🔗 각 데이터 별 결과 시각화

In [6]:
```
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns

# 파일이 저장된 디렉토리 경로 (예시)
directory_path = './combined_data'

# 디렉토리에서 모든 CSV 파일 리스트 가져오기
csv_files = [f for f in os.listdir(directory_path) if f.endswith('.csv')]
```

In [7]:
```
csv_files
```

Out[7]:
```
['dohoon_I_1_combined.csv',
 'dohoon_I_2_combined.csv',
 'dohoon_I_3_combined.csv',
 'dohoon_I_4_combined.csv',
 'dohoon_S_1_combined.csv',
 'dohoon_S_2_combined.csv',
 'dohoon_S_3_combined.csv',
 'dohoon_S_4_combined.csv',
 'jaeho_I_1_combined.csv',
 'jaeho_I_2_combined.csv',
 'jaeho_I_3_combined.csv',
 'jaeho_I_4_combined.csv',
 'jaeho_S_1_combined.csv',
 'jaeho_S_2_combined.csv',
 'jaeho_S_3_combined.csv',
 'jaeho_S_4_combined.csv',
 'jaewan_I_1_combined.csv',
 'jaewan_I_2_combined.csv',
 'jaewan_I_3_combined.csv',
 'jaewan_I_4_combined.csv',
 'jaewan_S_1_combined.csv',
 'jaewan_S_2_combined.csv',
 'jaewan_S_3_combined.csv',
 'jaewan_S_4_combined.csv',
 'jiyoung_I_1_combined.csv',
 'jiyoung_I_2_combined.csv',
 'jiyoung_I_3_combined.csv',
 'jiyoung_I_4_combined.csv',
 'jiyoung_S_1_combined.csv',
 'jiyoung_S_2_combined.csv',
 'jiyoung_S_3_combined.csv',
 'jiyoung_S_4_combined.csv',
 'minho_I_1_combined.csv',
 'minho_I_2_combined.csv',
 'minho_I_3_combined.csv',
 'minho_I_4_combined.csv',
 'minho_S_1_combined.csv',
 'minho_S_2_combined.csv',
 'minho_S_3_combined.csv',
 'minho_S_4_combined.csv',
 'nayoung_I_1_combined.csv',
 'nayoung_I_2_combined.csv',
 'nayoung_I_3_combined.csv',
 'nayoung_I_4_combined.csv',
 'nayoung_S_1_combined.csv',
 'nayoung_S_2_combined.csv',
 'nayoung_S_3_combined.csv',
 'nayoung_S_4_combined.csv',
 'seongjun_I_1_combined.csv',
 'seongjun_I_2_combined.csv',
 'seongjun_I_3_combined.csv',
 'seongjun_I_4_combined.csv',
 'seongjun_S_1_combined.csv',
 'seongjun_S_2_combined.csv',
 'seongjun_S_3_combined.csv',
 'seongjun_S_4_combined.csv',
 'seoyeong_I_1_combined.csv',
 'seoyeong_I_2_combined.csv',
 'seoyeong_I_3_combined.csv',
 'seoyeong_I_4_combined.csv',
```

```
 'seoyeong_S_1_combined.csv',
 'seoyeong_S_2_combined.csv',
 'seoyeong_S_3_combined.csv',
 'seoyeong_S_4_combined.csv',
 'taeyoon_I_1_combined.csv',
 'taeyoon_I_2_combined.csv',
 'taeyoon_I_3_combined.csv',
 'taeyoon_I_4_combined.csv',
 'taeyoon_S_1_combined.csv',
 'taeyoon_S_2_combined.csv',
 'taeyoon_S_3_combined.csv',
 'taeyoon_S_4_combined.csv']
```

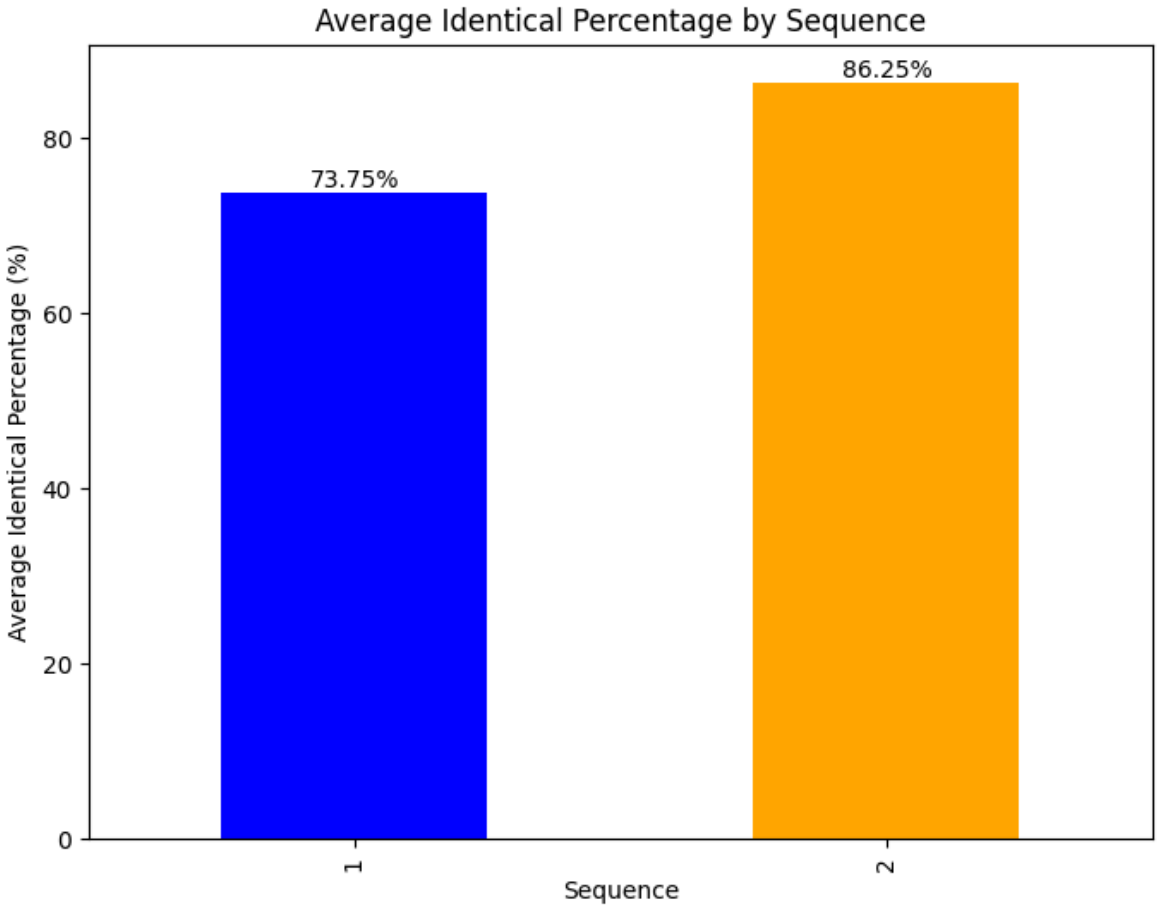
In [9]:

```python
# 각 파일의 Sequence 값을 설정하는 조건
sequence_map = {
    "dohoon_I": "2", "dohoon_S": "1",
    "jiyoung_I": "2", "jiyoung_S": "1",
    "jaeho_I": "2", "jaeho_S": "1",
    "jaewan_I": "1", "jaewan_S": "2",
    "minho_I": "1", "minho_S": "2",
    "nayoung_I": "1", "nayoung_S": "2",
    "seongjun_I": "1", "seongjun_S": "2",
    "taeyoon_I": "2", "taeyoon_S": "1"
}

# 결과를 저장할 리스트 초기화
comparison_results = []

# 각 CSV 파일 경로를 불러와 처리하는 코드
def process_and_add_sequence_with_path(csv_files, sequence_map, directory_path):
    for csv_file in csv_files:
        # 파일 경로 만들기
        file_path = os.path.join(directory_path, csv_file)

        # 파일 이름에서 접두사 추출 (예: dohoon_I, jiyoung_S 등)
        prefix = '_'.join(csv_file.split('_')[:2])

        # CSV 파일 불러오기
        try:
            df = pd.read_csv(file_path)
        except FileNotFoundError:
            print(f"File not found: {file_path}")
            continue

        # Sequence 값 설정
        sequence_value = sequence_map.get(prefix)

        if sequence_value:
            # Sequence 컬럼 추가
            df['Sequence'] = sequence_value

            # maxforce와 result 열이 있는지 확인
            if 'maxforce' in df.columns and 'result' in df.columns:
                # 상관계수 계산
                correlation = df['maxforce'].corr(df['result'])

                # 차이의 절대값 평균 계산
                mean_absolute_difference = (df['maxforce'] - df['result']).abs()

                # maxforce와 result가 동일한 값의 비율 계산
```

```
                identical_percentage = (df['maxforce'] == df['result']).mean() *

                # 결과 저장
                comparison_results.append({
                    'filename': csv_file,
                    'correlation': correlation,
                    'mean_absolute_difference': mean_absolute_difference,
                    'identical_percentage': identical_percentage,
                    'Sequence': sequence_value
                })

            # 수정된 파일을 같은 경로에 저장
            df.to_csv(file_path, index=False)
            print(f"Processed {file_path} and added Sequence {sequence_value}")

process_and_add_sequence_with_path(csv_files, sequence_map, directory_path)

# 결과를 데이터프레임으로 변환
results_df = pd.DataFrame(comparison_results)

# Sequence별 identical_percentage 평균 계산
sequence_grouped = results_df.groupby('Sequence')['identical_percentage'].mean()

# 그래프 그리기
plt.figure(figsize=(8, 6))
bars = sequence_grouped.plot(kind='bar', color=['blue', 'orange'])

# 각 막대에 퍼센티지 값 표시
for bar in bars.patches:
    plt.text(bar.get_x() + bar.get_width() / 2,
             bar.get_height(),
             f'{bar.get_height():.2f}%',  # 퍼센티지 값 표시
             ha='center',
             va='bottom')

# 그래프 제목 및 라벨 설정
plt.title('Average Identical Percentage by Sequence')
plt.xlabel('Sequence')
plt.ylabel('Average Identical Percentage (%)')

# 그래프 출력
plt.show()

# 결과 데이터프레임 출력
print(results_df)
```

```
Processed ./combined_data\dohoon_I_1_combined.csv and added Sequence 2
Processed ./combined_data\dohoon_I_2_combined.csv and added Sequence 2
Processed ./combined_data\dohoon_I_3_combined.csv and added Sequence 2
Processed ./combined_data\dohoon_I_4_combined.csv and added Sequence 2
Processed ./combined_data\dohoon_S_1_combined.csv and added Sequence 1
Processed ./combined_data\dohoon_S_2_combined.csv and added Sequence 1
Processed ./combined_data\dohoon_S_3_combined.csv and added Sequence 1
Processed ./combined_data\dohoon_S_4_combined.csv and added Sequence 1
Processed ./combined_data\jaeho_I_1_combined.csv and added Sequence 2
Processed ./combined_data\jaeho_I_2_combined.csv and added Sequence 2
Processed ./combined_data\jaeho_I_3_combined.csv and added Sequence 2
Processed ./combined_data\jaeho_I_4_combined.csv and added Sequence 2
Processed ./combined_data\jaeho_S_1_combined.csv and added Sequence 1
Processed ./combined_data\jaeho_S_2_combined.csv and added Sequence 1
Processed ./combined_data\jaeho_S_3_combined.csv and added Sequence 1
Processed ./combined_data\jaeho_S_4_combined.csv and added Sequence 1
Processed ./combined_data\jaewan_I_1_combined.csv and added Sequence 1
Processed ./combined_data\jaewan_I_2_combined.csv and added Sequence 1
Processed ./combined_data\jaewan_I_3_combined.csv and added Sequence 1
Processed ./combined_data\jaewan_I_4_combined.csv and added Sequence 1
Processed ./combined_data\jaewan_S_1_combined.csv and added Sequence 2
Processed ./combined_data\jaewan_S_2_combined.csv and added Sequence 2
Processed ./combined_data\jaewan_S_3_combined.csv and added Sequence 2
Processed ./combined_data\jaewan_S_4_combined.csv and added Sequence 2
Processed ./combined_data\jiyoung_I_1_combined.csv and added Sequence 2
Processed ./combined_data\jiyoung_I_2_combined.csv and added Sequence 2
Processed ./combined_data\jiyoung_I_3_combined.csv and added Sequence 2
Processed ./combined_data\jiyoung_I_4_combined.csv and added Sequence 2
Processed ./combined_data\jiyoung_S_1_combined.csv and added Sequence 1
Processed ./combined_data\jiyoung_S_2_combined.csv and added Sequence 1
Processed ./combined_data\jiyoung_S_3_combined.csv and added Sequence 1
Processed ./combined_data\jiyoung_S_4_combined.csv and added Sequence 1
Processed ./combined_data\minho_I_1_combined.csv and added Sequence 1
Processed ./combined_data\minho_I_2_combined.csv and added Sequence 1
Processed ./combined_data\minho_I_3_combined.csv and added Sequence 1
Processed ./combined_data\minho_I_4_combined.csv and added Sequence 1
Processed ./combined_data\minho_S_1_combined.csv and added Sequence 2
Processed ./combined_data\minho_S_2_combined.csv and added Sequence 2
Processed ./combined_data\minho_S_3_combined.csv and added Sequence 2
Processed ./combined_data\minho_S_4_combined.csv and added Sequence 2
Processed ./combined_data\nayoung_I_1_combined.csv and added Sequence 1
Processed ./combined_data\nayoung_I_2_combined.csv and added Sequence 1
Processed ./combined_data\nayoung_I_3_combined.csv and added Sequence 1
Processed ./combined_data\nayoung_I_4_combined.csv and added Sequence 1
Processed ./combined_data\nayoung_S_1_combined.csv and added Sequence 2
Processed ./combined_data\nayoung_S_2_combined.csv and added Sequence 2
Processed ./combined_data\nayoung_S_3_combined.csv and added Sequence 2
Processed ./combined_data\nayoung_S_4_combined.csv and added Sequence 2
Processed ./combined_data\seongjun_I_1_combined.csv and added Sequence 1
Processed ./combined_data\seongjun_I_2_combined.csv and added Sequence 1
Processed ./combined_data\seongjun_I_3_combined.csv and added Sequence 1
Processed ./combined_data\seongjun_I_4_combined.csv and added Sequence 1
Processed ./combined_data\seongjun_S_1_combined.csv and added Sequence 2
Processed ./combined_data\seongjun_S_2_combined.csv and added Sequence 2
Processed ./combined_data\seongjun_S_3_combined.csv and added Sequence 2
Processed ./combined_data\seongjun_S_4_combined.csv and added Sequence 2
Processed ./combined_data\taeyoon_I_1_combined.csv and added Sequence 2
Processed ./combined_data\taeyoon_I_2_combined.csv and added Sequence 2
Processed ./combined_data\taeyoon_I_3_combined.csv and added Sequence 2
Processed ./combined_data\taeyoon_I_4_combined.csv and added Sequence 2
```

```
Processed ./combined_data\taeyoon_S_1_combined.csv and added Sequence 1
Processed ./combined_data\taeyoon_S_2_combined.csv and added Sequence 1
Processed ./combined_data\taeyoon_S_3_combined.csv and added Sequence 1
Processed ./combined_data\taeyoon_S_4_combined.csv and added Sequence 1
```

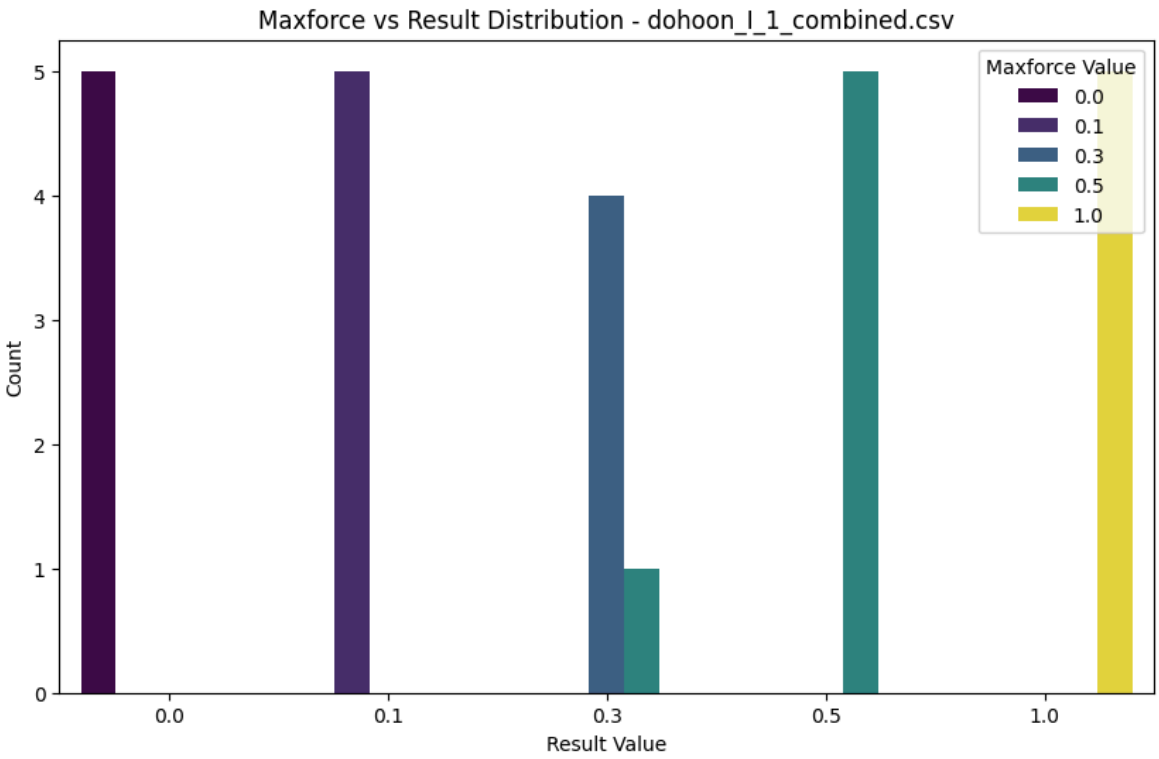


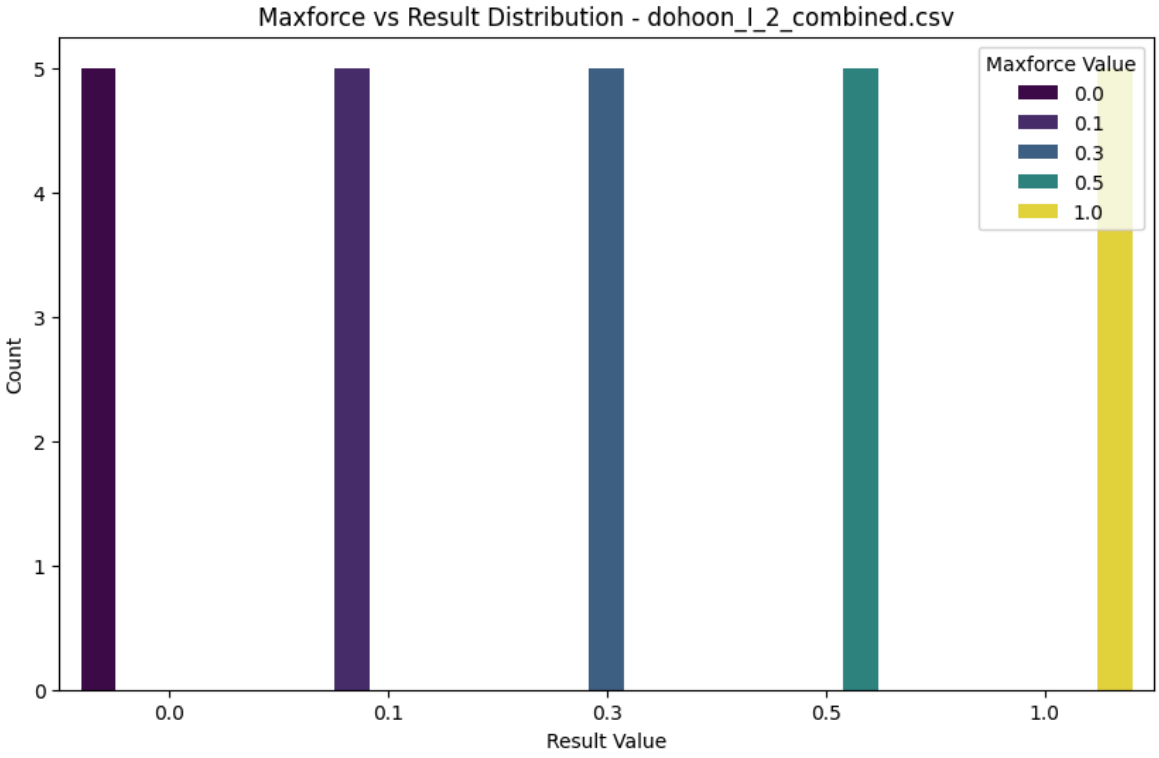
	filename	correlation	mean_absolute_difference	identical_percentage	Sequence
0	dohoon_I_1_combined.csv	0.993892	0.008	96.0	2
1	dohoon_I_2_combined.csv	1.000000	0.000	100.0	2
2	dohoon_I_3_combined.csv	0.993877	0.008	96.0	2
3	dohoon_I_4_combined.csv	1.000000	0.000	100.0	2
4	dohoon_S_1_combined.csv	0.994014	0.008	96.0	1
..	...	...	...	...	...
59	taeyoon_I_4_combined.csv	0.805299	0.116	72.0	2
60	taeyoon_S_1_combined.csv	0.427669	0.276	32.0	1
61	taeyoon_S_2_combined.csv	0.515988	0.236	48.0	1
62	taeyoon_S_3_combined.csv	0.767243	0.136	60.0	1
63	taeyoon_S_4_combined.csv	0.619546	0.184	56.0	1

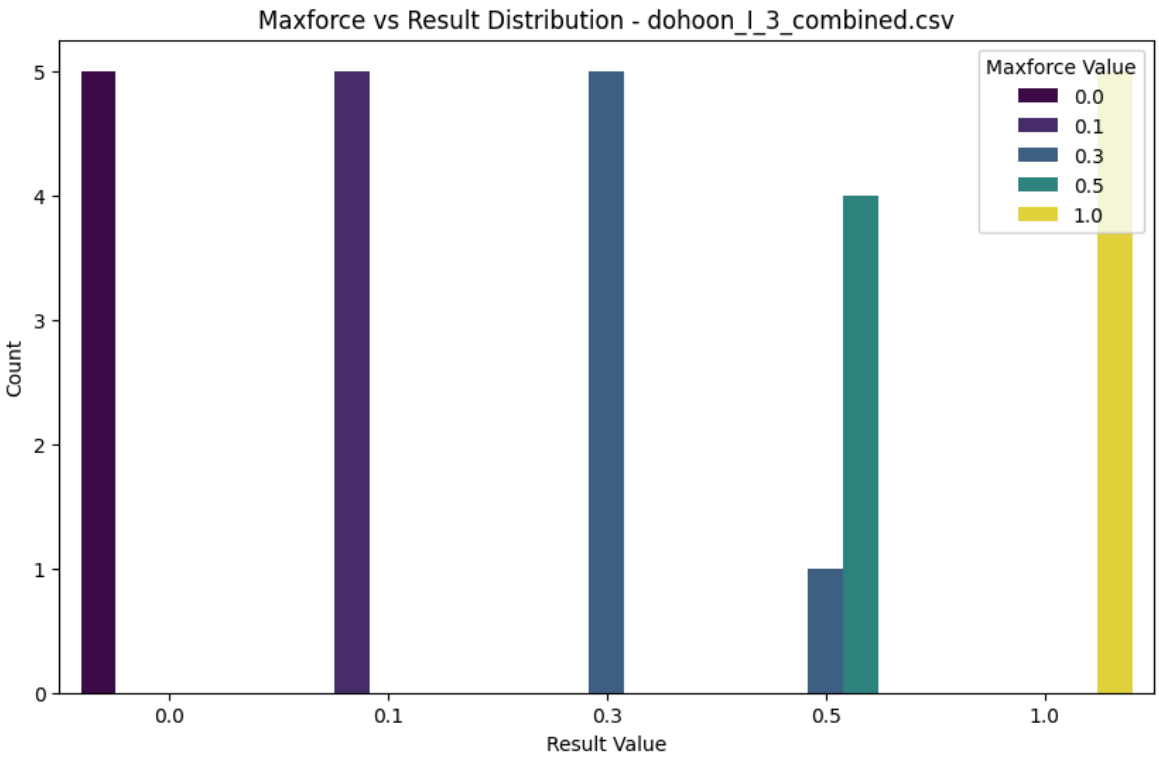
[64 rows x 5 columns]

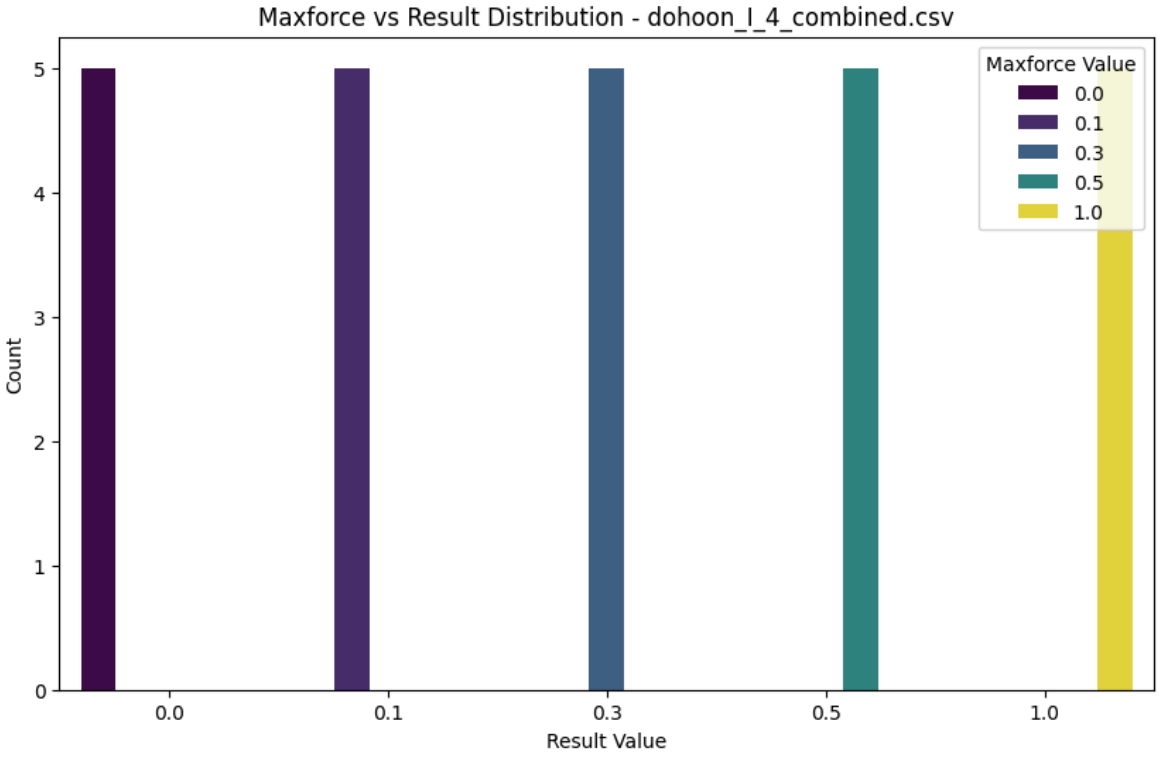
In [35]:

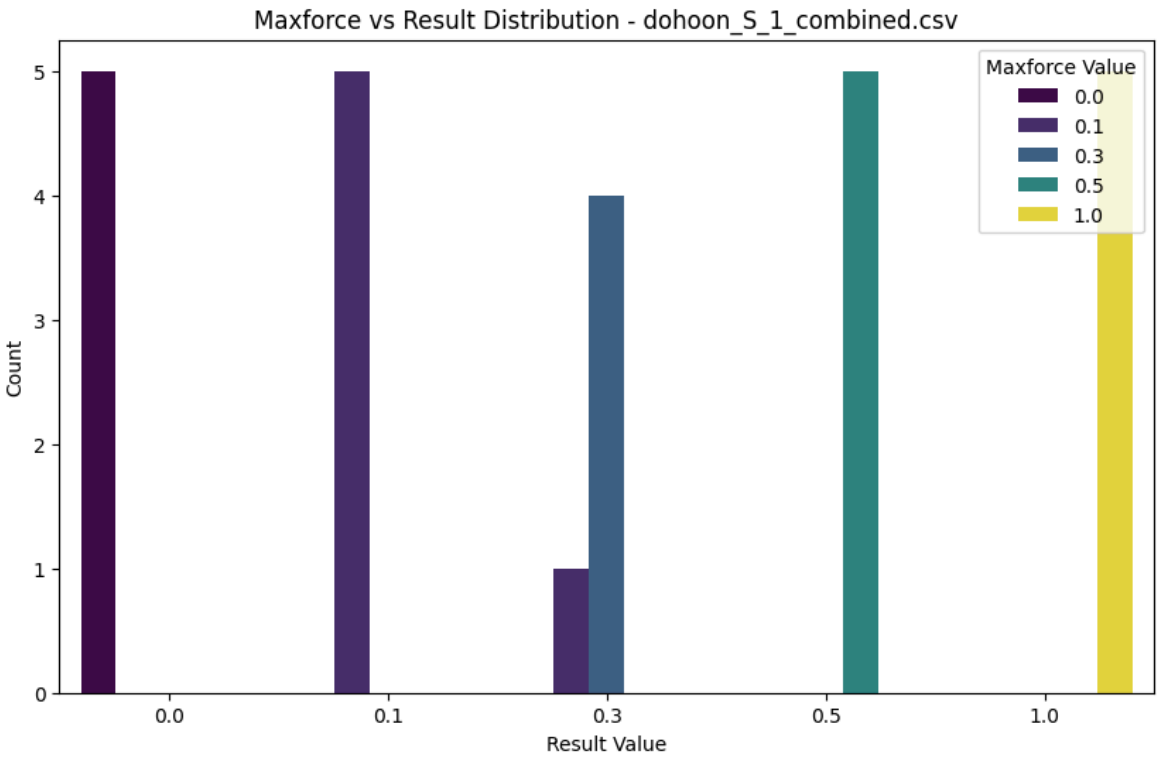
```
# 각 CSV 파일에 대해 작업 수행
for csv_file in csv_files:
    file_path = os.path.join(directory_path, csv_file)

    # CSV 파일 읽기
    df = pd.read_csv(file_path)

    # maxforce와 result 열이 있는지 확인
    if 'maxforce' in df.columns and 'result' in df.columns:
        maxforce_values = [0.0, 0.1, 0.3, 0.5, 1.0]
        result_values = [0.0, 0.1, 0.3, 0.5, 1.0]

        distribution = []

        for maxforce_value in maxforce_values:
            for result_value in result_values:
                count = len(df[(df['maxforce'] == maxforce_value) & (df['result'
                distribution.append({
                    'maxforce_value': maxforce_value,
                    'result_value': result_value,
                    'count': count
                })

        distribution_df = pd.DataFrame(distribution)

        # 막대그래프 생성
        plt.figure(figsize=(10, 6))
        sns.barplot(x='result_value', y='count', hue='maxforce_value', data=dist

        # 그래프 세부 설정
        plt.title(f'Maxforce vs Result Distribution - {csv_file}')
        plt.xlabel('Result Value')
        plt.ylabel('Count')
        plt.legend(title='Maxforce Value')

        # 그래프 출력
        plt.show()

    else:
        print(f"'maxforce' or 'result' column missing in {csv_file}")
```

