index

January 10, 2022

1 Bernoulli and Binomial Distribution - Lab

1.1 Introduction

In this lab, you'll practice your newly gained knowledge on the Bernoulli and Binomial Distribution.

1.2 Objectives

You will be able to:

- Apply the formulas for the Binomial and Bernoulli distribution to calculate the probability of a specific event
- Use `numpy` to randomly generate Binomial and Bernoulli trials
- Use `matplotlib` to show the output of generated Binomial and Bernoulli trials

1.3 Apply the formulas for the Binomial and Bernoulli distributions

When playing a game of bowling, what is the probability of throwing exactly 3 strikes in a game with 10 rounds? Assume that the probability of throwing a strike is 25% for each round. Use the formula for the Binomial distribution to get to the answer. You've created this before, so we provide you with the function for factorials again:

[2]:
```
def factorial(n):
    prod = 1
    while n >= 1:
        prod = prod * n
        n = n - 1
    return prod
```

[4]:
```
p_3_strikes =factorial(10)/(factorial(3)*factorial(7))*0.25**3*(1-0.25)**7
p_3_strikes #answer = 0.2502822
```

[4]:
```
0.25028228759765625
```

Now, create a function for the Binomial distribution with three arguments n, p and k just like in the formula:

$$P(Y = k) = \binom{n}{k} p^k (1-p)^{(n-k)}$$

[5]:
```
def binom_distr(n,p,k):
    coef = factorial(n)/(factorial(k)*factorial(n-k))
    return coef * p**k * (1-p)**(n-k)
```

Validate your previous result by applying your new function.

[6]:
```
# Your code here
print(binom_distr(10,0.25,3))
```

```
0.25028228759765625
```

Now write a `for` loop along with your function to compute the probability that you have five strikes or more in one game. You'll want to use `numpy` here!

[12]:
```
import numpy as np
# Your code here
## Mine
l = []
for item in range(5,11):
    l.append(binom_distr(10,0.25,item))
np.array(l).sum()

### From GitHub Solution:

# import numpy as np
# prob = 0
# for i in np.arange(5,11):
#       prob += binom_distr(10,0.25,i)
# prob
```

[12]:
```
0.07812690734863281
```

1.4 Use a simulation to get the probabilities for all the potential outcomes

Repeat the experiment 5000 times.

[47]:
```
# leave the random seed here for reproducibility of results
import collections
np.random.seed(123)
lis = []

for item in range(5000):
    lis.append(np.random.binomial(10, 0.25))
dic = collections.Counter(lis)
sorted_dict = dict(sorted(dic.items()))
keys = np.array(list(sorted_dict.keys()))
values = np.array(list(sorted_dict.values()))
print(keys)
```

```
print(values)
```

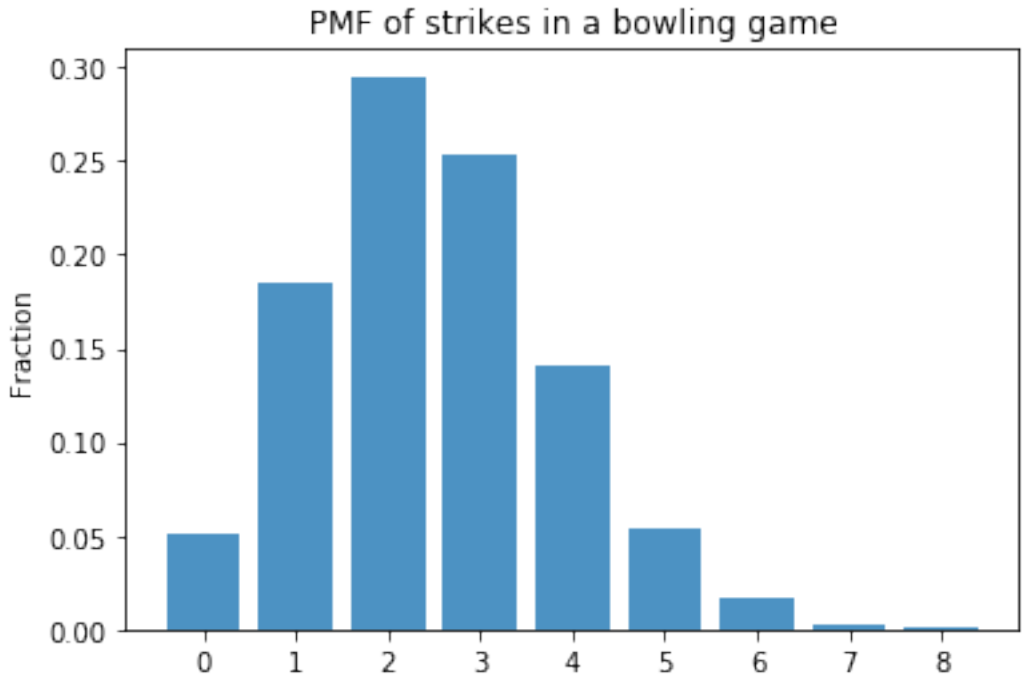
```
[0 1 2 3 4 5 6 7 8]
[ 310  941 1368 1286  707  297   78   11    2]
```

