Authors: Disha Satyanarayan (22202275) , Sangeeta Panigrahi (22202158)
Affiliations: University College Dublin, School of Mathematics & Statistics

SIMULATE SINGLE DECK BLACKJACK

INTRODUCTION

- Blackjack, aka "21" is an intricate blend of luck, strategy, mathematics
- Element of randomness - challenges players to make strategic decisions based on incomplete information
- Mathematical significance lies in the pursuit of optimal strategies that maximize the player's chances of winning
- Given the complexity of the game's strategic landscape, simulation (multiple times) serves as a valuable tool for gaining insights.
- This project aims to simulate different strategies of Blackjack and compare the results

MOTIVATION

- Simulate different strategies - gain insights into how different approaches fare in real gameplay.
- Analysing outcomes - help understand impact of each strategy on player's winning potential
- Comparison will highlight the importance of skill and calculations in making strategic decisions.
- Provides a clear picture of how different strategies stack up against each other

AIM

- **STRATEGY**: Create functions to deal/stay/hit and Implement different strategies. (Inexperienced/ Simple/ Complex)
- **SIMULATE**: Simulate one hand of blackjack using all 3 strategies
- **REPEAT**: Run multiple times & aggregate results

PROCESS

- Tool used – Python
- Number of decks: Single deck
- Created functions to perform different actions like calculate hand value, create deck and shuffle, hit, stay, deal cards, play, simulate, etc
- Using the functions, implemented 3 different strategies: Inexperienced, Simple, Complex
- Inexperienced player logic: The logic for the inexperienced player's decision to hit or stay is based on random choices. (random.choice(['hit', 'stand']))
- Simple logic: Player's decisions are based on the value of their hand. The player hits if their hand value is less than 17 and stands otherwise
- Complex logic: Demonstrates a more advanced simulation of Blackjack, incorporating splitting logic and a strategy based on dealer's card (upcard).
- The randomness required for the game is satisfied using 'random' package available in python - to generate pseudorandom numbers (random.shuffle, random.choice, etc)
- The code for each strategy asks the user to input the number of simulations required, accordingly runs it and outputs the result & plot.
- Final output plots a clustered bar plot showing the results of all 3 strategies after running each 100,000 times.
- An interactive HTML webpage of blackjack is also created – to play in real-time!