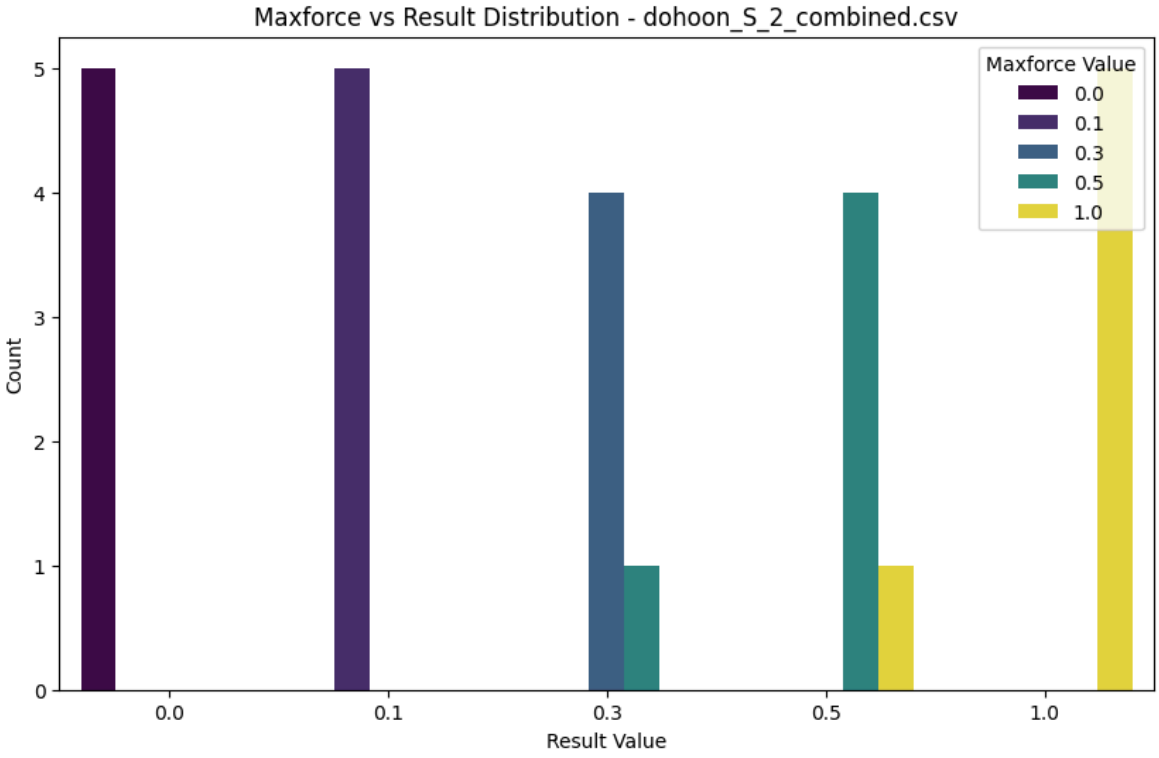

Maxforce vs Result Distribution - dohoon_I_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

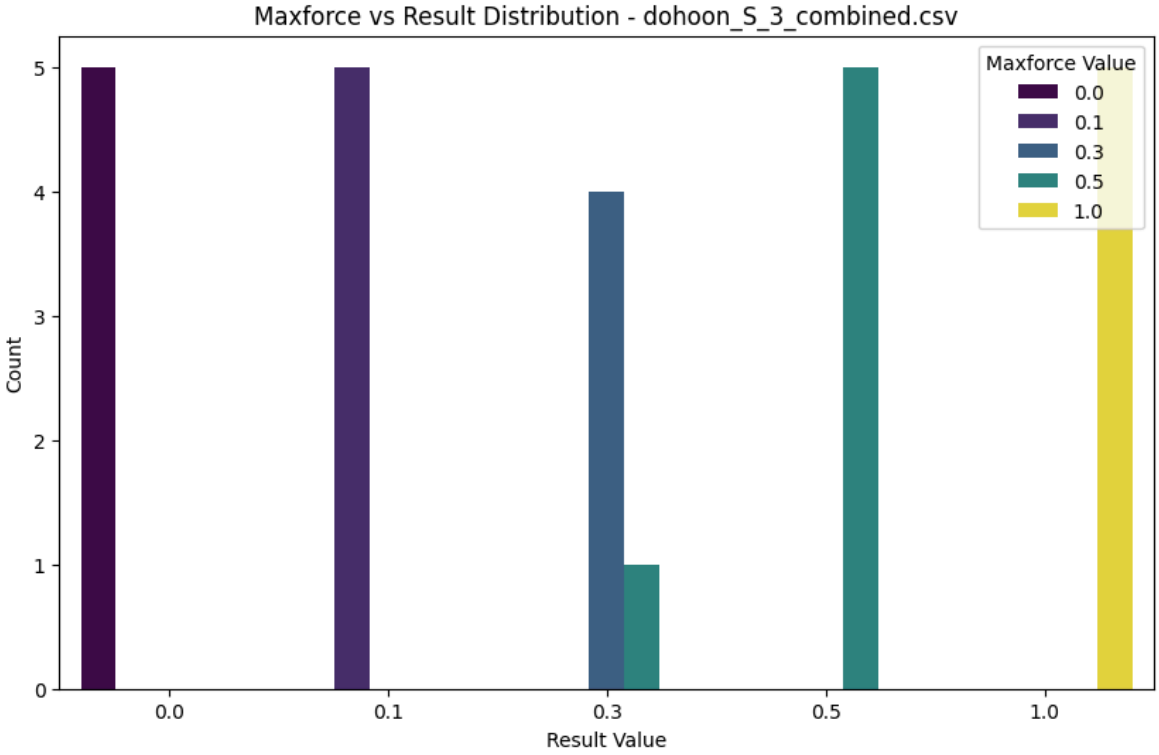

Maxforce vs Result Distribution - dohoon_I_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

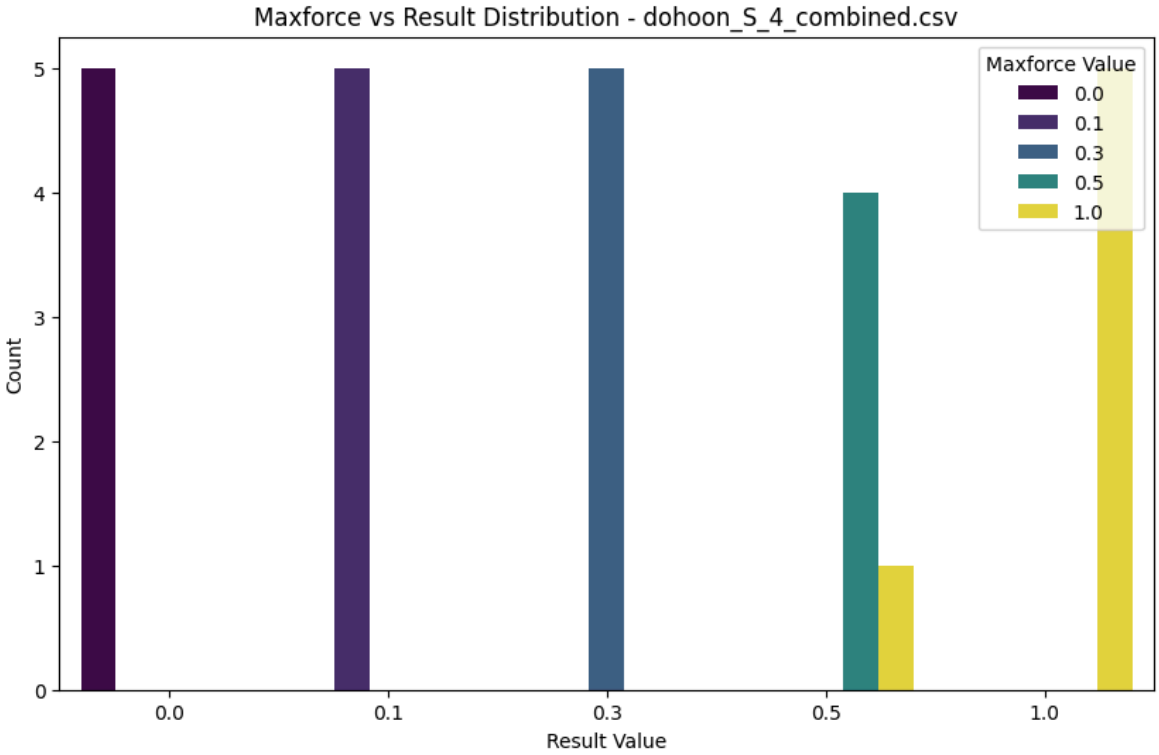

Maxforce vs Result Distribution - dohoon_I_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

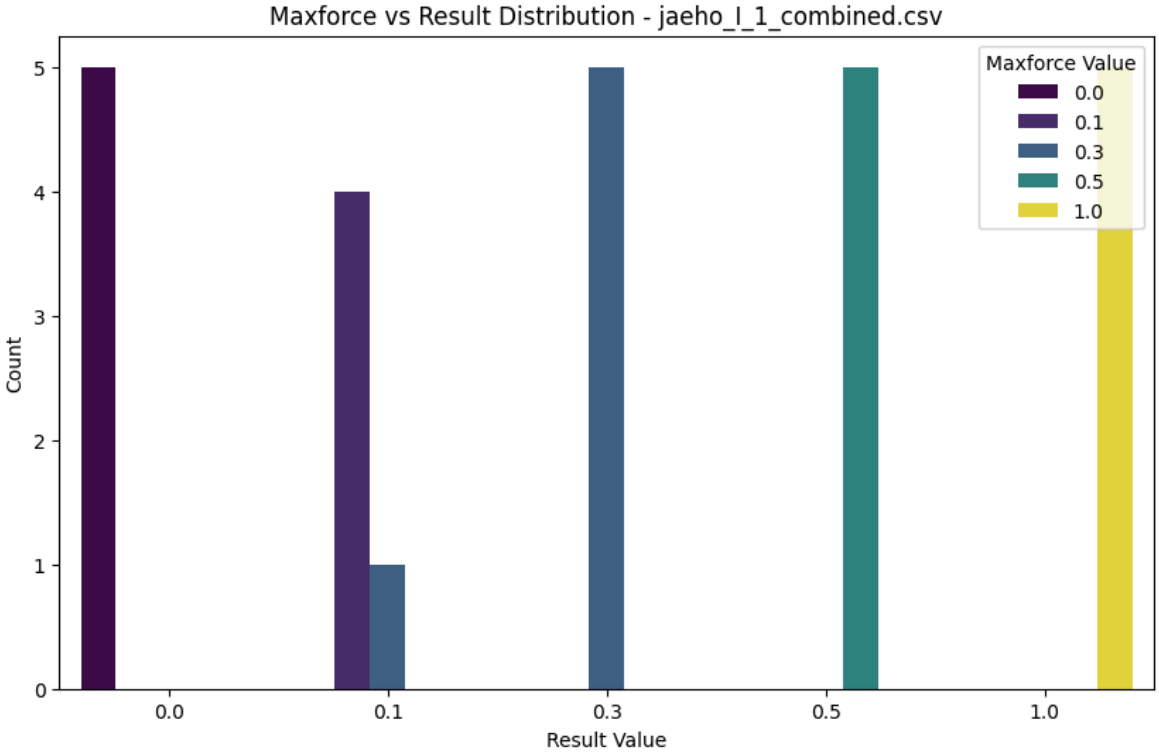

Maxforce vs Result Distribution - dohoon_I_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

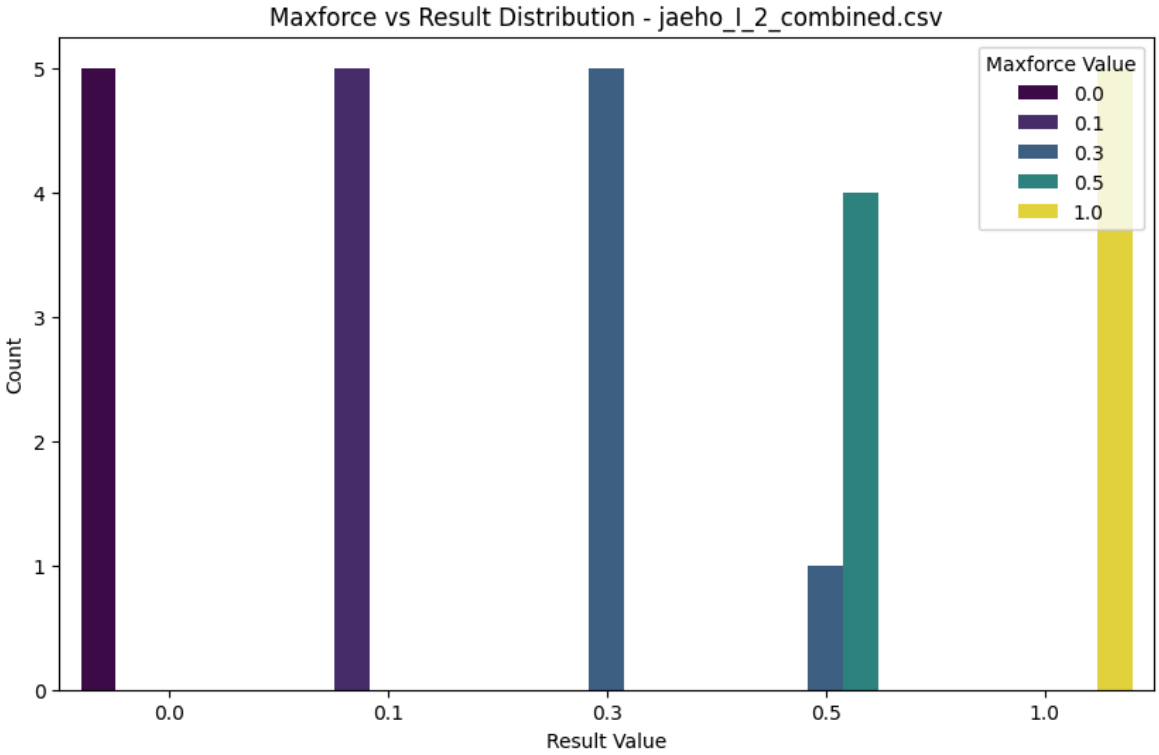

Maxforce vs Result Distribution - dohoon_S_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

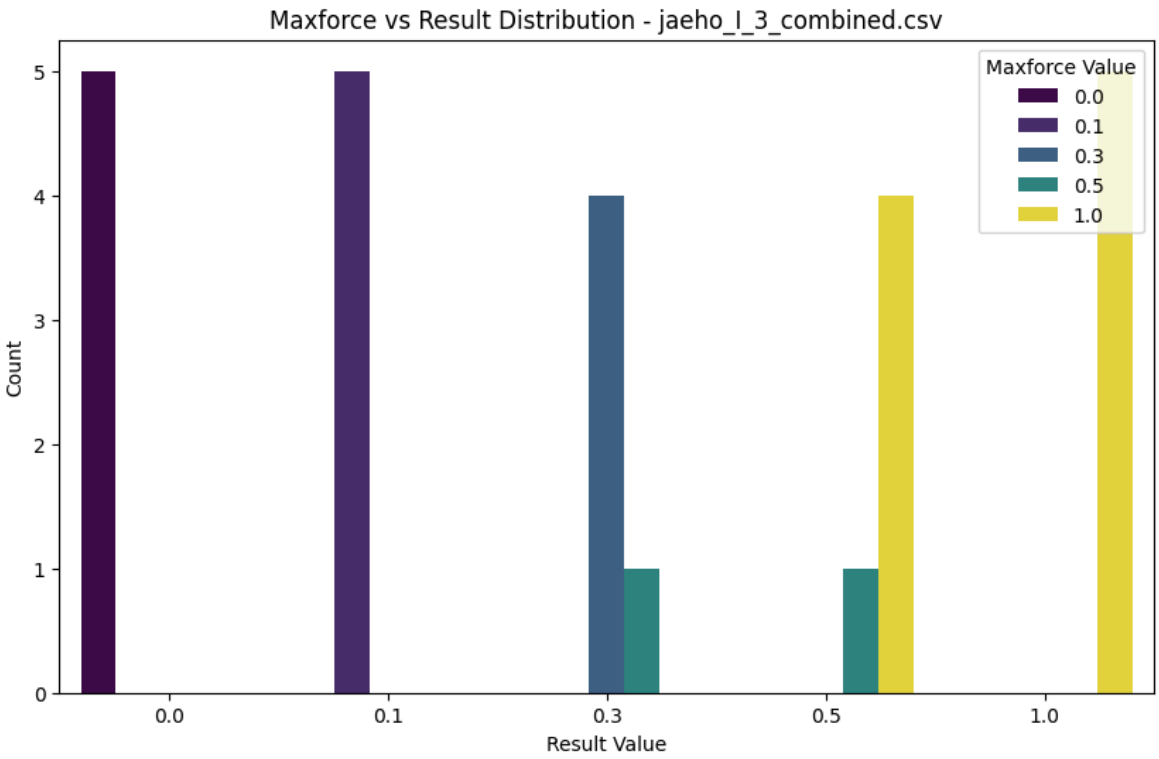

Maxforce vs Result Distribution - dohoon_S_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

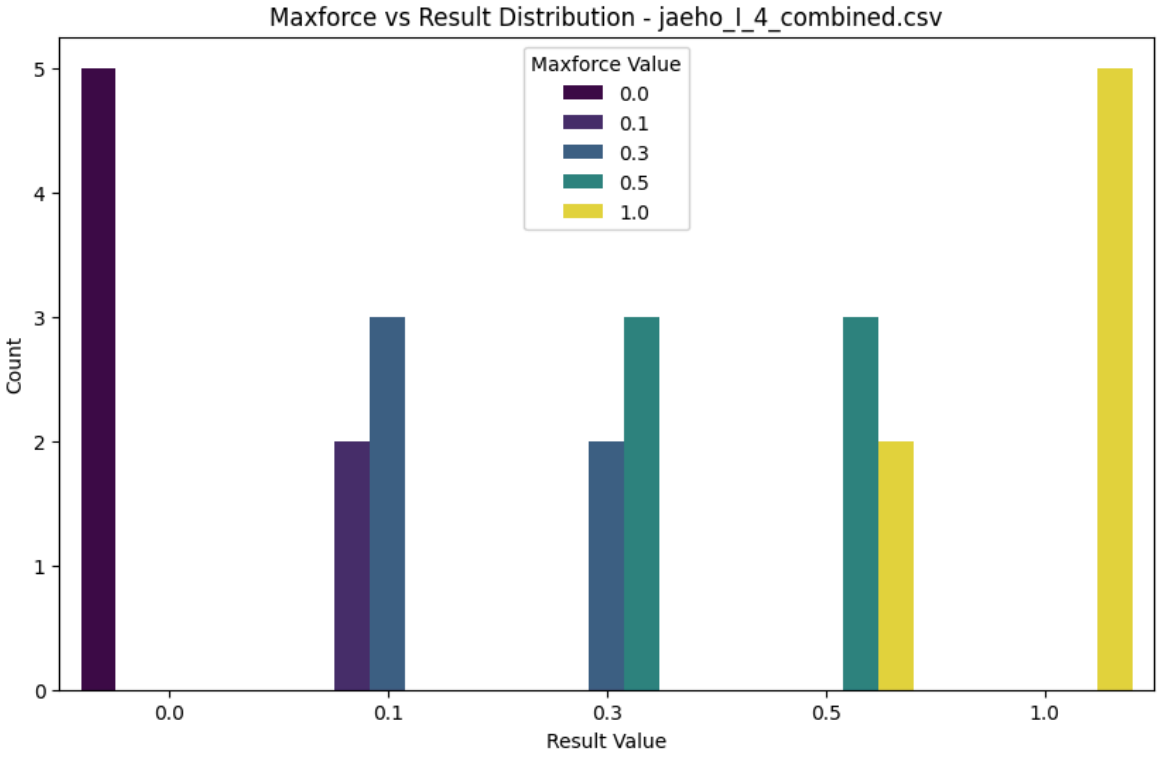

Maxforce vs Result Distribution - dohoon_S_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

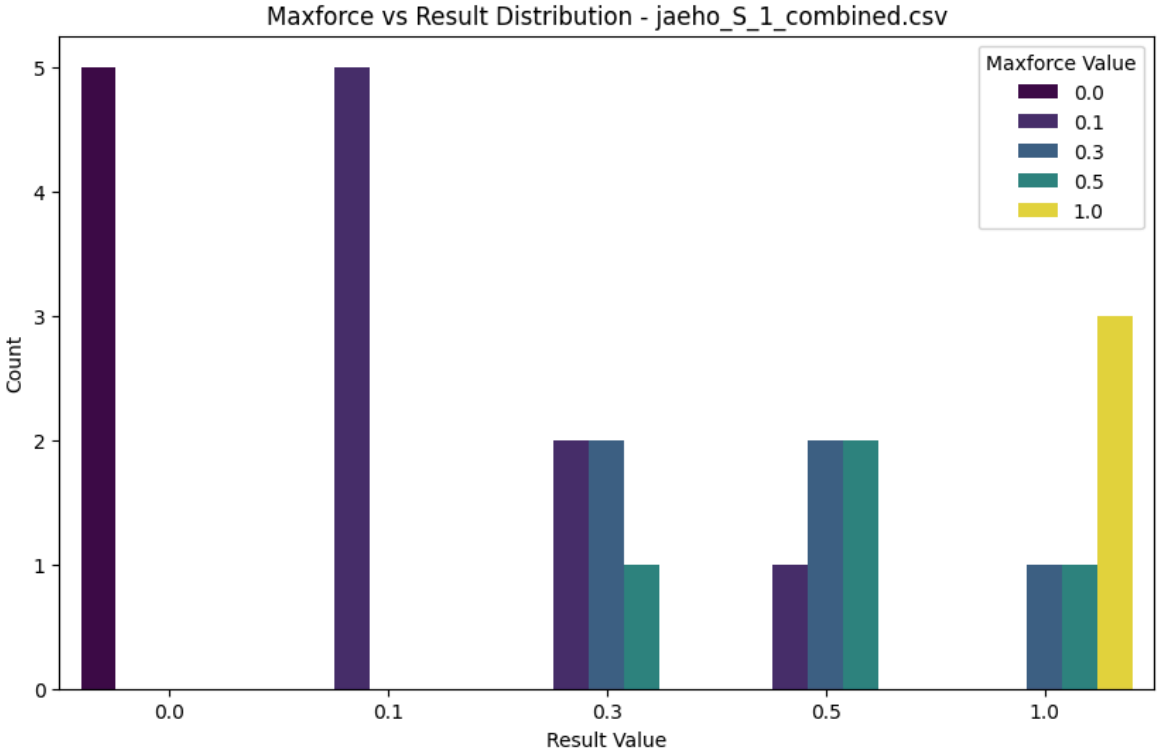

Maxforce vs Result Distribution - dohoon_S_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

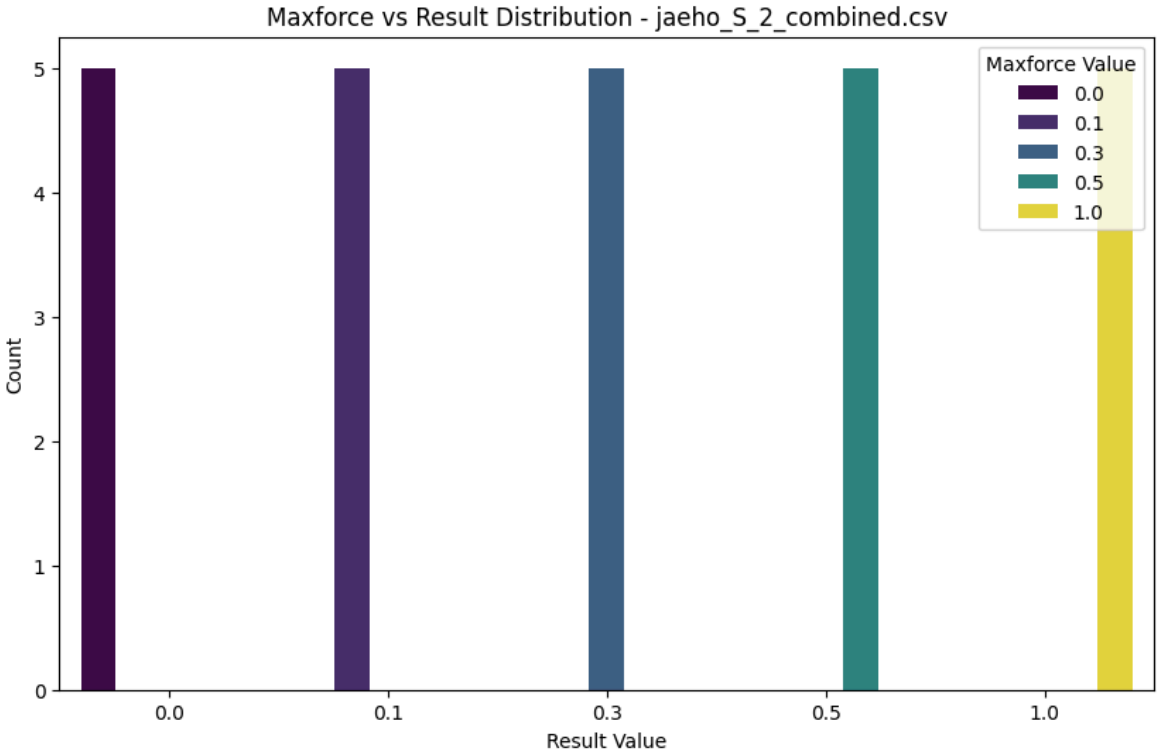

Maxforce vs Result Distribution - jaeho_l_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

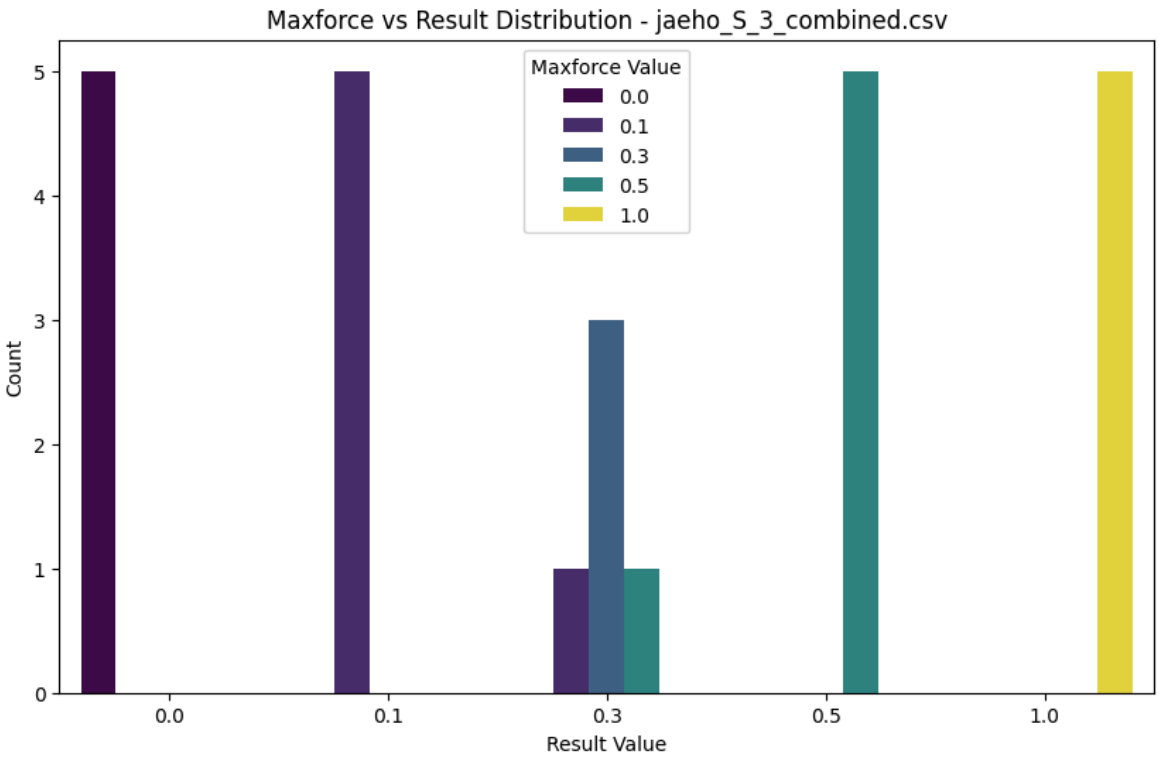

Maxforce vs Result Distribution - jaeho_l_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

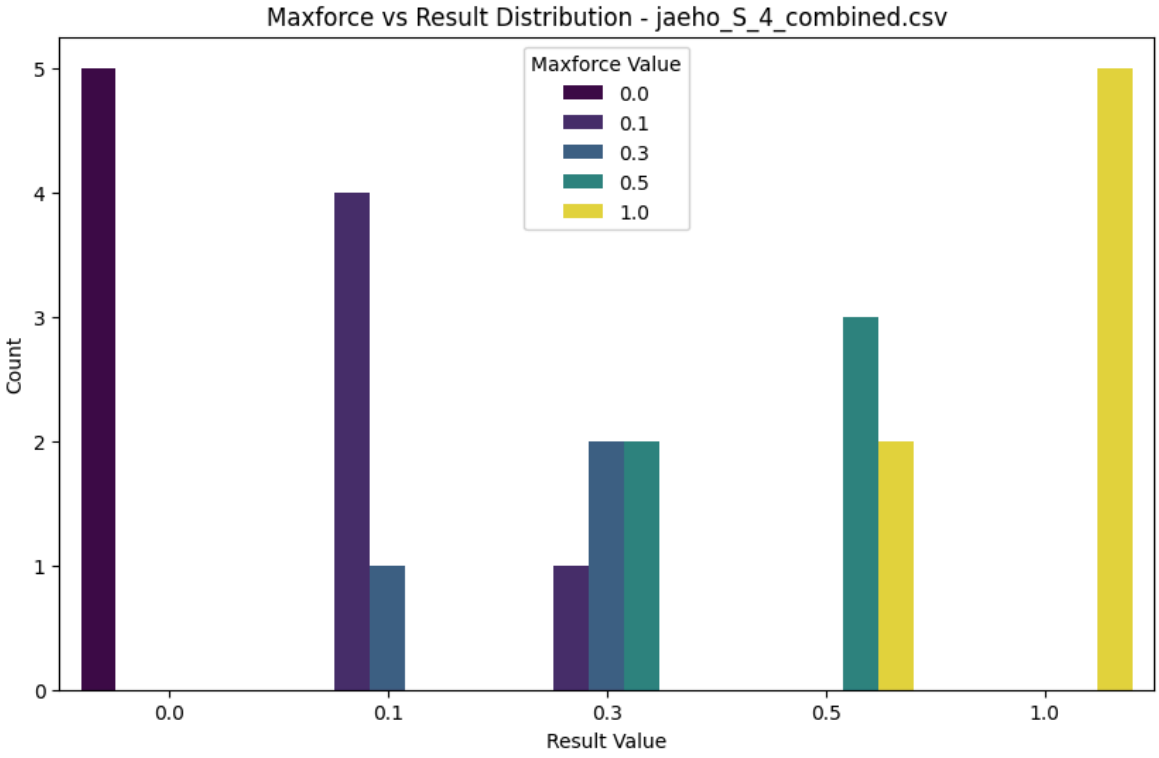

Maxforce vs Result Distribution - jaeho_l_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

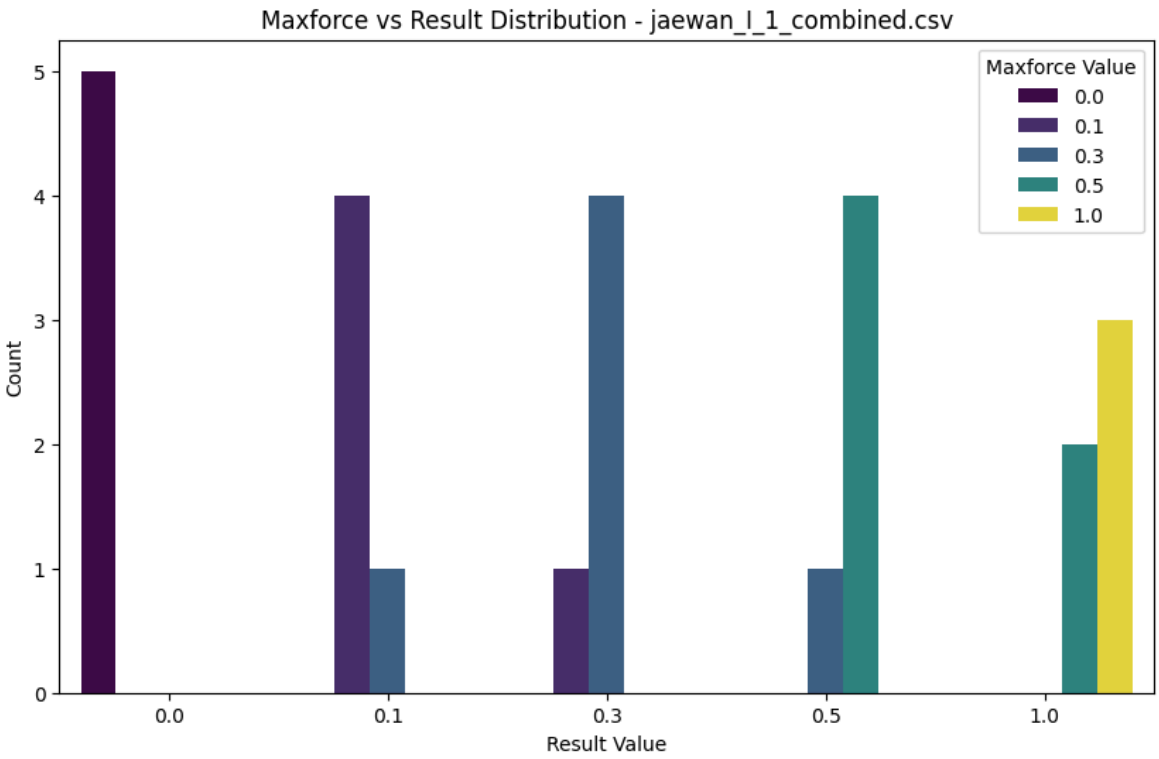

Maxforce vs Result Distribution - jaeho_l_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

