BDAT 610
Final Project

Casino Data Analysis

Aaron Fiedler

TectonArchitects

Background of Data

The data I am using comes from slot machines over the course of 27 years (1993-2020) in two casinos in Connecticut, Mohegan Sun and Foxwoods.

The main statistics shown in the data are dollar amounts of contributions and payout. Total handle, or total amount wagered, is broken into what the casino pays out vs holds. Contributions are split up into what is paid and what is earned in bonuses.

```
data = pd.read_csv('Desktop\MARYVILLE\BDAT_610\Slot_Machine_Data_Foxwoods_Mohegan_Sun.csv')
```

```
data.head(5)
```

	Casino	Date	Fiscal Year	Slot Machine Contributions to the State of Connecticut (1) (6)	Total Contributions (1)	eBonus Contribution (3)	eBonus Credits Redeemed at Slot Machines (2)	Weighted Average # of Machines	Payout %	Hold %	Win (9)	Handle	Footnotes
0	Foxwoods	12/15/2020 0:00	2020/2021	5368328	6085094	0	1960609	1875	91.51	8.49	21473310	253041236	(3) (4)
1	Mohegan Sun	12/15/2020 0:00	2020/2021	8059398	8087961	28563	3660386	2595	92.01	7.99	32237592	403375272	(2) (3)
2	Foxwoods	11/15/2020 0:00	2020/2021	5445009	5445009	0	1568948	2332	91.45	8.55	21780036	254706766	(3) (4)
3	Mohegan Sun	11/15/2020 0:00	2020/2021	7128718	7193431	64714	3395490	2541	92.34	7.66	28514871	372455663	(2) (3)
4	Foxwoods	10/15/2020 0:00	2020/2021	6525003	6525003	0	2569762	2333	91.53	8.47	26100010	308128405	(3) (4)

Conditional Statement

```
data_rank[["Casino", "Date", "Fiscal Year", "Payout %", "Payout Rank", "Handle", "Handle Rank"]].head(20)
```

	Casino	Date	Fiscal Year	Payout %	Payout Rank	Handle	Handle Rank
0	Foxwoods	12/15/2020 0:00	2020/2021	91.51	Bottom 25%	253041236	Bottom 25%
1	Mohegan Sun	12/15/2020 0:00	2020/2021	92.01	Top 25%	403375272	Bottom 25%
2	Foxwoods	11/15/2020 0:00	2020/2021	91.45	Bottom 25%	254706766	Bottom 25%
3	Mohegan Sun	11/15/2020 0:00	2020/2021	92.34	Top 25%	372455663	Bottom 25%
4	Foxwoods	10/15/2020 0:00	2020/2021	91.53	Bottom 25%	308128405	Bottom 25%
5	Mohegan Sun	10/15/2020 0:00	2020/2021	91.78	Middle 50%	435869683	Bottom 25%
6	Foxwoods	9/15/2020 0:00	2020/2021	91.27	Bottom 25%	316828736	Bottom 25%
7	Mohegan Sun	9/15/2020 0:00	2020/2021	91.28	Bottom 25%	461942777	Bottom 25%
8	Foxwoods	8/15/2020 0:00	2020/2021	91.36	Bottom 25%	342691930	Bottom 25%
9	Mohegan Sun	8/15/2020 0:00	2020/2021	91.35	Bottom 25%	526932118	Middle 50%
10	Foxwoods	7/15/2020 0:00	2020/2021	91.45	Bottom 25%	357082368	Bottom 25%
11	Mohegan Sun	7/15/2020 0:00	2020/2021	91.35	Bottom 25%	549085284	Middle 50%
12	Foxwoods	6/15/2020 0:00	2019/2020	91.09	Bottom 25%	377110769	Bottom 25%
13	Mohegan Sun	6/15/2020 0:00	2019/2020	91.21	Bottom 25%	517513274	Bottom 25%
14	Foxwoods	5/15/2020 0:00	2019/2020	97.07	Top 25%	11998084	Bottom 25%
15	Mohegan Sun	5/15/2020 0:00	2019/2020	93.43	Top 25%	10181232	Bottom 25%
16	Foxwoods	4/15/2020 0:00	2019/2020	0.00	Bottom 25%	0	Bottom 25%
17	Mohegan Sun	4/15/2020 0:00	2019/2020	0.00	Bottom 25%	0	Bottom 25%
18	Foxwoods	3/15/2020 0:00	2019/2020	91.91	Middle 50%	172833652	Bottom 25%
19	Mohegan Sun	3/15/2020 0:00	2019/2020	91.89	Middle 50%	240474475	Bottom 25%

