Analysis of Historical Over-Under NBA Odds

January 25, 2022

1 Historical Over/Under Analysis in the NBA

For this project, we have been using datasets from the 2012-2013 NBA Season to the 2021-2022 NBA Season. The data has been sourced from the following website: https://www.sportsbookreviewsonline.com/scoresoddsarchives/nba/nbaoddsarchives.htm

Note: The data from the 21-22 season only contains game information up to January 10, 2022.

What does betting on the Over/Under mean?

The easiest way to explain this would be through an arbitrary example. Let's say the Warriors and Lakers have a game tomorrow, and when the odds come out, the bookmakers set the line for the total points in the matchup at 223.5. A bet on the **over** would mean you believe the total points scored in the game would be *at least 224*. Conversely, a bet on the **under** would you expect the total score to be *less than 224*.

Bookmakers reserve the right to move the odds. For example, say that a few hours before game time, Steph Curry of the Warriors is ruled out of the matchup due to an injury. Since Curry is a one of the NBA's best players and scorers, Vegas/bookmakers move the line to 218, a 5.5 point decrease from the **OLV** (opening line value). If the game begins with the line at 218, that means the **CLV** (closing line value) is 218.

Project Goals:

1. What does line movement on the Over/Under indicate to bettors?
2. Is it more profitable to be on the Over or Under depending on line movement?

Keep in mind that in the long-term, professional sports bettors rarely maintain a winning rate above 55%, and typically the win-rate is around 53%-54%, per https://professionalgambler.org/

To begin, let's import our data into a pandas dataframe.

[50]:
```
import pandas as pd
import numpy as np

original = pd.read_excel([redacted]
 [redacted])

#replace pk with arbitrary float values
original.loc[original.Open == 'pk', 'Open'] = '0.0'
original.loc[original.Close == 'pk', 'Close'] = '0.0'
original.loc[original.Open == 'PK', 'Open'] = '0.0'
```

```
original.loc[original.Close == 'PK', 'Close'] = '0.0'

#making sure data type for number totals are either int or float
original = original.astype({"Final": int, "Open": float, "Close": float})

print(original)
```

```
       Date  Rot VH         Team  1st  2nd  3rd  4th  Final   Open  Close  \
0      1019  501  V     Brooklyn   25   34   26   19    104  240.5  233.5
1      1019  502  H    Milwaukee   37   29   31   30    127    0.0    1.5
2      1019  503  V  GoldenState   32   21   30   38    121  230.5  226.5
3      1019  504  H     LALakers   34   25   26   29    114    4.5    3.5
4      1020  505  V      Indiana   38   37   13   34    122  228.5    1.5
…       …   …  ..          …   …   …   …   …    …      …      …
24109   616  710  H   SanAntonio   32   29   26   27    114    0.0  189.0
24110   618  711  V   SanAntonio   25   25   25   20    100  189.5  192.0
24111   618  712  H        Miami   27   17   21   30    103    7.5    7.0
24112   620  713  V   SanAntonio   16   28   27   17     88  191.0  188.0
24113   620  714  H        Miami   18   28   26   23     95    6.0    5.5

       Total
0        231
1        231
2        235
3        235
4        245
…       …
24109    218
24110    203
24111    203
24112    183
24113    183

[24114 rows x 12 columns]
```

Let's understand the original data that we have in the table above. More specifically, we'll examine the first 2 rows, which contain the data for the 2021-2022 season opener, where Milwaukee defeated Brooklyn 127 to 104.

Here we see that the opening line for the Over/Under in this matchup was 240.5 (as seen in the **Open** column), and the closing line went down to 233.5 (as seen in the **Close** column). In the second row, we see in the **Open** and **Close** columns, that there are values 0.0 and 1.5 for each of those columns, respectively. Since these first two rows have the same value in the **Total** column (which should always be the case, since these two rows are from the same game), we will want to drop the second row from our dataframe, as it has redundant data without the necessary and accurate Opening and Closing line values.

The above will be important to understand when we need to drop specific rows from our dataframe in order to get accurate results.

1.1 General Examination of Over/Unders in the NBA over the Last 5 Years

Below we drop any rows from the original data where the over/under is below 150, as that is inaccurate data from duplicate rows.

Function 1, *OpeningOverUnderLines*, examines how frequently the Over hits based on the **opening** line.