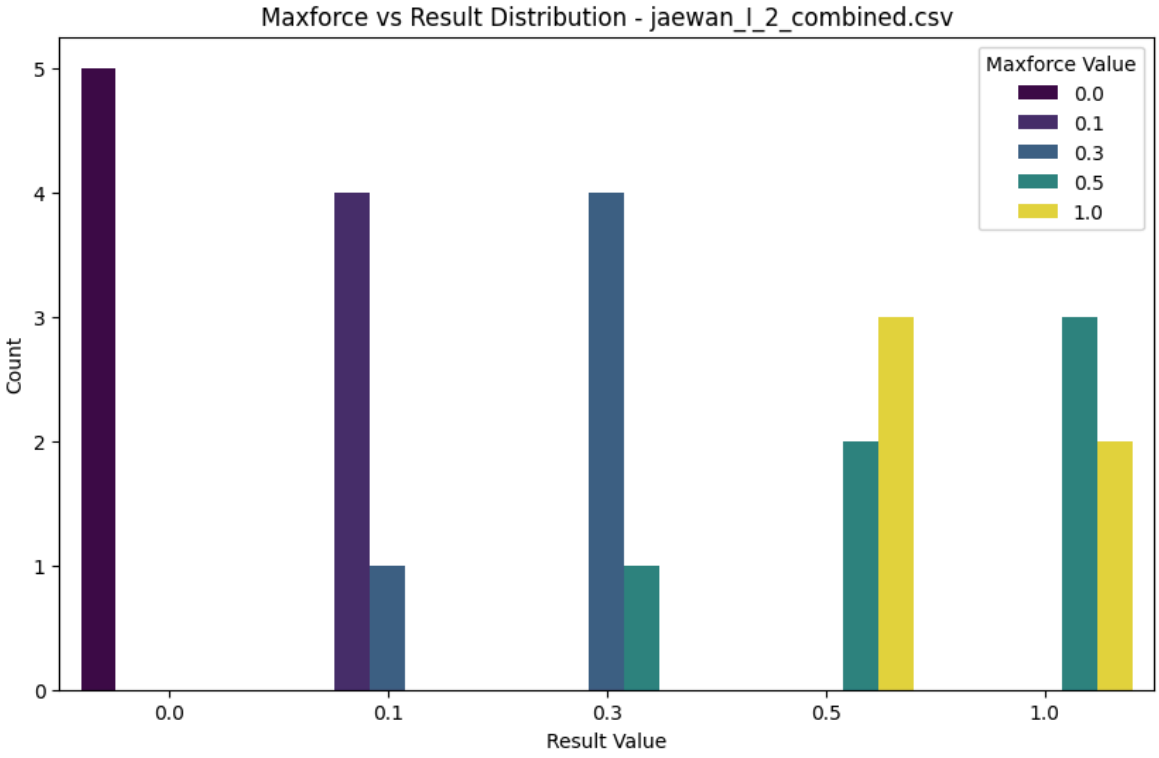

Maxforce vs Result Distribution - jaeho_S_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

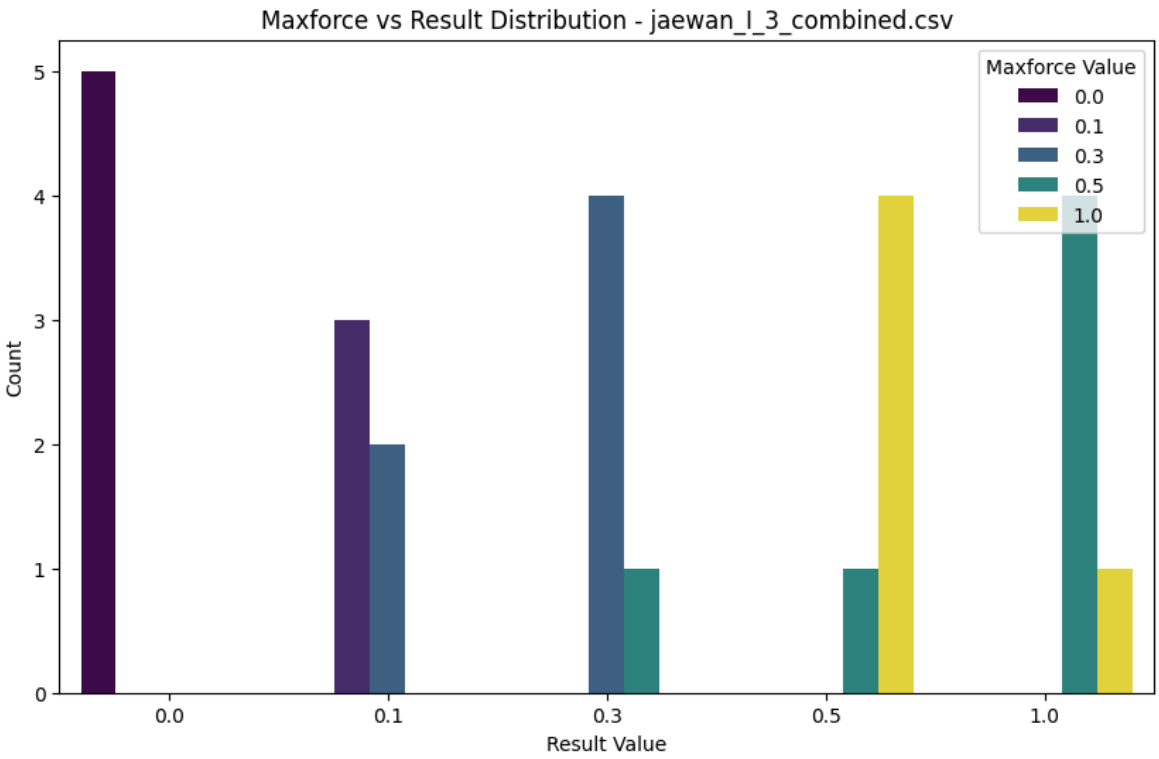

Maxforce vs Result Distribution - jaeho_S_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

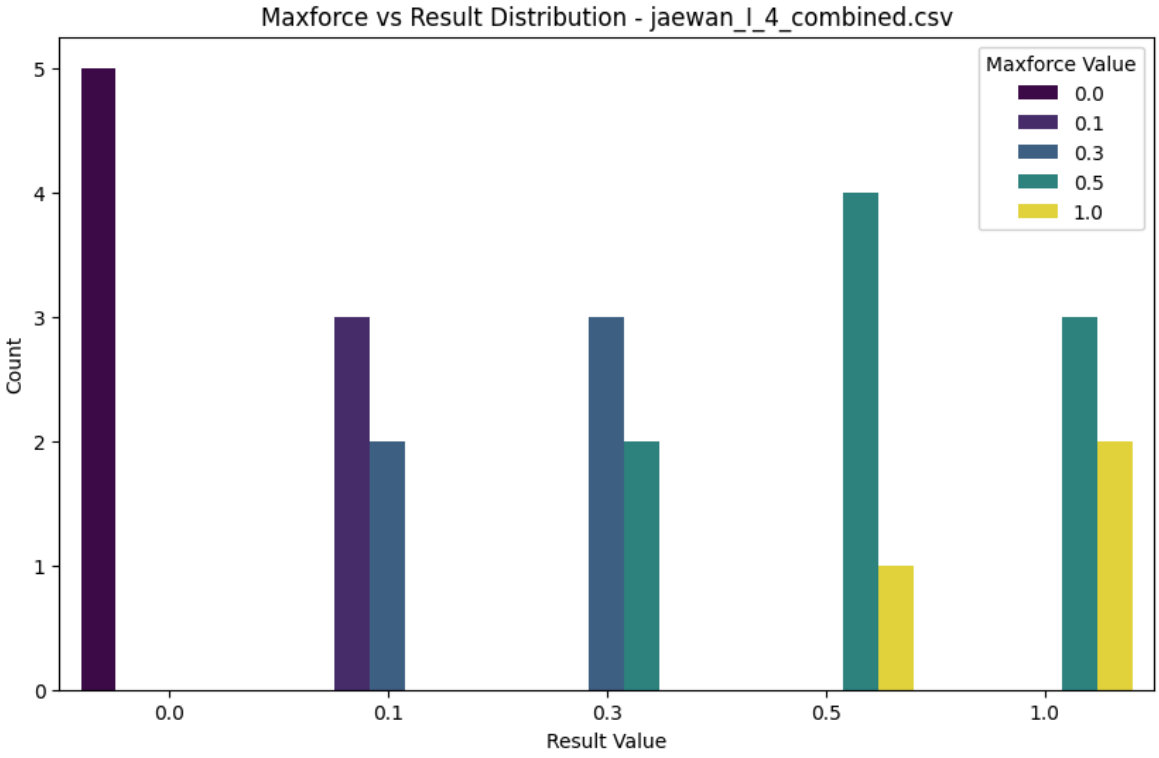

Maxforce vs Result Distribution - jaeho_S_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

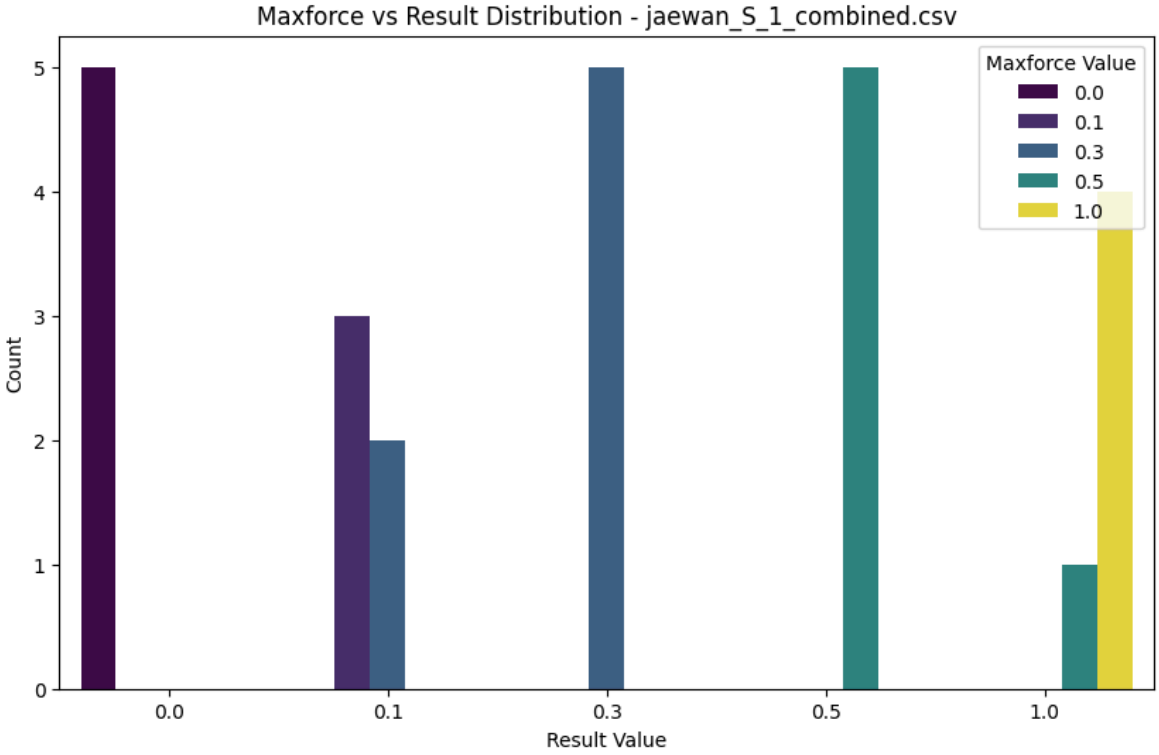

Maxforce vs Result Distribution - jaeho_S_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

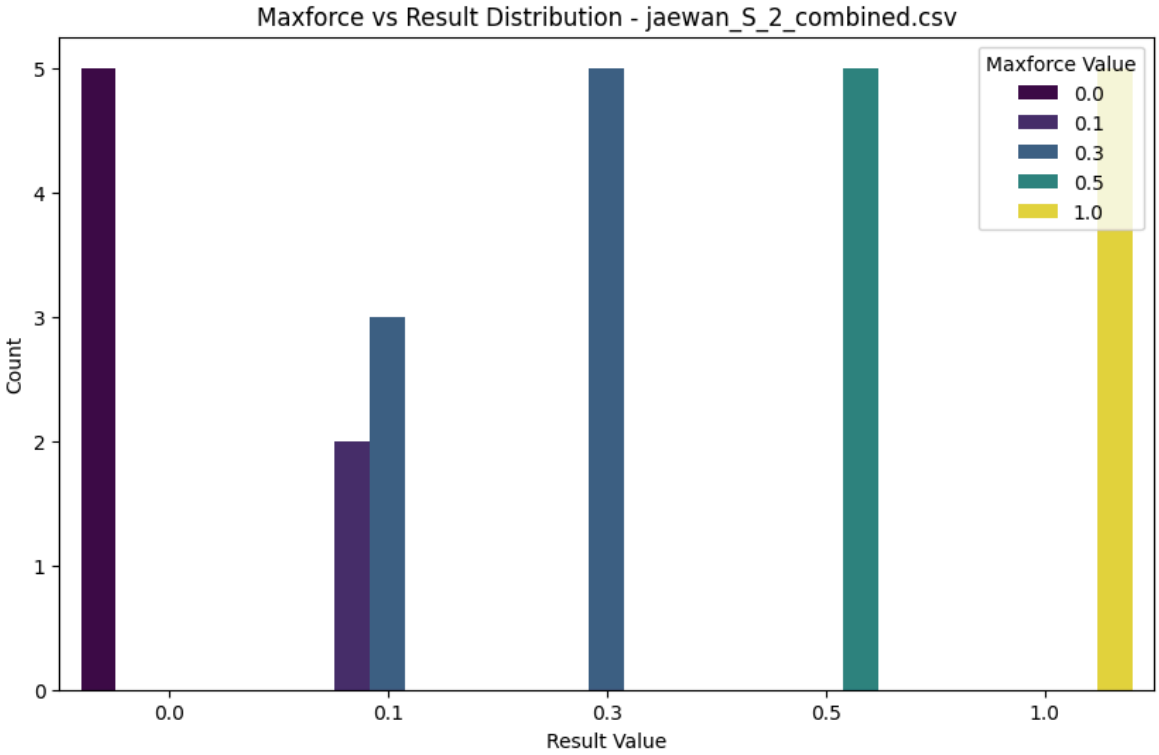

Maxforce vs Result Distribution - jaewan_I_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

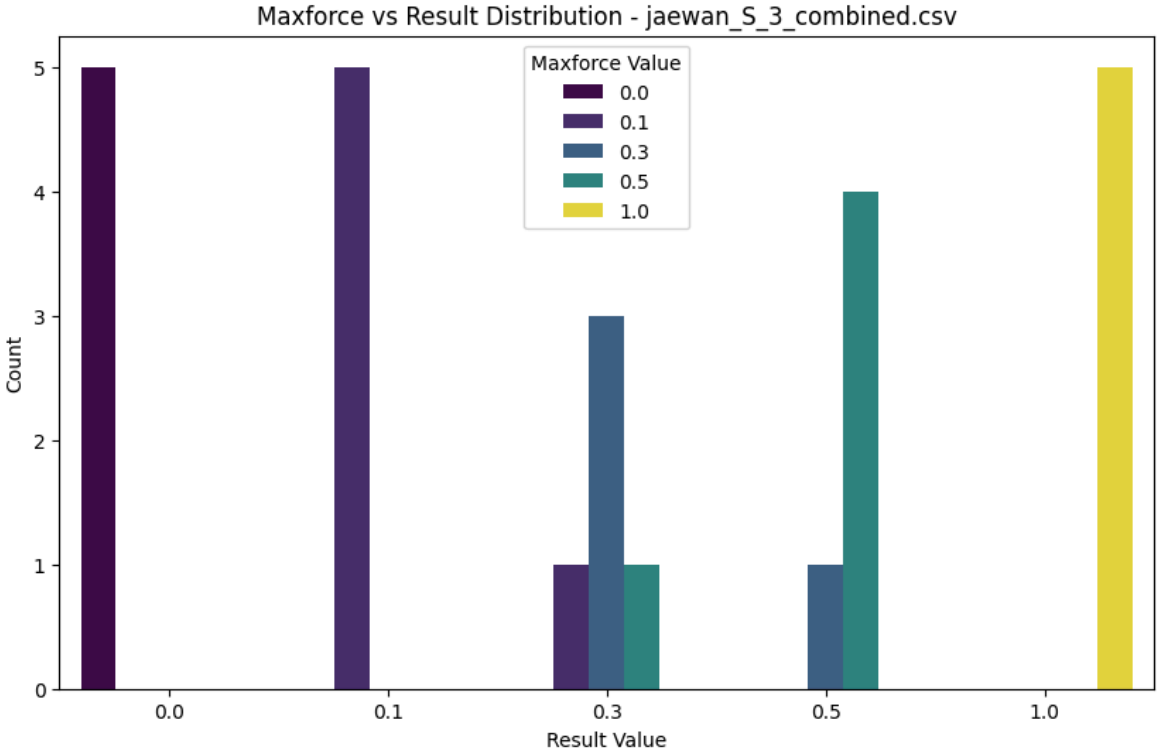

Maxforce vs Result Distribution - jaewan_I_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

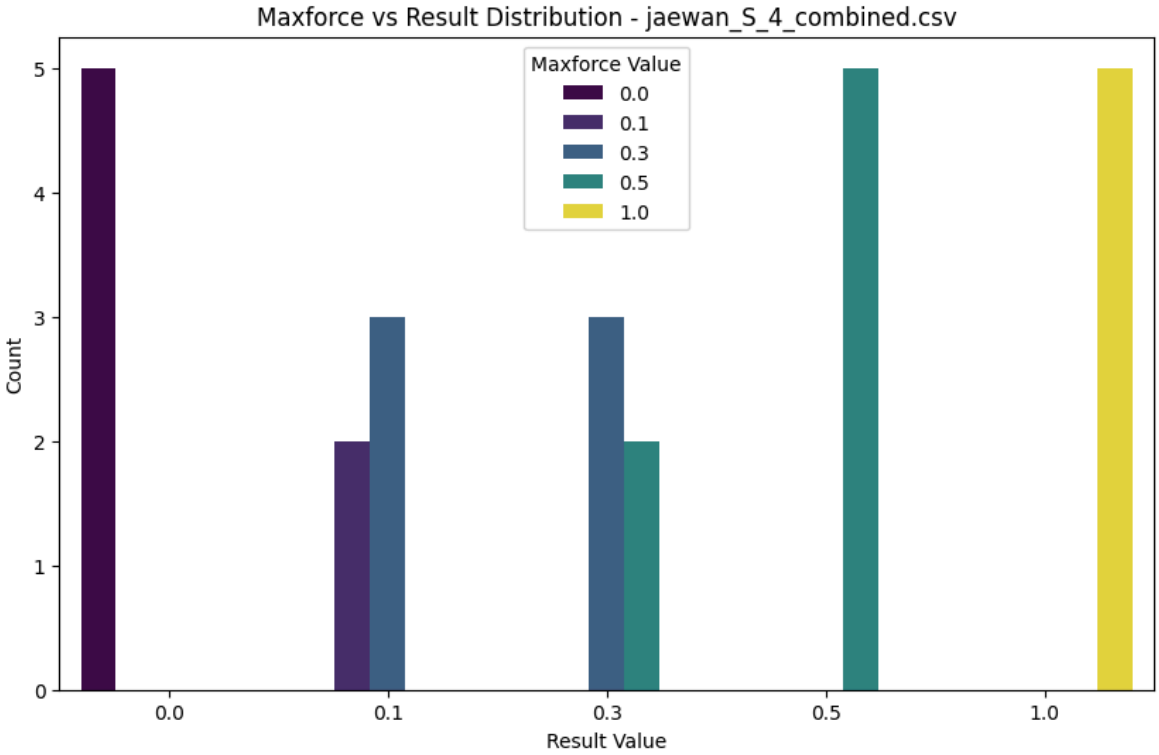

Maxforce vs Result Distribution - jaewan_I_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

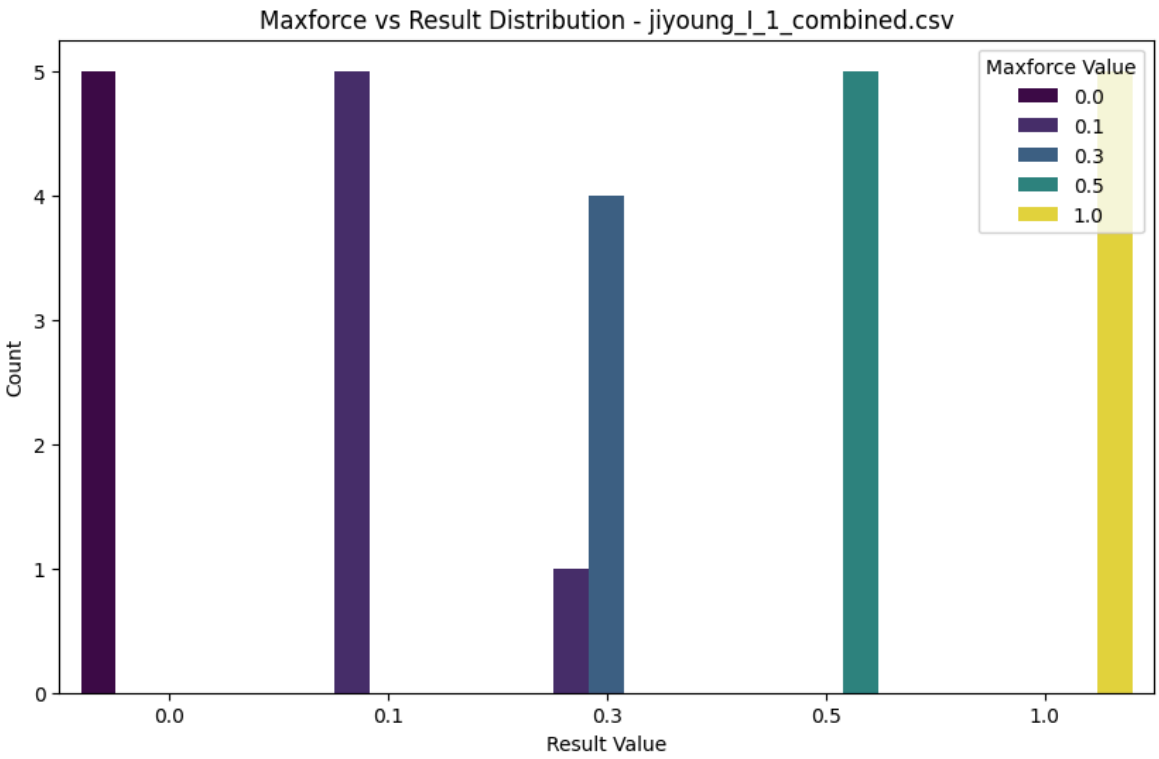

Maxforce vs Result Distribution - jaewan_I_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

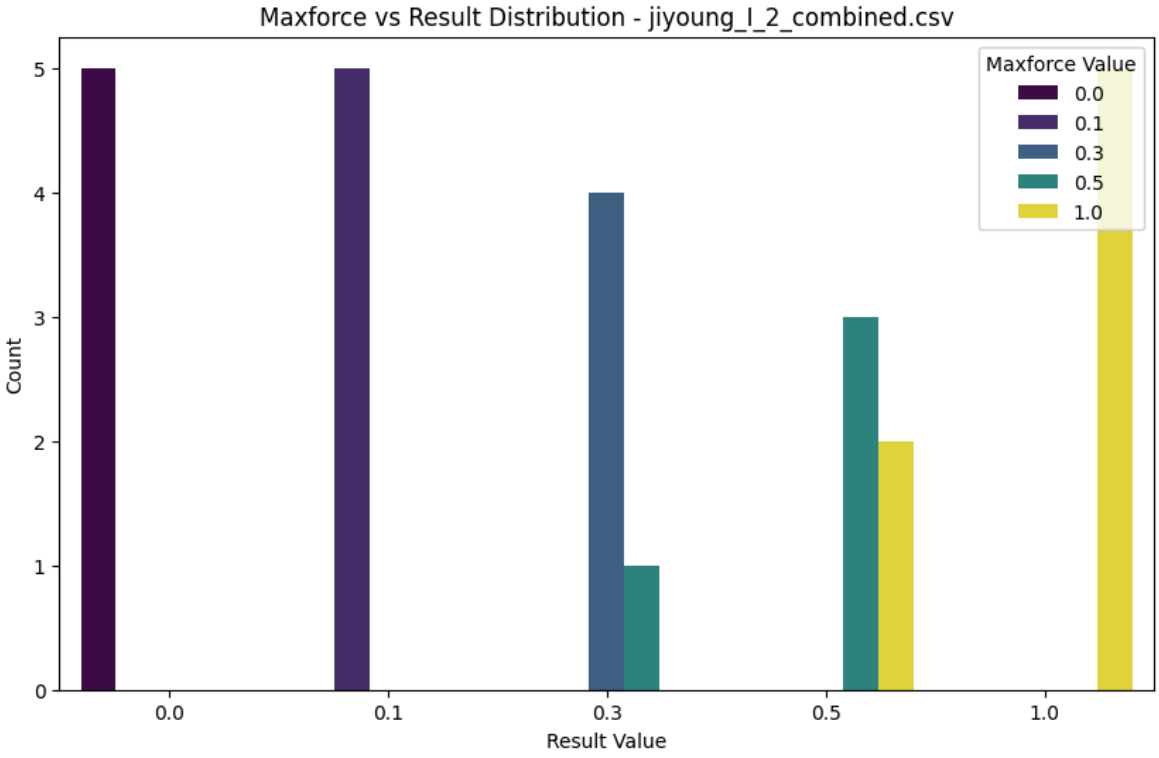

Maxforce vs Result Distribution - jaewan_S_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

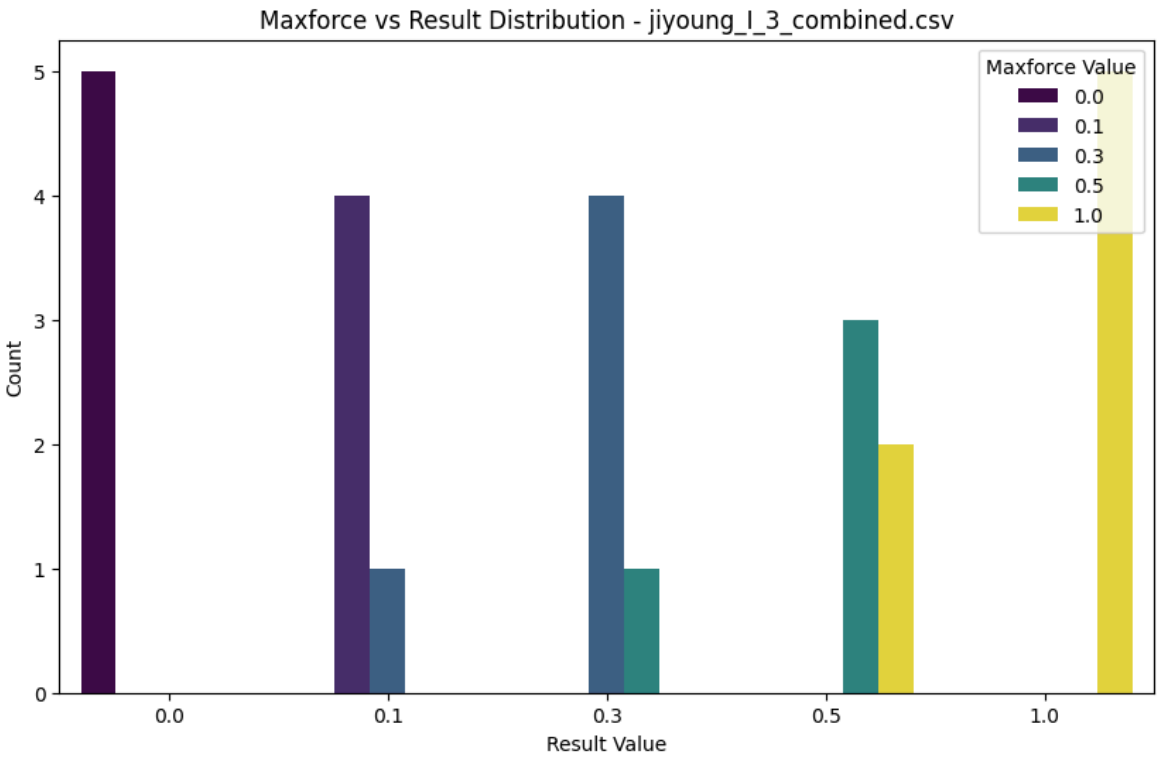

Maxforce vs Result Distribution - jaewan_S_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

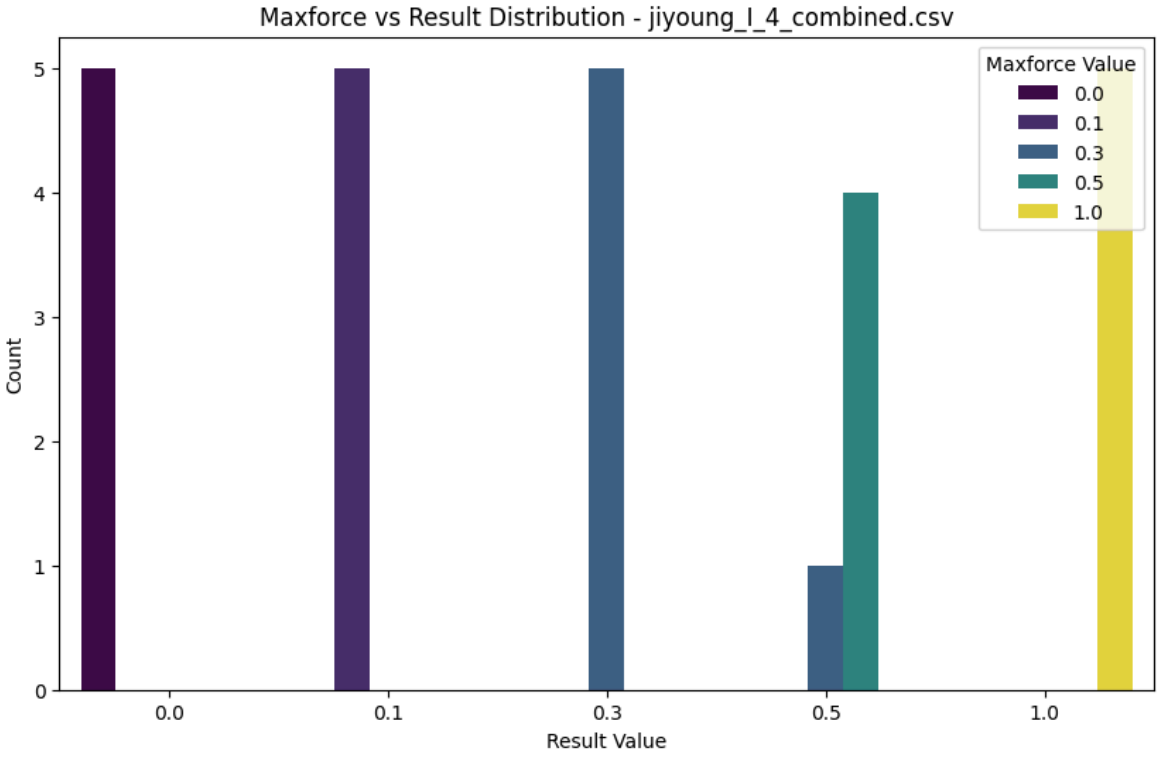

Maxforce vs Result Distribution - jaewan_S_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

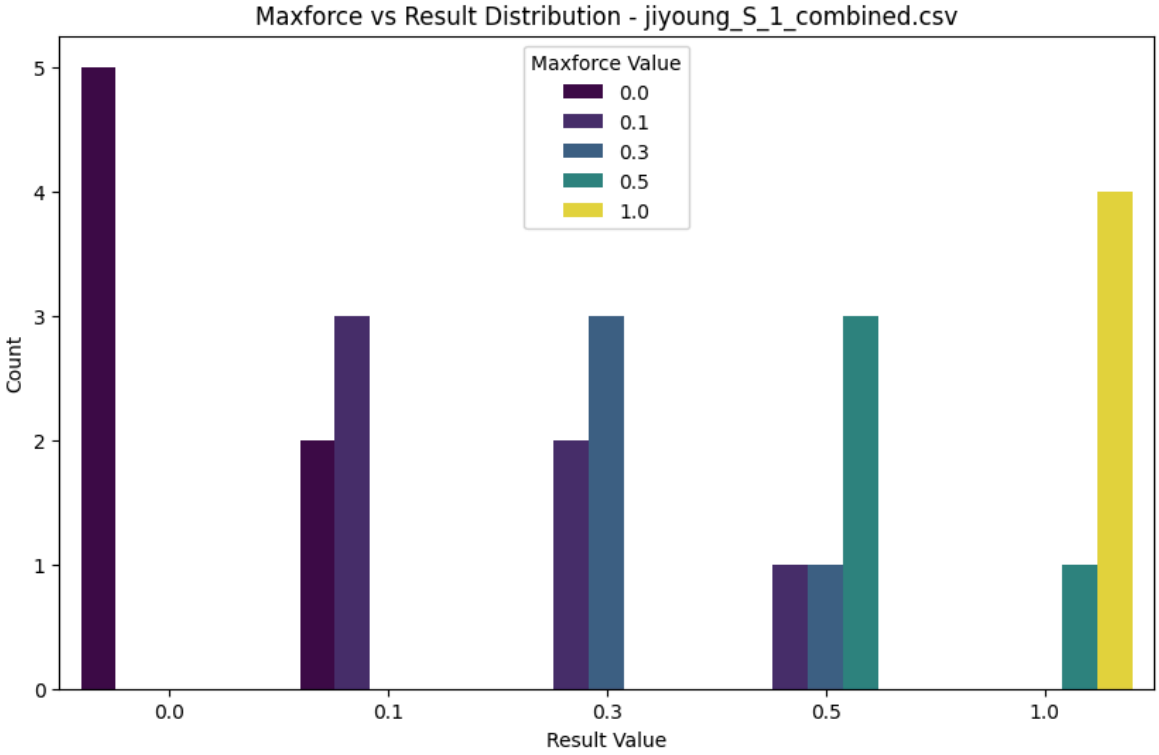

Maxforce vs Result Distribution - jaewan_S_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