I used a conditional statement to add bins for the ranks of ‘Payout %’ and ‘Handle’.
I classified them into three groups (lowest 25%, highest 25%, and middle 50%).
This helped me be able to easily interpret the values and see a ballpark of where they stand in relation to each other rather than having to read each large number.

```
# insert new rows for the bins

data_rank.insert(13,"Payout Rank", '')
data_rank.insert(14,"Handle Rank", '')

# Payout bins

for i in range(0, len(data_rank["Payout %"]), 1):
    if(data_rank["Payout %"][i] >= 91.970000):
        data_rank["Payout Rank"][i] = "Top 25%"
    elif(data_rank["Payout %"][i] <= 91.600000):
        data_rank["Payout Rank"][i] = "Bottom 25%"
    else:
        data_rank["Payout Rank"][i] = "Middle 50%"

# Handle bins

for i in range(0, len(data_rank["Handle"]), 1):
    if(data_rank["Handle"][i] >= 775952200):
        data_rank["Handle Rank"][i] = "Top 25%"
    elif(data_rank["Handle"][i] <= 526347800):
        data_rank["Handle Rank"][i] = "Bottom 25%"
    else:
        data_rank["Handle Rank"][i] = "Middle 50%"
```

```
data_rank = data
data_rank[["Payout %", "Handle"]].describe()
```

	Payout %	Handle
count	627.000000	6.270000e+02
mean	91.519825	6.386761e+08
std	5.198301	1.700733e+08
min	0.000000	0.000000e+00
25%	91.600000	5.263478e+08
50%	91.800000	6.424454e+08
75%	91.970000	7.759522e+08
max	97.070000	1.012606e+09

Iterative Calculation


```
handle_sum = 0
for i in data['Handle']:
    handle_sum = handle_sum + i

handle_sum_s = str(handle_sum)
print("Total Handle of the two casinos from 1993 to 2020: $" + handle_sum_s)
```

```
Total Handle of the two casinos from 1993 to 2020: $400449921538
```

$400,449,921,538

This calculation gives the total amount of money wagered across both casinos over the 27 year period. This is an informative value that gives us the scale to which money circulates through the casinos. Nearly half a trillion dollars has made its way through the casinos with 8.48% being held in revenue; and this is only slots, table games wouldn't be included.

According to WorldOMeter, the amount of money that flows through these casinos each year on average is greater than the GDP of 70 different countries in 2017, including Jamaica, Mongolia, and Chad.

Quite crazy to think about.

Meaningful Calculation

I calculated the averages of both handle and payout percentage across various months.

My reasoning for diving into this was to see any trends in gambling in different seasons. How would summer break effect that statistics? How about Christmas time?

