbol_or_wild('cherry', 6)=}")
```

```
calculate_payout_by_symbol_or_wild('logo-red', 100)=34800
calculate_payout_by_symbol_or_wild('logo-blue', 50)=40000
calculate_payout_by_symbol_or_wild('bar-3', 40)=77120
calculate_payout_by_symbol_or_wild('bar-2', 20)=174400
calculate_payout_by_symbol_or_wild('bar-1', 10)=234480
calculate_payout_by_symbol_or_wild('cherry', 6)=635592
```

Next it comes the mixed logo, and mixed bar. A payline should include at least two different logos or two bars to qualify for the awards. This can be implemented by modifying the previous payout functions.

```
# helper function to calculate adjusted payout for a particular symbol, including
possible wilds
def calculate_payout_mixed_logo_or_wild(mixed_logo_payout=4):
    hits = ('logo-red', 'logo-blue', 'wild-2', 'wild-3', 'wild-4', 'wild-5')
    payout_multiplier_sum = 0
    for reel0_symbol in hits:
        for reel1_symbol in hits:
            for reel2_symbol in hits:
                # at least two different logos are present
                reel_symbols = (reel0_symbol, reel1_symbol, reel2_symbol)
                if 'logo-red' not in reel_symbols or \
                        'logo-blue' not in reel_symbols:
                    continue

                current_multiplier = 1
                if reel0_symbol[:4] == 'wild':
                    current_multiplier *= int(reel0_symbol[5])
                if reel1_symbol[:4] == 'wild':
                    current_multiplier *= int(reel1_symbol[5])
                if reel2_symbol[:4] == 'wild':
                    current_multiplier *= int(reel2_symbol[5])
```

```
                # calculate the number of possibilites of
                # hitting this particular combination
                num_possibilities = reels[0][reel0_symbol] *\
                                    reels[1][reel1_symbol] *\
                                    reels[2][reel2_symbol]
                payout_multiplier_sum += num_possibilities * current_multiplier


    return payout_multiplier_sum * mixed_logo_payout
```

```
calculate_payout_mixed_logo_or_wild()
```

```
3120
```

```
# helper function to calculate adjusted payout for a particular symbol, including
possible wilds
def calculate_payout_mixed_bars_or_wild(mixed_bars_payout=2):
    hits = ('bar-1', 'bar-2', 'bar-3', 'wild-2', 'wild-3', 'wild-4', 'wild-5')
    payout_multiplier_sum = 0
    for reel0_symbol in hits:
        for reel1_symbol in hits:
            for reel2_symbol in hits:

                reel_symbols = (reel0_symbol, reel1_symbol, reel2_symbol)
                # at least two different bars are present
                if ('bar-1' in reel_symbols) +\
                   ('bar-2' in reel_symbols) +\
                   ('bar-3' in reel_symbols) < 2:
                    continue

                current_multiplier = 1
                if reel0_symbol[:4] == 'wild':
                    current_multiplier *= int(reel0_symbol[5])
                if reel1_symbol[:4] == 'wild':
                    current_multiplier *= int(reel1_symbol[5])
                if reel2_symbol[:4] == 'wild':
                    current_multiplier *= int(reel2_symbol[5])

                # calculate the number of possibilites of
                # hitting this particular combination
                num_possibilities = reels[0][reel0_symbol] *\
                                    reels[1][reel1_symbol] *\
                                    reels[2][reel2_symbol]
                payout_multiplier_sum += num_possibilities * current_multiplier

    return payout_multiplier_sum * mixed_bars_payout
```

```
calculate_payout_mixed_bars_or_wild()
```

```
223744
```

Finally, let's calculate RTP (Return to Player). A slots game with RTP < 1 (100%) would be profitable to the game host.