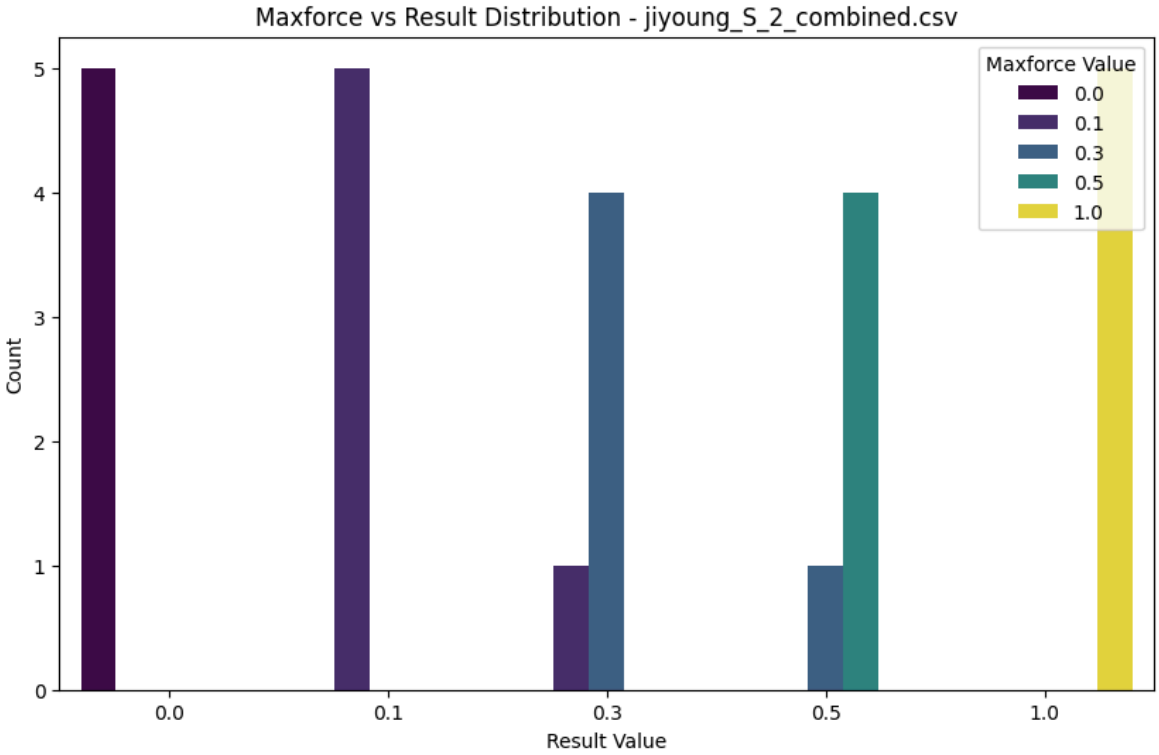

Maxforce vs Result Distribution - jiyoung_I_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

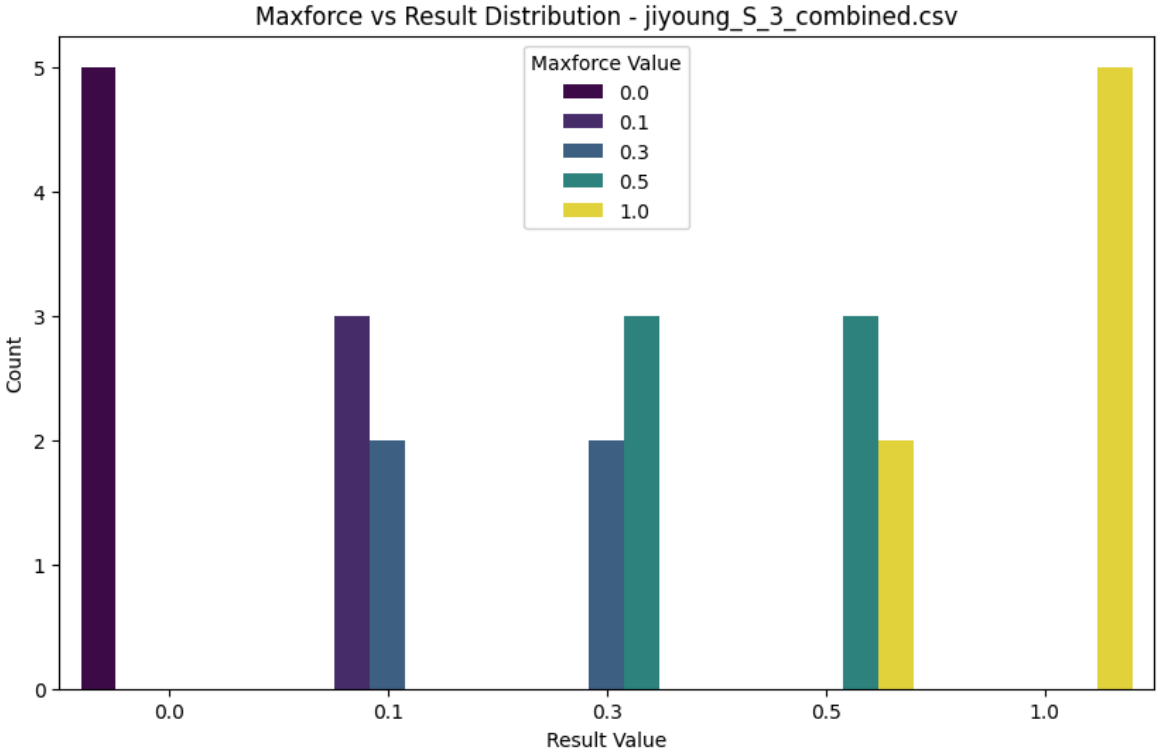

Maxforce vs Result Distribution - jiyoung_I_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

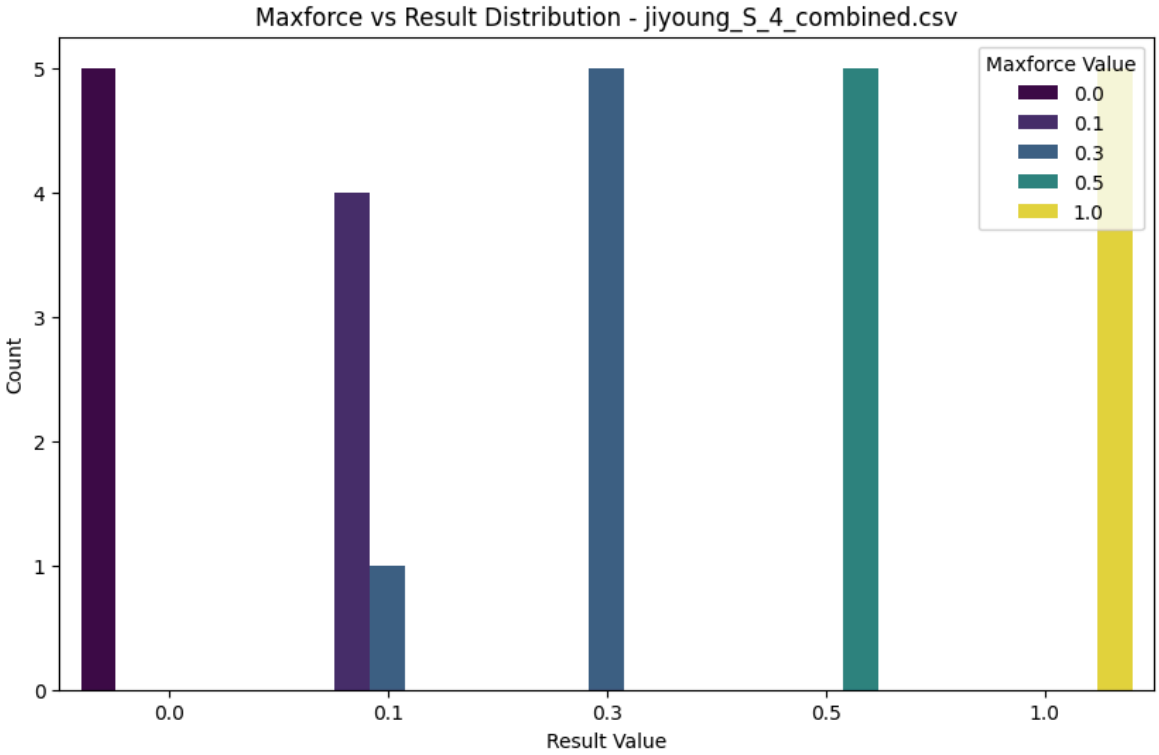

Maxforce vs Result Distribution - jiyoung_I_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

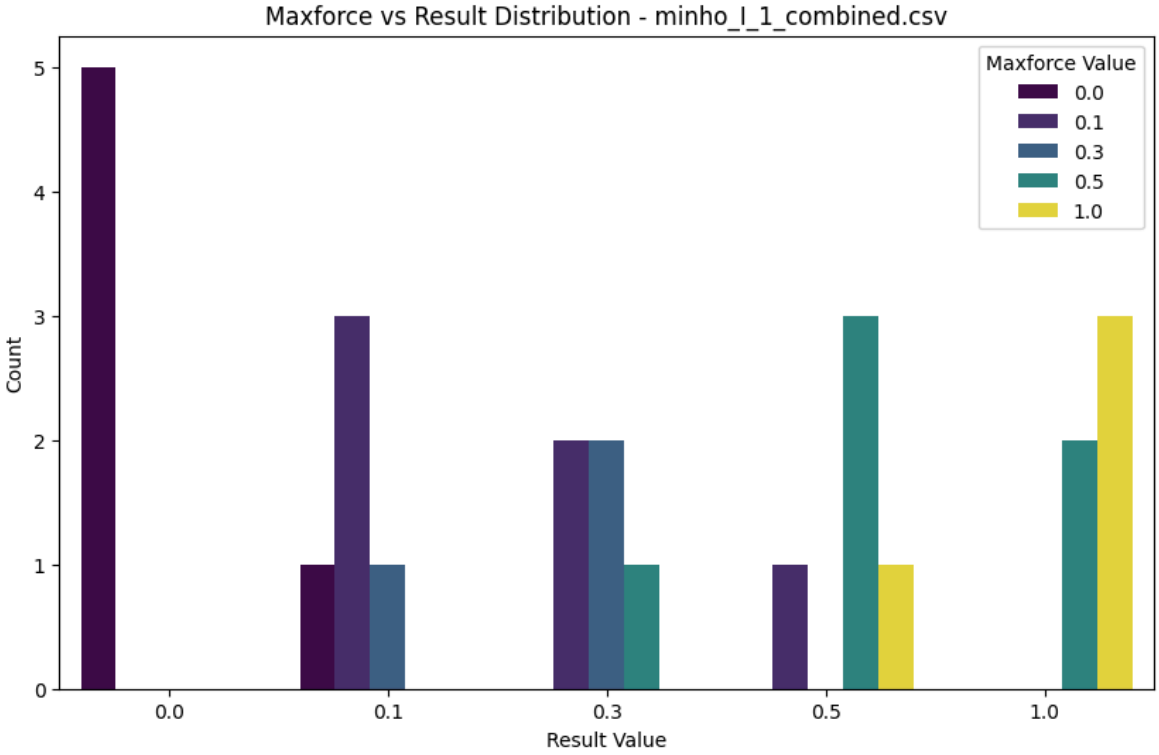

Maxforce vs Result Distribution - jiyoung_I_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

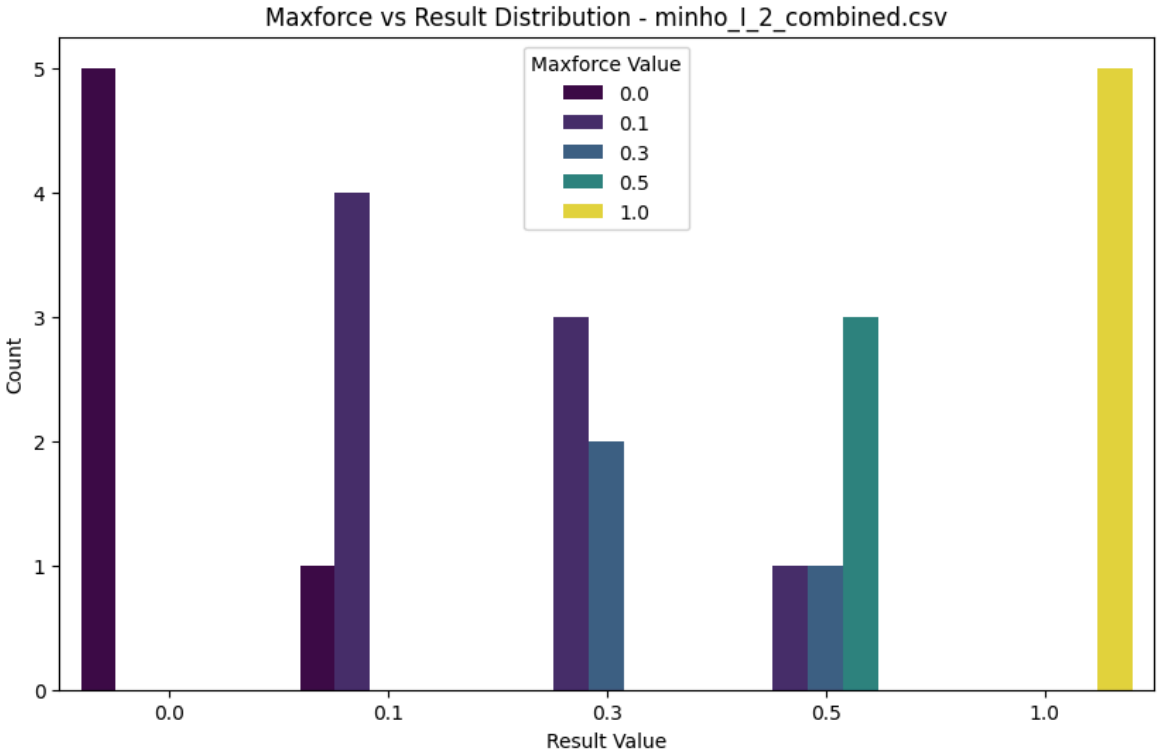

Maxforce vs Result Distribution - jiyoung_S_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

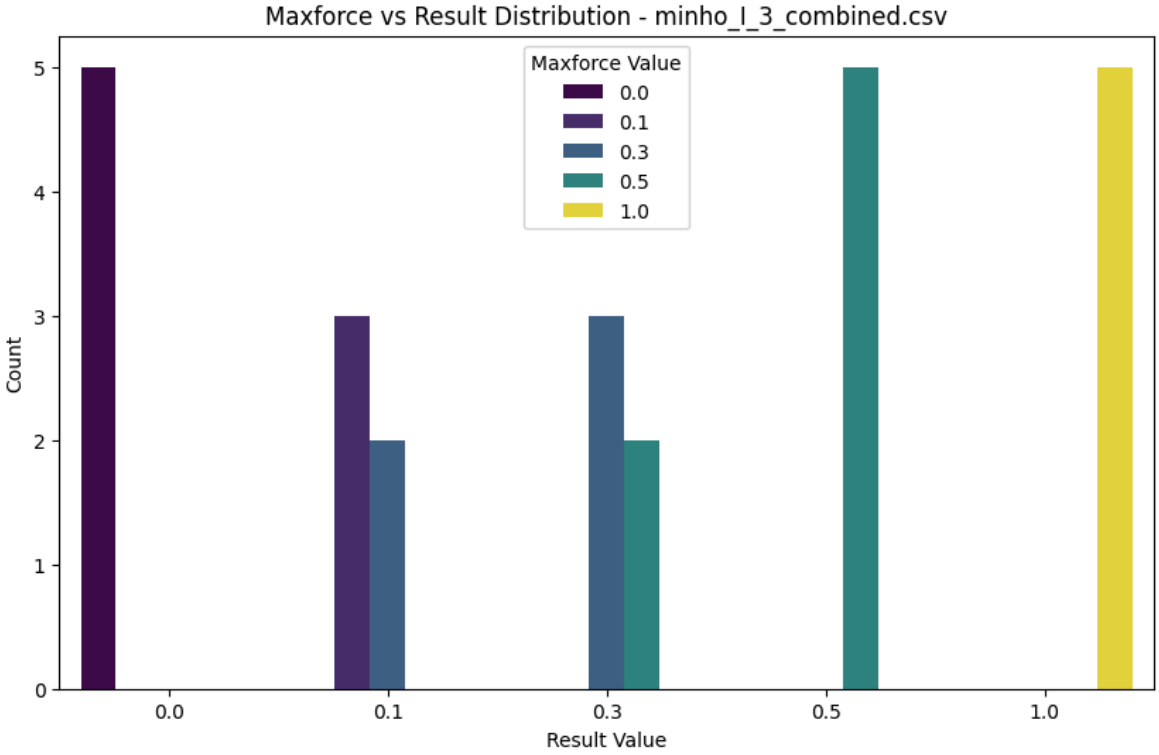

Maxforce vs Result Distribution - jiyoung_S_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

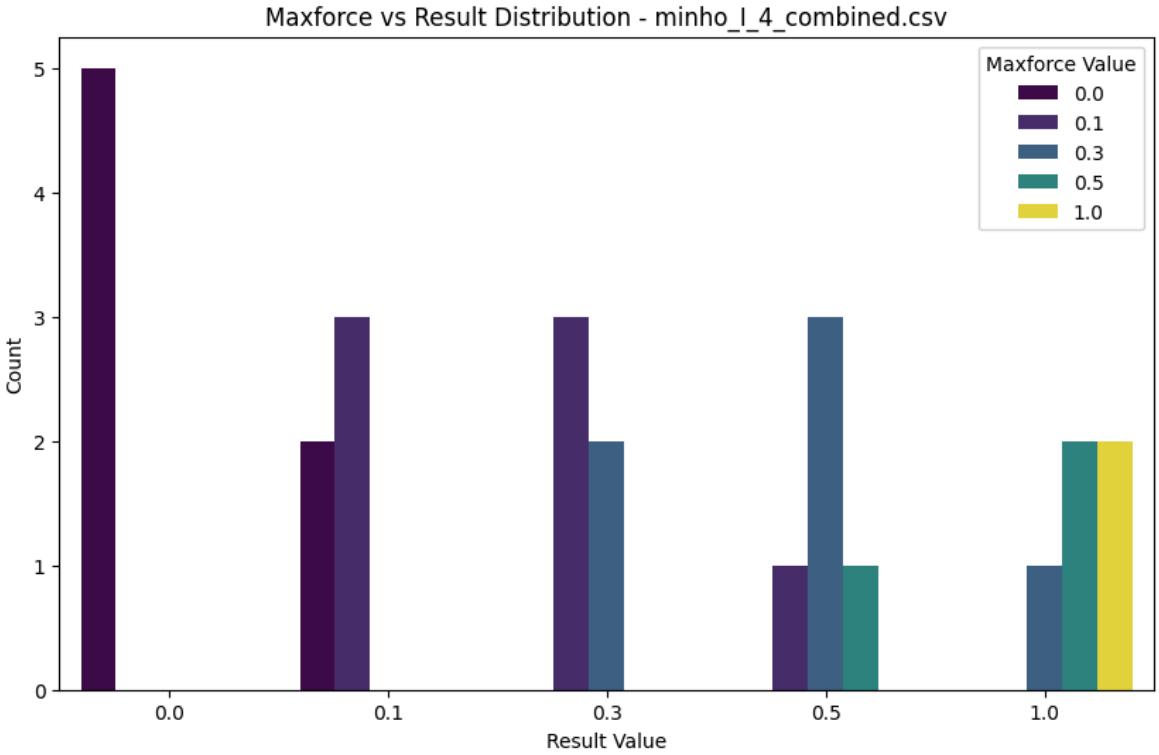

Maxforce vs Result Distribution - jiyoung_S_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

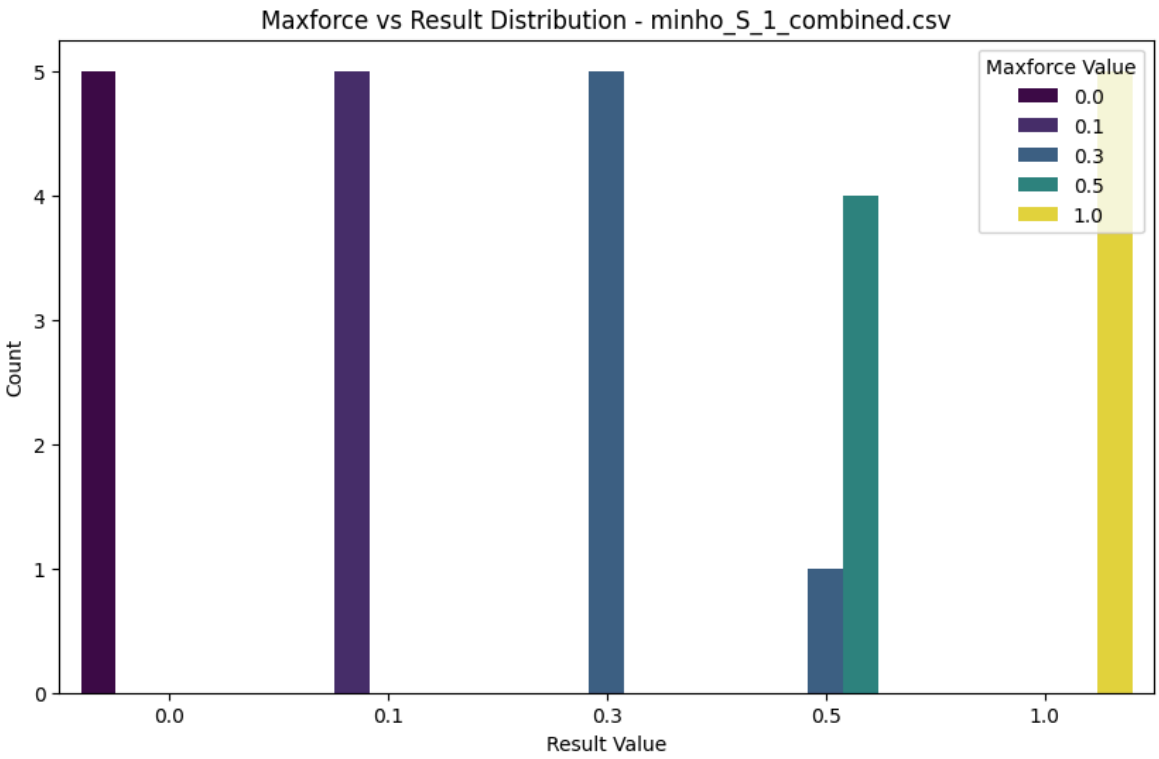

Maxforce vs Result Distribution - jiyoung_S_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

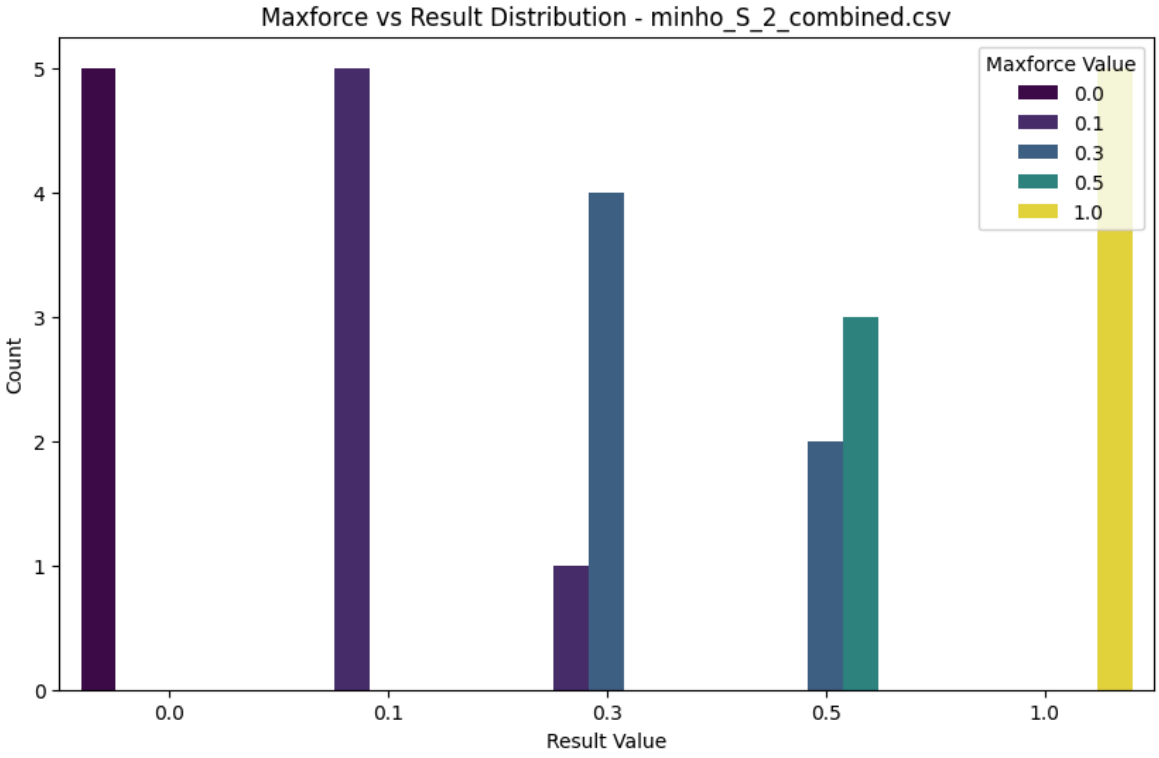

Maxforce vs Result Distribution - minho_I_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

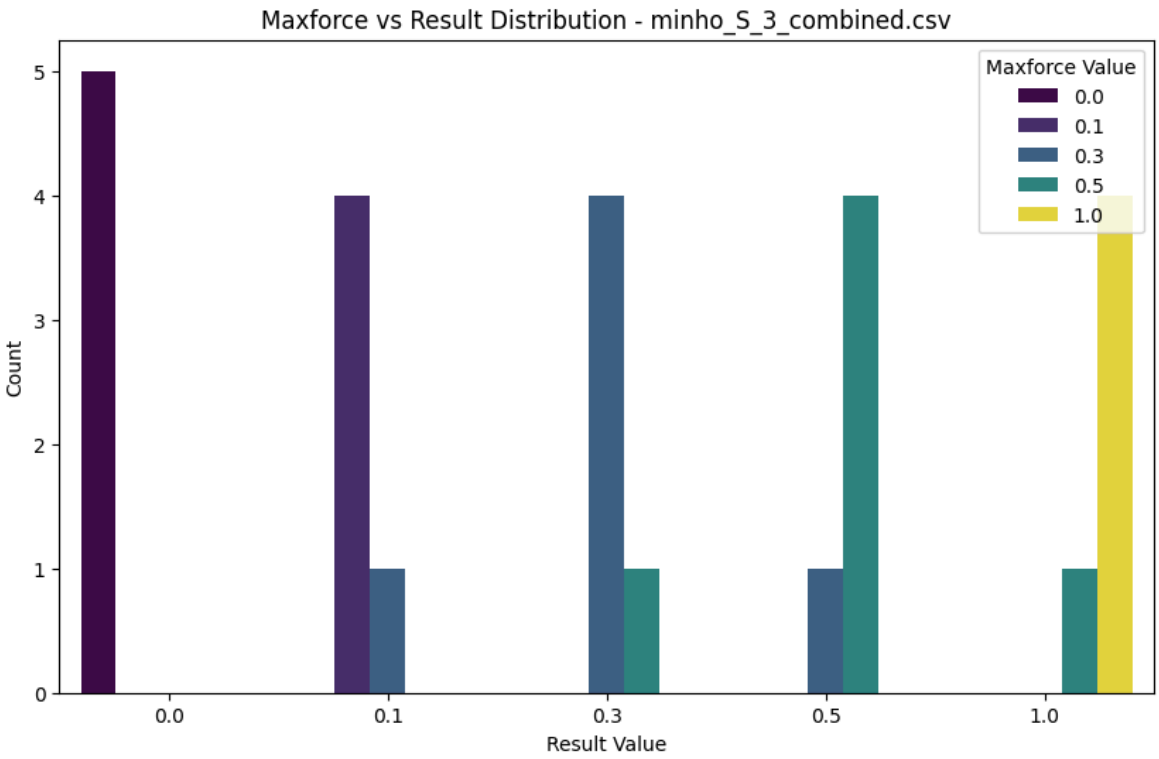

Maxforce vs Result Distribution - minho_I_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

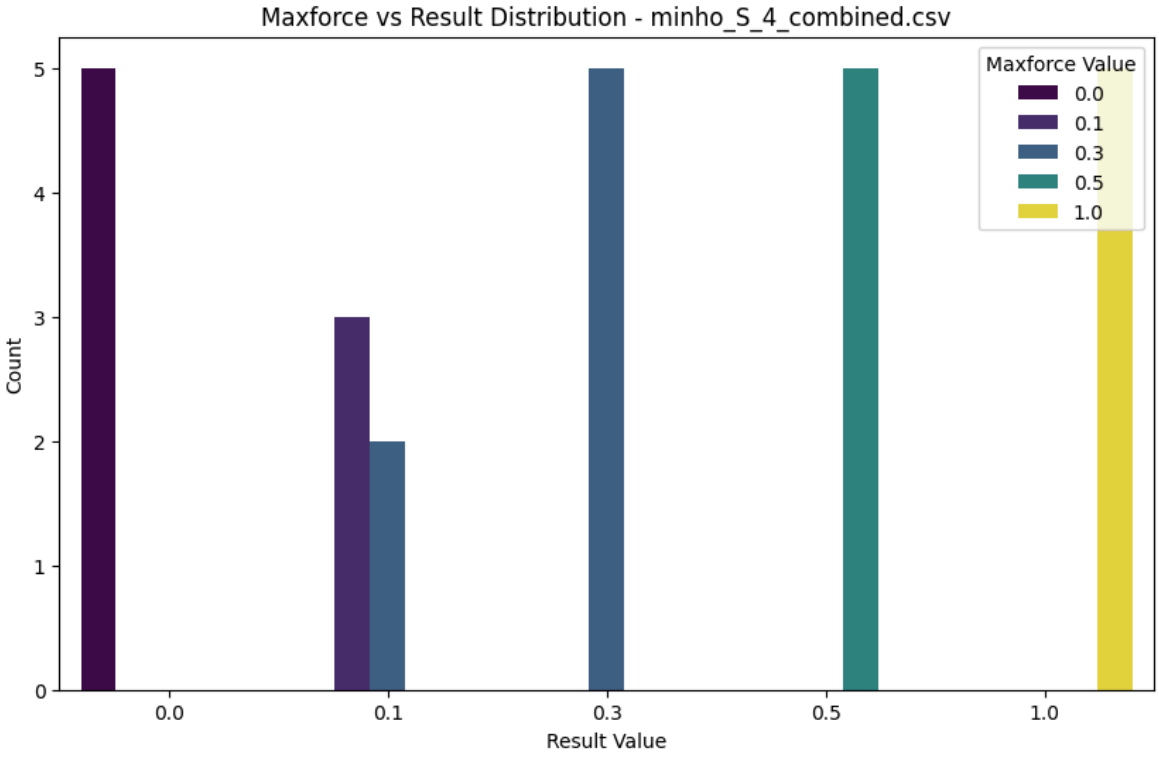

Maxforce vs Result Distribution - minho_I_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
Result Value
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0

