Winning Edge:
Texas Hold'em Data Analysis

Peter Sushko, Dylan Heino, John Tatka, Victor Tian, Eric Gibson

Background

WinningEdge™ is a strategy optimization tool for online, low-stakes Texas hold'em players. The application will provide real time probabilities and suggest action based on an ML algorithm trained using historical data. In using WinningEdge™, players will be able to make better informed decisions with the goal of winning and being profitable.

Data

- >500,000 low-stakes, heads up texas hold'em poker games purchased from 888poker.com
  - 888poker.com is one of the three largest online poker sites.
  - Data stored in >30,000 .txt files of various lengths.
- Data included in each game:
  - # of players, usernames, and who is small and big blind.
  - Game ID#
  - Time and Data
  - Chip stack sizes
  - All betting actions at each phase of game (preflop, flop, turn, river).
  - Bet amounts
  - Board cards at each phase of game. (if applicable)
  - Player cards. (if nobody folds)

Data Example:

This is an example of a single game played

- Here you can see the structure of the data given directly from 888poker.com

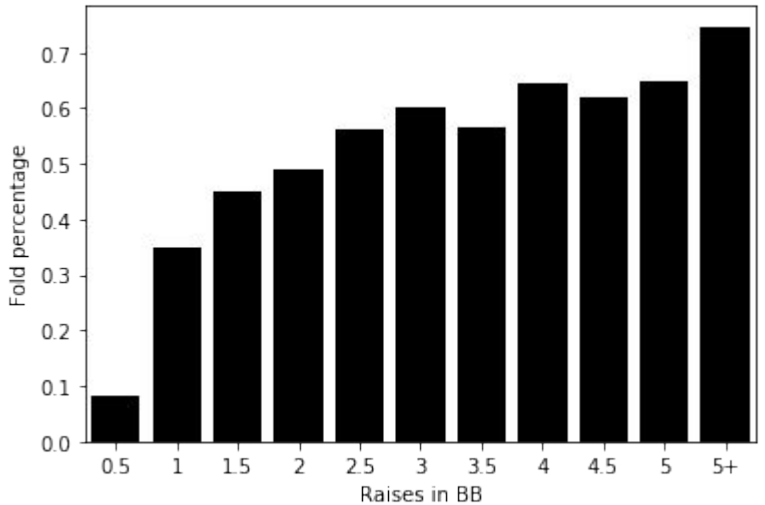
#Game No : 502745408
***** 888poker Hand History for Game 502745408 *****
$0.01/$0.02 Blinds No Limit Holdem - *** 06 06 2018 04:49:57
Table Bedford 6 Max (Real Money)
Seat 1 is the button
Total number of players : 2
Seat 1: ponte1001 ($0.80)
Seat 4: Bolorig888 ($1.01)
ponte1001 posts small blind [$0.01]
Bolorig888 posts big blind [$0.02]
** Dealing down cards **
ponte1001 calls [$0.01]
Bolorig888 checks
** Dealing flop ** [2c, Qh, Jd]
Bolorig888 checks
ponte1001 checks
** Dealing turn ** [9h]
Bolorig888 bets [$0.04]
ponte1001 calls [$0.04]
** Dealing river ** [Js]
Bolorig888 bets [$0.06]
ponte1001 calls [$0.06]
** Summary **
Bolorig888 shows [9c, Qs]
ponte1001 mucks [9s, 6h]
Bolorig888 collected [$0.23]

Use Case 1: Showdown Analysis

Which hands are more likely to make it to the end?

Use Case 2: Bet comparison

What size bet can make the opponent fold for the cheapest price?



Use Case 3: Catching Bluffs

You made it to the river. Opponent bets. Given the board and how the hand was played, how do you know if its a bluff?

Our first model considers only the board texture and aggression level of opponent.

57% accuracy is low.

To improve, we need better features.

