Project Darts

I. Introduction

a. Beschreibung und Ziel

II. Prototype

a. Code Backend

b. Bisherige Oberfläche

III. Aktueller Stand

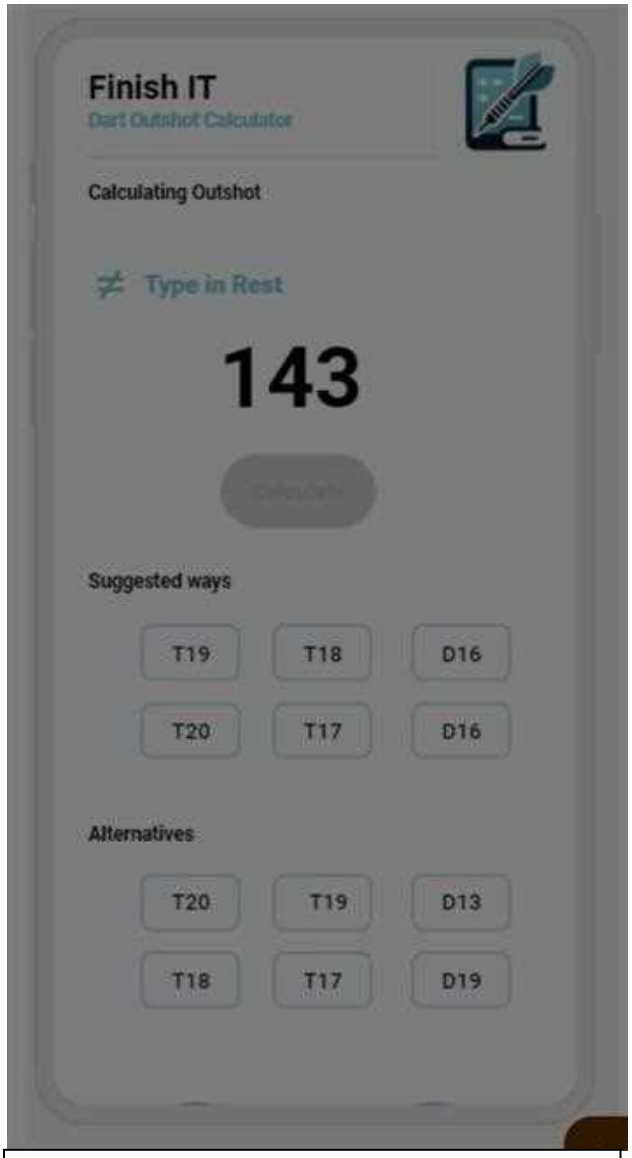
a. To-dos

1. Introduction

Beschreibung und Ziel

Erfolg im Dartsport beruht nicht nur auf Präzision und motorischen Fähigkeiten, sondern auch auf der zugrunde liegenden Statistik, auf der die Regeln basieren. Besonders im Bereich unter 170 Punkten, in dem es mehrere theoretische Möglichkeiten gibt, das Spiel zu beenden, erweisen sich einige Lösungswege als taktisch sinnvoller als andere. Zudem haben viele Spieler individuelle Präferenzen, da sie bestimmte Felder besser beherrschen.

Ziel des nachfolgenden Projekts ist es, alle sinnvollen Kombinationen darzustellen, sodass der Spieler sich seines bevorzugten Wegs sowie möglicher Alternativen bewusst wird.



Erstes design

b. .