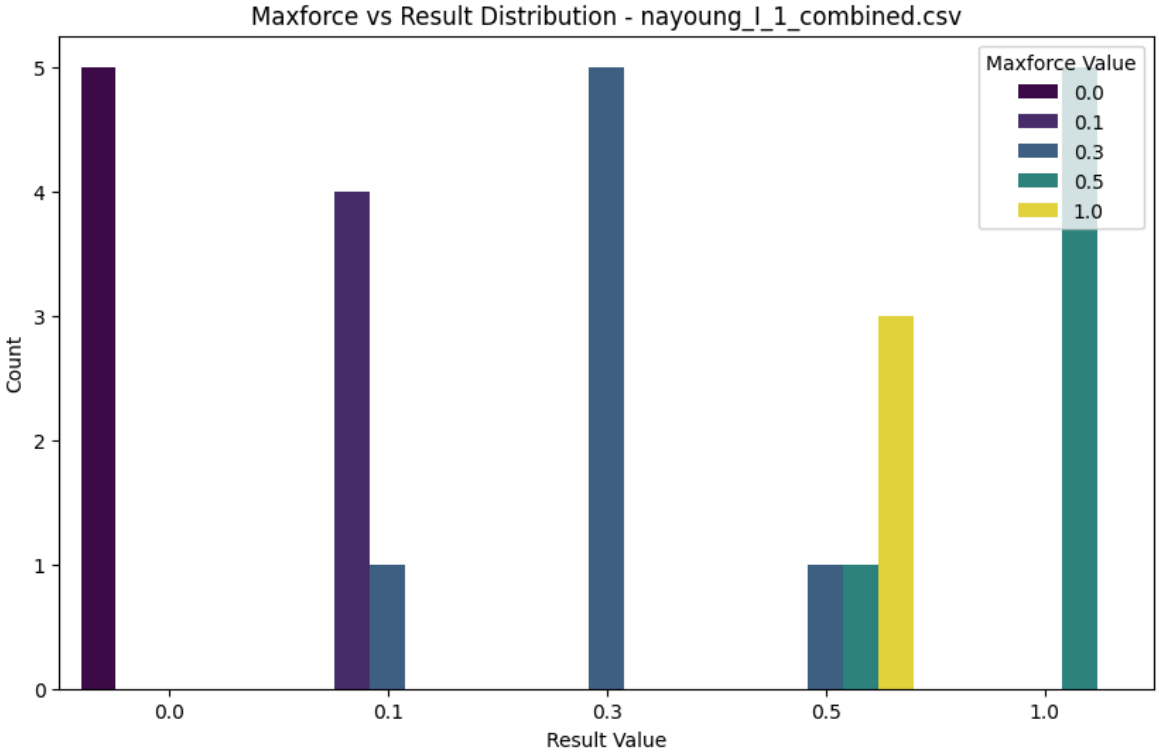

Maxforce vs Result Distribution - minho_I_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
Result Value
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0

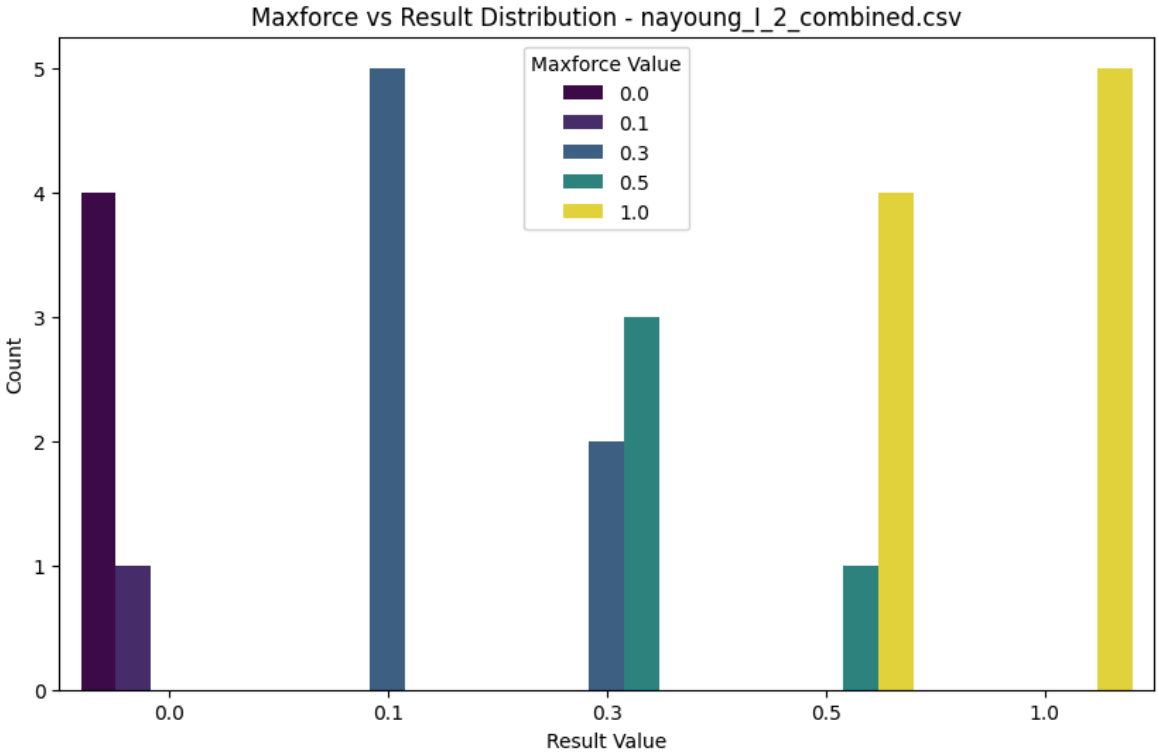

Maxforce vs Result Distribution - minho_S_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

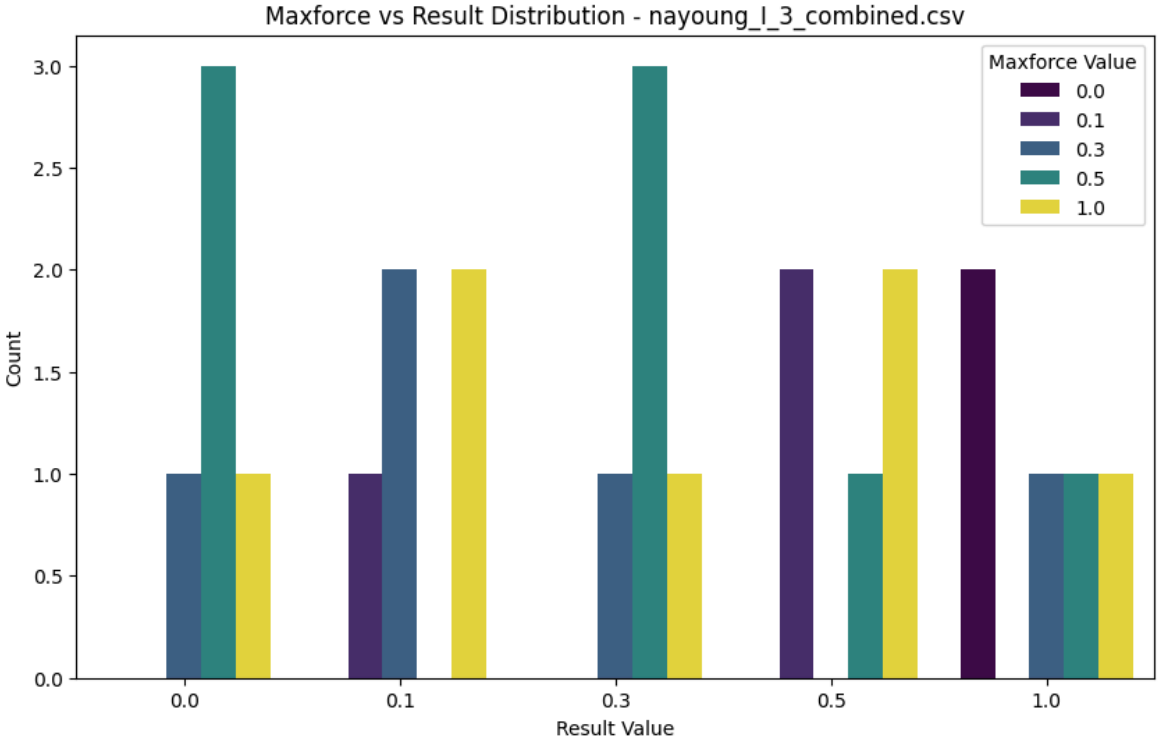

Maxforce vs Result Distribution - minho_S_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

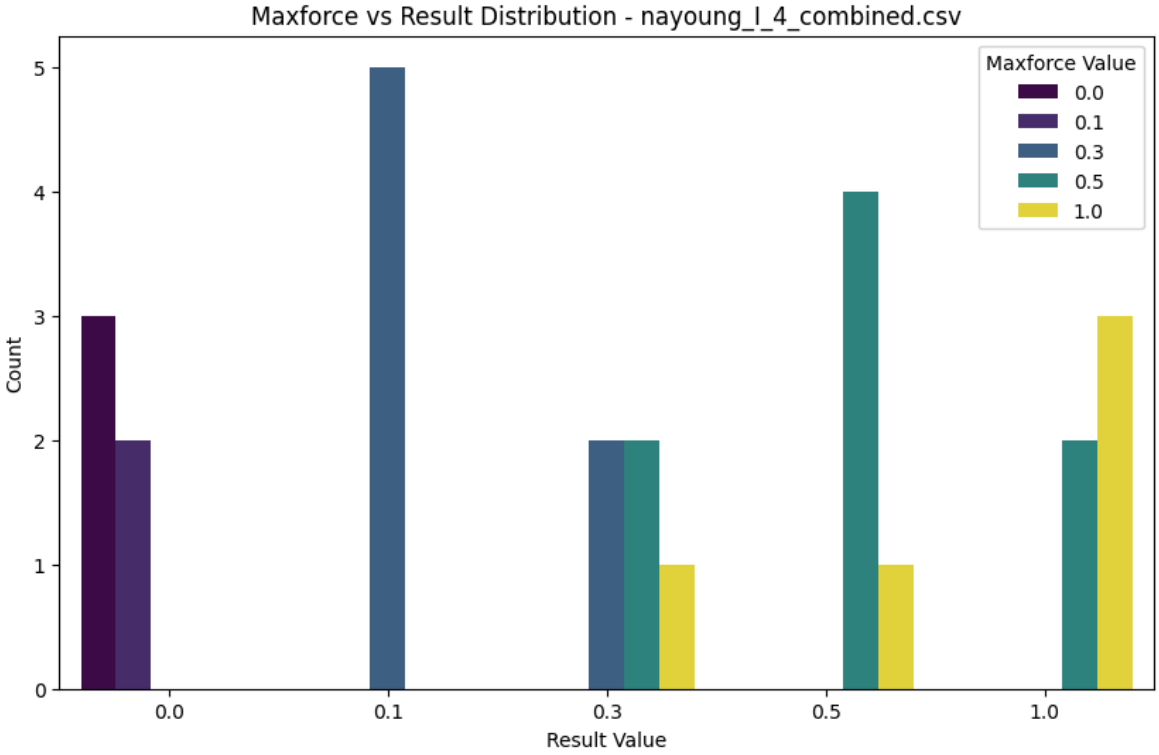

Maxforce vs Result Distribution - minho_S_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

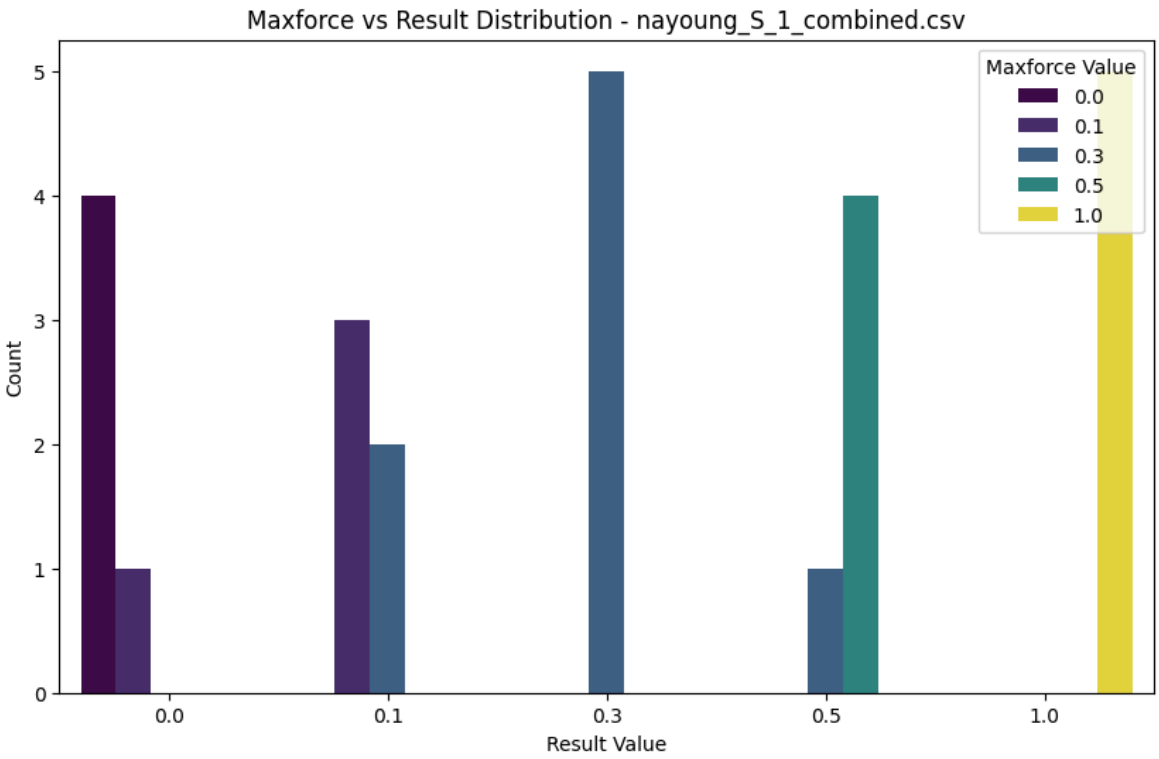

Maxforce vs Result Distribution - minho_S_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

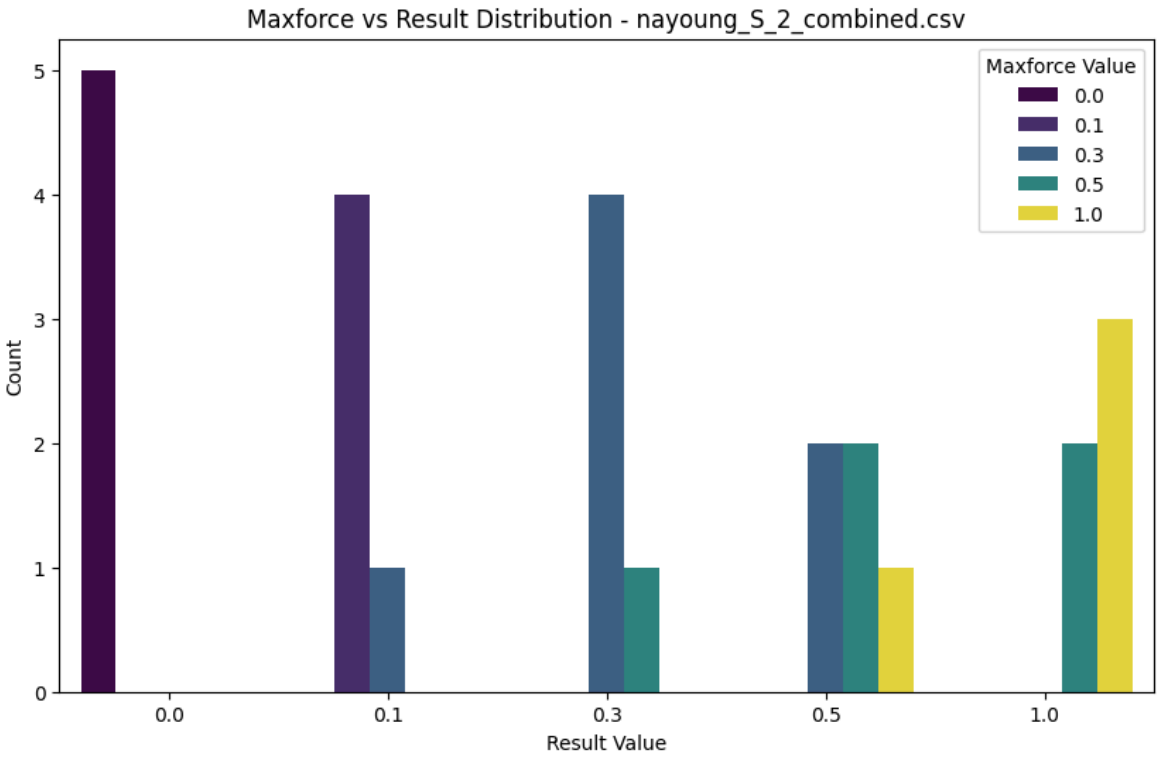

Maxforce vs Result Distribution - nayoung_I_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

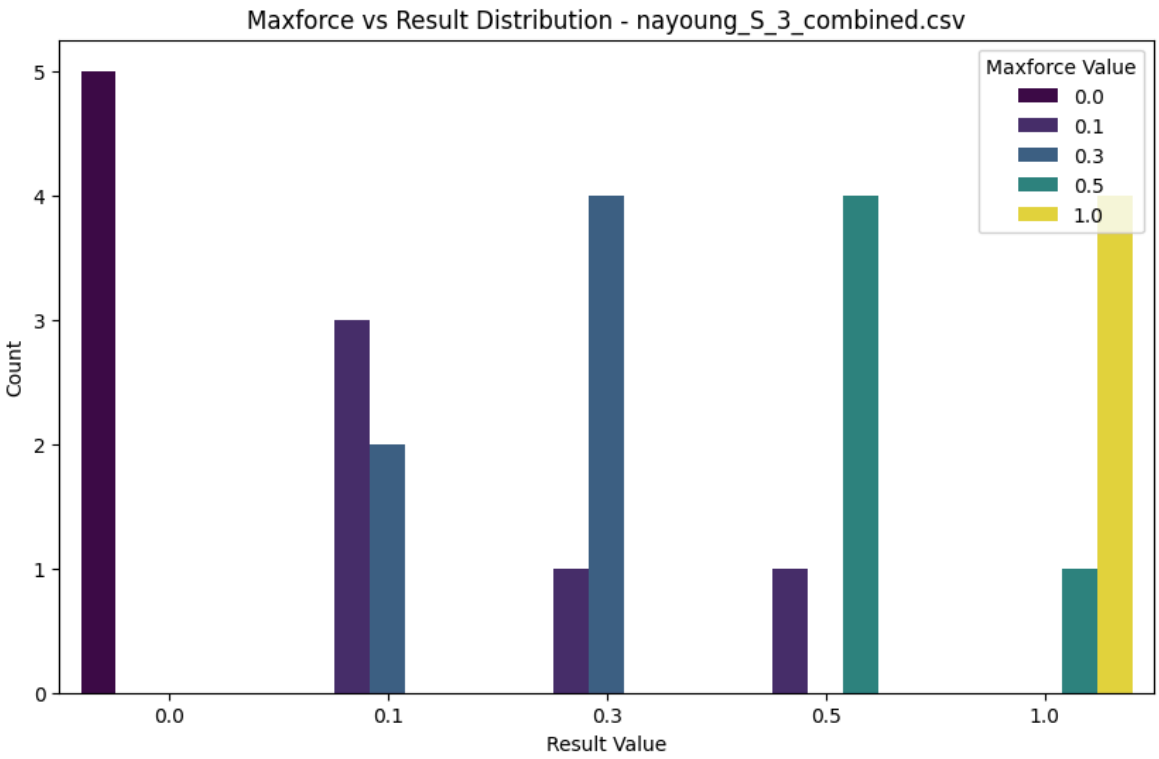

Maxforce vs Result Distribution - nayoung_I_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

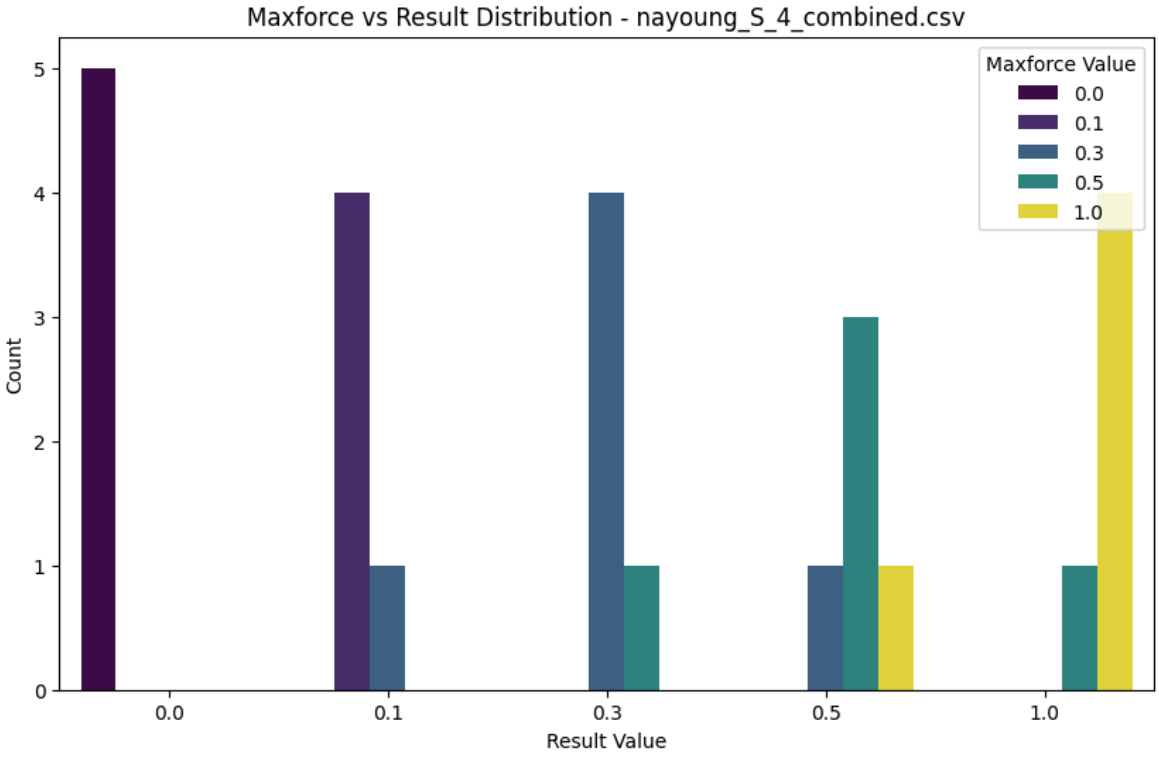

Maxforce vs Result Distribution - nayoung_l_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
Result Value
0.0
0.1
0.3
0.5
1.0

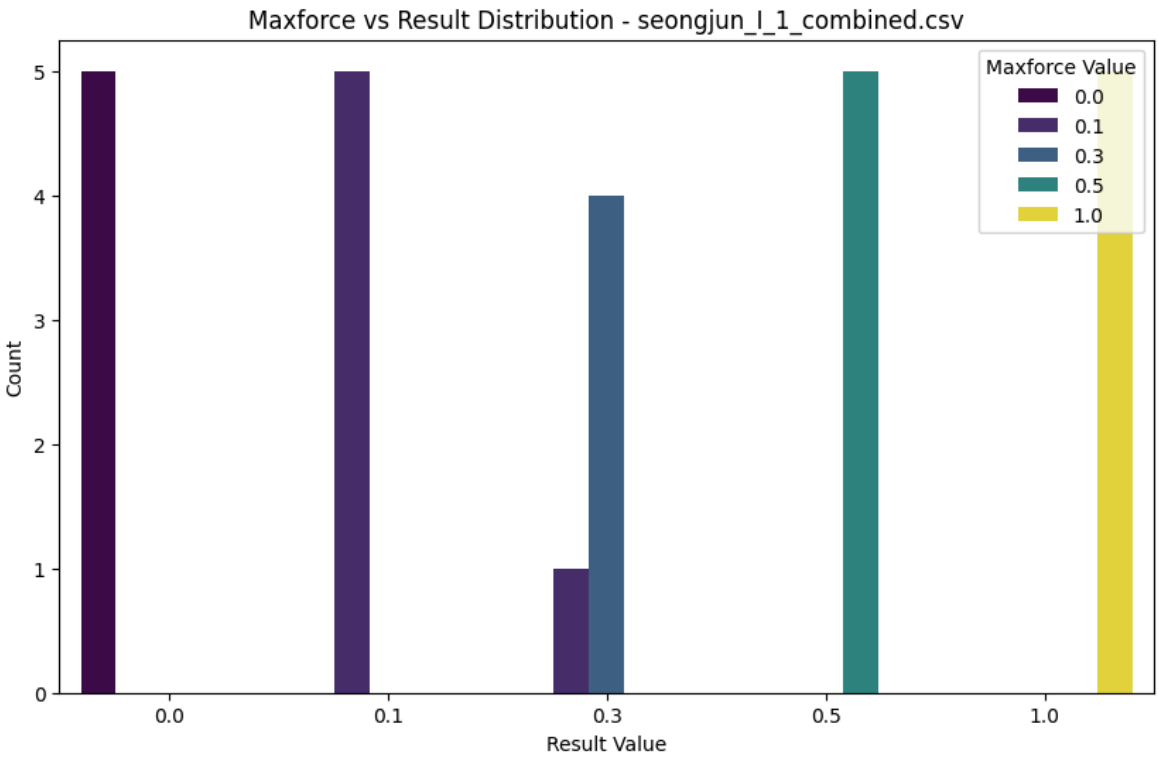

Maxforce vs Result Distribution - nayoung_l_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
Result Value
0.0
0.1
0.3
0.5
1.0

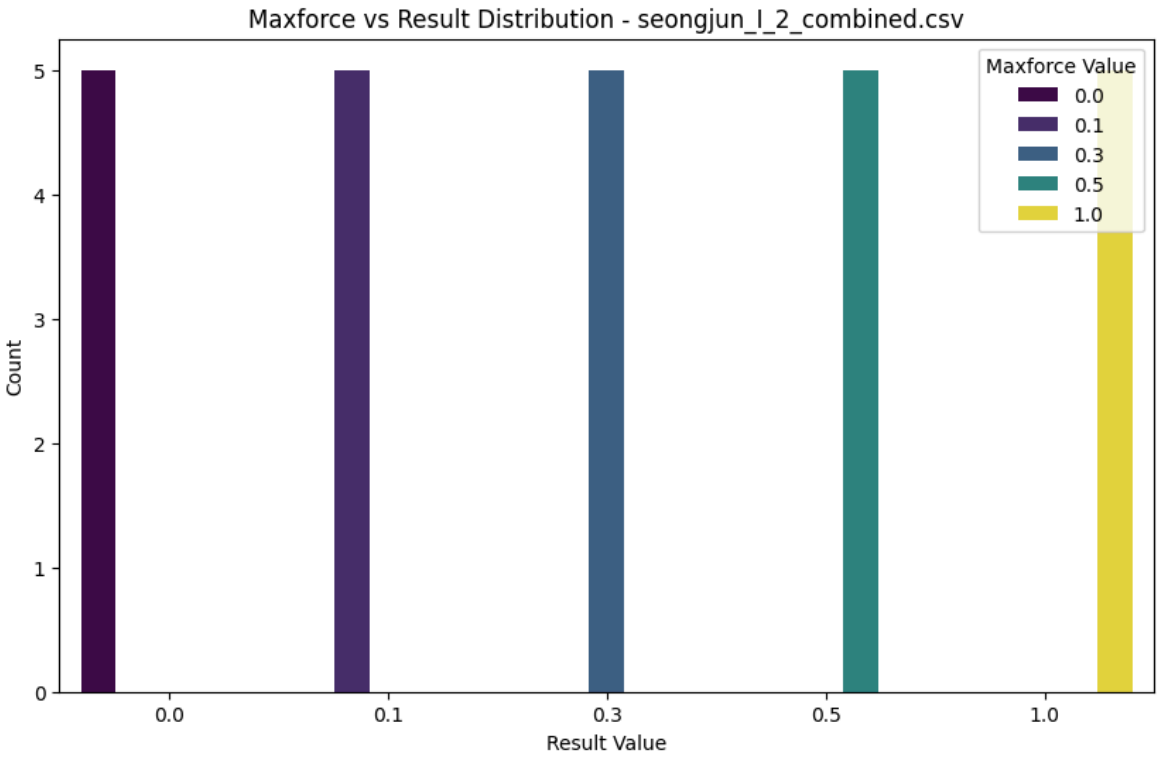

Maxforce vs Result Distribution - nayoung_S_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

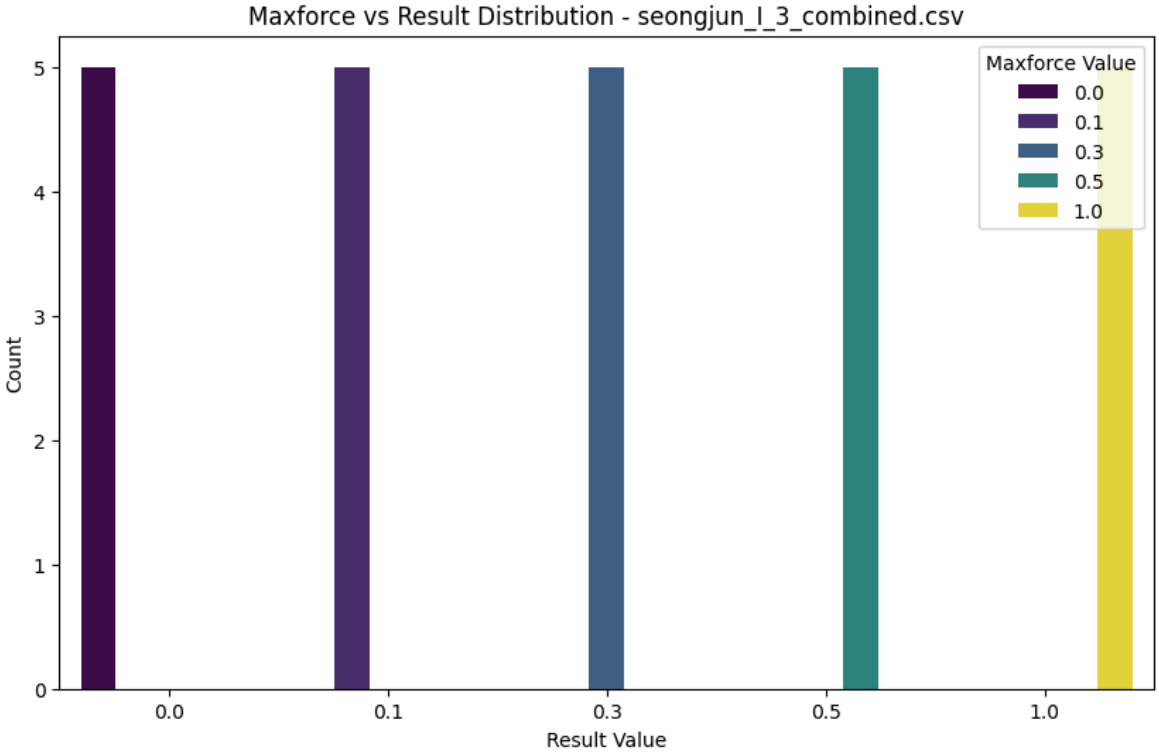

Maxforce vs Result Distribution - nayoung_S_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

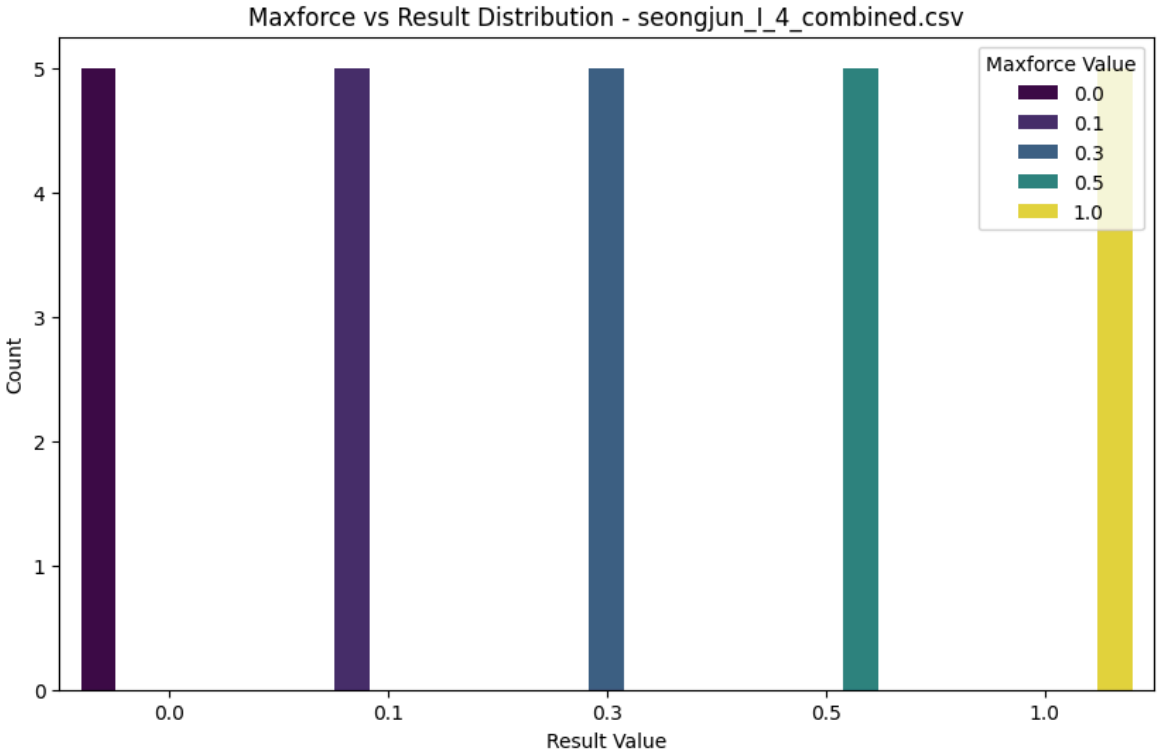

Maxforce vs Result Distribution - nayoung_S_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