TYPES OF STRATEGY

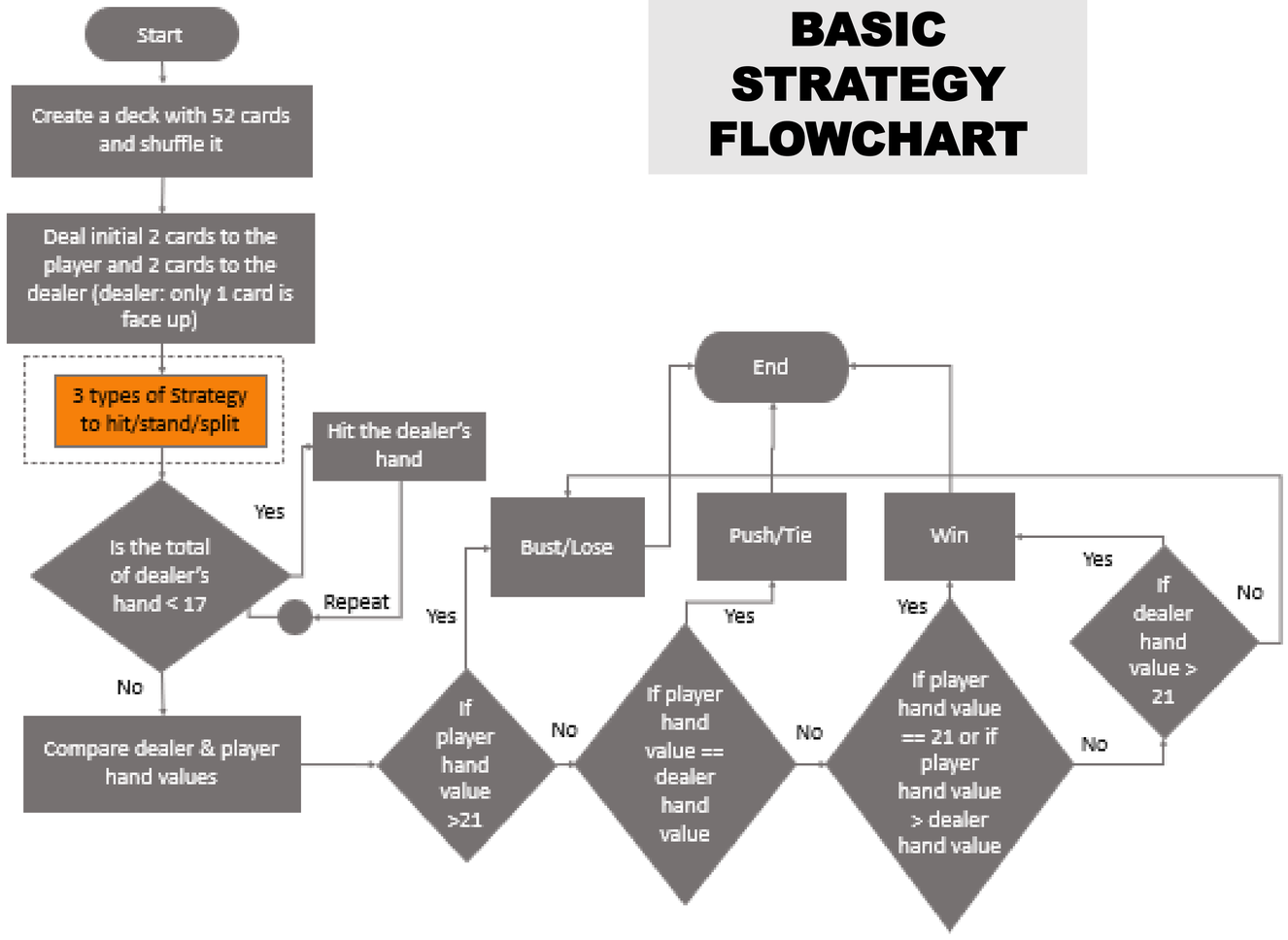
Soft hands: If the hand's value is over 21 and it contains an Ace (11), the Ace's value is adjusted to 1 to prevent busting (going over 21).

COMPLEX (with splitting) Strategy:
- If player's hand value >= 17, stand.
- If player's hand value < = 11 or less, hit.
- Split if hand contains a pair of 2s, 3s, 7s, or 8s:
  - Split the hand, simulate both split hands' outcomes. Compare outcomes, choose better one (non bust, higher)
- If player's hand value = 12, dealer's upcard is 4, 5, or 6, stand.
- If player's hand value >= 13 and <= 16, dealer's upcard is 2, 3, 4, 5, or 6, stand.
- If player's hand value is 9, dealer's upcard is 2, 3, 4, 5, or 6, hit.
- If player's hand value = 10, dealer's upcard != 10 or != Ace, hit.
- If player's handvalue = 11, dealer's upcard != Ace, hit.
- Otherwise, stand.

