Software Components:

1) Create Dataframe

What it does: Create Dataframe component takes in ~30,000 raw text files that contain various amounts of poker games played on 888poker.com during heads up Texas Hold'em. It contains many functions that splits the raw data into keys and values such as 'Flop actions', 'Flop', 'BB cards', 'Folded pre', and many more. These keys and values are then stored as a pandas dataframe, with each row representing a full game played.

Input: ~30,000 raw text files (.txt) stored in Peter's computer

Output: pandas dataframe of 520,810 rows (games) and 16 columns (different game metrics and values)

2) Evaluator

What it does: The evaluator component, forked from the deuces public repo, is used to evaluate a player's 2 card hand and the board in all states of the game (preflop, flop, turn, river). It takes lists of unique 32 bit integers encoding card information created by the Card component as inputs and uses the Lookup component to calculate either the rank class (e.g. Pair, Straight, Full House, etc.) of the hand, a rank from 1 to 7462 representing the strength of all possible poker hands (e.g. Royal Flush = rank 1), or the rank percentage (the % of all possible poker hands that the current hand beats).

Forked Repo: https://github.com/worldveil/deuces/tree/master

Input: Two lists containing the unique 32bit integers encoding the card information for the player hand and the board created using the Card component. Done in the form of:

```
hand1 = [
   Card.new('Ts'),
   Card.new('Ac')

board = [
   Card.new('2h'),
   Card.new('2s'),
   Card.new('Jc'),
   Card.new('Kh'),
   Card.new('Qc')
```

Output:

- Rank Class (str): Phrase describing the class of card a player has. (e.g. Full House, Royal Flush, Two Pair)

- Hand Rank (int): The rank of a player's cards out of all possible poker hands from 1 to 7462 in descending order of strength. (e.g. Royal Flush = rank 1)
- Rank Percentage (float): The % of all possible poker hands which the current hand beats. (e.g. Royal Flush = 1.0)

Ex of calculations per game)
rankclass1 = evaluator.class_to_string(evaluator.get_rank_class(evaluator.evaluate(hand1, board)))
rankperc1 = evaluator.get_rank_percentage(evaluator.evaluate(hand1, board))
rankclass1 = Straight
rankperc1 = 0.785580

3) Adding Flush

What it does: Adding Flush component only analyzes games that have made it to the river and classifies the likelihood of a flush on the board. Filtering so that only these games are included, a new dataframe called poker_dataframe_w_flush is created. It creates a new column in this dataframe called 'Board' which is the combination of the 'Flop', 'Turn', and 'River' columns and calculates the highest number of the same suit of the board. Based on this highest value, 'Flush Count' is added to the dataframe. Then based on this flush count value, a new column is created called 'Flush' which takes the value of 0, 1, 2, or 3. The conversion between 'Flush Count' and 'Flush' is as follows;

Flush Count	Flush
1 or 2	0
3	1
4	2
5	3

Input: A pkl file of the original data frame made in Create Dataframe component

Output: A new data frame called poker_dataframe_w_flush**,** which is a subset of the original data frame, with added columns, 'Board, 'Flush Count', and 'Flush'

4) Straight Risk Evaluator

What it does: The Straight Risk Evaluator component analyzes games and evaluates the risk of a straight occuring based on the board cards at the flop, turn, and river stages. A copy of the input dataframe is created and appended with the columns 'Flop Straight Risk', 'Turn Straight Risk', and 'River Straight Risk' and populates them with NaN. The component then evaluates the risk of a straight for each game which made it to the flop stage based on the Straight Risk Metric defined below and populates the Flop Straight Risk column with the risk number which is

calculated by analyzing the differences between the numerical value of the cards on the board extracted using the Card component. The component then does the same for the turn and the river for each game in the dataframe. Finally, the component converts the dataframe to a new .pkl file located in the current working directory.

Straight Risk Metric:
1 (Low) - 3 or more cards + board cards required to make straight.
2 (Medium) - 2 cards + board cards required to make straight.
3 (High) - 1 card + board cards required to make straight.
4 (Straight) - Straight is already present in the board cards.
NaN - Game did not reach this stage so straight risk cannot be calculated.

Input: A pkl file of the original dataframe made in using the Create Dataframe component.

Output: A new .pkl dataframe file called poker_df_straight_risk.pkl is exported to the current working directory containing the original dataframe with the appended 'Flop Straight Risk', 'Turn Straight Risk', and 'River Straight Risk' columns containing the straight risks for each game at each stage.

5) Adding Rank

What it does: The Adding Rank component analyzes each game in the poker dataframe generated by the Create Dataframe component using the Card and Evaluator components and determines the hand rank and the hand strength for the small and big blind players. The component does this by creating a copy of the original dataframe and removing all games that were not played to completion. The component then appends 'SB Handrank', 'BB Handrank', 'SB Hand Strength', and 'BB Hand Strength' columns to the dataframe and calculates them for each game using the Evaluator and Card modules. Finally, the component converts the new dataframe to a .pkl file located in the current working directory.

Definitions:

- SB - Small Blind player
- BB - Big Blind player
- Handrank - A phrase (string) representing the type of hand a player has. (e.g. "Royal Flush")
- Hand Strength - A float representing the % of all possible poker hands that the player's current hand beats.

Input: A pkl file of the original dataframe made in using the Create Dataframe component.