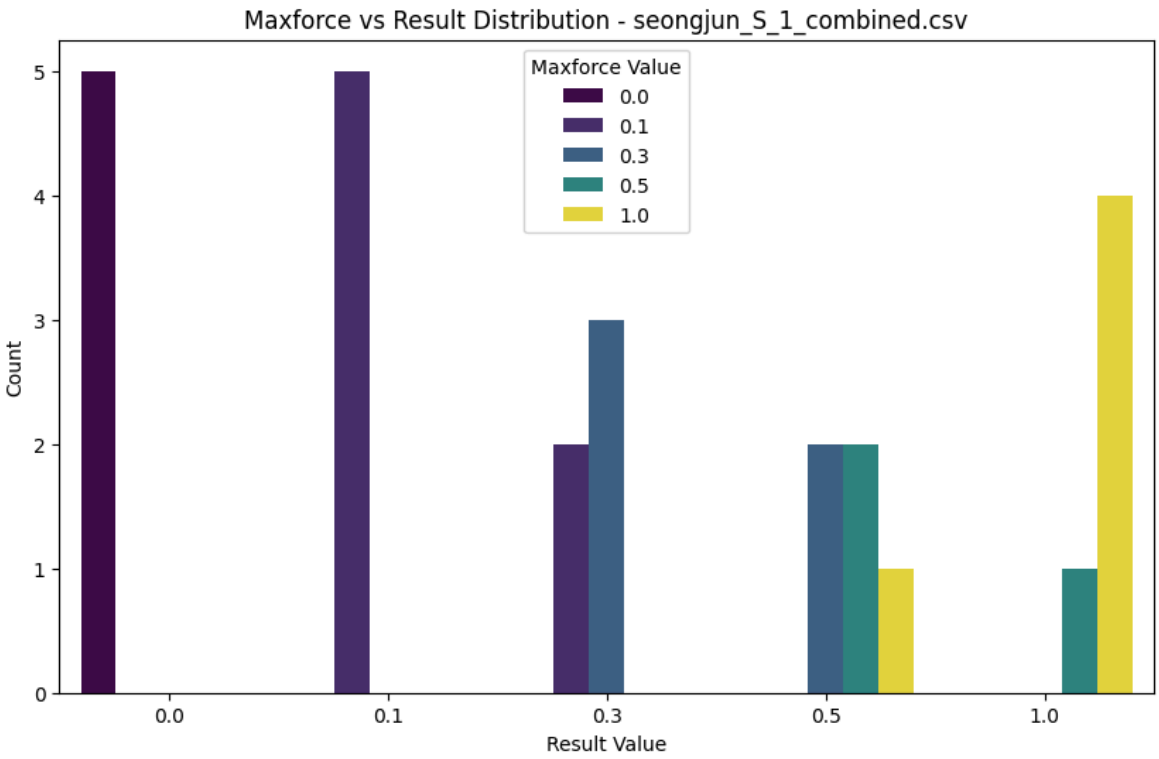

Maxforce vs Result Distribution - nayoung_S_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

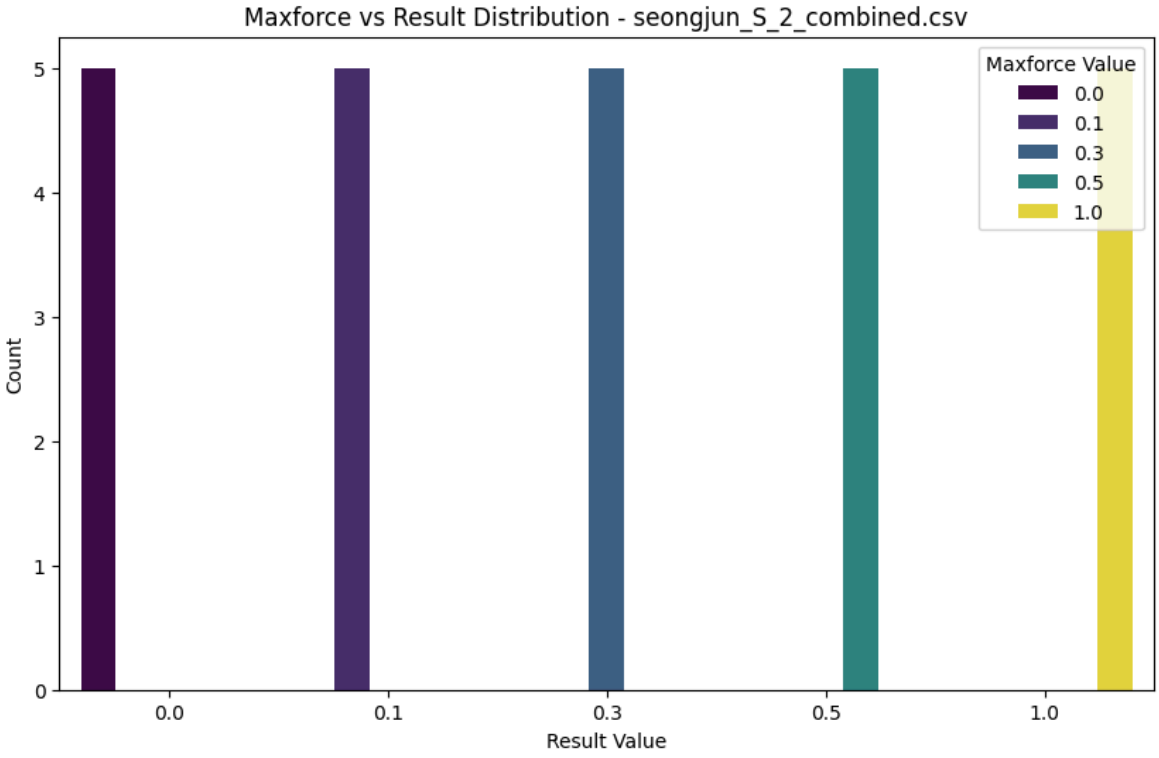

Maxforce vs Result Distribution - seongjun_I_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

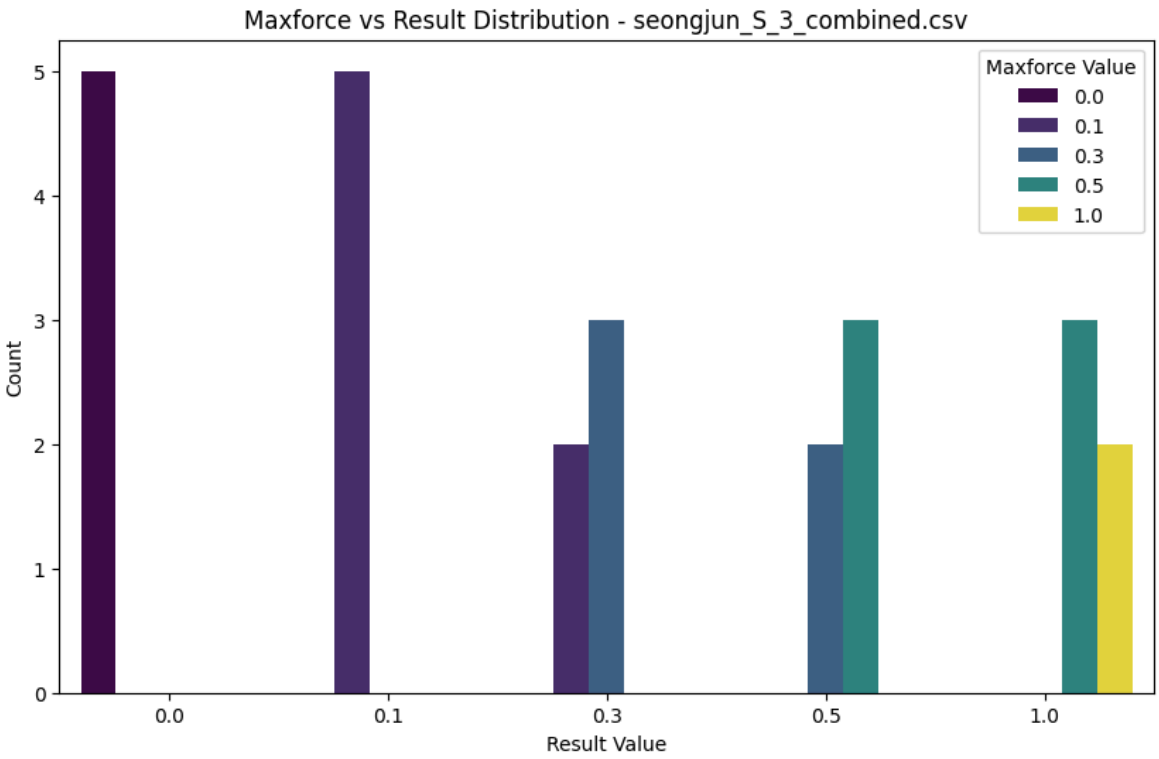

Maxforce vs Result Distribution - seongjun_I_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

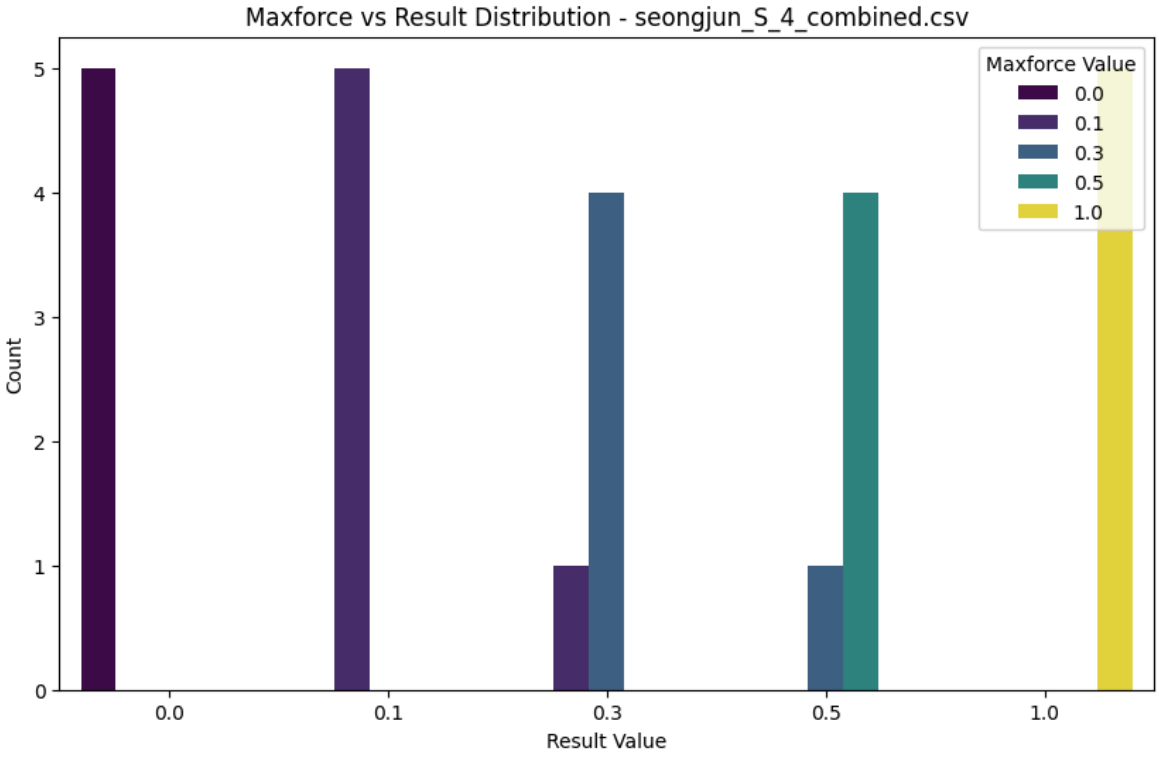

Maxforce vs Result Distribution - seongjun_l_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

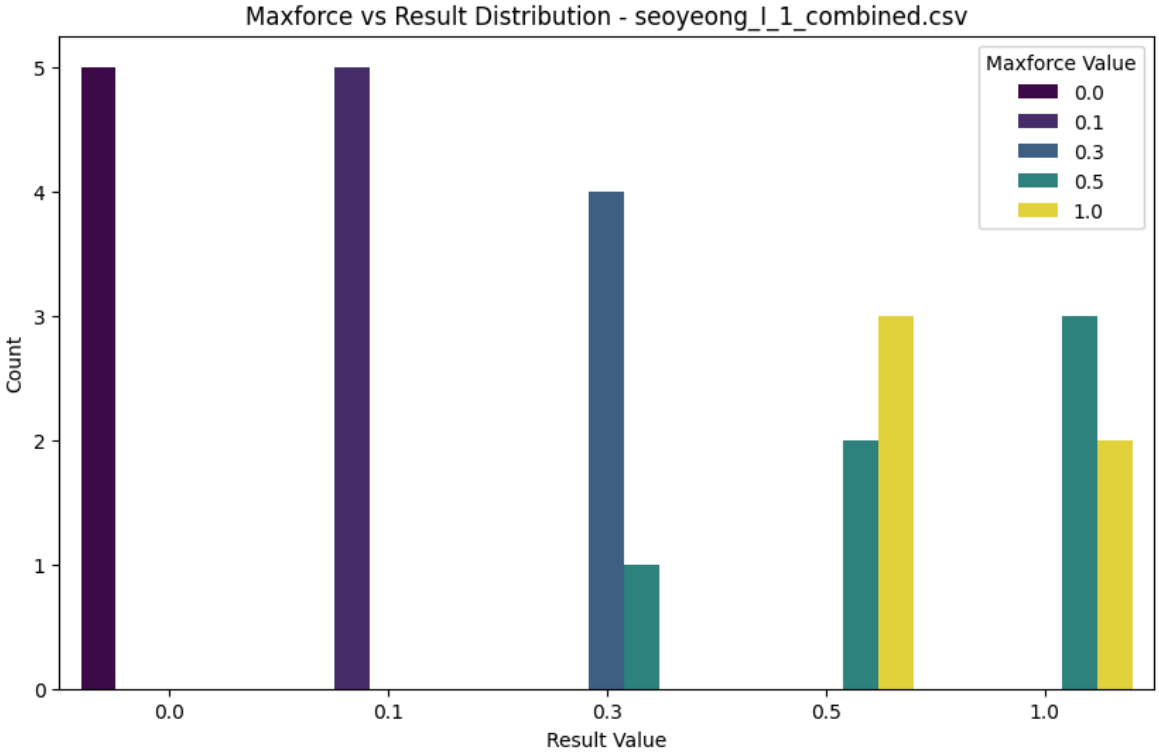

Maxforce vs Result Distribution - seongjun_l_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

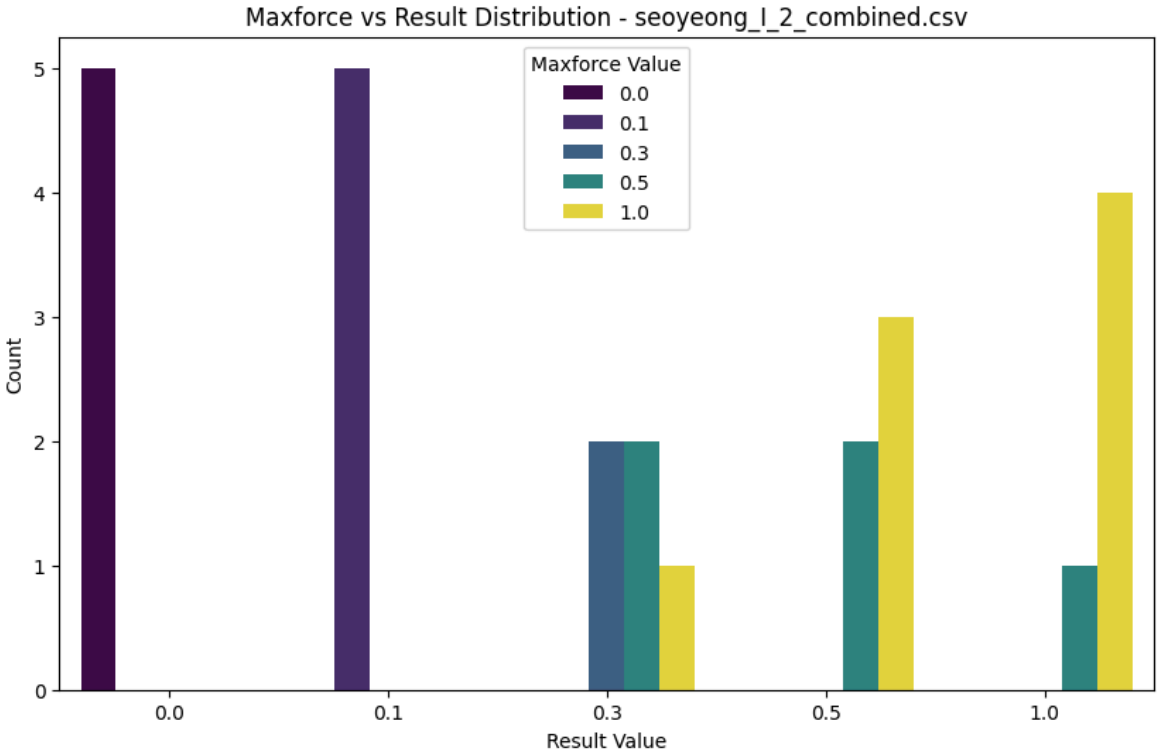

Maxforce vs Result Distribution - seongjun_S_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

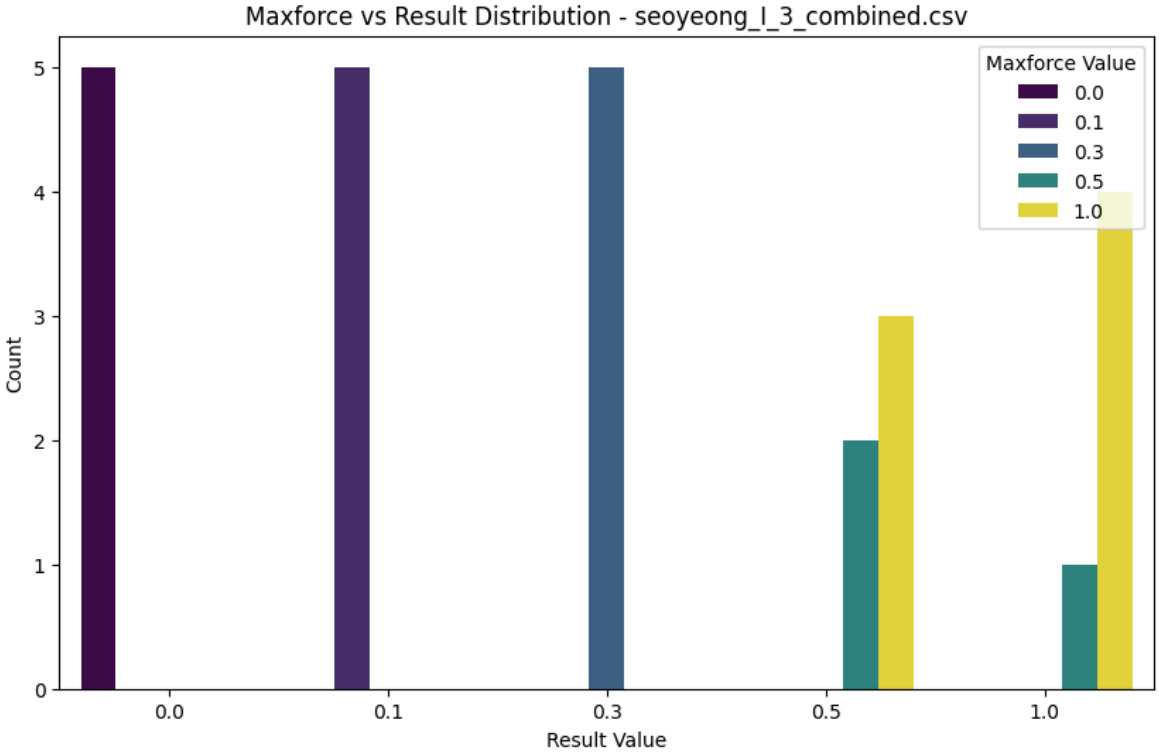

Maxforce vs Result Distribution - seongjun_S_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

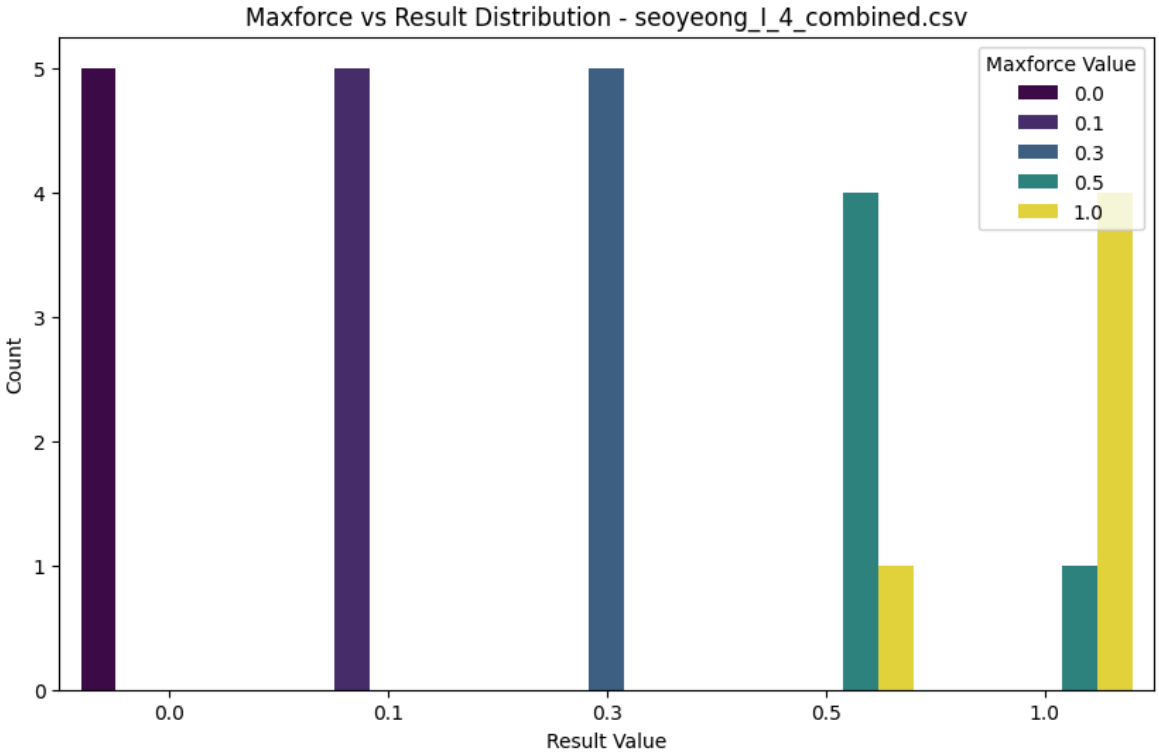

Maxforce vs Result Distribution - seongjun_S_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

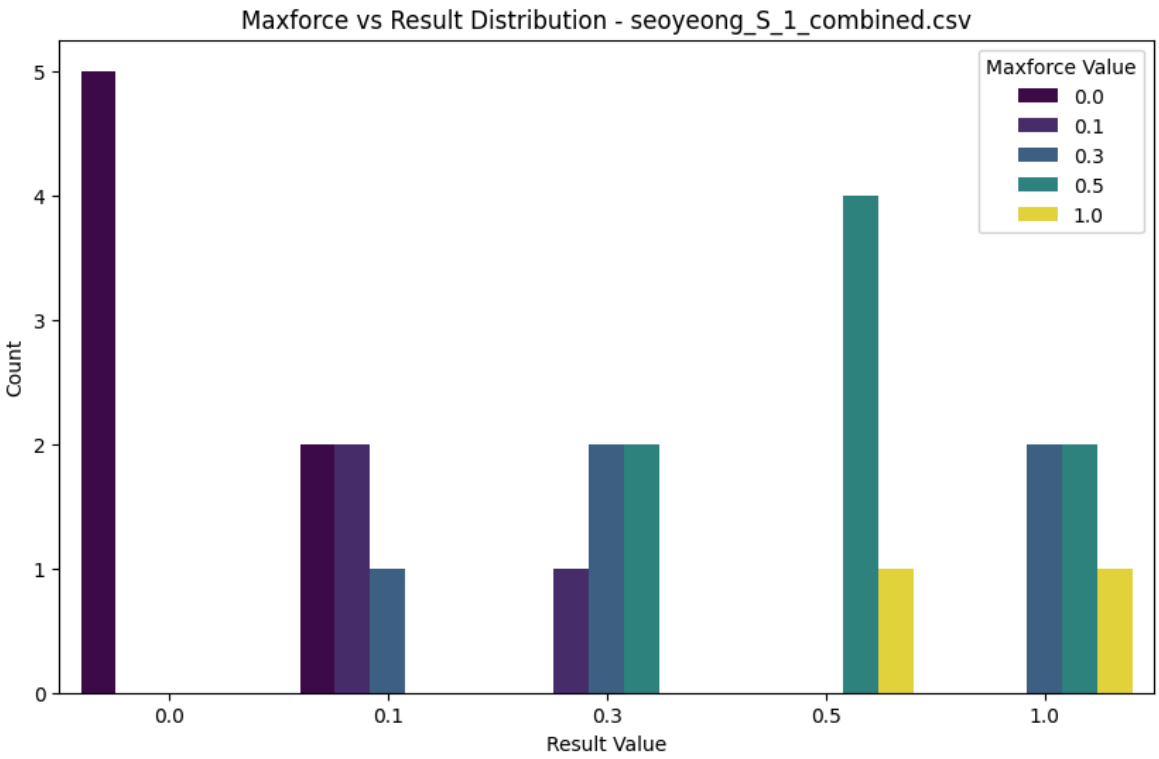

Maxforce vs Result Distribution - seongjun_S_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

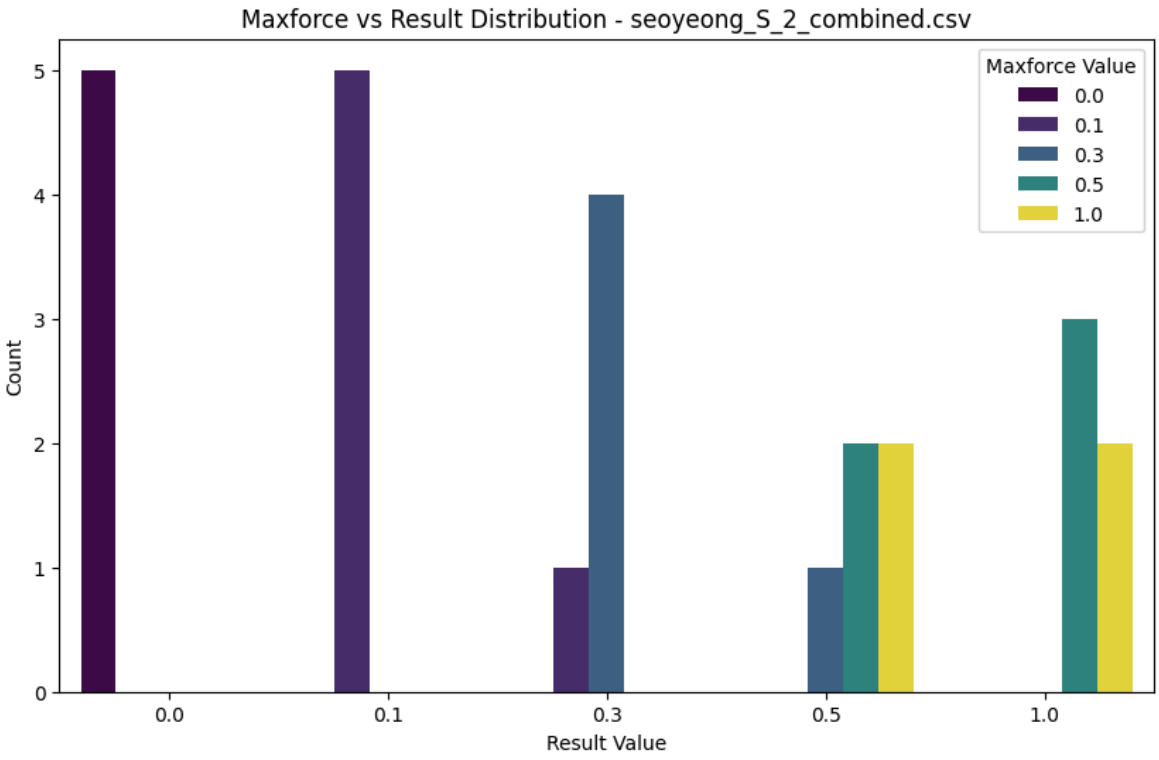

Maxforce vs Result Distribution - seoyeong_l_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

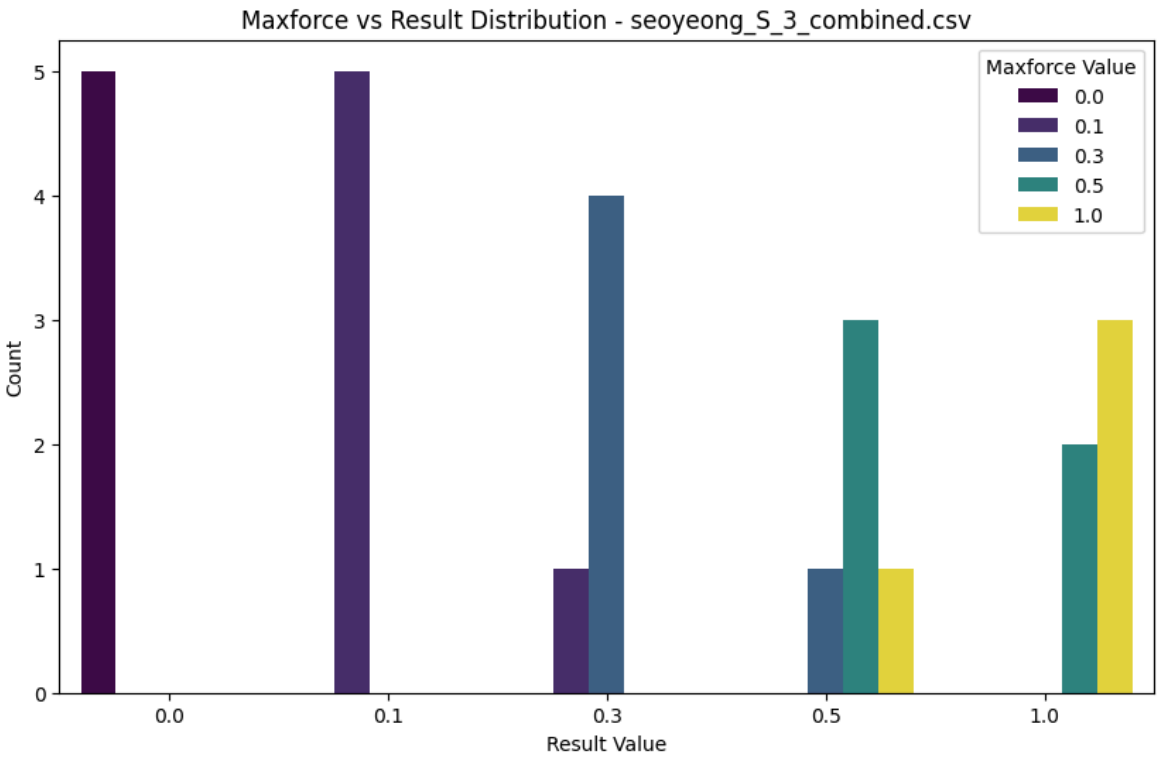

Maxforce vs Result Distribution - seoyeong_l_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