To my surprise, the holiday season of December, when people have time off work and probably more disposable income to spend, has the lowest amount wagered with the lowest handle. That time of the year does however produce the best payout percentage.

```
data['Date'] = pd.to_datetime(data.Date)

data_date = data[['Payout %','Handle','Casino','Date']]

data_date_mar = data_date[data_date.Date.dt.month == 3]
data_date_june = data_date[data_date.Date.dt.month == 6]
data_date_sept = data_date[data_date.Date.dt.month == 9]
data_date_dec = data_date[data_date.Date.dt.month == 12]
```

```
mar_handle = data_date_mar['Handle'].mean()
june_handle = data_date_june['Handle'].mean()
spet_handle = data_date_sept['Handle'].mean()
dec_handle = data_date_dec['Handle'].mean()
mar_handle_s = str(round(mar_handle,2))
print("Average March Handle = " + mar_handle_s)
june_handle_s = str(round(june_handle,2))
print("Average June Handle = " + june_handle_s)
sept_handle_s = str(round(spet_handle,2))
print("Average September Handle = " + sept_handle_s)
dec_handle_s = str(round(dec_handle,2))
print("Average December Handle = " + dec_handle_s)

print(' ')

mar_pay = data_date_mar['Payout %'].mean()
june_pay = data_date_june['Payout %'].mean()
sept_pay = data_date_sept['Payout %'].mean()
dec_pay = data_date_dec['Payout %'].mean()
mar_pay_s = str(round(mar_pay,2))
print("Average March Payout Percentage = " + mar_pay_s)
june_pay_s = str(round(june_pay,2))
print("Average June Payout Percentage = " + june_pay_s)
sept_pay_s = str(round(sept_pay,2))
print("Average September Payout Percentage = " + sept_pay_s)
dec_pay_s = str(round(dec_pay,2))
print("Average December Payout Percentage = " + dec_pay_s)
```

```
Average March Handle = 644473569.6
Average June Handle = 637767701.83
Average September Handle = 646902060.81
Average December Handle = 597562355.75

Average March Payout Percentage = 91.74
Average June Payout Percentage = 91.83
Average September Payout Percentage = 91.75
Average December Payout Percentage = 91.98
```

