Homework 1

Deadline: 2019/03/12 (Tuesday) 23:59

Problem 1: The Blackjack Game (hw1_blackjack.py)

One approach to strengthen your programming skills is the game development. In this problem, therefore, you are asked to develop a Blackjack (二十一點) game, which allows a user to play against a computer dealer (莊家). First of all, let me remind you the rules of Blackjack as follows:

- First, both player and dealer receive two cards from a shuffled deck of 52 cards (一副洗好的牌).
- After the first two cards are delivered to dealer and player, the player is asked if they want another card (called "hitting" (再抽)), or if he/she are happy with the cards they have already (called "staying" (停止)). The goal is to make the sum of the player's card values as close to 21 as possible, without going over. If the player makes 21 exactly, he/she has Blackjack, which cannot be beat. If the player goes over 21, he/she "busts" (爆掉) and lose the game. The player is allowed to stop hitting at any time point.
- The number cards (2 through 10) are worth the number displayed, face cards (JACK, QUEEN, and KING) are worth 10, and an ACE can be worth either 1 or 11. For example, if the first two cards in the player's hand are a JACK and an ACE, you needs to count the ACE as 11 because 10 + 11 = 21 and have Blackjack. But if the player has already had the cards worth 18, and decides to hit and gets an ACE, you should count such ACE as 1, because counting it as 11 would put the player at 29 and get busted.
- Once the player's hand is finished, the dealer will try to do the same things. The dealer must keep hitting until he/she gets to 17. If the dealer gets above 17 without busting, then they can stay.

Finally, the game is settling by simple rules: (1) if the player has blackjack, he/she wins the game, unless the dealer also has Blackjack, in which case the game is a tie. (2) If the dealer busts and the player does not, the player wins. (3) If the player busts, the dealer wins. (4) If the player and the dealer both do not bust, whoever is closest to 21 wins.

To help you implement the Blackjack game, we divide the Blackjack game into six of the division of labor in terms of the program logic, as described in the following. The program logic forms the game engine, enforcing the rules of the game.

(a) **Preprocessing.** Create the deck of cards by combing the 13 ranks (ACE, 2, 3, 4, 5, 6, 7, 8, 9, 10, JACK, QUEEN, and KING) with 4 suits (SPADE, HEART, DIAMOND, and CLUB). There are 52 cards. Then before the game starts, you need to shuffle the cards.

(b) **Settle the Stage.** Send the player's and dealer's first two cards by random. In other words, four

cards that will be owned by the player and the dealer respectively are taken from the 52 cards.

(c) **Compute the Total Value.** Your program has to reason the total value of cards in the hand. Two things must be done here. First, your program needs to reason the value of a card. For example, ACE = 1 or 11, JACK/QUEEN/KING = 10. Second, by default you program considers ACE as 11. Then your program needs to count up the number of ACEs in the hand, then checks to see that the total value in the hand is higher than 21. If it is, and there is an ACE in the hand, your program needs to consider the ACE as 1, and re-computes the total value. If not, your program obtains the current total value of the player. Otherwise, if there is a second ACE in the hand, your program again considers the second ACE as 1 and checks again. We continue to do this until we have exhausted all the ACEs in the player's hand. Finally, your program obtains the total value in the player's hand.

(d) **Game Logic.** The next thing we code is the logic of gameplay. Your program has to ask the player whether he/she would like to hit or stay, and continue to ask them until they bust, or they decide to stay. One piece of information that's crucial to the player's decision is their current cards in the hands. Therefore, we need to print the player's cards in the hand and current total value each time before asking for their response and input. As long as the player's hand isn't a bust, we ask for the player's input: hit (再抽) = 1, stay (停止) = 0. If the player wants to hit, we again deliver him/her a new card by randomly drawn a card from the **remaining** cards in the deck, and immediately print the newly drawn card. If they ask to stay, the action of the player will be terminated, and your program moves on to the dealer. In the meanwhile, your program needs to calculate the player's value, and the dealer's current value (on his/her first two cards). If the player's hand isn't a bust, we print the dealer's total value and current cards. Then, while the dealer's hand is worth less than 17, the dealer is made to hit. Each time each dealer hits, you need to print the new drawn card. By design, this loop halts when the dealer exceeds 17.