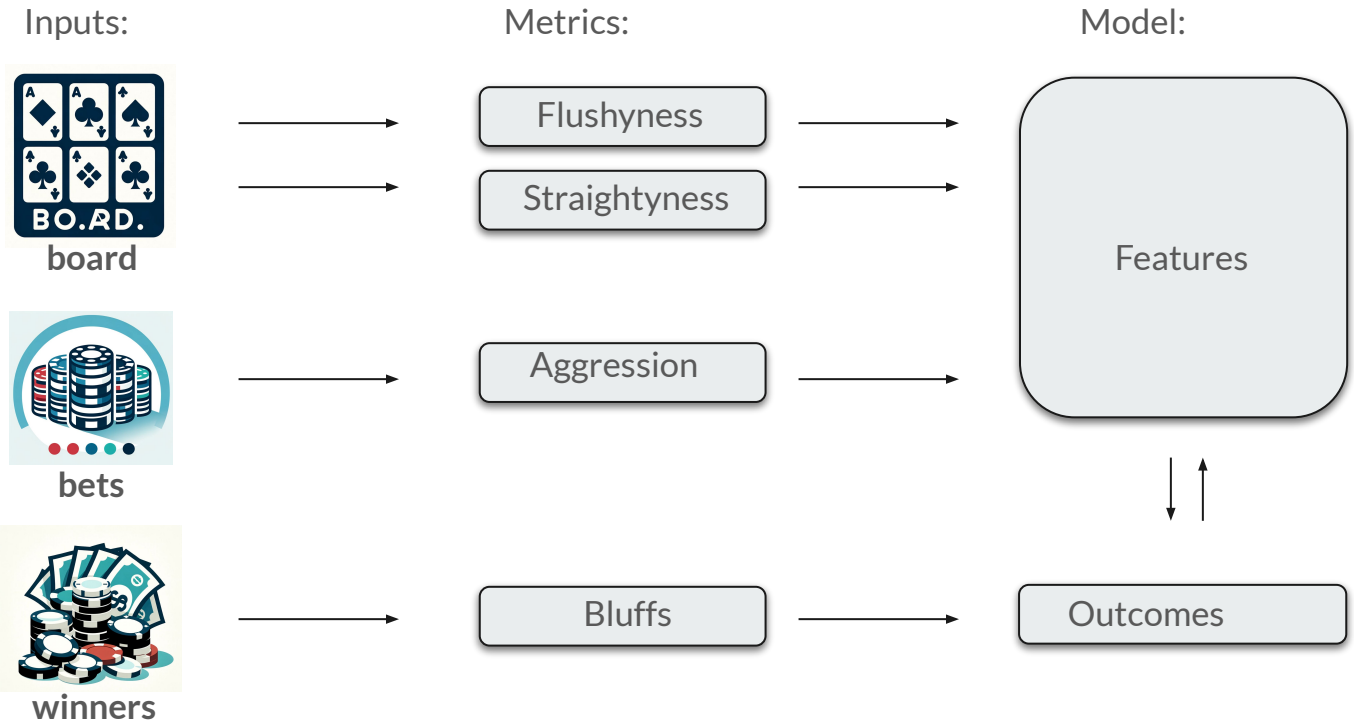
In [3]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

precision = precision_score(y_test, y_pred)
print(f"Precision: {precision * 100:.2f}%")