Vectors, Matrices, and Arrays // Multiply matrices

```
# multiply matrices

mat3 = mat1 * mat2
mat3
```

```
array([[1539779706936184,      40262456250],
       [3262483468300392,      83656551240],
       [1386880633230894,      50791043952],
       [2679234112349753,      72456287211],
       [2010538767010215,      60891323330],
       [3903592653758893,      90346824126],
       [2189566642362624,      62032097028],
       [4651107805646660,     101491283880],
       [2537415789582520,      66135777015],
       [6003204833480264,     114975610293],
       [2724412060681728,      68392148985],
       [6520484935595268,     120129288003],
       [3169078253488406,      49177715048],
       [5887299834289442,     101020063260],
       [   1252408000256,        425017289],
       [   1703737544112,        797210142],
       [                0,                0],
       [                0,        141498240],
       [ 606657007040076,      48417759869],
       [1191093881648025,      81134643200],
       [3868479255583488,     123485178206],
       [5957373575799840,     185935228698],
       [3339197606107334,     110384461272],
       [5794603741775121,     180900528381]], dtype=int64)
```

```
#vectors

data24 = data.head(24)
```

```
data24['Payout %'].to_numpy()
```

```
array([91.51, 92.01, 91.45, 92.34, 91.53, 91.78, 91.27, 91.28, 91.36,
       91.35, 91.45, 91.35, 91.09, 91.21, 97.07, 93.43,  0.  ,  0.  ,
       91.91, 91.89, 91.68, 91.7 , 91.92, 91.72])
```

```
data24['Hold %'].to_numpy()
```

```
array([8.49, 7.99, 8.55, 7.66, 8.47, 8.22, 8.73, 8.72, 8.64, 8.65, 8.55,
       8.65, 8.91, 8.79, 2.93, 6.57, 0.  , 0.  , 8.09, 8.11, 8.32, 8.3 ,
       8.08, 8.28])
```

```
#matrices

mat1 = data24[['Total Contributions (1)',
               'Weighted Average # of Machines']].to_numpy()
mat1
```

```
array([[ 6085094,     1875],
       [ 8087961,     2595],
       [ 5445009,     2332],
       [ 7193431,     2541],
       [ 6525003,     2333],
       [ 8955871,     2522],
       [ 6910884,     2244],
       [10068580,     2520],
       [ 7404364,     2233],
       [11392748,     2523],
       [ 7629646,     2241],
       [11875177,     2529],
       [ 8403574,     1463],
       [11376133,     2220],
       [  104384,     1207],
       [  167341,     1191],
       [       0,     3461],
       [    8503,     4160],
       [ 3510063,     3461],
       [ 4953099,     4160],
       [ 9020544,     3461],
       [11115210,     4182],
       [ 8314541,     3401],
       [10953047,     4129]], dtype=int64)
```

```
mat2 = data24[['Handle', 'Win (9)']].to_numpy()
mat2
```

```
array([[253041236,  21473310],
       [403375272,  32237592],
       [254706766,  21780036],
       [372455663,  28514871],
       [308128405,  26100010],
       [435869683,  35823483],
       [316828736,  27643537],
       [461942777,  40274319],
       [342691930,  29617455],
       [526932118,  45570991],
       [357082368,  30518585],
       [549085284,  47500707],
       [377110769,  33614296],
       [517513274,  45504533],
       [ 11998084,    352127],
       [ 10181232,    669362],
       [        0,         0],
       [        0,     34014],
       [172833652,  13989529],
       [240474475,  19503520],
       [428852102,  35679046],
       [535965904,  44460839],
       [401609374,  32456472],
       [529040343,  43812189]], dtype=int64)
```

```
# arrays

data24[['Payout %', 'Hold %']].to_numpy()
```

```
array([[91.51,  8.49],
       [92.01,  7.99],
       [91.45,  8.55],
       [92.34,  7.66],
       [91.53,  8.47],
       [91.78,  8.22],
       [91.27,  8.73],
       [91.28,  8.72],
       [91.36,  8.64],
       [91.35,  8.65],
       [91.45,  8.55],
       [91.35,  8.65],
       [91.09,  8.91],
       [91.21,  8.79],
       [97.07,  2.93],
       [93.43,  6.57],
       [ 0.  ,  0.  ],
       [ 0.  ,  0.  ],
       [91.91,  8.09],
       [91.89,  8.11],
       [91.68,  8.32],
       [91.7 ,  8.3 ],
       [91.92,  8.08],
       [91.72,  8.28]])
```

```
data24[['Total Contributions (1)',
        'eBonus Contribution (3)']].to_numpy()
```

```
array([[ 6085094,        0],
       [ 8087961,    28563],
       [ 5445009,        0],
       [ 7193431,    64714],
       [ 6525003,        0],
       [ 8955871,        0],
       [ 6910884,        0],
       [10068580,        0],
       [ 7404364,        0],
       [11392748,        0],
       [ 7629646,        0],
       [11875177,        0],
       [ 8403574,        0],
       [11376133,        0],
       [  104384,    16352],
       [  167341,        0],
       [       0,        0],
       [    8503,        0],
       [ 3510063,    12681],
       [ 4953099,    77218],
       [ 9020544,   100782],
       [11115210,        0],
       [ 8314541,   200423],
       [10953047,        0]], dtype=int64)
```

Meaningful Data Transformations

```
# only data for Mohegan Sun Casino

data_mohegan = data[data.Casino == 'Mohegan Sun']
data_mohegan.head(10)
```