(e) **Determine the Winner.** At this point, the game is nearly finished. All that remains is to check the player's total value and the dealer's total value against the list of scoring rules we outlined above. To start, we obtain the label, and number for the score of the dealer's hand. Next, we simply enumerate the possible end states, and ask which of them our game satisfies. Based on the outcome, we print to the screen declaring the winner.

(f) **Ask the Player to play or not.** If the player does not want to play again, quit the Blackjack game. If the player wants to play again, your program needs to rearrange the deck of cards to let the player play one more time.

You are asked to (1) write functions for each component of the game, and
(2) write comments to describe the meaning of each part.

Sample Input and Output

Example 1

```
c:\Python35-32\workspace>python hw4_p2.py

Your current value is 19
with the hand: 10-HEART, 9-HEART

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 11
with the hand: 3-CLUB, 8-DIAMOND

Dealer draws QUEEN-HEART

Dealer's current value is Blackjack! (21)
with the hand: 3-CLUB, 8-DIAMOND, QUEEN-HEART

*** Dealer wins! ***

Want to play again? (y/n): y

-------------------------------------


Your current value is 11
with the hand: 8-CLUB, 3-SPADE

Hit or stay? (Hit = 1, Stay = 0): 1
You draw 6-DIAMOND

Your current value is 17
with the hand: 8-CLUB, 3-SPADE, 6-DIAMOND

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 16
with the hand: JACK-HEART, 6-CLUB

Dealer draws 3-HEART

Dealer's current value is 19
with the hand: JACK-HEART, 6-CLUB, 3-HEART

*** Dealer wins! ***

Want to play again? (y/n): y

-------------------------------------


Your current value is 8
with the hand: 5-HEART, 3-DIAMOND

Hit or stay? (Hit = 1, Stay = 0): 1
You draw ACE-CLUB

Your current value is 19
with the hand: 5-HEART, 3-DIAMOND, ACE-CLUB

Hit or stay? (Hit = 1, Stay = 0): 1
You draw 8-SPADE

Your current value is 17
with the hand: 5-HEART, 3-DIAMOND, ACE-CLUB, 8-SPADE

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 15
with the hand: KING-HEART, 5-SPADE

Dealer draws KING-CLUB

Dealer's current value is Bust! (>21)
with the hand: KING-HEART, 5-SPADE, KING-CLUB

*** You beat the dealer! ***

Want to play again? (y/n): n
```

Example 2

```
c:\Python35-32\workspace>python hw4_p2.py

Your current value is 11
with the hand: 5-DIAMOND, 6-HEART

Hit or stay? (Hit = 1, Stay = 0): 1
You draw 10-HEART

Your current value is Blackjack! (21)
with the hand: 5-DIAMOND, 6-HEART, 10-HEART

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 13
with the hand: 9-HEART, 4-SPADE

Dealer draws QUEEN-DIAMOND

Dealer's current value is Bust! (>21)
with the hand: 9-HEART, 4-SPADE, QUEEN-DIAMOND

*** You beat the dealer! ***

Want to play again? (y/n): y

-------------------------------------


Your current value is 11
with the hand: 9-DIAMOND, 2-DIAMOND

Hit or stay? (Hit = 1, Stay = 0): 1
You draw 9-SPADE

Your current value is 20
with the hand: 9-DIAMOND, 2-DIAMOND, 9-SPADE

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 16
with the hand: JACK-HEART, 6-CLUB

Dealer draws 2-HEART

Dealer's current value is 18
with the hand: JACK-HEART, 6-CLUB, 2-HEART

*** You beat the dealer! ***

Want to play again? (y/n): y

-------------------------------------


Your current value is 20
with the hand: QUEEN-SPADE, 10-SPADE

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 10
with the hand: 8-SPADE, 2-DIAMOND

Dealer draws JACK-DIAMOND

Dealer's current value is 20
with the hand: 8-SPADE, 2-DIAMOND, JACK-DIAMOND

*** You tied the dealer, nobody wins. ***

Want to play again? (y/n): n
```