```
# helper function to calculate RTP
def calculate_RTP():
    # 3 reels of 100 possiblities each
    base_num_possibilities = 100 * 100 * 100

    return sum([
        calculate_payout_three_wilds(wild_payout=300),
        calculate_payout_by_symbol_or_wild('logo-red', 100),
        calculate_payout_by_symbol_or_wild('logo-blue', 50),
        calculate_payout_by_symbol_or_wild('bar-3', 40),
        calculate_payout_by_symbol_or_wild('bar-2', 20),
        calculate_payout_by_symbol_or_wild('bar-1', 10),
        calculate_payout_by_symbol_or_wild('cherry', 6),
        calculate_payout_mixed_logo_or_wild(mixed_logo_payout=4),
        calculate_payout_mixed_bars_or_wild(mixed_bars_payout=2)
    ]) / base_num_possibilities
```

```
calculate_RTP()
```

```
1.453256
```

And put everything together into a nice table.

```
# helper function to create a table for showing all the data
def results_table():
    import tabulate
    data = [
        ["Payout Type", "Payout"],
        ["Three Wilds", calculate_payout_three_wilds(wild_payout=300)],
        ["Red Logo", calculate_payout_by_symbol_or_wild('logo-red', 100)],
        ["Blue Logo", calculate_payout_by_symbol_or_wild('logo-blue', 50)],
        ["Bar 3", calculate_payout_by_symbol_or_wild('bar-3', 40)],
```

```
        ["Bar 2", calculate_payout_by_symbol_or_wild('bar-2', 20)],
        ["Bar 1", calculate_payout_by_symbol_or_wild('bar-1', 10)],
        ["Cherry", calculate_payout_by_symbol_or_wild('cherry', 6)],
        [
            "Mixed Logos",
            calculate_payout_mixed_logo_or_wild(mixed_logo_payout=4)
        ],
        [
            "Mixed Bars",
            calculate_payout_mixed_bars_or_wild(mixed_bars_payout=2)
        ],
        ["", ""],
        ["RTP", calculate_RTP()]
    ]
    table = tabulate.tabulate(data)
    return table
```

```
print(results_table())
```

```
-----------  --------
Payout Type  Payout
Three Wilds  30000
Red Logo     34800
Blue Logo    40000
Bar 3        77120
Bar 2        174400
Bar 1        234480
Cherry       635592
Mixed Logos  3120
Mixed Bars   223744

RTP          1.453256
-----------  --------
```

Current the RTP is 145.33%, which is too high. We can see from the table that a large amount of payout comes from cherries. Red Logo, Blue Logo and Bar 3, despite being high value, is only contributing a small amount of payout. Let's revise the reels table to decrease the number of cherries and increase the number of other logos.

Here is a new symbols table for a possible setup of reels. The number of cherries are decreased significantly while the number of other symbols are increased a bit.

```
reels = [
    {
        'wild-5': 0,
```

```
        'wild-4': 0,
        'wild-3': 1,
        'wild-2': 1,
        'logo-red': 8,
        'logo-blue': 13,
        'bar-3': 12,
        'bar-2': 16,
        'bar-1': 22,
        'cherry': 27
    },
    {
        'wild-5': 0,
        'wild-4': 1,
        'wild-3': 0,
        'wild-2': 0,
        'logo-red': 8,
        'logo-blue': 13,
        'bar-3': 12,
        'bar-2': 16,
        'bar-1': 22,
        'cherry': 28
    },
    {
        'wild-5': 1,
        'wild-4': 0,
        'wild-3': 0,
        'wild-2': 0,
        'logo-red': 8,
        'logo-blue': 13,
        'bar-3': 12,
        'bar-2': 16,
        'bar-1': 22,
        'cherry': 28
    }
]

# assert every reel has exactly 100 possible outcomes
for reel in reels:
    assert sum(reel[logo] for logo in reel) == 100
```

Let's display the information of the reels in a table as well.

```
# helper function to create a table for showing the number of symbols on each reel
def reels_table():
    import tabulate
    data = [
        ["Reel #", 0, 1, 2],
        ["5x Wild", reels[0]['wild-5'], reels[1]['wild-5'], reels[2]['wild-5']],
        ["4x Wild", reels[0]['wild-4'], reels[1]['wild-4'], reels[2]['wild-4']],
        ["3x Wild", reels[0]['wild-3'], reels[1]['wild-3'], reels[2]['wild-3']],
        ["2x Wild", reels[0]['wild-2'], reels[1]['wild-2'], reels[2]['wild-2']],
        [
```