	Casino	Date	Fiscal Year	Slot Machine Contributions to the State of Connecticut (1) (6)	Total Contributions (1)	eBonus Contribution (3)	eBonus Credits Redeemed at Slot Machines (2)	Weighted Average # of Machines	Payout %	Hold %	Win (9)	Handle	Footnotes
1	Mohegan Sun	12/15/2020 0:00	2020/2021	8059398	8087961	28563	3660386	2595	92.01	7.99	32237592	403375272	(2) (3)
3	Mohegan Sun	11/15/2020 0:00	2020/2021	7128718	7193431	64714	3395490	2541	92.34	7.66	28514871	372455663	(2) (3)
5	Mohegan Sun	10/15/2020 0:00	2020/2021	8955871	8955871	0	3768396	2522	91.78	8.22	35823483	435869683	(2) (3)
7	Mohegan Sun	9/15/2020 0:00	2020/2021	10068580	10068580	0	3578691	2520	91.28	8.72	40274319	461942777	(2) (3)
9	Mohegan Sun	8/15/2020 0:00	2020/2021	11392748	11392748	0	3811825	2523	91.35	8.65	45570991	526932118	(2) (3)
11	Mohegan Sun	7/15/2020 0:00	2020/2021	11875177	11875177	0	3566228	2529	91.35	8.65	47500707	549085284	(2) (3)
13	Mohegan Sun	6/15/2020 0:00	2019/2020	11376133	11376133	0	2939292	2220	91.21	8.79	45504533	517513274	(2) (3)
15	Mohegan Sun	5/15/2020 0:00	2019/2020	167341	167341	0	24218						
17	Mohegan Sun	4/15/2020 0:00	2019/2020	8503	8503	0	0						
19	Mohegan Sun	3/15/2020 0:00	2019/2020	4875880	4953099	77218	2454261						

```
# only data for Foxwoods Casino

data_foxwoods = data[data.Casino == 'Foxwoods']
data_foxwoods.head(10)
```

	Casino	Date	Fiscal Year	Slot Machine Contributions to the State of Connecticut (1) (6)	Total Contributions (1)	eBonus Contribution (3)	eBonus Credits Redeemed at Slot Machines (2)	Weighted Average # of Machines	Payout %	Hold %	Win (9)	Handle	Footnotes
0	Foxwoods	12/15/2020 0:00	2020/2021	5368328	6085094	0	1960609	1875	91.51	8.49	21473310	253041236	(3) (4)
2	Foxwoods	11/15/2020 0:00	2020/2021	5445009	5445009	0	1568948	2332	91.45	8.55	21780036	254706766	(3) (4)
4	Foxwoods	10/15/2020 0:00	2020/2021	6525003	6525003	0	2569762	2333	91.53	8.47	26100010	308128405	(3) (4)
6	Foxwoods	9/15/2020 0:00	2020/2021	6910884	6910884	0	2023444	2244	91.27	8.73	27643537	316828736	(3) (4)
8	Foxwoods	8/15/2020 0:00	2020/2021	7404364	7404364	0	2744859	2233	91.36	8.64	29617455	342691930	(3) (4)
10	Foxwoods	7/15/2020 0:00	2020/2021	7629646	7629646	0	2992245	2241	91.45	8.55	30518585	357082368	(3) (4)
12	Foxwoods	6/15/2020	2019/2020	8403574	8403574	0	2287114	1463	91.09	8.91	33614296	377110769	(3) (4)
					104384	16352	104143	1207	97.07	2.93	352127	11998084	(3) (4)
					0	0	0	3461	0.00	0.00	0	0	(3) (4)
					3510063	12681	1589574	3461	91.91	8.09	13989529	172833652	(3) (4)

```
# keep the data simple and consice to give a basic preview/explination

data_simple = data[['Casino', 'Date', 'Payout %', 'Handle']]
data_simple.head(10)
```

	Casino	Date	Payout %	Handle
0	Foxwoods	12/15/2020 0:00	91.51	253041236
1	Mohegan Sun	12/15/2020 0:00	92.01	403375272
2	Foxwoods	11/15/2020 0:00	91.45	254706766
3	Mohegan Sun	11/15/2020 0:00	92.34	372455663
4	Foxwoods	10/15/2020 0:00	91.53	308128405
5	Mohegan Sun	10/15/2020 0:00	91.78	435869683
6	Foxwoods	9/15/2020 0:00	91.27	316828736
7	Mohegan Sun	9/15/2020 0:00	91.28	461942777
8	Foxwoods	8/15/2020 0:00	91.36	342691930
9	Mohegan Sun	8/15/2020 0:00	91.35	526932118