Example 3

```
c:\Python35-32\workspace>python hw4_p2.py

Your current value is 8
with the hand: 5-CLUB, 3-DIAMOND

Hit or stay? (Hit = 1, Stay = 0): 1
You draw 4-DIAMOND

Your current value is 12
with the hand: 5-CLUB, 3-DIAMOND, 4-DIAMOND

Hit or stay? (Hit = 1, Stay = 0): 1
You draw 8-DIAMOND

Your current value is 20
with the hand: 5-CLUB, 3-DIAMOND, 4-DIAMOND, 8-DIAMOND

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 12
with the hand: JACK-CLUB, 2-CLUB

Dealer draws QUEEN-HEART

Dealer's current value is Bust! (>21)
with the hand: JACK-CLUB, 2-CLUB, QUEEN-HEART

*** You beat the dealer! ***

Want to play again? (y/n): y

-------------------------------------


Your current value is 14
with the hand: 3-HEART, ACE-CLUB

Hit or stay? (Hit = 1, Stay = 0): 1
You draw 4-CLUB

Your current value is 18
with the hand: 3-HEART, ACE-CLUB, 4-CLUB

Hit or stay? (Hit = 1, Stay = 0): 1
You draw ACE-SPADE

Your current value is 19
with the hand: 3-HEART, ACE-CLUB, 4-CLUB, ACE-SPADE

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 20
with the hand: 10-SPADE, QUEEN-CLUB

*** Dealer wins! ***

Want to play again? (y/n): y

-------------------------------------


Your current value is Blackjack! (21)
with the hand: ACE-CLUB, QUEEN-DIAMOND

Hit or stay? (Hit = 1, Stay = 0): 1
You draw 4-HEART

Your current value is 15
with the hand: ACE-CLUB, QUEEN-DIAMOND, 4-HEART

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 14
with the hand: 4-SPADE, 10-SPADE

Dealer draws JACK-SPADE

Dealer's current value is Bust! (>21)
with the hand: 4-SPADE, 10-SPADE, JACK-SPADE

*** You beat the dealer! ***

Want to play again? (y/n): n
```

Example 4

```
c:\Python35-32\workspace>python hw4_p2.py

Your current value is Blackjack! (21)
with the hand: ACE-CLUB, JACK-HEART

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 8
with the hand: 2-CLUB, 6-CLUB

Dealer draws 4-CLUB
Dealer draws 2-DIAMOND
Dealer draws 2-SPADE
Dealer draws 3-HEART

Dealer's current value is 19
with the hand: 2-CLUB, 6-CLUB, 4-CLUB, 2-DIAMOND, 2-SPADE, 3-HEART

*** You beat the dealer! ***

Want to play again? (y/n): y

-------------------------------------

Your current value is 20
with the hand: JACK-DIAMOND, QUEEN-HEART

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 20
with the hand: QUEEN-DIAMOND, KING-DIAMOND

*** You tied the dealer, nobody wins. ***

Want to play again? (y/n): y

-------------------------------------


Your current value is 12
with the hand: ACE-SPADE, ACE-HEART

Hit or stay? (Hit = 1, Stay = 0): 1
You draw 9-HEART

Your current value is Blackjack! (21)
with the hand: ACE-SPADE, ACE-HEART, 9-HEART

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is 16
with the hand: QUEEN-HEART, 6-DIAMOND

Dealer draws 4-SPADE

Dealer's current value is 20
with the hand: QUEEN-HEART, 6-DIAMOND, 4-SPADE

*** You beat the dealer! ***

Want to play again? (y/n): y

-------------------------------------


Your current value is 18
with the hand: 8-HEART, 10-CLUB

Hit or stay? (Hit = 1, Stay = 0): 0

Dealer's current value is Blackjack! (21)
with the hand: ACE-DIAMOND, JACK-CLUB

*** Dealer wins! ***

Want to play again? (y/n): y

-------------------------------------


Your current value is 19
with the hand: 8-SPADE, ACE-HEART

Hit or stay? (Hit = 1, Stay = 0): 1
You draw 4-SPADE

Your current value is 13
with the hand: 8-SPADE, ACE-HEART, 4-SPADE

Hit or stay? (Hit = 1, Stay = 0): 1
You draw QUEEN-HEART

Your current value is Bust! (>21)
with the hand: 8-SPADE, ACE-HEART, 4-SPADE, QUEEN-HEART

*** Dealer wins! ***

Want to play again? (y/n): n
```