2. Prototype

Code Backend

```
from flask import Flask, render_template, request
import numpy as np

app = Flask(__name__)

def fliplr(arr):
return np.fliplr(arr)

def combvec(*args):
return np.array(np.meshgrid(*args, indexing='ij')).T.reshape(-1, len(args))

def calculate_output(V):

U = np.concatenate((np.arange(1, 21), np.array([25])))
D1 = 2 * np.concatenate((np.arange(1, 21), np.array([25]))) T = 3
* np.arange(7, 21)
S = fliplr(combvec(D1, U, T))
T = S[:, 1] + S[:, 0] + S[:, 2]
S = np.hstack((S, T[:, np.newaxis])) Z =
S[:, 3] == V
S = np.hstack((S, Z[:, np.newaxis]))
S1 = S[S[:, 4] == 1, :]
#S1 spiegelt alle 3 1 2 Kombinationen wieder

#Erstellung von S2
U = 2*np.concatenate((np.arange(11, 21), np.array([25]))) D1 =
2 * np.concatenate((np.arange(1, 21), np.array([25]))) T = U
S = fliplr(combvec(D1, U, T))
T = S[:, 1] + S[:, 0] + S[:, 2]
S = np.hstack((S, T[:, np.newaxis])); Z =
S[:, 3] == V
S = np.hstack((S, Z[:, np.newaxis]))
S2 = S[S[:, 4] == 1, :]
# S2 spiegelt alle 2 2 2 Kombinationen wieder

T= 2*np.concatenate((np.arange(11, 21), np.array([25]))) D1 = 2
* np.concatenate((np.arange(1, 21), np.array([25]))) U =
np.concatenate((np.arange(1, 21), np.array([25])))
S = fliplr(combvec(D1, U, T))
T = S[:, 1] + S[:, 0] + S[:, 2]
S = np.hstack((S, T[:, np.newaxis])); Z =
S[:, 3] == V
S = np.hstack((S, Z[:, np.newaxis]))
S3 = S[S[:, 4] == 1, :]
#S3 spiegelt alle 2 1 2 Kombinationen wieder

# Erstellung S4
U = 3 * np.arange(7, 21)
D1 = 2 * np.concatenate((np.arange(1, 21), np.array([25]))) T = 3
* np.arange(7, 21)
S = fliplr(combvec(D1, U, T))
T = S[:, 1] + S[:, 0] + S[:, 2]
S = np.hstack((S, T[:, np.newaxis])); Z =
S[:, 3] == V
S = np.hstack((S, Z[:, np.newaxis]))
S4 = S[S[:, 4] == 1, :]
# S4 spiegelt alle 3 2 3 Kombinationen wieder

#Erstellung von S5
U = 2*np.concatenate((np.arange(11, 21), np.array([25]))) D1 =
2 * np.concatenate((np.arange(1, 21), np.array([25]))) T = 3 *
np.arange(7, 21)
S = fliplr(combvec(D1, U, T))
T = S[:, 1] + S[:, 0] + S[:, 2]
S = np.hstack((S, T[:, np.newaxis])); Z =
S[:, 3] == V
S = np.hstack((S, Z[:, np.newaxis]))
S5 = S[S[:, 4] == 1, :]
#S5 spiegelt alle 3 2 2 Kombinationen wieder

# Erstellung von S6
U=np.concatenate((np.arange(1, 21), np.array([25])))
B=50.*np.ones(1)
D1 = 2 * np.concatenate((np.arange(1, 21), np.array([25])))
S = fliplr(combvec(D1, U, B))
T = S[:, 1] + S[:, 0] + S[:, 2]
S = np.hstack((S, T[:, np.newaxis])); Z =
S[:, 3] == V
S = np.hstack((S, Z[:, np.newaxis]))
S6= S[S[:, 4] == 1, :]
# Erstellung 50 1 2
# Übergangsweise Lösung da man bei verschiedenen Kombinationen beim Verfehlen von Bull noch einen Outshot hat #
langfristig Bull und 25 nicht als Single und Dopple behandeln sondern extra

# Erstellung S7
U= 3 * np.arange(7, 21) B=50.*np.ones(1)
D1 = 2 * np.concatenate((np.arange(1, 21), np.array([25])))
S = fliplr(combvec(D1, U, B))
T = S[:, 1] + S[:, 0] + S[:, 2]
S = np.hstack((S, T[:, np.newaxis])); Z =
S[:, 3] == V
S = np.hstack((S, Z[:, np.newaxis]))
S7= S[S[:, 4] == 1, :]
#  S7 steht für Bull 3 2
```