Meaningful Data Transformations (cont.)

```
# drop all info prior to the date in which a U.S. state, for the fist time,
# legalized forms of online gambling. This changed the game for the industry.

data['Date'] = pd.to_datetime(data.Date)

data_recent = data[~(data['Date'] < '2010-11-22')]
data_recent
```

	Casino	Date	Fiscal Year	Slot Machine Contributions to the State of Connecticut (1) (6)	Total Contributions (1)	eBonus Contribution (3)	eBonus Credits Redeemed at Slot Machines (2)	Weighted Average # of Machines	Payout %	Hold %	Win (9)	Handle	Footnotes
0	Foxwoods	2020-12-15	2020/2021	5368328	6085094	0	1960609	1875	91.51	8.49	21473310	253041236	(3) (4)
1	Mohegan Sun	2020-12-15	2020/2021	8059398	8087961	28563	3660386	2595	92.01	7.99	32237592	403375272	(2) (3)
2	Foxwoods	2020-11-15	2020/2021	5445009	5445009	0	1568948	2332	91.45	8.55	21780036	254706766	(3) (4)
3	Mohegan Sun	2020-11-15	2020/2021	7128718	7193431	64714	3395490	2541	92.34	7.66	28514871	372455663	(2) (3)
4	Foxwoods	2020-10-15	2020/2021	6525003	6525003	0	2569762	2333	91.53	8.47	26100010	308128405	(3) (4)
...	...	...	...	...	...	...	...	...	...	...	...	...	...
237	Mohegan Sun	2011-02-17	2010/2011	14511935	15101403	589468	5550499	6367	91.90	8.10	58047738	716293719	(2) (3)
238	Foxwoods	2011-01-17	2010/2011	12135930	12656351	520421	4751590	6571	91.29	8.71	48543718	557241710	(3) (4)
239	Mohegan Sun	2011-01-17	2010/2011	13635323	14315598	680276	5720874	6372	92.09	7.91	54541290	689190773	(2) (3)
240	Foxwoods	2010-12-17	2010/2011	11279583	11957744	678161	5194151	6583	91.82	8.18	45118333	551657703	(3) (4)
241	Mohegan Sun	2010-12-17	2010/2011	13361550	13484207	122656	3430167	6404	92.06	7.94	53446202	673412992	(2) (3)

242 rows × 13 columns

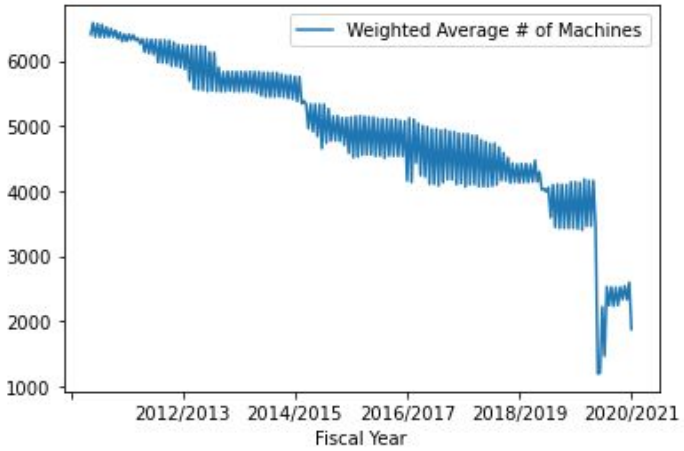
```
# keep columns touching on contributions

data_cont = data[['Casino', 'Date','Slot Machine Contributions to the State of Connecticut (1) (6)',
                  'Total Contributions (1)', 'eBonus Contribution (3)', 'eBonus Credits Redeemed at Slot Machines (2)',
                  'Weighted Average # of Machines']]
data_cont.head(10)
```