recall = recall_score(y_test, y_pred)
print(f"Recall: {recall * 100:.2f}%")
```

```
Accuracy: 57.31%
Precision: 52.57%
Recall: 31.56%
```

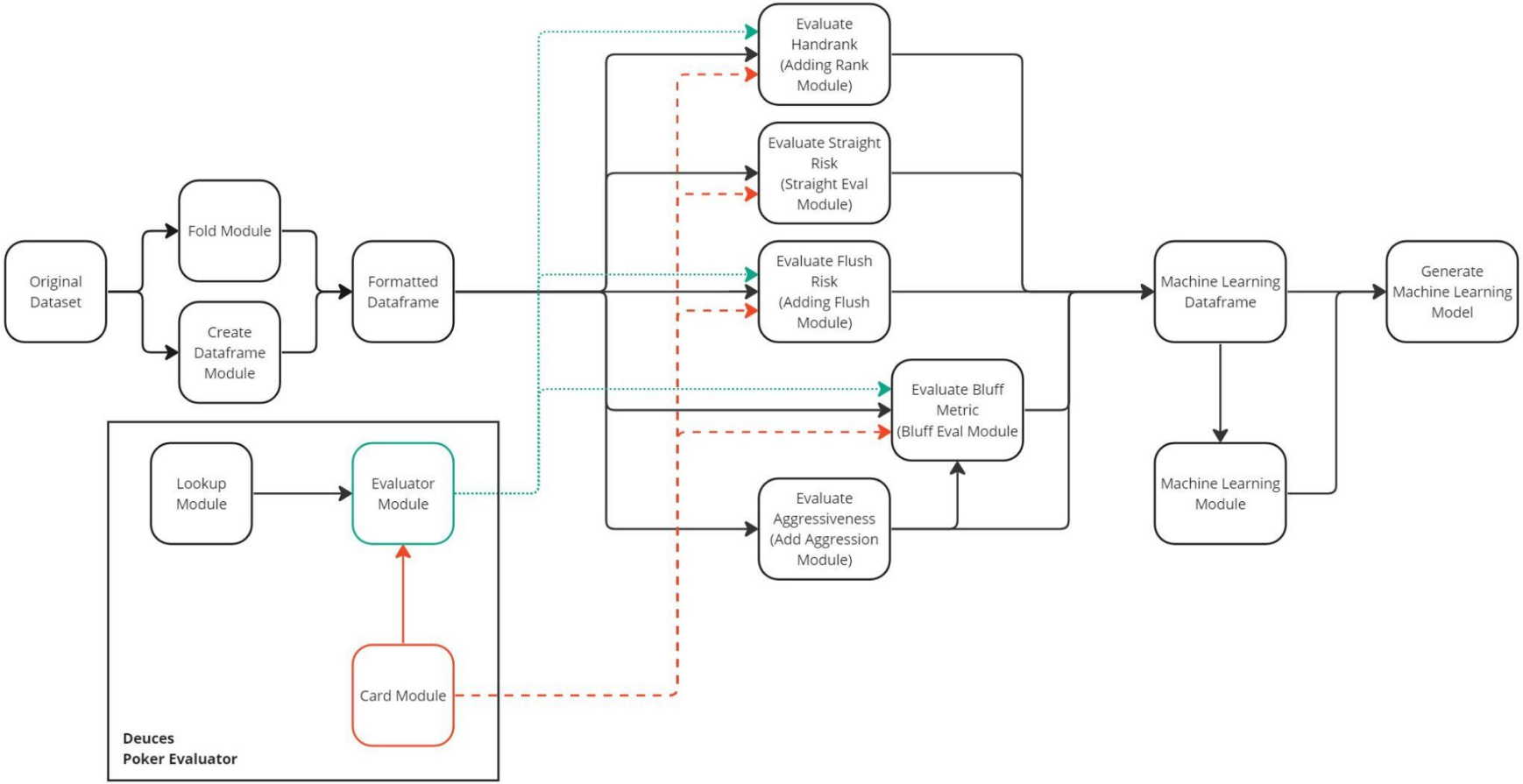
In [4]:
```
coefficients = model.coef_
intercept = model.intercept_

print("Coefficients:", coefficients)
print("Intercept:", intercept)
```

```
Coefficients: [[ 0.2130902   0.21575425  0.00345047 -0.33398528 -0.1852743  -0.56403836
  -0.23594029  0.11906768]]
Intercept: [-0.15363636]
```

Model: Under the hood



Design:


Original Dataset
Fold Module
Create Dataframe Module
Formatted Dataframe
Evaluate Handrank (Adding Rank Module)
Evaluate Straight Risk (Straight Eval Module)
Evaluate Flush Risk (Adding Flush Module)
Evaluate Bluff Metric (Bluff Eval Module
Evaluate Aggressiveness (Add Aggression Module)
Machine Learning Dataframe
Machine Learning Module
Generate Machine Learning Model
Lookup Module
Evaluator Module
Card Module
Deuces
Poker Evaluator

Evaluator Demo

```
%cd ..
from winedge import Card, Evaluator

# for debugging purposes, also include ```lookup```
```

```
\\wsl.localhost\Ubuntu-22.04\home\dheino13\WinningEdge
```

Input hands and board

Provide hands information of you and your opponent, as well as the community cards.

Name the cards in the following way:

- For ranks, use `['23456789TJQKA']`.
- For suits, use `['shdc']`
- For example, ten of heart would be named `['Th']`, whereas eight of spade is denoted as `['8s']`

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

hand2 = [
    Card.new('2c'),
    Card.new('Js')
]

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

```
# Initialize evaluator. This step calls the lookup function and generate
# the LookupTable for hand ranks.
evaluator = Evaluator()
```