Problem 2: Kobe Shot Data Analysis (hw1_kobe.py)

Kobe Bryant marked his retirement from the NBA by scoring 60 points in his final game as a Los Angeles Laker on Wednesday, April 12, 2016. Drafted into the NBA at the age of 17, Kobe earned the sport's highest accolades throughout his long career. Using 20 years of data on Kobe's swishes and misses, many people like to analyze how his shots find the bottom of the net.

In this homework, based on the data we provided you "**kobe.csv**", you can try to be the Kobe shot data analyst. Please answer the following questions based on the dataset. You are asked to write a function for each of the following question.

(1) 計算 Kobe 對戰火箭隊(HOU)的平均兩分球與三分球命中率。
Hint: opponent, shot_type, shot_made_flag

(2) 列出 Kobe 對戰過的球隊中，使得 Kobe 平均得分最低的前五支球隊，並顯示平均得分。
Hint: opponent, shot_made_flag, shot_type

(3) 列出 Kobe 在季後賽最後 3 分鐘內得分最高的前五場球賽，並顯示其得分。
Hint: opponent, playoffs, shot_made_flag, shot_type, minutes_remaining

(4) 列出 Kobe 各球季季後賽中，比賽最後 1 分鐘內的 Jump Shot 命中率，按球季先後排序。
Hint: action_type, playoffs, game_date, seconds_remaining, shot_made_flag, shot_type

(5) 計算 Kobe「得分命中率 33%以上的最長連續場數」前 3 名，列出場數以及起訖日期。
Hint: game_date, shot_type, shot_made_flag

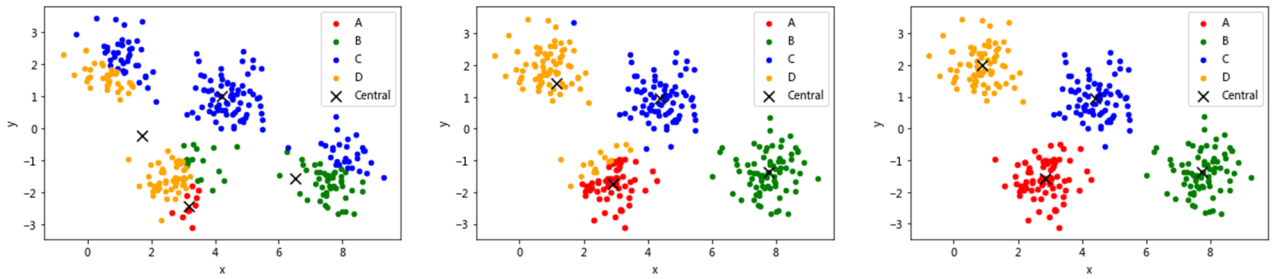
(6) 計算 Kobe「上半場得分多於下半場」且命中率最低的前 3 名場次，列出日期、對手、上下半場得分差、該場得分，按命中率由上而下排序。
Hint: period, shot_made_flag, opponent, game_date

(7) 計算 Kobe「投籃連續失手最多球」之前 3 名場次，列出日期、對手、連續失手球數、及該場得分，按失手次數由上而下排序。
Hint: game_id, shot_made_flag, opponent