	Casino	Date	Slot Machine Contributions to the State of Connecticut (1) (6)	Total Contributions (1)	eBonus Contribution (3)	eBonus Credits Redeemed at Slot Machines (2)	Weighted Average # of Machines
0	Foxwoods	2020-12-15	5368328	6085094	0	1960609	1875
1	Mohegan Sun	2020-12-15	8059398	8087961	28563	3660386	2595
2	Foxwoods	2020-11-15	5445009	5445009	0	1568948	2332
3	Mohegan Sun	2020-11-15	7128718	7193431	64714	3395490	2541
4	Foxwoods	2020-10-15	6525003	6525003	0	2569762	2333
5	Mohegan Sun	2020-10-15	8955871	8955871	0	3768396	2522
6	Foxwoods	2020-09-15	6910884	6910884	0	2023444	2244
7	Mohegan Sun	2020-09-15	10068580	10068580	0	3578691	2520
8	Foxwoods	2020-08-15	7404364	7404364	0	2744859	2233
9	Mohegan Sun	2020-08-15	11392748	11392748	0	3811825	2523

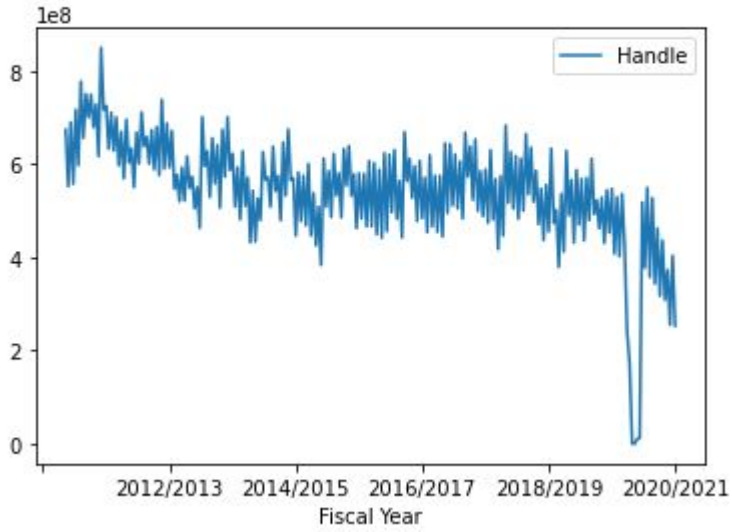
Visualizations

```
data_recent.plot('Fiscal Year',
                 'Weighted Average # of Machines').invert_xaxis()
```



This line graph shows the trent over the last 10 years of number of machines active in the two casinos. Obviously it took a hit during Covid, but it is interesting that it was steady declining well before the pandemic. This is likely due to the surge of online casinos.

```
data_recent.plot('Fiscal Year',
                 'Handle').invert_xaxis()
```

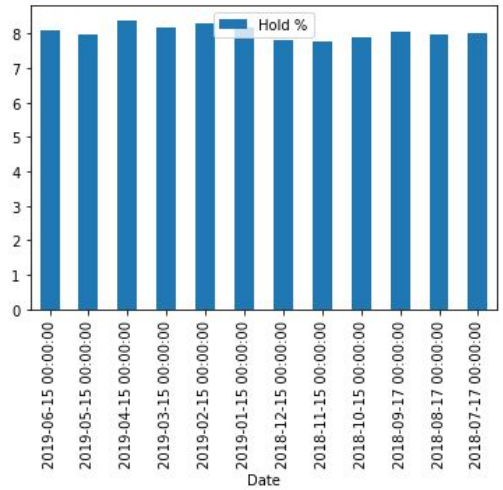


This is similar to the graph to the left in the sense that it shows interaction and participation in the casino over the last ten years. In the same fashion as the number of declining machines, the amount wagered is also slightly decreasing. I would be curious to see which causes the other. My guess is dropping handle causes fewer machines.

Visualizations (cont.)