Function 2, *ClosingOverUnderLines*, examines how frequently the Over hits based on the **closing** line

[51]:
```
#dropping rows where the over/under is below 150, aka inaccurate data, for the␣
 ↪open
openingLV = original.drop(original[original.Open < 150].index)

#function 1
def OpeningOverUnderLines(df):
    count = 0
    tot = len(df.index)
    for index, col in df.iterrows():
        if col['Total'] > col['Open']:
            count += 1
    return count/tot * 100
print(OpeningOverUnderLines(openingLV), "percent of the Overs hit across the␣
 ↪league from the Opening Line Value")


closingLV = original.drop(original[original.Close < 150].index)
#function 2
def ClosingOverUnderLines(df):
    count = 0
    tot = len(df.index)
    for index, col in df.iterrows():
        if col['Total'] > col['Close']:
            count += 1
    return count/tot * 100
print(ClosingOverUnderLines(closingLV), "percent of the Overs hit across the␣
 ↪league from the Closing Line value")
```

```
48.983486847564514 percent of the Overs hit across the league from the Opening
Line Value
49.25360756344336 percent of the Overs hit across the league from the Closing
Line value
```

Based on the data above, we can conclude that the Under is slightly more statistically likely to hit, based on opening and closing line value. However, the above **does not** return anything of significance.

1.2 Analysis of Opening/Closing Line Value

Below, we will analyze the three cases in Opening and Closing line value:

```
Case 1: OLV = CLV
Case 2: OLV > CLV
Case 3: OLV < CLV
```

Note that in our functions below, we compare the **Total** of the NBA matchup result to the **Close**, or CLV. This is because in order to assess line movement and potential betting opportunities, we need to wait for the line movement to change, or not change.

First, we drop the inaccurate data, and create our new dataframe **filtered**

[52]:
```
filtered = original.drop(original[original.Close < 150].index)
filtered = filtered.drop(filtered[filtered.Open < 150].index)
```

1.2.1 Case 1: Opening Line Value = Closing Line Value

[53]:
```
def ZeroPercentChange(df):
    a = 0
    b = 0
    for index, col in df.iterrows():
        if col['Open'] == col['Close']:
            if col['Total'] > col['Close']:
                # a = instances of the final score being > than the CLV
                a += 1
            else:
                # b = instances of final score being < than the CLV
                b += 1
    return a, b+a, a/(a + b) * 100

print(ZeroPercentChange(filtered))
```

```
(438, 883, 49.60362400906002)
```

Above, we see that when the $OLV = CLV$, the Over hits from the CLV at a 49.6% rate, which is a losing formula in the long-term.

1.2.2 Case 2: Opening Line Value > Closing Line Value

Let's perform further analysis to see if the Over/Under is more likely to hit based on a percentage decrease.

For example, consider the scenario where the OLV is set at 210 and the CLV is 204. We seek to understand the following question:

```
What percentage range decrease from the OLV do there exist profitable ranges to take the Over/U
```

The framework of the iteration below is based off of the following:

$$OLV - (OLV \cdot \beta) \leq CLV \leq OLV - (OLV \cdot \alpha)$$

where α is equal to the lower percentage decrease from the OLV, and β is equal to the higher percentage decrease from the OLV.

[67]:
```
def PercentDecreaseFromOLV(df, lowEnd, highEnd):
    a = 0
    b = 0
    for index, col in df.iterrows():
        if col['Open'] > col['Close']:
            if (col['Open'] - col['Open'] * (lowEnd)) >= col['Close']:
                if (col['Open'] - col['Open'] * (highEnd)) <= col['Close']:
                    if col['Total'] > col['Close']:
                        #a = instances of the final score being > than the CLV
                        a += 1
                    else:
                        # b = instances of final score being < than the CLV
                        b += 1
    return a, b+a, a/(a + b) * 100

print(PercentDecreaseFromOLV(filtered, .005, .04))
```

```
(1795, 3734, 48.07177289769684)
```

The above function displays that when then CLV decrease is between 0.5% and 4% of the OLV, the Over on the CLV hits at a rate of 48.07%.

Below, we test different ranges of CLV decreases to assess whether an optimal range exists, with frequency.

[69]:
```
print(PercentDecreaseFromOLV(filtered, .00001, .005), "0.001% and 0.5% lower␣
 ↪than OLV")
print(PercentDecreaseFromOLV(filtered, .00501, .01), "0.501% and 1% lower than␣
 ↪OLV")
print(PercentDecreaseFromOLV(filtered, .01001, 0.012), "1.001% and 1.5% lower␣
 ↪than OLV")
print(PercentDecreaseFromOLV(filtered, .01501, 0.02), "1.501% and 2% lower than␣
 ↪OLV")
print(PercentDecreaseFromOLV(filtered, .02001, 0.025), "2.001% and 2.5% lower␣
 ↪than OLV")
print(PercentDecreaseFromOLV(filtered, .02501, 0.03), "2.501% and 3% lower than␣
 ↪OLV")
print(PercentDecreaseFromOLV(filtered, .03001, 0.035), "3.001% and 3.5% lower␣
 ↪than OLV")
print(PercentDecreaseFromOLV(filtered, .03501, 0.04), "3.501% and 4% lower than␣
 ↪OLV")
print(PercentDecreaseFromOLV(filtered, .04001, 0.1), "4.001% and 10% lower than␣
 ↪OLV")
```

```
(723, 1478, 48.91745602165088) 0.001% and 0.5% lower than OLV
(727, 1491, 48.75922199865862) 0.501% and 1% lower than OLV
```

```
(198, 432, 45.83333333333333) 1.001% and 1.5% lower than OLV
(296, 641, 46.17784711388456) 1.501% and 2% lower than OLV
(170, 337, 50.445103857566764) 2.001% and 2.5% lower than OLV
(84, 161, 52.17391304347826) 2.501% and 3% lower than OLV
(33, 65, 50.76923076923077) 3.001% and 3.5% lower than OLV
(9, 20, 45.0) 3.501% and 4% lower than OLV
(20, 27, 74.07407407407408) 4.001% and 10% lower than OLV
```