Output: A new .pkl dataframe file called poker_dataframe_w_rank.pkl is exported to the current working directory containing the original dataframe minus all games that were not played to completion with the additional 'SB Handrank', 'BB Handrank', 'SB Hand Strength', and 'BB Hand Strength' columns containing those values for each game.

6) Bluff Eval

What it does: The Bluff Eval components analyzes each game in the poker dataframe generated by the Create Dataframe component using the Card and Evaluator components and calculates the Normalized Aggressiveness and Bluff Metric for the river phase of the game for the small and big blind players. The component does this by creating a copy of the original dataframe and removing all games that were not played to completion. The component then appends 'aggSB', 'aggBB', 'SB Bluff Metric', and 'BB Bluff Metric' columns to the dataframe and calculates them for each game using the Evaluator and Card modules. Finally, the component coverts the new dataframe to a .pkl file located in the current working directory.

Definitions:

- SB - Small Blind player
- BB - Big Blind player
- Agg (normalized aggressiveness) - The normalized aggressiveness for a player in their position (SB or BB) for the river phase of the game. Aggressiveness is calculated as the number of times a player bets or raises during that phase of the game +1. This number is then normalized by dividing it by the average aggressiveness for a player in that position (SB, BB).
    - Agg = (sum of bets/raises +1)/(average agg for players in same position)
- Bluff Metric - A metric to quantify if a player was bluffing during the river phase of a game. It is calculated by multiplying the agg (normalized aggressiveness) of the player by (1 minus that players hand strength)
    - Bluff Metric = agg * (1 - hand strength)
    - (1 - hand strength) = % of all possible poker hands that the player's hand loses to.
- Hand Strength - A float representing the % of all possible poker hands that the player's current hand beats.
    - Calculated using the Evaluator and Card components

Input: A pkl file of the original dataframe made in using the Create Dataframe component.

Output: A new .pkl dataframe file called poker_df_w_bluff.pkl is exported to the current working directory containing the original dataframe minus all games that were not played to completion with the additional 'aggSB', 'aggBB', 'SB Bluff Metric', and 'BB Bluff Metric' columns containing those values for each game.

7) Add Aggression Column

What it does: The Add Aggression Column component analyzes each game in the poker dataframe generated by the Create Dataframe component and categorizes the aggressiveness

of the small and big blind players at each stage of the game (preflop, flop, turn, river). The component then creates a modified copy of the dataframe containing four new columns: ‘Aggr preflop’, ‘Aggr flop’, ‘Aggr turn’, ‘Aggr river’. Each column contains the aggressiveness category for each game at that stage. The aggressiveness is categorized using the player actions at the respective phase of the game. Finally, the component returns the modified dataframe to the dataframe to the current working directory.

Definitions:
SB - Small Blind Player
BB - Big Blind Player
Aggressiveness: 1 (aggressive) if the player raised or bet, otherwise 0 (passive).

Aggressiveness Catagories:
00 - Both players were passive.
10 - SB player was aggressive, BB player was passive.
01 - BB player was aggressive, SB player was passive.
11 - Both players were aggressive

Input: A pkl file of the original dataframe made using the Create Dataframe component.

Output: A new .pkl dataframe file called poker_dataframe_with_agg_columns.pkl is exported to the current working directory containing the original dataframe with the additional ‘Aggr preflop’, ‘Aggr flop’, ‘Aggr turn’, and ‘Aggr river’ columns containing those values for each game.

8) Add Fold Column

What it does: The Add Fold Column component appends two new columns representing whether the small blind or big blind player folded throughout the game to the original dataframe made using the Create Dataframe component. The new columns ‘fold_sb’ and ‘fold_bb’ are populated with false (boolean) unless the small blind or big blind player folded throughout the game. The component does not create a new dataframe and instead modifies the existing dataframe.

Input: A pkl file of the original dataframe made in using the Create Dataframe component.

Output: The input poker dataframe.pkl is replaced with the modified dataframe containing the fold_sb and fold_bb columns.

Interactions to accomplish use cases:

Use Case 1: User Interaction

The user will input their cards, board, and previous personal and opponent actions during the river stage of the game. These cards will be run through Evaluator to get the rank class and rank percentage of their hand. Once this is done, our ML algorithm, which has been trained using the data frames created in Create Dataframe and Adding Flush as well as many more, will analyze the players hand, previous rounds actions, and compare the current game to the hundred of thousands of games stored in our data frames to give insights into the likelihood of winning against a random opponent, as well as recommended actions.

Preliminary Plan:

Week 8:

- Finalize base dataframe
- Finalize choice of open source hand evaluators
- Brainstorm additional columns we need for ML algorithm to be effective and informative
- Begin heavy work into component specification and functional specifications docs

Week 9:

- Begin writing components as jupyter notebooks that add columns to original base dataframe
- Split work between project members and form teams to tackle different sections of to-dos

Week 10:

- Implement all the components and notebooks as python scripts
- Run dataframes through ML algorithm and begin testing examples to see efficacy
- Create tests for our scripts
- Clean up main repo to only include necessary scripts, tests, docs, notebooks and data frames

Week 11:

- Create demos in jupyter notebook for presentation
- Finish documentation of modules, functions, and scripts
- Polish project, such as comments and coding style (Pylint)