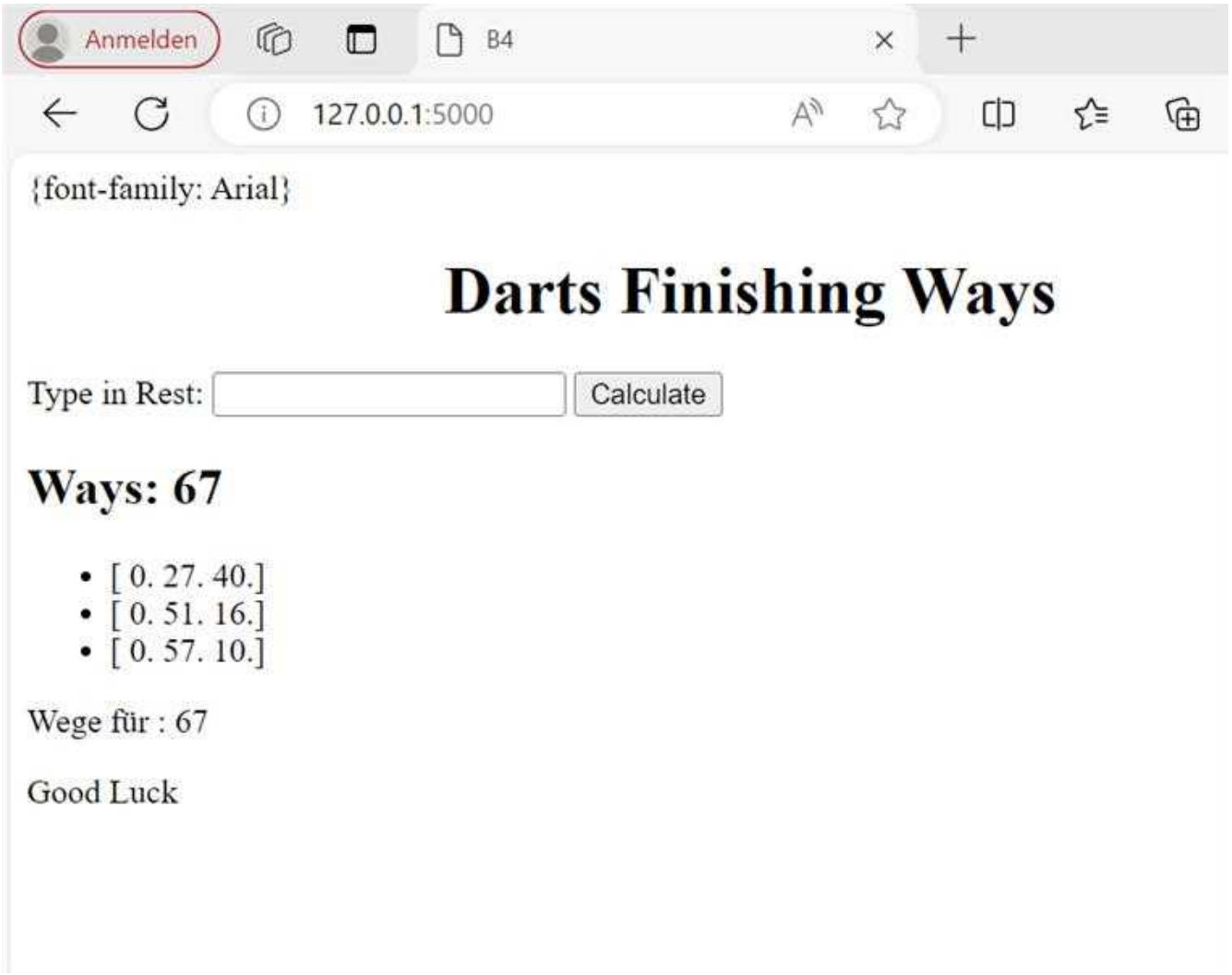
```
#Erstellung S8
U= 3 * np.arange(7, 21) B=0.*np.ones(1)
D1 = 2 * np.concatenate((np.arange(1, 21), np.array([25])))
S = fliplr(combvec(D1, U, B))
T = S[:, 1] + S[:, 0] + S[:, 2]
S = np.hstack((S, T[:, np.newaxis])); Z =
S[:, 3] == V
S = np.hstack((S, Z[:, np.newaxis]))
```

```
S8= S[S[:, 4] == 1, :]
# S8 spiegelt Tripple Double Combinationen wieder

## 022
U= 2*np.concatenate((np.arange(11, 21), np.array([25])))
B=0.*np.ones(1)
D1 = 2 * np.concatenate((np.arange(1, 21), np.array([25])))
S = fliplr(combvec(D1, U, B))
T = S[:, 1] + S[:, 0] + S[:, 2]
S = np.hstack((S, T[:, np.newaxis])); Z =
S[:, 3] == V
S = np.hstack((S, Z[:, np.newaxis]))
S9= S[S[:, 4] == 1, :]

## 0 1 2
U= np.concatenate((np.arange(1, 21), np.array([25])))
B=0.*np.ones(1)
D1 = 2 * np.concatenate((np.arange(1, 21), np.array([25])))
S = fliplr(combvec(D1, U, B))
T = S[:, 1] + S[:, 0] + S[:, 2]
S = np.hstack((S, T[:, np.newaxis])); Z =
S[:, 3] == V
S = np.hstack((S, Z[:, np.newaxis]))
S10= S[S[:, 4] == 1, :]


## 002
U= 0.*np.ones(1) B=U
D1 = 2 * np.concatenate((np.arange(1, 21), np.array([25])))
S = fliplr(combvec(D1, U, B))
T = S[:, 1] + S[:, 0] + S[:, 2]
S = np.hstack((S, T[:, np.newaxis])); Z =
S[:, 3] == V
S = np.hstack((S, Z[:, np.newaxis]))
S11 = S[S[:, 4] == 1, :]
## definieren und zuordnen welche finishing wege wo gebraicht werden
output_value=None
if V < 41 and V % 2 == 0: # kleiner 41 und durch 2 Teilbar ist ein 1 Dart finish
output_value = S11
elif V < 41 and V % 2 == 1:  # wenn nicht durch 2 Teilbar werden 2 darts benötigt
output_value = S10
elif 41 <= V <= 49:
output_value = S10
elif V == 50:
output_value = np.vstack((S11, S10[:3])) # bei 50 beides möglich
elif 51 <= V <= 60:
output_value = S10
elif 61 <= V < 86 and V % 2 == 0:
output_value = np.vstack((S10, S9, S8, S7,S6)) # hier reicht eine sigle Kombination nicht aus daher werden Tripple Dopple und Bull hinzugenommen
elif 61 <= V < 86 and V % 2 == 1:
output_value = np.vstack((S10, S9, S8, S6))
elif 86 <= V <= 98:
output_value = np.vstack((S10, S9, S8,S3))
elif V == 99:
output_value = np.vstack((S10, S9, S8, S1, S2, S3)) ## 99 einziges 3 Dart finish daher wird 3 1 2 und Bull mitgenommen
elif V == 100:
output_value = np.vstack((S10, S9, S8))

elif 100 < V <= 120:
output_value = np.vstack((S10, S9, S8, S1, S3))
elif 120 < V  <=4000:
output_value = np.vstack((S10, S9, S8, S1, S2, S3, S1,S7,S5,S4)) ### wird im 2ten Schritt besser gefiltert

output_value = output_value[:, :3] ##
output_value = output_value
V_minus_50_times_3 = (V - 50) * 3 # hierbei geht es dabei das man sich ein Bullseye finish stellen kann wenn der erste statt in der Tripple in der Single l

if (V == 131) or (133 <= V <= 134) or (136 <= V <= 170):