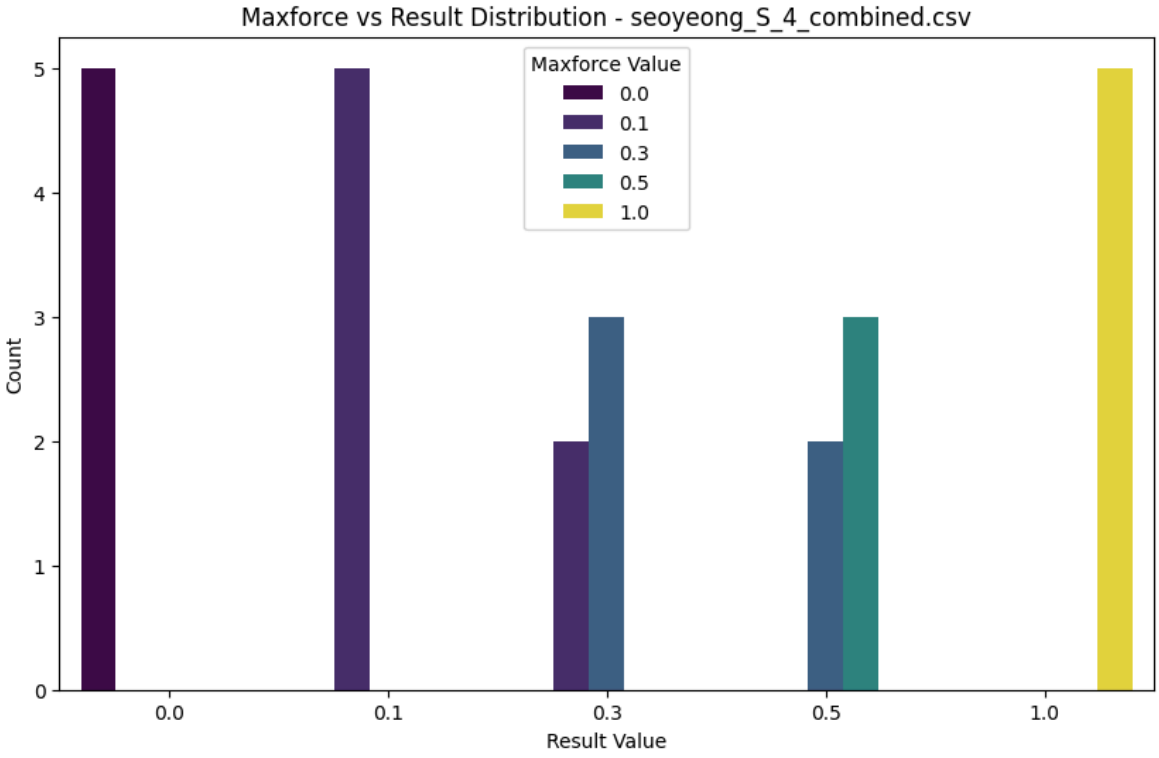

Maxforce vs Result Distribution - seoyeong_l_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

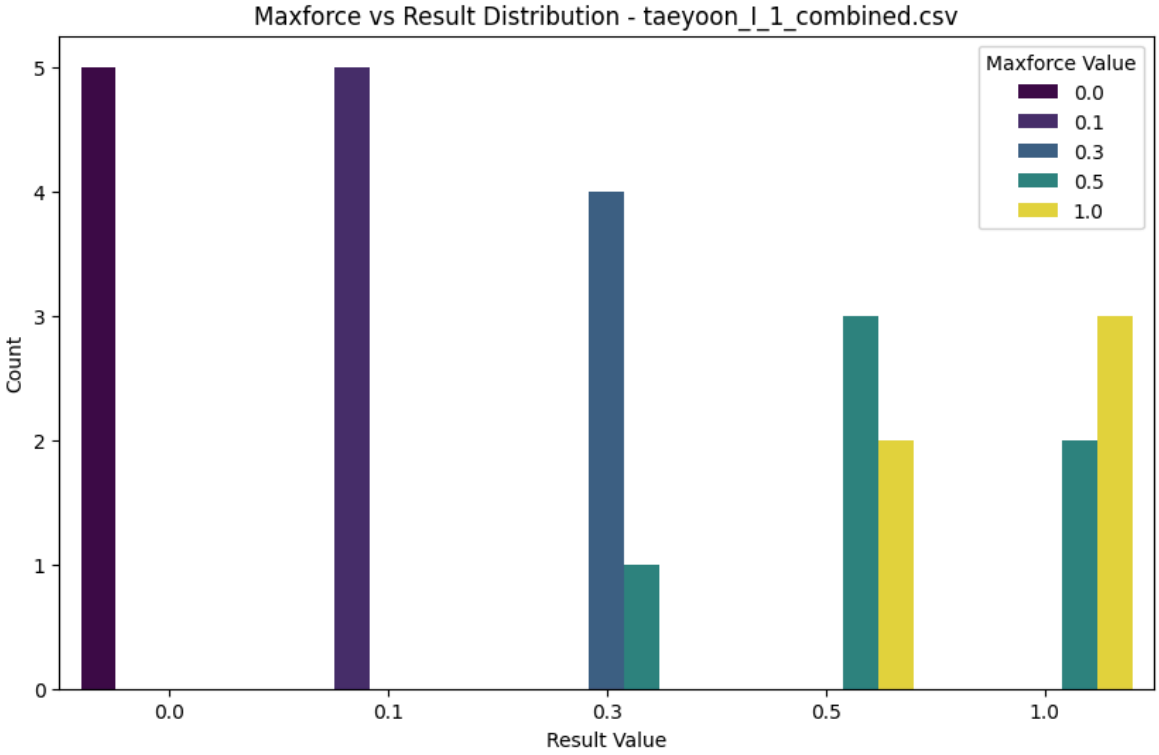

Maxforce vs Result Distribution - seoyeong_l_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

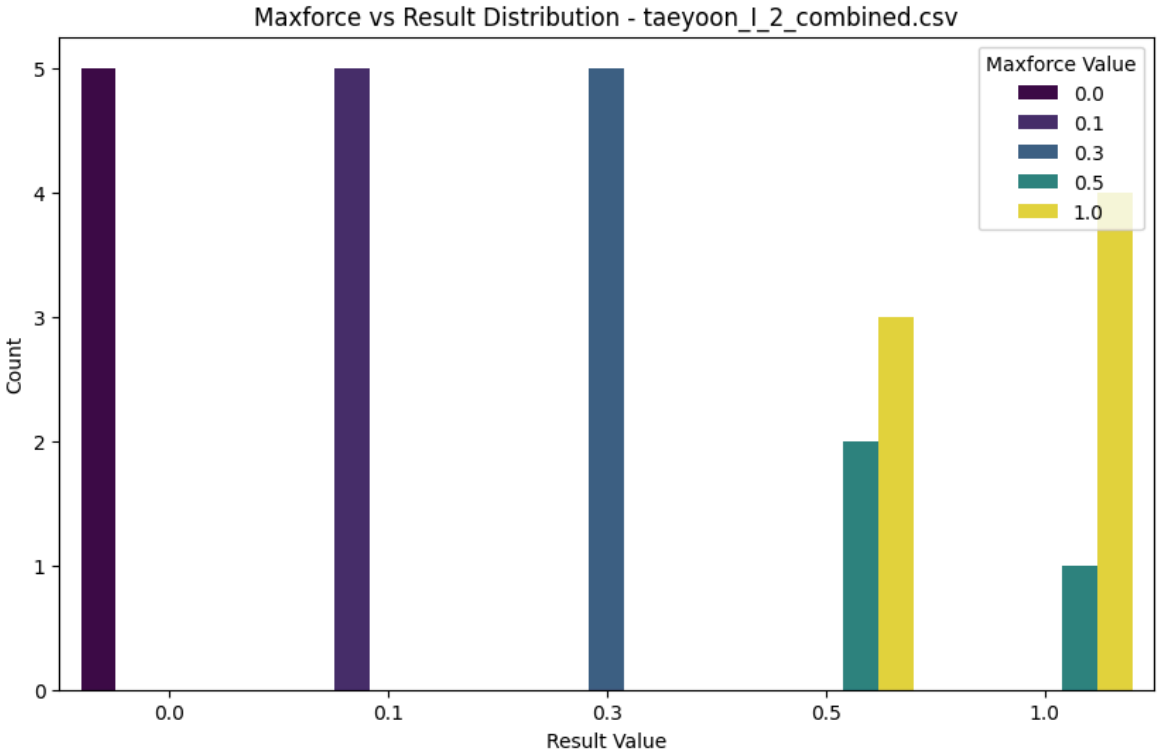

Maxforce vs Result Distribution - seoyeong_S_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

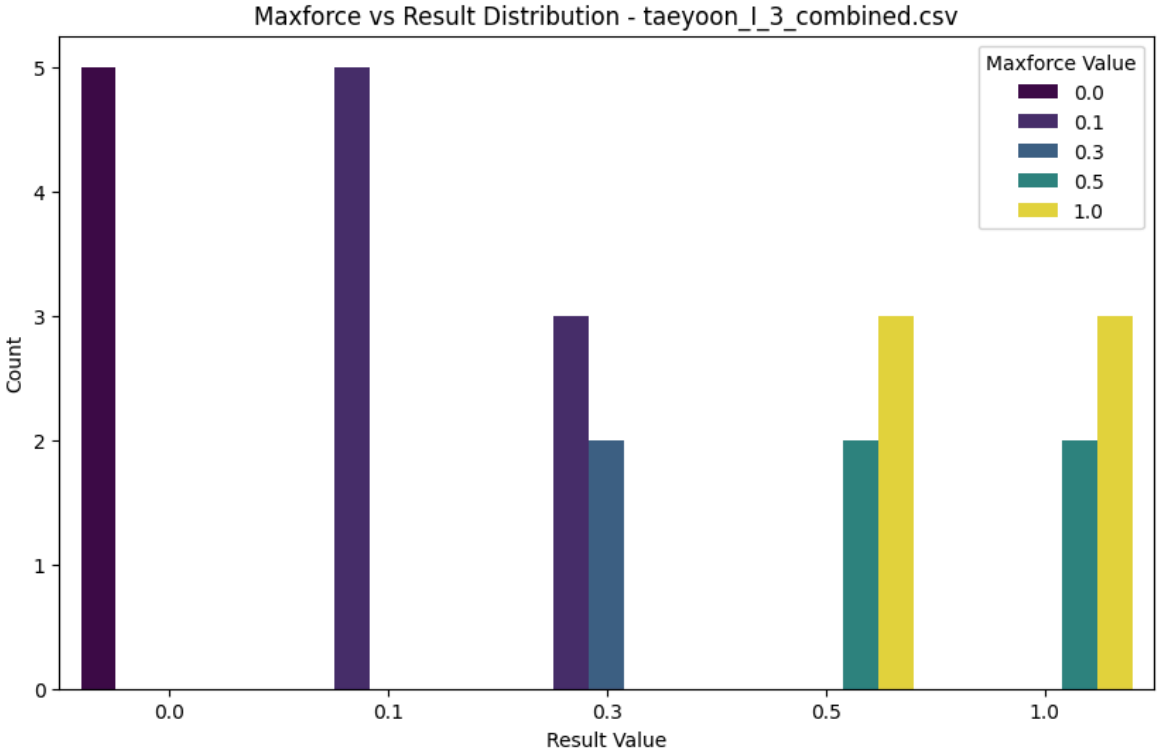

Maxforce vs Result Distribution - seoyeong_S_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

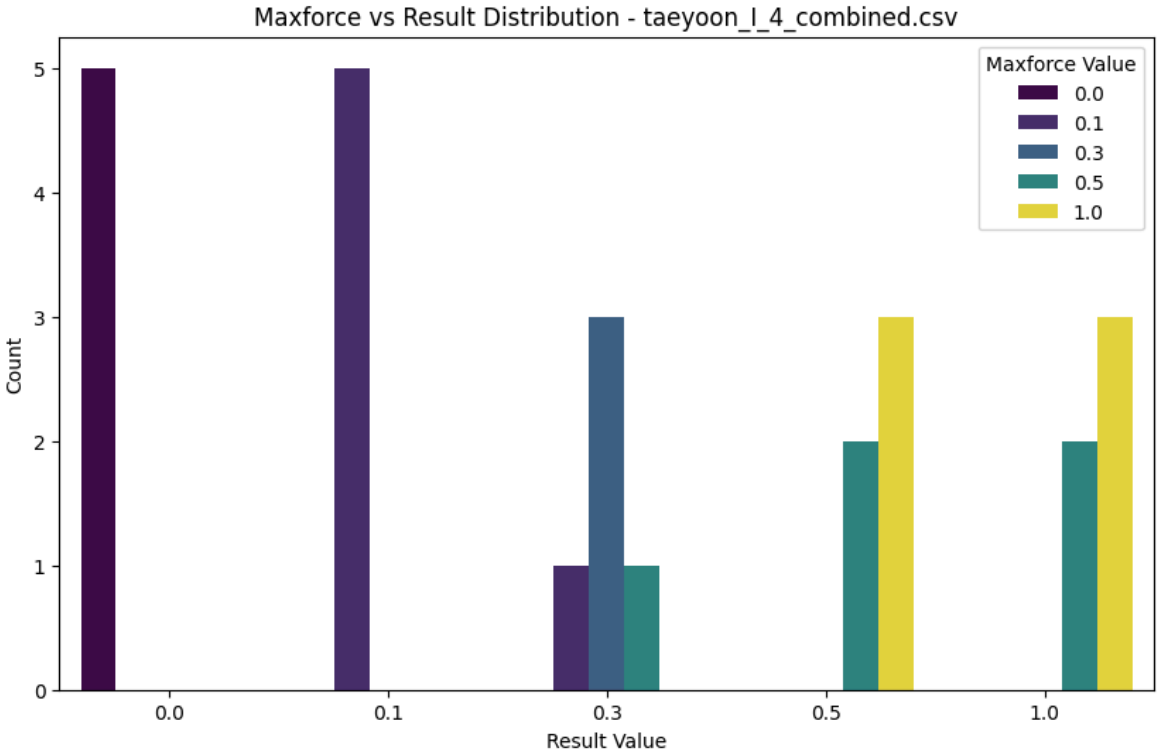

Maxforce vs Result Distribution - seoyeong_S_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

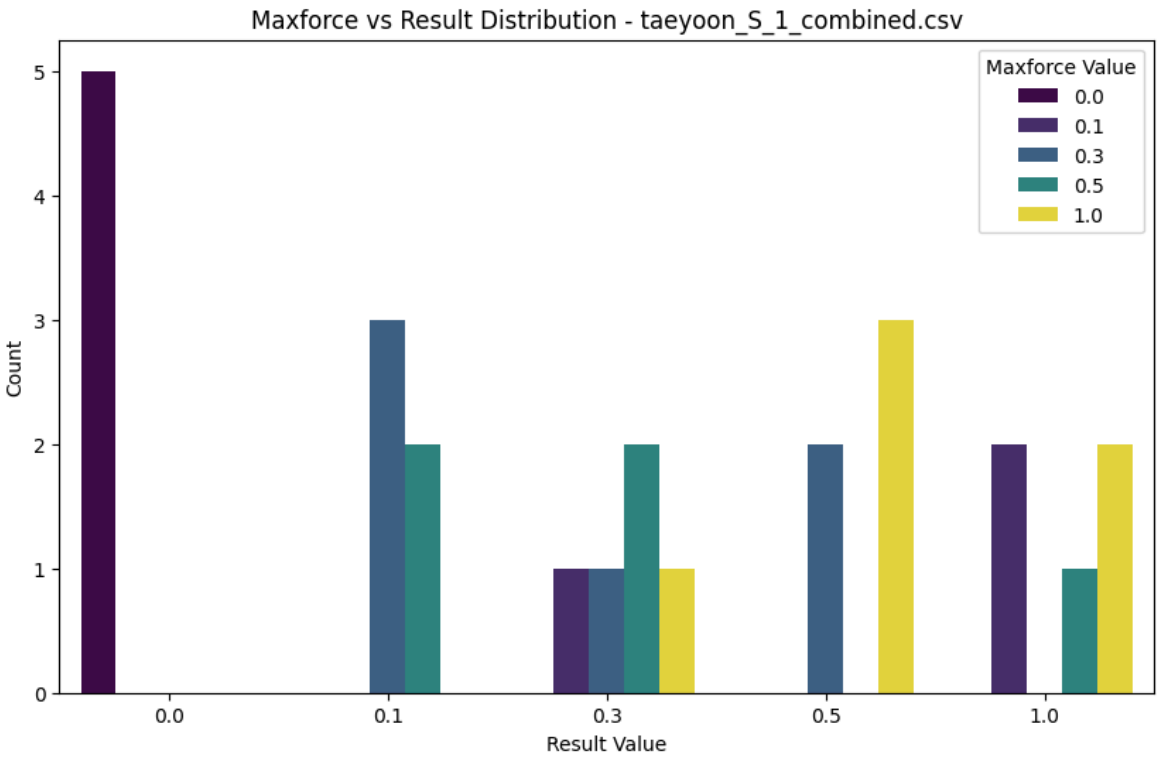

Maxforce vs Result Distribution - seoyeong_S_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

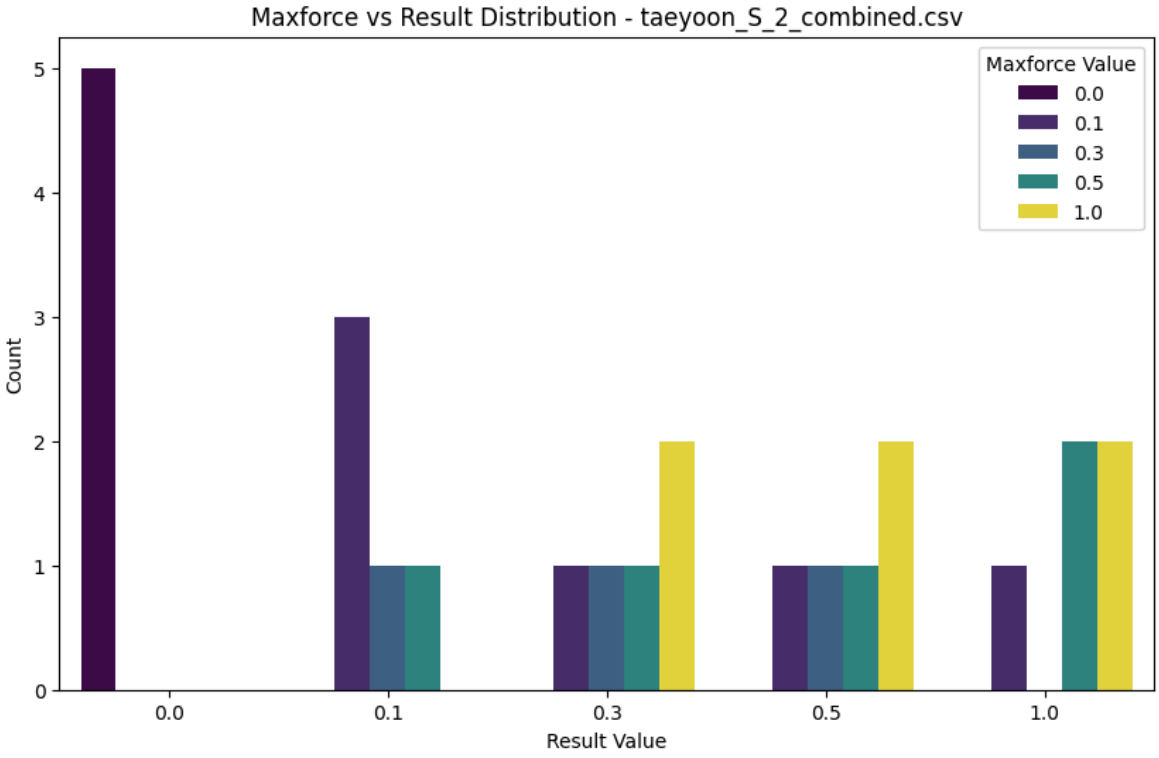

Maxforce vs Result Distribution - taeyoon_l_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

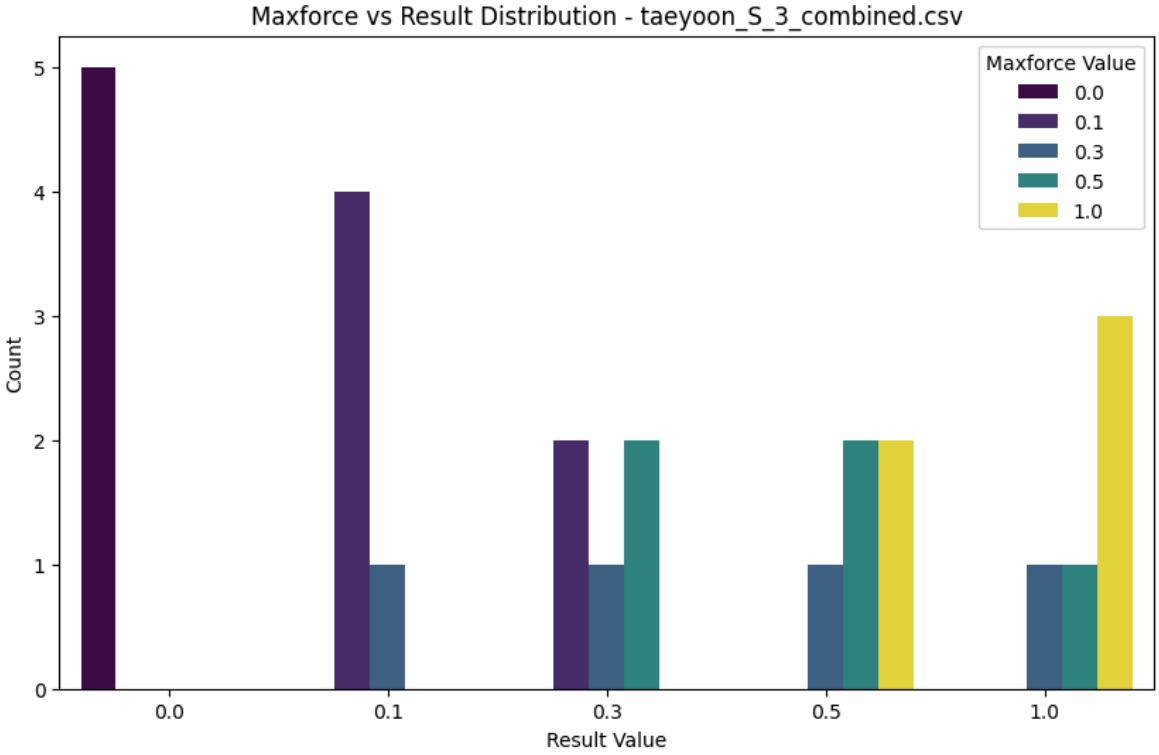

Maxforce vs Result Distribution - taeyoon_l_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value

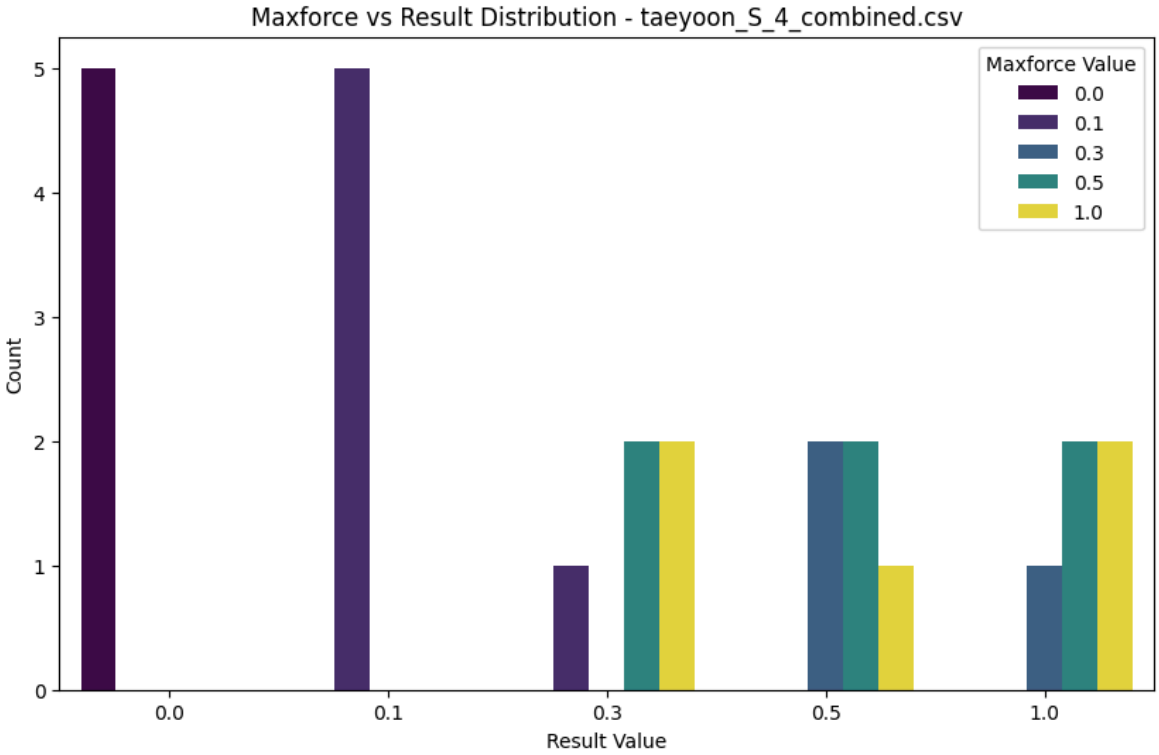

Maxforce vs Result Distribution - taeyoon_I_3_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value


Maxforce vs Result Distribution - taeyoon_I_4_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value


Maxforce vs Result Distribution - taeyoon_S_1_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value


Maxforce vs Result Distribution - taeyoon_S_2_combined.csv
Maxforce Value
0.0
0.1
0.3
0.5
1.0
Count
0
1
2
3
4
5
0.0
0.1
0.3
0.5
1.0
Result Value





In [36]:
```
pd.describe_option()
```

```
compute.use_bottleneck : bool
    Use the bottleneck library to accelerate if it is installed,
    the default is True
    Valid values: False,True
    [default: True] [currently: True]
compute.use_numba : bool
    Use the numba engine option for select operations if it is installed,
    the default is False
    Valid values: False,True
    [default: False] [currently: False]
compute.use_numexpr : bool
    Use the numexpr library to accelerate computation if it is installed,
    the default is True
    Valid values: False,True
    [default: True] [currently: True]
display.chop_threshold : float or None
    if set to a float value, all float values smaller than the given threshold
    will be displayed as exactly 0 by repr and friends.
    [default: None] [currently: None]
display.colheader_justify : 'left'/'right'
    Controls the justification of column headers. used by DataFrameFormatter.
    [default: right] [currently: right]
display.date_dayfirst : boolean
    When True, prints and parses dates with the day first, eg 20/01/2005
    [default: False] [currently: False]
display.date_yearfirst : boolean
    When True, prints and parses dates with the year first, eg 2005/01/20
    [default: False] [currently: False]
display.encoding : str/unicode
    Defaults to the detected encoding of the console.
    Specifies the encoding to be used for strings returned by to_string,
    these are generally strings meant to be displayed on the console.
    [default: UTF-8] [currently: UTF-8]
display.expand_frame_repr : boolean
    Whether to print out the full DataFrame repr for wide DataFrames across
    multiple lines, `max_columns` is still respected, but the output will
    wrap-around across multiple "pages" if its width exceeds `display.width`.
    [default: True] [currently: True]
display.float_format : callable
    The callable should accept a floating point number and return
    a string with the desired format of the number. This is used
    in some places like SeriesFormatter.
    See formats.format.EngFormatter for an example.
    [default: None] [currently: None]
display.html.border : int
    A ``border=value`` attribute is inserted in the ``<table>`` tag
    for the DataFrame HTML repr.
    [default: 1] [currently: 1]
display.html.table_schema : boolean
    Whether to publish a Table Schema representation for frontends
    that support it.
    (default: False)
    [default: False] [currently: False]
display.html.use_mathjax : boolean
    When True, Jupyter notebook will process table contents using MathJax,
    rendering mathematical expressions enclosed by the dollar symbol.
    (default: True)
    [default: True] [currently: True]
display.large_repr : 'truncate'/'info'
    For DataFrames exceeding max_rows/max_cols, the repr (and HTML repr) can
```

```
    show a truncated table, or switch to the view from
    df.info() (the behaviour in earlier versions of pandas).
    [default: truncate] [currently: truncate]
display.max_categories : int
    This sets the maximum number of categories pandas should output when
    printing out a `Categorical` or a Series of dtype "category".
    [default: 8] [currently: 8]
display.max_columns : int
    If max_cols is exceeded, switch to truncate view. Depending on
    `large_repr`, objects are either centrally truncated or printed as
    a summary view. 'None' value means unlimited.

    In case python/IPython is running in a terminal and `large_repr`
    equals 'truncate' this can be set to 0 or None and pandas will auto-detect
    the width of the terminal and print a truncated object which fits
    the screen width. The IPython notebook, IPython qtconsole, or IDLE
    do not run in a terminal and hence it is not possible to do
    correct auto-detection and defaults to 20.
    [default: 20] [currently: 100]
display.max_colwidth : int or None
    The maximum width in characters of a column in the repr of
    a pandas data structure. When the column overflows, a "..."
    placeholder is embedded in the output. A 'None' value means unlimited.
    [default: 50] [currently: 50]
display.max_dir_items : int
    The number of items that will be added to `dir(...)`. 'None' value means
    unlimited. Because dir is cached, changing this option will not immediately
    affect already existing dataframes until a column is deleted or added.

    This is for instance used to suggest columns from a dataframe to tab
    completion.
    [default: 100] [currently: 100]
display.max_info_columns : int
    max_info_columns is used in DataFrame.info method to decide if
    per column information will be printed.
    [default: 100] [currently: 100]
display.max_info_rows : int
    df.info() will usually show null-counts for each column.
    For large frames this can be quite slow. max_info_rows and max_info_cols
    limit this null check only to frames with smaller dimensions than
    specified.
    [default: 1690785] [currently: 1690785]
display.max_rows : int
    If max_rows is exceeded, switch to truncate view. Depending on
    `large_repr`, objects are either centrally truncated or printed as
    a summary view. 'None' value means unlimited.

    In case python/IPython is running in a terminal and `large_repr`
    equals 'truncate' this can be set to 0 and pandas will auto-detect
    the height of the terminal and print a truncated object which fits
    the screen height. The IPython notebook, IPython qtconsole, or
    IDLE do not run in a terminal and hence it is not possible to do
    correct auto-detection.
    [default: 60] [currently: None]
display.max_seq_items : int or None
    When pretty-printing a long sequence, no more then `max_seq_items`
    will be printed. If items are omitted, they will be denoted by the
    addition of "..." to the resulting string.

    If set to None, the number of items to be printed is unlimited.
```

```
    [default: 100] [currently: 100]
display.memory_usage : bool, string or None
    This specifies if the memory usage of a DataFrame should be displayed when
    df.info() is called. Valid values True,False,'deep'
    [default: True] [currently: True]
display.min_rows : int
    The numbers of rows to show in a truncated view (when `max_rows` is
    exceeded). Ignored when `max_rows` is set to None or 0. When set to
    None, follows the value of `max_rows`.
    [default: 10] [currently: 10]
display.multi_sparse : boolean
    "sparsify" MultiIndex display (don't display repeated
    elements in outer levels within groups)
    [default: True] [currently: True]
display.notebook_repr_html : boolean
    When True, IPython notebook will use html representation for
    pandas objects (if it is available).
    [default: True] [currently: True]
display.pprint_nest_depth : int
    Controls the number of nested levels to process when pretty-printing
    [default: 3] [currently: 3]
display.precision : int
    Floating point output precision in terms of number of places after the
    decimal, for regular formatting as well as scientific notation. Similar
    to ``precision`` in :meth:`numpy.set_printoptions`.
    [default: 6] [currently: 6]
display.show_dimensions : boolean or 'truncate'
    Whether to print out dimensions at the end of DataFrame repr.
    If 'truncate' is specified, only print out the dimensions if the
    frame is truncated (e.g. not display all rows and/or columns)
    [default: truncate] [currently: truncate]
display.unicode.ambiguous_as_wide : boolean
    Whether to use the Unicode East Asian Width to calculate the display text
    width.
    Enabling this may affect to the performance (default: False)
    [default: False] [currently: False]
display.unicode.east_asian_width : boolean
    Whether to use the Unicode East Asian Width to calculate the display text
    width.
    Enabling this may affect to the performance (default: False)
    [default: False] [currently: False]
display.width : int
    Width of the display in characters. In case python/IPython is running in
    a terminal this can be set to None and pandas will correctly auto-detect
    the width.
    Note that the IPython notebook, IPython qtconsole, or IDLE do not run in a
    terminal and hence it is not possible to correctly detect the width.
    [default: 80] [currently: 150]
future.infer_string Whether to infer sequence of str objects as pyarrow string dt
ype, which will be the default in pandas 3.0 (at which point this option will be
deprecated).
    [default: False] [currently: False]
future.no_silent_downcasting Whether to opt-in to the future behavior which will
*not* silently downcast results from Series and DataFrame `where`, `mask`, and `c
lip` methods. Silent downcasting will be removed in pandas 3.0 (at which point th
is option will be deprecated).
    [default: False] [currently: False]
io.excel.ods.reader : string
    The default Excel reader engine for 'ods' files. Available options:
    auto, odf, calamine.
```

```
    [default: auto] [currently: auto]
io.excel.ods.writer : string
    The default Excel writer engine for 'ods' files. Available options:
    auto, odf.
    [default: auto] [currently: auto]
io.excel.xls.reader : string
    The default Excel reader engine for 'xls' files. Available options:
    auto, xlrd, calamine.
    [default: auto] [currently: auto]
io.excel.xlsb.reader : string
    The default Excel reader engine for 'xlsb' files. Available options:
    auto, pyxlsb, calamine.
    [default: auto] [currently: auto]
io.excel.xlsm.reader : string
    The default Excel reader engine for 'xlsm' files. Available options:
    auto, xlrd, openpyxl, calamine.
    [default: auto] [currently: auto]
io.excel.xlsm.writer : string
    The default Excel writer engine for 'xlsm' files. Available options:
    auto, openpyxl.
    [default: auto] [currently: auto]
io.excel.xlsx.reader : string
    The default Excel reader engine for 'xlsx' files. Available options:
    auto, xlrd, openpyxl, calamine.
    [default: auto] [currently: auto]
io.excel.xlsx.writer : string
    The default Excel writer engine for 'xlsx' files. Available options:
    auto, openpyxl, xlsxwriter.
    [default: auto] [currently: auto]
io.hdf.default_format : format
    default format writing format, if None, then
    put will default to 'fixed' and append will default to 'table'
    [default: None] [currently: None]
io.hdf.dropna_table : boolean
    drop ALL nan rows when appending to a table
    [default: False] [currently: False]
io.parquet.engine : string
    The default parquet reader/writer engine. Available options:
    'auto', 'pyarrow', 'fastparquet', the default is 'auto'
    [default: auto] [currently: auto]
io.sql.engine : string
    The default sql reader/writer engine. Available options:
    'auto', 'sqlalchemy', the default is 'auto'
    [default: auto] [currently: auto]
mode.chained_assignment : string
    Raise an exception, warn, or no action if trying to use chained assignment,
    The default is warn
    [default: warn] [currently: warn]
mode.copy_on_write : bool
    Use new copy-view behaviour using Copy-on-Write. Defaults to False,
    unless overridden by the 'PANDAS_COPY_ON_WRITE' environment variable
    (if set to "1" for True, needs to be set before pandas is imported).
    [default: False] [currently: False]
mode.data_manager : string
    Internal data manager type; can be "block" or "array". Defaults to "block",
    unless overridden by the 'PANDAS_DATA_MANAGER' environment variable (needs
    to be set before pandas is imported).
    [default: block] [currently: block]
    (Deprecated, use `` instead.)
mode.sim_interactive : boolean
```

```
    Whether to simulate interactive mode for purposes of testing
    [default: False] [currently: False]
mode.string_storage : string
    The default storage for StringDtype. This option is ignored if
    ``future.infer_string`` is set to True.
    [default: python] [currently: python]
mode.use_inf_as_na : boolean
    True means treat None, NaN, INF, -INF as NA (old way),
    False means None and NaN are null, but INF, -INF are not NA
    (new way).

    This option is deprecated in pandas 2.1.0 and will be removed in 3.0.
    [default: False] [currently: False]
    (Deprecated, use `` instead.)
plotting.backend : str
    The plotting backend to use. The default value is "matplotlib", the
    backend provided with pandas. Other backends can be specified by
    providing the name of the module that implements the backend.
    [default: matplotlib] [currently: matplotlib]
plotting.matplotlib.register_converters : bool or 'auto'.
    Whether to register converters with matplotlib's units registry for
    dates, times, datetimes, and Periods. Toggling to False will remove
    the converters, restoring any converters that pandas overwrote.
    [default: auto] [currently: auto]
styler.format.decimal : str
    The character representation for the decimal separator for floats and comple
x.
    [default: .] [currently: .]
styler.format.escape : str, optional
    Whether to escape certain characters according to the given context; html or
latex.
    [default: None] [currently: None]
styler.format.formatter : str, callable, dict, optional
    A formatter object to be used as default within ``Styler.format``.
    [default: None] [currently: None]
styler.format.na_rep : str, optional
    The string representation for values identified as missing.
    [default: None] [currently: None]
styler.format.precision : int
    The precision for floats and complex numbers.
    [default: 6] [currently: 6]
styler.format.thousands : str, optional
    The character representation for thousands separator for floats, int and comp
lex.
    [default: None] [currently: None]
styler.html.mathjax : bool
    If False will render special CSS classes to table attributes that indicate Ma
thjax
    will not be used in Jupyter Notebook.
    [default: True] [currently: True]
styler.latex.environment : str
    The environment to replace ``\begin{table}``. If "longtable" is used results
    in a specific longtable environment format.
    [default: None] [currently: None]
styler.latex.hrules : bool
    Whether to add horizontal rules on top and bottom and below the headers.
    [default: False] [currently: False]
styler.latex.multicol_align : {"r", "c", "l", "naive-l", "naive-r"}
    The specifier for horizontal alignment of sparsified LaTeX multicolumns. Pipe
    decorators can also be added to non-naive values to draw vertical
```

```
    rules, e.g. "\|r" will draw a rule on the left side of right aligned merged c
ells.
    [default: r] [currently: r]
styler.latex.multirow_align : {"c", "t", "b"}
    The specifier for vertical alignment of sparsified LaTeX multirows.
    [default: c] [currently: c]
styler.render.encoding : str
    The encoding used for output HTML and LaTeX files.
    [default: utf-8] [currently: utf-8]
styler.render.max_columns : int, optional
    The maximum number of columns that will be rendered. May still be reduced to
    satisfy ``max_elements``, which takes precedence.
    [default: None] [currently: None]
styler.render.max_elements : int
    The maximum number of data-cell (<td>) elements that will be rendered before
    trimming will occur over columns, rows or both if needed.
    [default: 262144] [currently: 262144]
styler.render.max_rows : int, optional
    The maximum number of rows that will be rendered. May still be reduced to
    satisfy ``max_elements``, which takes precedence.
    [default: None] [currently: None]
styler.render.repr : str
    Determine which output to use in Jupyter Notebook in {"html", "latex"}.
    [default: html] [currently: html]
styler.sparse.columns : bool
    Whether to sparsify the display of hierarchical columns. Setting to False wil
l
    display each explicit level element in a hierarchical key for each column.
    [default: True] [currently: True]
styler.sparse.index : bool
    Whether to sparsify the display of a hierarchical index. Setting to False wil
l
    display each explicit level element in a hierarchical key for each row.
    [default: True] [currently: True]
```