```
            "Red Logo",
            reels[0]['logo-red'],
            reels[1]['logo-red'],
            reels[2]['logo-red']
        ],
        [
            "Blue Logo",
            reels[0]['logo-blue'],
            reels[1]['logo-blue'],
            reels[2]['logo-blue']
        ],
        ["Bar 3", reels[0]['bar-3'], reels[1]['bar-3'], reels[2]['bar-3']],
        ["Bar 2", reels[0]['bar-2'], reels[1]['bar-2'], reels[2]['bar-2']],
        ["Bar 1", reels[0]['bar-1'], reels[1]['bar-1'], reels[2]['bar-1']],
        ["Cherry",  reels[0]['cherry'], reels[1]['cherry'], reels[2]['cherry']],
        ["Total", 100, 100, 100]
    ]
    table = tabulate.tabulate(data)
    return table
```

```
print(reels_table())
```

```
---------  ---  ---  ---
Reel #       0    1    2
5x Wild      0    0    1
4x Wild      0    1    0
3x Wild      1    0    0
2x Wild      1    0    0
Red Logo     8    8    8
Blue Logo   13   13   13
Bar 3       12   12   12
Bar 2       16   16   16
Bar 1       22   22   22
Cherry      27   28   28
Total      100  100  100
---------  ---  ---  ---
```

And calculate the RTP table again.

```
print(results_table())
```

```
-----------  --------
Payout Type  Payout
Three Wilds  30000
```

```
Red Logo     192800
Blue Logo    270400
Bar 3        180960
Bar 2        174400
Bar 1        188540
Cherry       202152
Mixed Logos  37856
Mixed Bars   262304

RTP          1.539412
-----------  --------
```

This time it is even worse. Maybe a roughly equal chance of hitting a symbol on each reel is not a good idea.

Let's try something different. What if the symbols of each kind are more concentrated in one or two reels?

```
reels = [
    {
        'wild-5': 0,
        'wild-4': 0,
        'wild-3': 1,
        'wild-2': 1,
        'logo-red': 3,
        'logo-blue': 23,
        'bar-3': 5,
        'bar-2': 5,
        'bar-1': 52,
        'cherry': 10
    },
    {
        'wild-5': 0,
        'wild-4': 1,
        'wild-3': 0,
        'wild-2': 0,
        'logo-red': 4,
        'logo-blue': 5,
        'bar-3': 5,
        'bar-2': 31,
        'bar-1': 6,
        'cherry': 48
    },
    {
        'wild-5': 1,
        'wild-4': 0,
        'wild-3': 0,
        'wild-2': 0,
        'logo-red': 17,
        'logo-blue': 5,
        'bar-3': 16,
        'bar-2': 5,
        'bar-1': 6,
```

```
        'cherry': 50
    }
]

# assert every reel has exactly 100 possible outcomes
for reel in reels:
    assert sum(reel[logo] for logo in reel) == 100
```

```
print(reels_table())
```

```
---------  ---  ---  ---
Reel #       0    1    2
5x Wild      0    0    1
4x Wild      0    1    0
3x Wild      1    0    0
2x Wild      1    0    0
Red Logo     3    4   17
Blue Logo   23    5    5
Bar 3        5    5   16
Bar 2        5   31    5
Bar 1       52    6    6
Cherry      10   48   50
Total      100  100  100
---------  ---  ---  ---
```

```
print(results_table())
```

```
-----------  -------
Payout Type  Payout
Three Wilds  30000
Red Logo     130800
Blue Logo    121000
Bar 3        71600
Bar 2        68000
Bar 1        61700
Cherry       256800
Mixed Logos  28212
Mixed Bars   174328

RTP          0.94244
-----------  -------
```

After a lot of tweaking and fine-tuning, I brought the RTP to 94.24%. This is much harder than I expected it to be. But this is an eye-opening exercise. Changing a symbol on the reel to another decreases the payout of one type of combo, but increases the payout of another. It is interesting to play around the reels table while trying to make the overall payout to stay within the budget.

Anyway, here is my first analysis of a slots game. For further improvements, I should probably write a couple of helper functions that tells me the effect of adding or removing a symbol on the payout budget. This would save a lot of time for tweaking the numbers to find the right amount of symbols on each reel.