selected_rows = output_value[
((output_value[:, 0] >= 45) &
(output_value[:, 1] >= 45) &
(output_value[:, 2] % 4 == 0) &
(output_value[:, 0] != 50) &
(output_value[:, 1] != 50)) |
((output_value[:, 0] >= 51) &
(output_value[:, 1] >= 51)) |
((output_value[:, 1] == output_value[:, 0]) |
(output_value[:, 2] == output_value[:, 1]))
]

elif (V == 132) or (V == 135):
selected_rows = output_value[
((output_value[:, 0] >= 45) &
(output_value[:, 1] >= 45) &
(output_value[:, 2] % 4 == 0) &
(output_value[:, 0] != 50) &
(output_value[:, 1] != 50)) |
((output_value[:, 0] >= 51) &
(output_value[:, 1] >= 51)) |
((output_value[:, 1] == output_value[:, 0]) |
((output_value[:, 0] == 50) &
(output_value[:, 2] % 4 == 0)) |
(output_value[:, 2] == output_value[:, 1]))]
elif (61 <= V <= 75):
selected_rows = output_value[
(output_value[:, 0] == 0) &
(output_value[:, 1] > 48) |
(output_value[:, 0] == 0) & (output_value[:, 1] ==
V_minus_50_times_3) | (output_value[:, 0] == 0) &
(output_value[:, 2] == 40) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 32) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 16) |
```

```
(output_value[:, 0] == 0) &
(output_value[:, 1] == 25) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 24) |
(output_value[:,    0]    ==    0)    &
(output_value[:,     2]     ==     36)|
(output_value[:, 0] == 0) &
(output_value[:, 2] ==output_value[:, 1])]

elif (76 <= V <= 80):
selected_rows = output_value[
(output_value[:, 0] == 0) &
(output_value[:, 1] > 48) &  (output_value[:, 1] != 50) |
(output_value[:, 0] == 0) &  (output_value[:, 1] ==
V_minus_50_times_3) | (output_value[:, 0] == 0) &
(output_value[:, 2] == 40) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 32) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 16) |
(output_value[:, 0] == 0) &
(output_value[:, 1] == 25) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 24) |
(output_value[:,    0]    ==    0)    &
(output_value[:,     2]     ==     36)|
(output_value[:, 0] == 0) &
(output_value[:, 2] ==output_value[:, 1])

]
elif (81 <= V <= 85):
selected_rows = output_value[
(output_value[:, 0] == 0) &
(output_value[:, 1] > 48) &  (output_value[:, 1] != 50) |
(output_value[:, 0] == 0) &  (output_value[:, 1] ==
V_minus_50_times_3) | (output_value[:, 0] == 0) &
(output_value[:, 2] == 40) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 32) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 16) |
(output_value[:, 0] == 0) &
(output_value[:, 1] == 25) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 24) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 36) |
(output_value[:, 1] == 50) |
(output_value[:, 0] == 50) &
(output_value[:, 2] == 16)
]
elif (2 <= V <= 40):