In [37]:
```
pd.set_option('display.max_columns', 100)
```

In [38]:
```
pd.set_option('display.width', 150)
```

In [39]:
```
pd.set_option('display.max_rows', None)
```

🔗 세션별 정확도(I-Impedence Model, S-Siffness shifting algorithm)

In [40]:
```
# 결과를 저장할 리스트 초기화
comparison_results = []

# 각 CSV 파일에 대해 작업 수행
for csv_file in csv_files:
    file_path = os.path.join(directory_path, csv_file)

    # CSV 파일 읽기
    df = pd.read_csv(file_path)

    # maxforce와 result 열이 있는지 확인
    if 'maxforce' in df.columns and 'result' in df.columns:
        # 상관계수 계산
        correlation = df['maxforce'].corr(df['result'])

        # 차이의 절대값 평균 계산
```

```
        mean_absolute_difference = (df['maxforce'] - df['result']).abs().mean()

        # maxforce와 result가 동일한 값의 비율 계산
        identical_percentage = (df['maxforce'] == df['result']).mean() * 100

        # 결과 저장
        comparison_results.append({
            'filename': csv_file,
            'correlation': correlation,
            'mean_absolute_difference': mean_absolute_difference,
            'identical_percentage': identical_percentage
        })

        # # 그래프 생성
        # plt.figure(figsize=(10, 6))

        # # 산점도 (scatter plot)
        # plt.scatter(df.index, df['maxforce'], label='Maxforce', color='blue',
        # plt.scatter(df.index, df['result'], label='Result', color='orange', al

        # # 선 그래프 (line plot)
        # plt.plot(df.index, df['maxforce'], color='blue', alpha=0.3)
        # plt.plot(df.index, df['result'], color='orange', alpha=0.3)

        # # 제목 및 라벨 추가
        # plt.title(f'Result vs Maxforce - {csv_file}')
        # plt.xlabel('Index')
        # plt.ylabel('Values')
        # plt.legend()

    else:
        print(f"'maxforce' or 'result' column missing in {csv_file}")

# 결과를 데이터프레임으로 변환
results_df = pd.DataFrame(comparison_results)

# 결과 출력
print(results_df)
```

	filename	correlation	mean_absolute_difference	identical_percentage
0	dohoon_I_1_combined.csv	0.993892	0.008	96.0
1	dohoon_I_2_combined.csv	1.000000	0.000	100.0
2	dohoon_I_3_combined.csv	0.993877	0.008	96.0
3	dohoon_I_4_combined.csv	1.000000	0.000	100.0
4	dohoon_S_1_combined.csv	0.994014	0.008	96.0
5	dohoon_S_2_combined.csv	0.960668	0.028	92.0
6	dohoon_S_3_combined.csv	0.993892	0.008	96.0
7	dohoon_S_4_combined.csv	0.965275	0.020	96.0
8	jaeho_I_1_combined.csv	0.993883	0.008	96.0
9	jaeho_I_2_combined.csv	0.993877	0.008	96.0
10	jaeho_I_3_combined.csv	0.901517	0.088	80.0
11	jaeho_I_4_combined.csv	0.918953	0.088	68.0
12	jaeho_S_1_combined.csv	0.843943	0.104	68.0
13	jaeho_S_2_combined.csv	1.000000	0.000	100.0
14	jaeho_S_3_combined.csv	0.987500	0.016	92.0
15	jaeho_S_4_combined.csv	0.920785	0.072	76.0
16	jaewan_I_1_combined.csv	0.907143	0.064	80.0
17	jaewan_I_2_combined.csv	0.746634	0.136	68.0
18	jaewan_I_3_combined.csv	0.656684	0.184	56.0
19	jaewan_I_4_combined.csv	0.796246	0.112	68.0
20	jaewan_S_1_combined.csv	0.946608	0.036	88.0
21	jaewan_S_2_combined.csv	0.983258	0.024	88.0
22	jaewan_S_3_combined.csv	0.981416	0.024	88.0
23	jaewan_S_4_combined.csv	0.974279	0.040	80.0
24	jiyoung_I_1_combined.csv	0.994014	0.008	96.0
25	jiyoung_I_2_combined.csv	0.935393	0.048	88.0
26	jiyoung_I_3_combined.csv	0.930587	0.056	84.0
27	jiyoung_I_4_combined.csv	0.993877	0.008	96.0
28	jiyoung_S_1_combined.csv	0.929386	0.068	72.0

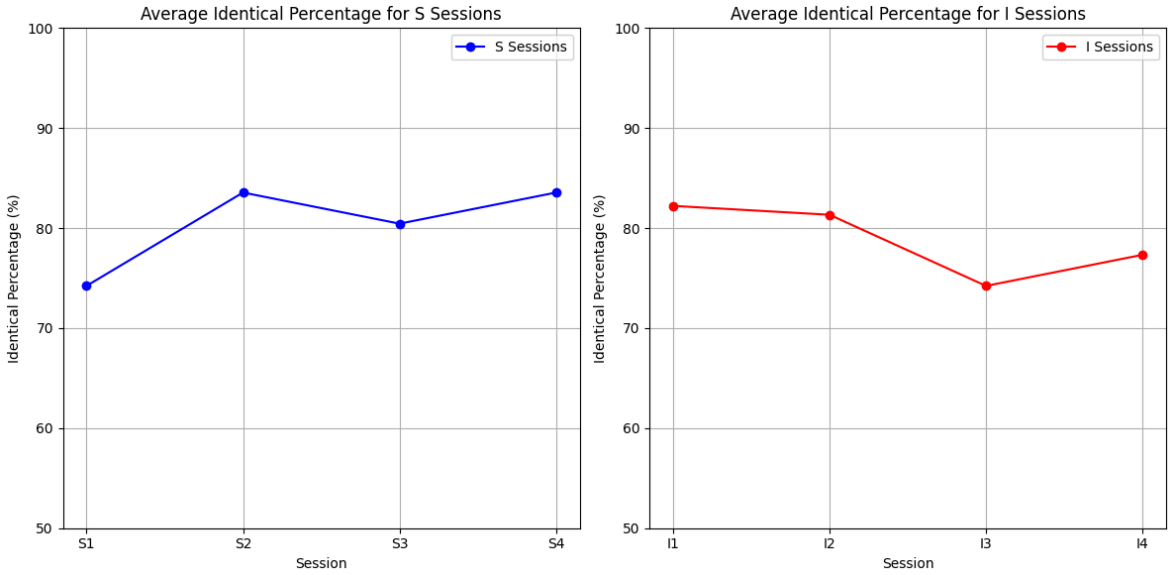
```
29   jiyoung_S_2_combined.csv     0.988428                    0.016
92.0
30   jiyoung_S_3_combined.csv     0.922031                    0.080
72.0
31   jiyoung_S_4_combined.csv     0.993883                    0.008
96.0
32     minho_I_1_combined.csv     0.829396                    0.112
64.0
33     minho_I_2_combined.csv     0.960760                    0.052
76.0
34     minho_I_3_combined.csv     0.978370                    0.032
84.0
35     minho_I_4_combined.csv     0.843053                    0.140
52.0
36     minho_S_1_combined.csv     0.993877                    0.008
96.0
37     minho_S_2_combined.csv     0.983317                    0.024
88.0
38     minho_S_3_combined.csv     0.939945                    0.044
84.0
39     minho_S_4_combined.csv     0.988290                    0.016
92.0
40   nayoung_I_1_combined.csv     0.631618                    0.176
60.0
41   nayoung_I_2_combined.csv     0.887813                    0.148
48.0
42   nayoung_I_3_combined.csv    -0.199253                    0.452
16.0
43   nayoung_I_4_combined.csv     0.755987                    0.152
48.0
44   nayoung_S_1_combined.csv     0.979792                    0.028
84.0
45   nayoung_S_2_combined.csv     0.848014                    0.092
72.0
46   nayoung_S_3_combined.csv     0.915326                    0.060
80.0
47   nayoung_S_4_combined.csv     0.900424                    0.064
80.0
48  seongjun_I_1_combined.csv     0.994014                    0.008
96.0
49  seongjun_I_2_combined.csv     1.000000                    0.000
100.0
50  seongjun_I_3_combined.csv     1.000000                    0.000
100.0
51  seongjun_I_4_combined.csv     1.000000                    0.000
100.0
52  seongjun_S_1_combined.csv     0.900718                    0.072
76.0
53  seongjun_S_2_combined.csv     1.000000                    0.000
100.0
54  seongjun_S_3_combined.csv     0.887474                    0.092
72.0
55  seongjun_S_4_combined.csv     0.988428                    0.016
92.0
56  seoyeong_I_1_combined.csv     0.755282                    0.128
72.0
57  seoyeong_I_2_combined.csv     0.814612                    0.124
72.0
58  seoyeong_I_3_combined.csv     0.866834                    0.080
84.0
```

```
59  seoyeong_I_4_combined.csv     0.920382                    0.040
92.0
60  seoyeong_S_1_combined.csv     0.669061                    0.156
56.0
61  seoyeong_S_2_combined.csv     0.782878                    0.116
72.0
62  seoyeong_S_3_combined.csv     0.867168                    0.076
80.0
63  seoyeong_S_4_combined.csv     0.979068                    0.032
84.0
64   taeyoon_I_1_combined.csv     0.834818                    0.088
80.0
65   taeyoon_I_2_combined.csv     0.866834                    0.080
84.0
66   taeyoon_I_3_combined.csv     0.803441                    0.124
68.0
67   taeyoon_I_4_combined.csv     0.805299                    0.116
72.0
68   taeyoon_S_1_combined.csv     0.427669                    0.276
32.0
69   taeyoon_S_2_combined.csv     0.515988                    0.236
48.0
70   taeyoon_S_3_combined.csv     0.767243                    0.136
60.0
71   taeyoon_S_4_combined.csv     0.619546                    0.184
56.0
```

In [41]:

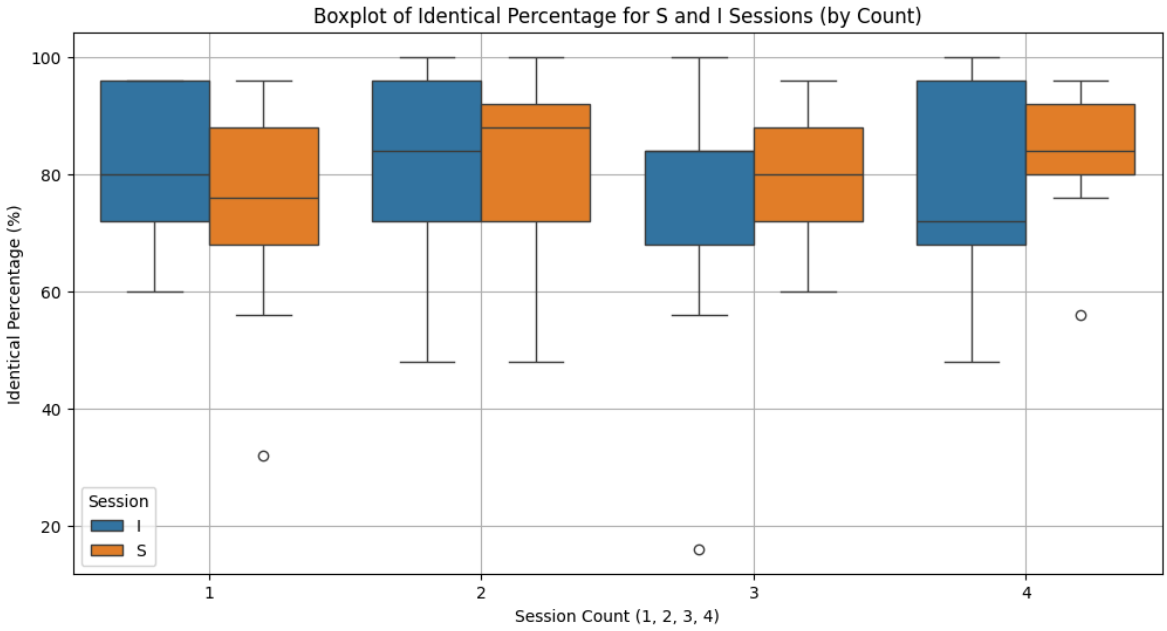
```python
# S와 I에 해당하는 identical_percentage 저장 딕셔너리
identical_percentages = {'S1': [], 'S2': [], 'S3': [], 'S4': [],
                         'I1': [], 'I2': [], 'I3': [], 'I4': []}

# 각 CSV 파일에 대해 작업 수행
for csv_file in csv_files:
    file_path = os.path.join(directory_path, csv_file)

    # CSV 파일 읽기
    df = pd.read_csv(file_path)

    # maxforce와 result 열이 있는지 확인
    if 'maxforce' in df.columns and 'result' in df.columns:
        # maxforce와 result가 동일한 값의 비율 계산
        identical_percentage = (df['maxforce'] == df['result']).mean() * 100

        # 파일 이름에서 S1, S2, S3, S4 또는 I1, I2, I3, I4를 추출하여 해당 리스트
        if '_S_1_' in csv_file:
            identical_percentages['S1'].append(identical_percentage)
        elif '_S_2_' in csv_file:
            identical_percentages['S2'].append(identical_percentage)
        elif '_S_3_' in csv_file:
            identical_percentages['S3'].append(identical_percentage)
        elif '_S_4_' in csv_file:
            identical_percentages['S4'].append(identical_percentage)
        elif '_I_1_' in csv_file:
            identical_percentages['I1'].append(identical_percentage)
        elif '_I_2_' in csv_file:
            identical_percentages['I2'].append(identical_percentage)
        elif '_I_3_' in csv_file:
            identical_percentages['I3'].append(identical_percentage)
        elif '_I_4_' in csv_file:
            identical_percentages['I4'].append(identical_percentage)
```

```
    else:
        print(f"'maxforce' or 'result' column missing in {csv_file}")

# S1, S2, S3, S4, I1, I2, I3, I4의 평균 계산
average_S1 = sum(identical_percentages['S1']) / len(identical_percentages['S1'])
average_S2 = sum(identical_percentages['S2']) / len(identical_percentages['S2'])
average_S3 = sum(identical_percentages['S3']) / len(identical_percentages['S3'])
average_S4 = sum(identical_percentages['S4']) / len(identical_percentages['S4'])

average_I1 = sum(identical_percentages['I1']) / len(identical_percentages['I1'])
average_I2 = sum(identical_percentages['I2']) / len(identical_percentages['I2'])
average_I3 = sum(identical_percentages['I3']) / len(identical_percentages['I3'])
average_I4 = sum(identical_percentages['I4']) / len(identical_percentages['I4'])
# S와 I의 평균을 리스트로 저장
averages_S = [average_S1, average_S2, average_S3, average_S4]
averages_I = [average_I1, average_I2, average_I3, average_I4]

# 두 개의 그래프 생성
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# S 세션 그래프 (왼쪽) - 파란색
axs[0].plot(['S1', 'S2', 'S3', 'S4'], averages_S, marker='o', color='blue', labe
axs[0].set_title('Average Identical Percentage for S Sessions')
axs[0].set_xlabel('Session')
axs[0].set_ylabel('Identical Percentage (%)')
axs[0].grid(True)
axs[0].legend()
axs[0].set_ylim(50, 100)  # y축 범위를 50%~100%로 설정

# I 세션 그래프 (오른쪽) - 빨간색
axs[1].plot(['I1', 'I2', 'I3', 'I4'], averages_I, marker='o', color='red', label
axs[1].set_title('Average Identical Percentage for I Sessions')
axs[1].set_xlabel('Session')
axs[1].set_ylabel('Identical Percentage (%)')
axs[1].grid(True)
axs[1].legend()
axs[1].set_ylim(50, 100)  # y축 범위를 50%~100%로 설정

# 그래프 출력
plt.tight_layout()
plt.show()
```



In [42]:

```
# 각 횟수별로 identical_percentage 저장 딕셔너리
identical_percentages = {'Session': [], 'Percentage': [], 'Count': []}

# 각 CSV 파일에 대해 작업 수행
for csv_file in csv_files:
    file_path = os.path.join(directory_path, csv_file)

    # CSV 파일 읽기
    df = pd.read_csv(file_path)

    # maxforce와 result 열이 있는지 확인
    if 'maxforce' in df.columns and 'result' in df.columns:
        # maxforce와 result가 동일한 값의 비율 계산
        identical_percentage = (df['maxforce'] == df['result']).mean() * 100

        # 파일 이름에서 S1, S2, S3, S4 또는 I1, I2, I3, I4를 추출하여 데이터 추가
        if '_S_1_' in csv_file:
            identical_percentages['Session'].append('S')
            identical_percentages['Count'].append('1')
        elif '_S_2_' in csv_file:
            identical_percentages['Session'].append('S')
            identical_percentages['Count'].append('2')
        elif '_S_3_' in csv_file:
            identical_percentages['Session'].append('S')
            identical_percentages['Count'].append('3')
        elif '_S_4_' in csv_file:
            identical_percentages['Session'].append('S')
            identical_percentages['Count'].append('4')
        elif '_I_1_' in csv_file:
            identical_percentages['Session'].append('I')
            identical_percentages['Count'].append('1')
        elif '_I_2_' in csv_file:
            identical_percentages['Session'].append('I')
            identical_percentages['Count'].append('2')
        elif '_I_3_' in csv_file:
            identical_percentages['Session'].append('I')
            identical_percentages['Count'].append('3')
        elif '_I_4_' in csv_file:
            identical_percentages['Session'].append('I')
            identical_percentages['Count'].append('4')

        identical_percentages['Percentage'].append(identical_percentage)

# 데이터프레임으로 변환
df_percentages = pd.DataFrame(identical_percentages)

# Boxplot 생성
plt.figure(figsize=(12, 6))
sns.boxplot(x='Count', y='Percentage', hue='Session', data=df_percentages)

# 그래프 세부 설정
plt.title('Boxplot of Identical Percentage for S and I Sessions (by Count)')
plt.xlabel('Session Count (1, 2, 3, 4)')
plt.ylabel('Identical Percentage (%)')
plt.grid(True)

# 그래프 출력
plt.show()
```

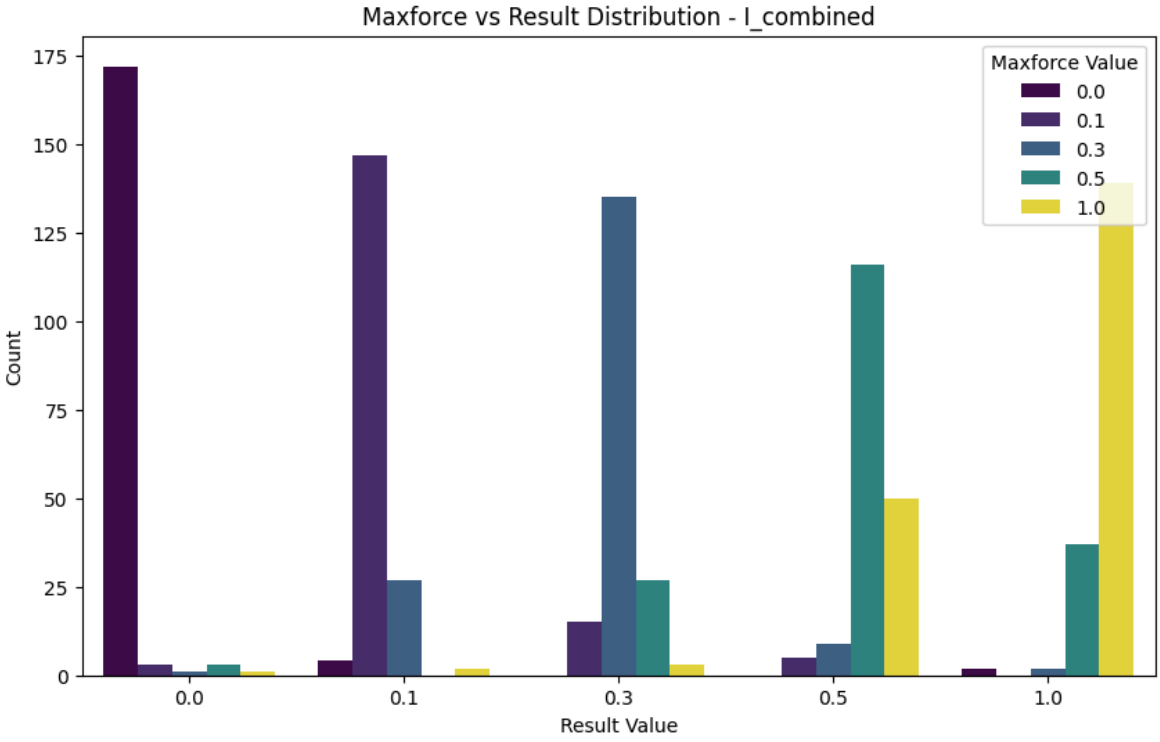


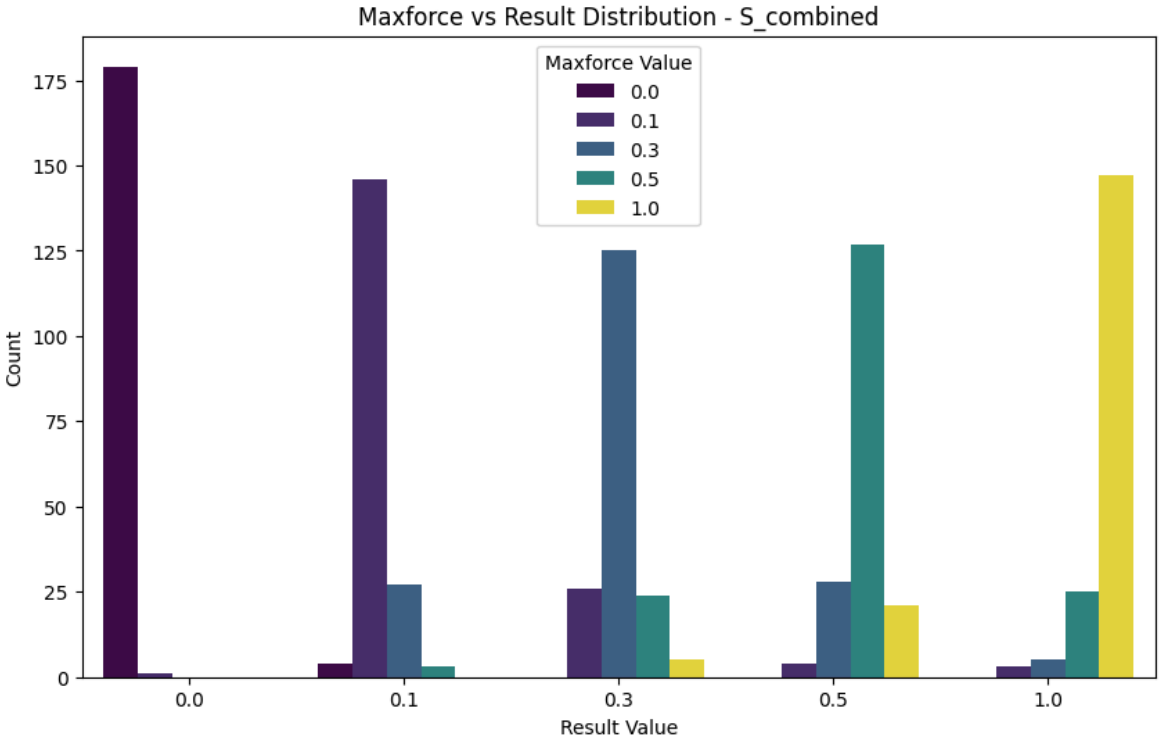
In [ ]:

🔗 렌더링별 각 데이터 합산 결과

In [43]:
```
import re

# E_combined와 S_combined 파일 리스트 필터링
i_combined_files = [f for f in os.listdir(directory_path) if re.search(r'I_\d+_c
s_combined_files = [f for f in os.listdir(directory_path) if re.search(r'S_\d+_c

# E_combined와 S_combined 데이터프레임 각각 합치기
df_i_combined = pd.concat([pd.read_csv(os.path.join(directory_path, file)) for f
df_s_combined = pd.concat([pd.read_csv(os.path.join(directory_path, file)) for f

# 데이터프레임을 합친 후 각각 동일한 작업 수행
for df_combined, combined_name in [(df_i_combined, 'I_combined'), (df_s_combined

    # maxforce와 result 열이 있는지 확인
    if 'maxforce' in df_combined.columns and 'result' in df_combined.columns:
        maxforce_values = [0.0, 0.1, 0.3, 0.5, 1.0]
        result_values = [0.0, 0.1, 0.3, 0.5, 1.0]

        distribution = []

        for maxforce_value in maxforce_values:
            for result_value in result_values:
                count = len(df_combined[(df_combined['maxforce'] == maxforce_val
                distribution.append({
                    'maxforce_value': maxforce_value,
                    'result_value': result_value,
                    'count': count
                })

        distribution_df = pd.DataFrame(distribution)

        # 막대그래프 생성
        plt.figure(figsize=(10, 6))
        sns.barplot(x='result_value', y='count', hue='maxforce_value', data=dist
```

```
        # 그래프 세부 설정
        plt.title(f'Maxforce vs Result Distribution - {combined_name}')
        plt.xlabel('Result Value')
        plt.ylabel('Count')
        plt.legend(title='Maxforce Value')

        # 그래프 출력
        plt.show()

    else:
        print(f"'maxforce' or 'result' column missing in {combined_name}")
```





In [44]:
```
# 결과를 저장할 리스트 초기화
comparison_results = []
```

```
# 각 데이터프레임에 대해 작업 수행
for df_combined, combined_name in [(df_i_combined, 'I_combined'), (df_s_combined

    # maxforce와 result 열이 있는지 확인
    if 'maxforce' in df_combined.columns and 'result' in df_combined.columns:
        # 상관계수 계산
        correlation = df_combined['maxforce'].corr(df_combined['result'])

        # 차이의 절대값 평균 계산
        mean_absolute_difference = (df_combined['maxforce'] - df_combined['resul

        # maxforce와 result가 동일한 값의 비율 계산
        identical_percentage = (df_combined['maxforce'] == df_combined['result']

        # 결과 저장
        comparison_results.append({
            'filename': combined_name,
            'correlation': correlation,
            'mean_absolute_difference': mean_absolute_difference,
            'identical_percentage': identical_percentage
        })

    else:
        print(f"'maxforce' or 'result' column missing in {combined_name}")

# 결과를 데이터프레임으로 변환
results_df = pd.DataFrame(comparison_results)

# 결과 출력
print(results_df)
```

```
     filename  correlation  mean_absolute_difference  identical_percentage
0  I_combined     0.855124                  0.079889             78.777778
1  S_combined     0.892093                  0.063333             80.444444
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: