Mobile App to reduce Lottery Addiction

by Susan Fisher

A medical institute wants to create a mobile app to help lottery addicts. The app will estimate chances of winning based on various scenarios. The Data Science group will create the logical core and compute probabilities for the app.

The Data Science group will build functions that enable players to answer the following questions:

1. **The probability of winning the big prize in a single ticket i.e. all six numbers on the ticket match all the numbers drawn.**
2. **Function to compare players' ticket numbers to historical data.**
3. **The probability of winning the big prize in any number of tickets.**
4. **The probability of having 1, 2, 3, 4, or 5 winning numbers on a single ticket.**

The data set is Canada's national 6/49 lottery game data, from 1982 to 2018. For each lottery game, a single ticket has six numbers, and six numbers are drawn from a selection of 49 numbers, ranging from 1 to 49.

The data can be downlowded from kaggle at: https://www.kaggle.com/datascienceai/lottery-dataset

Data Exploration

In [1]:

```
# Read in the csv file as a dataframe, "lottery."
import numpy as np
import pandas as pd

lottery = pd.read_csv('C:/Users/Name/Documents/Python Scripts/DataSets/649.csv')
```

In [2]:

```
lottery.head(2)
```

Out[2]:

	PRODUCT	DRAW NUMBER	SEQUENCE NUMBER	DRAW DATE	NUMBER DRAWN 1	NUMBER DRAWN 2	NUMBER DRAWN 3	NUMBER DRAWN 4	NUMBER DRAWN 5	NUMBER DRAWN 6	BONUS NUMBE
0	649	1	0	6/12/1982	3	11	12	14	41	43	1
1	649	2	0	6/19/1982	8	33	36	37	39	41	

In [3]:

```
lottery.tail(3)
```

Out[3]:

	PRODUCT	DRAW NUMBER	SEQUENCE NUMBER	DRAW DATE	NUMBER DRAWN 1	NUMBER DRAWN 2	NUMBER DRAWN 3	NUMBER DRAWN 4	NUMBER DRAWN 5	NUMBER DRAWN 6	BO NUM
3662	649	3589	0	6/13/2018	6	22	24	31	32	34	
3663	649	3590	0	6/16/2018	2	15	21	31	38	49	
3664	649	3591	0	6/20/2018	14	24	31	35	37	48	

In [4]:

```
lottery.columns
```

Out[4]:

```
Index(['PRODUCT', 'DRAW NUMBER', 'SEQUENCE NUMBER', 'DRAW DATE',
       'NUMBER DRAWN 1', 'NUMBER DRAWN 2', 'NUMBER DRAWN 3', 'NUMBER DRAWN 4',
       'NUMBER DRAWN 5', 'NUMBER DRAWN 6', 'BONUS NUMBER'],
      dtype='object')
```

In [5]:

```
lottery.shape
```

Out[5]:

```
(3665, 11)
```

Core Functions

The core functions, factorial and combinations, will be used repeatedly in the project.

In [6]:

```
# Function to compute Factorial
# Factorial example: 5! = 5*4*3*2*1 = 120
def factorial(n):
    final_product = 1
    for x in range(n, 0, -1):
        final_product *= x
    return final_product
```

In [7]:

```
# Function to compute Combinations
# Combinations: the number of combinations independent of order
def combinations(n, k):
    num = factorial(n)
    den = factorial(k) * factorial(n-k)
    return num/den
```

Create Functions to answer various players' questions

(1) Create function to compute probability of winning the big prize with a single ticket.

For each drawing, six numbers are drawn from 49, and if all six numbers match the players's ticket, then the player wins the big prize.

The Engineering team has informed the Data Science group of the following details of the app:

- **Inside the app, the player inputs six different numbers from 1 to 49.**
- **Under the hood, the six numbers will come as a Python list, which will serve as the single input to our function.**
- **The function needs to print the probability in a friendly way - in a way that will be understood by those who don't have probability background.**

Below, one_ticket_probability() function takes in a list of six unique numbers and prints the probability of winning in a friendly way.

In [8]:

```
# Function that computes probability of winning big prize on a single ticket,
#   or getting all 6 numbers on the ticket match the winning numbers

def one_ticket_probability(user_numbers):
    total_outcomes = combinations(49,6)
    p_one_ticket = 1/total_outcomes * 100
    return print("Your chances of winning the big prize with your numbers {} is {:0.7
f}%.  In other words, you have a one in {:,} chance of winning.".format(
        user_numbers, p_one_ticket, int(total_outcomes)))
```

In [9]:

```
# Test one_ticket_probability function
one_ticket_probability([5, 11, 17, 23, 39, 42])
```

```
Your chances of winning the big prize with your numbers [5, 11, 17, 23, 39, 42] is 0.0
000072%.  In other words, you have a one in 13,983,816 chance of winning.
```

(2) Create function to determine if players' ticket numbers have won the big prize in the past.

The function compares the players' number combination to Canada's historical 6/49 lottery data, and will return the number of times that combination has won the big prize in the past. The data contains 3,665 drawings from 1982 to 2018. Each row of data is a drawing where the each winning number is in a column, "NUMBER DRAWN 1," "NUMBER DRAWN 2"..., "NUMBER DRAWN 6."

For the player's ticket combination, the function also returns the probability of winning the big prize in the next drawing.

The engineering team informed us that we need to be aware of the following:

- **Inside the app, the player inputs six different numbers from 1 to 49.**
- **Under the hood, the six numbers will come as a Python list and serve as an input to our function.**

The engineering team wants us to write a function that prints:

- **the number of times the combination selected occurred in the Canada data set.**
- **the probability of winning the big prize in the next drawing with that combination.**

First, a function to extract all six winning numbers and convert them to a Python set.

In [10]:

```
def extract_numbers(row):
    winning_numbers = set()
    for x in range(4,10):
        winning_numbers.add(row.iloc[x])
    return winning_numbers

winning_numbers = lottery.apply(extract_numbers, axis=1)
winning_numbers.head()
```

Out[10]:

```
0    {3, 41, 11, 12, 43, 14}
1    {33, 36, 37, 39, 8, 41}
2     {1, 6, 39, 23, 24, 27}
3     {3, 9, 10, 43, 13, 20}
4    {34, 5, 14, 47, 21, 31}
dtype: object
```

Below, the function, check_historical_occurence, takes in the players' six numbers and the historical winning numbers, and prints the number of past wins and probability of winning in the future.

In [11]:

```
def check_historical_occurence(user_numbers, winning_numbers):
```

```
def check_historical_occurence(user_numbers, winning_numbers):
    # user numbers: a Python list
    # winniner numbers: pandas series

    user_numbers = set(user_numbers)
    check_occurrence = user_numbers == winning_numbers
    n_occurrences = check_occurrence.sum()

    if n_occurrences == 0:
        return print("For your number combination {}, in the past, there were no winn
ers.  Your chances of winning the big prize in the next drawing using your numbers {}
are 0.0000072%, or 1 in 13,983,816 chance to win.".format(user_numbers,user_numbers))
    else:
        return print("For your number combindation {}, in the past, there were {} win
ners.  Your chances of winning the big prize in the next drawing using your numbers {
} are 0.0000072%, or 1 in 13,983,816 chance to win.".format(user_numbers, n_occurrenc
es, user_numbers))
```

In [12]:

```
# Test check_historical_occurence function

# random numbers
test1 = check_historical_occurence([5, 11, 17, 23, 39, 42], winning_numbers)

# line #1 of historical data: should be at least 1 match
test2 = check_historical_occurence([3, 41, 11, 12, 43, 14], winning_numbers)
print(test1, test2, sep='\n')
```

```
For your number combination {5, 39, 42, 11, 17, 23}, in the past, there were no winne
rs.  Your chances of winning the big prize in the next drawing using your numbers {5,
39, 42, 11, 17, 23} are 0.0000072%, or 1 in 13,983,816 chance to win.
For your number combindation {3, 41, 11, 12, 43, 14}, in the past, there were 1 winne
rs.  Your chances of winning the big prize in the next drawing using your numbers {3,
41, 11, 12, 43, 14} are 0.0000072%, or 1 in 13,983,816 chance to win.
None
None
```

(3) Create function to compute probability of winning the big prize for any number of tickets.

If a player wants to puchase more than one lottery ticket, or five, or ten, etc. then player can find the probability of winning the big prize based on number of lottery tickets.

The engineering team passed on the following information:

- **The user will input the number of different tickets they want to play (without inputting the specific combinations they intend to play).**
- **Our function will see an integer between 1 and 13,983,816 (the maximum number of different tickets).**
- **The function should print information about the probability of winning the big prize depending on the number of different tickets played.**

The function, multi_ticket_probability, takes in the number of lottery tickets, and prints the probability of winning the big prize based on the number of tickets.

In [13]:

```
# Function takes in an integer from 1 to 13,983,816 (max # of combinations)
def multi_ticket_probability(n_tickets):
    total_outcomes = combinations(49,6)
    p_n_tickets = n_tickets/total_outcomes * 100
    return print("Your chances of winning the big prize with {} of tickets is {:0.7f}
%.  In other words, you have a one in {:,} chance of winning.".format(
        n_tickets,p_n_tickets, int(total_outcomes)))
```

In [14]:

```
# Test multi_ticket_probability function:
```

```
n_tickets = [1, 10, 100, 1000, 10**5, 6991908, 13983816]
for number in n_tickets:
    multi_ticket_probability(number)
    print('-----------------')
```

```
Your chances of winning the big prize with 1 of tickets is 0.0000072%.  In other word
s, you have a one in 13,983,816 chance of winning.
-----------------
Your chances of winning the big prize with 10 of tickets is 0.0000715%.  In other word
s, you have a one in 13,983,816 chance of winning.
-----------------
Your chances of winning the big prize with 100 of tickets is 0.0007151%.  In other wor
ds, you have a one in 13,983,816 chance of winning.
-----------------
Your chances of winning the big prize with 1000 of tickets is 0.0071511%.  In other wo
rds, you have a one in 13,983,816 chance of winning.
-----------------
Your chances of winning the big prize with 100000 of tickets is 0.7151124%.  In other
words, you have a one in 13,983,816 chance of winning.
-----------------
Your chances of winning the big prize with 6991908 of tickets is 50.0000000%.  In othe
r words, you have a one in 13,983,816 chance of winning.
-----------------
Your chances of winning the big prize with 13983816 of tickets is 100.0000000%.  In ot
her words, you have a one in 13,983,816 chance of winning.
-----------------
```

(4) Create function to compute probability of matching 2, 3, 4, or 5 winning numbers on a single ticket.

In the 6/49 lottery, there is a smaller prize for matching two, three, four, or five numbers.

The engineering details are:

- **Inside the app, the user inputs:**
  - **six different numbers from 1 to 49**
  - **an integer between 2 and 5 that represents the number of winning numbers expected**
- **the function prints information about the probability of having the inputted number of winning numbers.**

The function, probability_less_6, only takes in the number of winning numbers that the player expects, number between 2 - 5.
This probability is independent of the players' number combination.

The function computes the probability of that a player's ticket matches exactly the given number of winning numbers i.e. *exactly* two numbers, *exactly* three numbers..., or *exactly* five numbers, and not *at least* two numbers or *at least* five numbers.

In [15]:

```
# Function probably of having x number of winning numbers
def probability_less_6(n_win_numbers):
    n_combinations = combinations(6,n_win_numbers)
    n_combinations_remaining = combinations(43, 6 - n_win_numbers)
    success_outcomes = n_combinations * n_combinations_remaining
    total_combinations = combinations(49,6)
    p_n_win_numbers = success_outcomes / total_combinations * 100
    print("The probability that you will receive {} winning numbers on a single ticke
t is {:0.4f}%.  Or you have a 1 in {} chance of winning.".format(n_win_numbers, p_n_w
in_numbers, int(n_combinations)))
```

In [16]:

```
# Test probability_less_6 function
n_win_numbers = [2,3,4,5]
for number in n_win_numbers:
    probability_less_6(number)
```

```
    print('-----------------')
```

```
The probability that you will receive 2 winning numbers on a single ticket is 13.2378
%.  Or you have a 1 in 15 chance of winning.
-----------------
The probability that you will receive 3 winning numbers on a single ticket is 1.7650%
.  Or you have a 1 in 20 chance of winning.
-----------------
The probability that you will receive 4 winning numbers on a single ticket is 0.0969%
.  Or you have a 1 in 15 chance of winning.
-----------------
The probability that you will receive 5 winning numbers on a single ticket is 0.0018%
.  Or you have a 1 in 6 chance of winning.
-----------------
```

Summary

For the first version of the app, the following functions were built:

1. **one_ticket_probability() — calculates the probability of winning the big prize with a single ticket**
2. **check_historical_occurrence() — checks whether a certain combination has occurred in the Canada lottery data set**
3. **multi_ticket_probability() — calculates the probability for any number of tickets between 1 and 13,983,816**
4. **probability_less_6() — calculates the probability of having two, three, four or five winning numbers**

Subsequent versions of the app could include:

- **Combine the one_ticket_probability() and check_historical_occurrence() to return probability with historical occurrence.**
- **Create a function similar to probability_less_6() that computes the probabilty of *at least* two, three, four, or five numbers. (the number of successful outcomes for having at least four winning numbers is sum of**
  - **number of successful outcomes for having *exactly* four winning numbers**
  - **number of successful outcomes for having *exactly* five winning numbers**
  - **number of successful outcomes for having *exactly* six winning numbers**