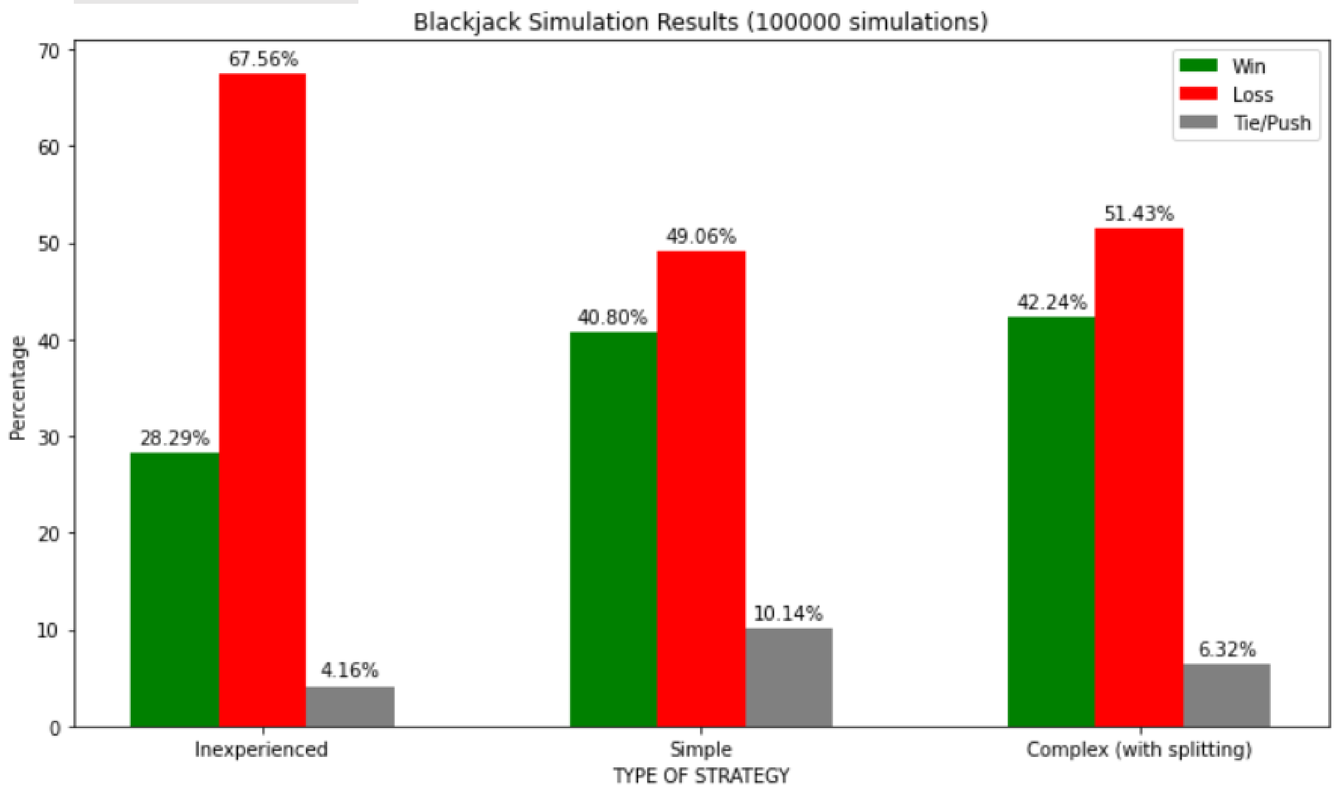
SIMPLE Strategy:
Player repeatedly draws cards ("hits") until player hand value >= 17.
Else stand

INEXPERIENCED Player's Strategy:
Player randomly chooses either to "hit" (draw a card) or "stand" (end their turn) using random.choice().

BASIC STRATEGY FLOWCHART



RESULTS



These simulation results provide insights into the performance of each simulator's strategies and their respective impact on the gameplay outcomes.

- Inexperienced player strategy: Randomness often leads to unfavourable decisions; player tends to bust more frequently (high loss percentage) and make suboptimal decisions (lower win percentage).
- The simple logic strategy increases the player's chances of having a stronger hand but also exposes them to a higher risk of busting.
  - Higher win percentage – due to player's tendency to reach higher hand values.
  - Higher tie percentage - due to the simple strategy leading to similar hand values more often.
- Complex logic (with Splitting Strategy):
  - Higher Win Percentage**:** The complex strategy likely makes better use of opportunities to split pairs, which can result in stronger hands - more potential wins.
  - More risks involved - the strategy might take calculated risks that, leading to busting. For instance, hitting on certain hands could backfire, contributing to a higher loss percentage.
  - Lower Tie Percentage**:** The complex strategy likely prioritizes actions that avoid ties by actively seeking wins or losses.

REFERENCES

- Creating And Hosting Blackjack Simulator Using Python and Streamlit**,** Michael-Camden-Smith, https://python.plainenglish.io/a-lesson-in-blackjack-and-python-hosted-by-streamlit-b927147bec8d
- Cai, Xinyi. "The House Edge and the Basic Strategy of Blackjack." *SHS Web of Conferences*. Vol. 148. EDP Sciences, 2022.
- Kakvi, Saqib A. "Reinforcement learning for blackjack." Entertainment Computing ICEC 2009: 8th International Conference, Paris, France, September 3 5, 2009. Proceedings 8 . Springer Berlin Heidelberg,2009.
- The Game of Blackjack and Analysis of Counting Cards Ariell Zimran, Anna Klis, Alejandra Fuster and Christopher Rivelli December 2, 2009.