Problem 3: K-means Clustering Implementation (hw1_kmeans.ipynb)

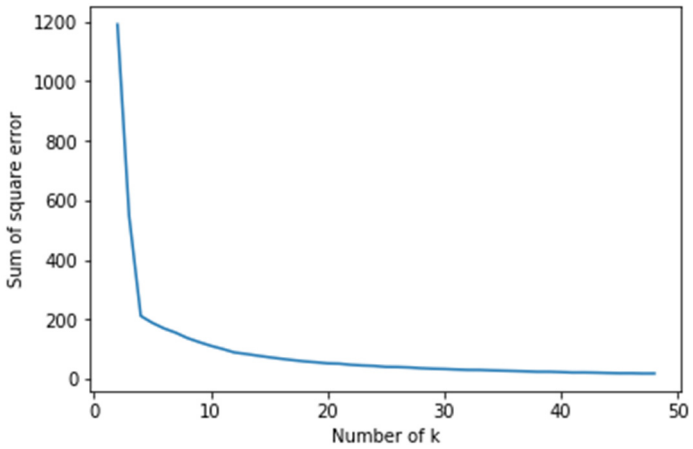
Your task is to use Python (along with numpy and Pandas) to implement the well-known clustering algorithm, K-means, based on a synthetic dataset **cdata.csv**. This dataset contains two data columns, "X" and "Y", and one "cluster" column (1, 2, 3, and 4). In implementing K-means, you need to use "X" and "Y" as **features** for clustering while the "cluster" column is for your validation. Note that it is not necessary to perfectly clustering all of the data points into clusters. Also note that the "cluster" column cannot be used in clustering.

(1) Randomly select data points as the initialized centroids. By default, please set K=4. Report and plot the process until convergence. The centroids also need to be plotted. An example is shown below. Note that it may not have 3 rounds (it can be 4 or 5 rounds, depend on initialized centroids).

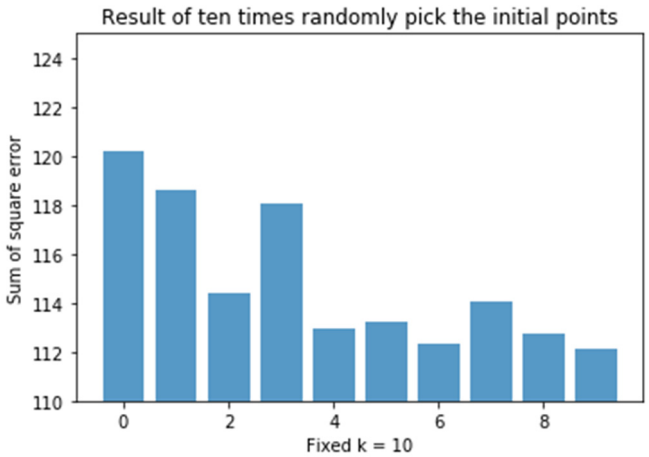


Round 1 Round 2 Round 3

(2) Re-execute your K-means clustering algorithm by changing K from 2 to 50 (from 2 to 10 is also okay). Plot the K value (x-axis) vs. the value of Sum of Squared Error (SSE) (y-axis) as below. Note that it is reasonable and acceptable if the curve is 凹凸不平. ☺



(3) Try 10 times of randomly initialized centroids, and plot their SSE values (y-axis) such as below.



Important Notes

This is a homework for each **individual**. You are asked to write comments to describe the meaning of each part of your codes in hw1.py.

How to Submit Your Homework?

Submission in NCKU Moodle. Before submitting your homework, please zip your coding files in a zip file, and name the file as "StudentID_hw1.zip". For example, if your StudentID is H12345678, then name your file name as "H12345678_ hw**1**.zip". Then submit your zipped file using NCKU Moodle platform http://moodle.ncku.edu.tw .

Have Questions about This Homework?

Please feel free to visit TAs, and ask/discuss any questions in their office hours. We will be more than happy to help you.