```
# evaluator.evaluate(hand, board) provide hand ranks
# other functions provide different ways of representing hand strengths

rankclass1 = evaluator.class_to_string(evaluator.get_rank_class(evaluator.evaluate(hand1, board)))
rankperc1 = evaluator.get_rank_percentage(evaluator.evaluate(hand1, board))

rankclass2 = evaluator.class_to_string(evaluator.get_rank_class(evaluator.evaluate(hand2, board)))
rankperc2 = evaluator.get_rank_percentage(evaluator.evaluate(hand2, board))
```

```
print("Rank class for your hand is: %s" % rankclass1)
print("The hand strength is: %f" % rankperc1)

print("Rank class for your opponent's hand is: %s" % rankclass2)
print("The hand strength is: %f" % rankperc2)

if rankperc1 > rankperc2:
    print('\nYou won!')

elif rankperc1 < rankperc2:
    print('\nYou lost!')

else:
    print("\nIt's a draw!")
```

```
Rank class for your hand is: Straight
The hand strength is: 0.785580
Rank class for your opponent's hand is: Full House
The hand strength is: 0.957920

You lost!
```

Repository Structure:

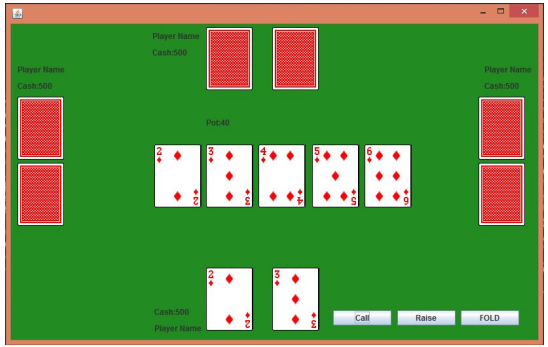
- Dataframe directory
  - Formatted dataframe
  - Create dataframe module
- ML directory
  - ML dataframes
  - ML module
- Notebooks directory
  - Module prototypes
- Winedge directory
  - Data and dataframe analysis modules

```
../WinningEdge/
├── LICENSE
├── README.md
├── dataframe
│   ├── convert_data_to_dataframe.ipynb
│   ├── create_dataframe.py
│   ├── poker_dataframe_bugfixed.pkl
│   └── poker_dataframe_with_agg_columns.pkl
├── doc
│   ├── Component Specification.pdf
│   ├── Functional Specifications.pdf
│   ├── Technology Review Slides.pdf
│   └── use-case.md
├── environment.yml
├── examples
│   ├── README.md
│   └── eval_comp_demo.ipynb
├── ml
│   ├── README.txt
│   ├── df_filtered_w_flush.pkl
│   ├── df_filtered_w_flush_strength.pkl
│   ├── make_ml_df.ipynb
│   ├── make_regression_model.ipynb
│   ├── needs_labels.pkl
│   └── ready_to_model.pkl
├── notebooks
│   ├── AddingFlush.ipynb
│   ├── AddingRank.ipynb
│   ├── WIP.ipynb
│   ├── bluff_metric_prototype.ipynb
│   ├── straight_risk_prototype.ipynb
│   └── testing_create_dataframe.ipynb
├── setup.py
└── winedge
    ├── Folded.ipynb
    ├── __init__.py
    ├── __pycache__
    ├── add_aggression_column.py
    ├── add_fold_column.py
    ├── adding_flush.py
    ├── adding_rank.py
    ├── bluff_eval.py
    ├── card.py
    ├── evaluator.py
    ├── lookup.py
    ├── straight_risk_eval.py
    └── tests.py
```

Lessons Learned:

- Understanding limitations and possible biases of data prior to modelling
  - Player cards only visible for completed games introduces heavy bias to ML model
- Importance of a comprehensive dataset
- Benefits of Modular Design
  - Breaking problem into smaller parts (Dataframe, Evaluator, etc.)
- Importance of complete data
  - Handling missing / invalid data

Future Work:



- Refine ML algorithm to increase accuracy
- Purchase more comprehensive dataset to train ML and eliminate biases
- Implement **GUI** to utilize software in real time while gaming
- Implement continuous training so training dataset can be expanded in real time while in use
- Expand scope of software to higher stakes game and >2 players

Thanks!