selected_rows = output_value
elif (86 <= V <= 98) and V % 2 == 0 or (V == 100):
selected_rows = output_value[(output_value[:, 0] == 0) & (output_value[:, 1] >=
48) &  (output_value[:, 1] != 50) |
(output_value[:, 0] == 0) &  (output_value[:, 1] ==
V_minus_50_times_3) | (output_value[:, 0] == 0) &
(output_value[:, 2] == 40) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 32) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 16) |
(output_value[:, 0] == 0) &
(output_value[:, 1] == 25) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 24) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 36) |
(output_value[:, 1] == 50) |
(output_value[:, 0] == output_value[:, 2]) &
(output_value[:, 2] % 4 == 0) |
(output_value[:, 0] == 32) &
(output_value[:, 2] == 40)|
(output_value[:, 0] == 36) &
(output_value[:, 2] == 40)|
(output_value[:, 0] == 32) &
(output_value[:, 2] == 40)|
(output_value[:, 1] == (output_value[:, 2] / 2)) &
(output_value[:, 0] % 4 == 0)]

elif (86 <= V <= 98) and V % 2 == 1:
selected_rows = output_value[(output_value[:, 0] == 0) &  # Überprüfung auf Null in der ersten Spalte
(output_value[:, 1] >= 48) &  (output_value[:, 1] != 50) |
(output_value[:, 0] == 0) &  (output_value[:, 1] ==
V_minus_50_times_3) | (output_value[:, 0] == 0) &
(output_value[:, 2] == 40) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 32) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 16) |
(output_value[:, 0] == 0) &
(output_value[:, 1] == 25) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 24) |
(output_value[:, 0] == 0) &
(output_value[:, 2] == 36) |
(output_value[:, 0] == output_value[:, 2]) &
(output_value[:, 2] % 4 == 0)&
(output_value[:, 1] != 25) |
(output_value[:, 0] == 32) &
(output_value[:, 2] == 40)&
```

```
(output_value[:, 1] != 25)|
(output_value[:, 0] == 36)
& (output_value[:, 2] ==
40)& (output_value[:, 1] !=
25)| (output_value[:, 0] ==
32) &
(output_value[:, 2] == 40)&  (output_value[:, 1] != 25)|
(output_value[:, 1] == (output_value[:, 2] / 2)) &
(output_value[:, 0] % 4 == 0) &
(output_value[:, 1] != 25) ]
elif (99 <= V <= 120):
selected_rows=output_value[(output_value[:, 0] >= 39) &
(output_value[:, 2] == 32)& (output_value[:, 1] !=
25)|
(output_value[:, 0] >= 39) & (output_value[:, 2]
== 40)& (output_value[:, 1] != 25)|
(output_value[:, 0] >= 39) & (output_value[:, 2]
== 24)& (output_value[:, 1] != 25)|
(output_value[:, 0] == 0)]
elif V==121 or 122 or 123 or 125 or 126 or 129:
            selected_rows = output_value[np.where(((output_value[:, 0] / 3) == (V - 104)) &  (output_value[:, 1] != 50) &(output_value[:, 2] != 50)
                                                                                                                                &(output_value[