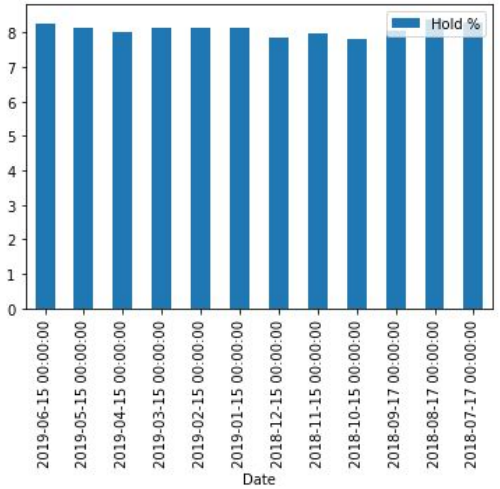
```
data_1819 = data[(data['Fiscal Year'] == '2018/2019')]
data_1819_fox = data_1819[data_1819.Casino == 'Foxwoods']
data_1819_fox.plot.bar('Date', 'Hold %')
```

```
<AxesSubplot:xlabel='Date'>
```




```
data_1819 = data[(data['Fiscal Year'] == '2018/2019')]
data_1819_sun = data_1819[data_1819.Casino == 'Mohegan Sun']
data_1819_sun.plot.bar('Date', 'Hold %')
```

```
<AxesSubplot:xlabel='Date'>
```

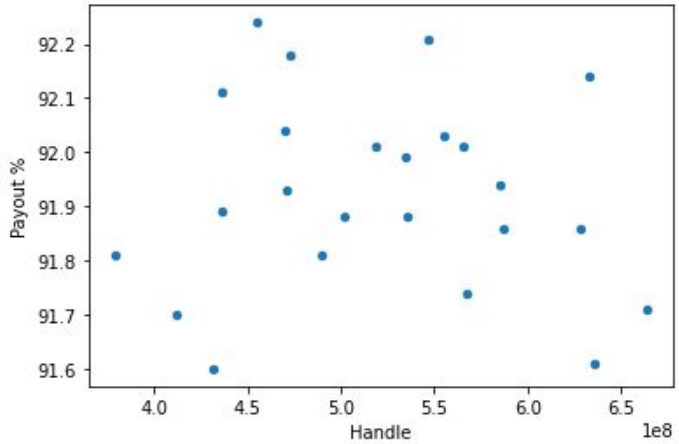


In these two bar graphs, I took a year long section of time from 2018/2019 (most recent data not affected by pandemic) and compared the percentage that the casino takes in as revenue and doesn't pay out. So as a gambling customer, we would want this number to be low. In both cases, the casinos seem to hold the lowest amount in the final few months of the calendar year. The fall/early season is the best time to be a gambler playing slots. Again, note that it isn't a crazy difference, but when you are gambling, every percentage point counts.

Visualizations (cont.)

```
data_1819.plot.scatter('Handle', 'Payout %')
```

```
<AxesSubplot:xlabel='Handle', ylabel='Payout %'>
```



This scatter plot is somewhat of an honesty test for the casinos. A non-rigged casino would see a truly random result. This is what we see here. Basically, what we can conclude from this is that there is no correlation between what the casino pays out based on what they bring in. If casinos changed the payouts based on customer's actions, essentially playing the system unfairly for their benefit, that would be bad sign. This is a great result both for the casino and gambler.

Reflection

- I learned a lot from this final project and from the class overall. It is fun doing practice problems but it is also fun to look at things from a real world perspective and have this feeling of closure/accomplishment to end the class. I really enjoyed how the class was laid out.
- As far as this project goes, it was neat to peak into the statistics of the payouts and to see actually how much money flows through these casinos. I solidified my rule of no slots. Payout is much higher playing basically any table game if you approach with a good strategy, and in my eyes are more entertaining to play.
- Casinos are pretty calculated operations and have patterns for everything; what months they pay the highest percentage back to customers, weather to open more machines or close, etc. It would be interesting to sit down and learn the reasoning behind these decisions. With such an insane amount of money flowing through these machines, as seen in prior slides, they must have a smart team dedicated to making sure all goes as planned and they make as much profit as they can while still providing a good experience.