[48]:
```
# the results should look like this:
# [0 1 2 3 4 5 6 7 8]
# [ 310  941 1368 1286  707  297   78   11    2]
```

1.5 Visualize these results

Create the PMF using these empirical results (that is, the proportions based on the values we obtained running the experiment 5000 times).

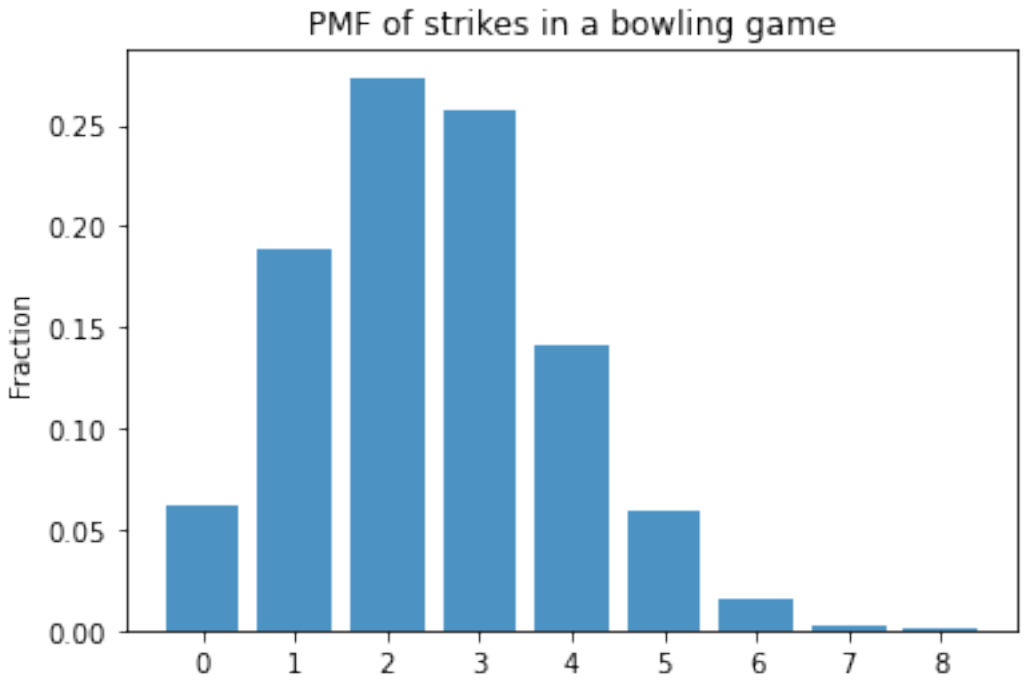
[49]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.bar(keys, values/5000, align = "center", alpha = 0.8)
plt.xticks(keys)
plt.ylabel('Fraction')
plt.title('PMF of strikes in a bowling game');
```



[50]:
```
### From GitHub Solution


np.random.seed(123)
```

```
n = 5000
iteration = []
for loop in range(n):
    iteration.append(np.random.binomial(10, 0.25))
    np_it = np.array(iteration)

values, counts = np.unique(np_it, return_counts=True)
print(values)
print(counts)

import matplotlib.pyplot as plt
%matplotlib inline
plt.bar(values, counts/5000, align='center', alpha=0.8)
plt.xticks(values)
plt.ylabel('Fraction')
plt.title('PMF of strikes in a bowling game');
```

```
[0 1 2 3 4 5 6 7 8]
[ 310  941 1368 1286  707  297   78   11    2]
```



You should see that, with a 25% strike hit rate, even when simulating 5000 times, an almost perfect and/or perfect game of 9 and 10 strikes didn't even occur once! If you change the random seed, however, you'll see that perfect games will show up occasionally.

Next, let's create the CDF based on these results. You can use `np.cumsum` to obtain cumulative

probabilities.

[ ]:
```
# Your code here
## Mine

keys = np.array(list(sorted_dict.keys()))
values_cum = np.cumsum(np.array(list(sorted_dict.values())))
```

1.6 Summary

Congratulations! In this lab, you practiced your newly gained knowledge of the Bernoulli and Binomial Distribution.