From the results above, we see that most optimal range for taking bets are the following:

1. 53.82 hit rate on the UNDER when CLV drops between 1.501% and 2% from the OLV
2. 53.37 hit rate on the UNDER when CLV drops between 1.001% and 1.5% from the OLV
3. 52.17 hit rate on the OVER when CLV drops between 2.501% and 3% from the OLV

1.2.3 Case 3: Opening Line Value < Closing Line Value

Similar study to the section above, but now we are assessing the cases where $CLV > OLV$

[62]:
```
def PercentIncreaseFromOLV(df, lowEnd, highEnd):
    a = 0
    b = 0
    for index, col in df.iterrows():
        if col['Open'] < col['Close']:
            if (col['Open'] + col['Open'] * (lowEnd)) <= col['Close']:
                if (col['Open'] + col['Open'] * (highEnd)) >= col['Close']:
                    if col['Total'] > col['Close']:
                        # a = instances of the final score being > than the CLV
                        a += 1
                    else:
                        # b = instances of final score being < than the CLV
                        b += 1
    return a, b+a, a/(a + b) * 100

print(PercentIncreaseFromOLV(filtered, .005, .04))
```

```
(1815, 3597, 50.45871559633027)
```

The above function displays that when the CLV increase is between 0.5% and 4% of the OLV, the Over on the CLV hits at a rate of 50.45%.

Like above, let's test different ranges of CLV increases to assess whether an optimal range exists, with frequency.

[63]:
```
print(PercentIncreaseFromOLV(filtered, .00001, .005), "0.001% and 0.5% higher␣
 ↪than OLV")
print(PercentIncreaseFromOLV(filtered, .00501, .01), "0.501% and 1% higher than␣
 ↪OLV")
```

```
print(PercentIncreaseFromOLV(filtered, .01001, 0.015), "1.001% and 1.5% higher␣
↪than OLV")
print(PercentIncreaseFromOLV(filtered, .01501, 0.02), "1.501% and 2% higher␣
↪than OLV")
print(PercentIncreaseFromOLV(filtered, .02001, 0.025), "2.001% and 2.5% higher␣
↪than OLV")
print(PercentIncreaseFromOLV(filtered, .02501, 0.03), "2.501% and 3% higher␣
↪than OLV")
print(PercentIncreaseFromOLV(filtered, .03001, 0.035), "3.001% and 3.5% higher␣
↪than OLV")
print(PercentIncreaseFromOLV(filtered, .03501, 0.04), "3.501% and 4% higher␣
↪than OLV")
print(PercentIncreaseFromOLV(filtered, .0401, 0.1), "4.001% and 10% higher than␣
↪OLV")
```

```
(745, 1519, 49.045424621461486) 0.001% and 0.5% higher than OLV
(687, 1389, 49.46004319654428) 0.501% and 1% higher than OLV
(518, 1043, 49.664429530201346) 1.001% and 1.5% higher than OLV
(345, 659, 52.35204855842185) 1.501% and 2% higher than OLV
(153, 286, 53.4965034965035) 2.001% and 2.5% higher than OLV
(71, 137, 51.82481751824818) 2.501% and 3% higher than OLV
(27, 54, 50.0) 3.001% and 3.5% higher than OLV
(9, 18, 50.0) 3.501% and 4% higher than OLV
(9, 17, 52.94117647058824) 4.001% and 10% higher than OLV
```

From the results above, we see that most optimal range for taking bets are the following:

1. 53.49% hit rate on the OVER when CLV increases between 2.001% and 2.5% from the OLV
2. 52.35% hit rate on OVER when CLV increases between 1.501% and 2% from the OLV

1.3 Conclusions

After our analysis, we have identified 5 different long-term profitable ranges, in order of hit rate:

The first three have hit rates of above 53% in the long-term:

1. UNDER - When CLV **drops** between 1.501% and 2% from the OLV
2. OVER - When CLV **increases** between 2.001% and 2.5% from the OLV
3. UNDER - When CLV **drops** between 1.001% and 1.5% from the OLV

The next two have hit rates between 52% and 53% in the long-term:

4. OVER - When CLV **drops** between 2.501% and 3% from the OLV
5. OVER - When CLV **increases** between 1.501% and 2% from the OLV