                                ((output_value[:, 0] / 3) == (V - 110)) &  (output_value[:, 1] != 50) &(output_value[:, 2] != 50)&(output_value[:, 2]
                                ((output_value[:, 0] / 3) == (V - 100)) &  (output_value[:, 1] != 50)&(output_value[:, 2] != 50)&(output_value[:, 2]
                                ((output_value[:, 0] / 3) >= (V - 99)) &  (output_value[:, 1] != 50)&(output_value[:, 2] != 50) &(output_value[:, 2]
                                ((output_value[:, 0] / 3) == (V - 101)) &  (output_value[:, 1] != 50)&(output_value[:, 2] != 50)&(output_value[:, 2]
                                ((output_value[:, 0] / 3) == (V - 107)) &  (output_value[:, 1] != 50)&(output_value[:, 2] != 50)&(output_value[:, 2]
((output_value[:, 0]==50) & (output_value[:, 2] % 4 == 0)))]

elif V==124 or 127 or 128 or 130 :
            selected_rows = output_value[np.where(((output_value[:, 0] / 3) == (V - 104)) &  (output_value[:, 1] != 50) &(output_value[:, 2] != 50)
                                                                                                                                &(output_value[
                                ((output_value[:, 0] / 3) == (V - 110)) &  (output_value[:, 1] != 50) &(output_value[:, 2] != 50)&(output_value[:, 2]
                                ((output_value[:, 0] / 3) == (V - 100)) &  (output_value[:, 1] != 50)&(output_value[:, 2] != 50)&(output_value[:, 2]
                                ((output_value[:, 0] / 3) >= (V - 99)) &  (output_value[:, 1] != 50)&(output_value[:, 2] != 50) &(output_value[:, 2]
                                ((output_value[:, 0] / 3) == (V - 101)) &  (output_value[:, 1] != 50)&(output_value[:, 2] != 50)&(output_value[:, 2]
                                ((output_value[:, 0] / 3) == (V - 107)) &  (output_value[:, 1] != 50)&(output_value[:, 2] != 50)&(output_value[:, 2]
else:
selected_rows = []
return selected_rows

def print_solution(V):
if V == 159 or V == 162 or V == 163 or V == 165 or V == 166 or V == 168 or V == 169 or V > 170 or V == 1:
return "No possible outshot"
else:
return "Good Luck"
@app.route('/',
methods=['GET', 'POST'])
def index():
output_value = None
print_solution_message = None

if request.method == 'POST':
V = float(request.form['input_value'])
output_value = calculate_output(V)
print_solution_message = print_solution(V)



return render_template('index.html', output_value=output_value, print_solution_message=print_solution_message)

if __name__ == '__main__':
app.run(debug=True)
```

Bisherige Oberfläche



3. To-dos

c. Teilweise noch zu viele Möglichkeiten trotz Einschränkungen
d. Ranking der Kombination Reihenfolge erstellen
e. App Development