Collision Avoidance Challenge - EDA

This notebook is used to perform exploratory data analysis (EDA) on the Collision Avoidance Challenge dataset.

Setup

Imports

In [279…
```
import os
import sys
import numpy as np
import yaml
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import zipfile
from scipy.stats import yeojohnson
```

Set up the environment

In [279…
```
sns.set_theme(style="whitegrid")
plt.rcParams['figure.figsize'] = (12, 6)
%matplotlib inline
```

Add the src directory to the path

In [279…
```
module_path = os.path.abspath(os.path.join('..', 'src'))
if module_path not in sys.path:
    sys.path.append(module_path)
```

Extract the zip file

In [279…
```
file_name = 'test_data.csv.zip'
zip_path = '../data/'
extract_to = '../data/'

os.makedirs(extract_to, exist_ok=True)

with zipfile.ZipFile(zip_path + file_name, 'r') as zip_ref:
    zip_ref.extractall(extract_to)
```

Read the data

In [279…
```
csv_file = os.path.join(extract_to, file_name)
df = pd.read_csv(csv_file)
```

Copy the collumns.yml file to the working columns (processed_columns.yml) file

In [280…
```
!cp ../columns.yml ../filtered_columns.yml
```

Basic Information

First Lines

In [280…
```
display(df.head())
```

	event_id	time_to_tca	mission_id	risk	max_risk_estimate	max_risk_scaling	miss_distance	relative_speed	relative_position_r	rel
0	0	6.842095	19	-7.296967	-7.208941	1.787894	31816.0	7929.0	-365.5	
1	0	6.571818	19	-7.282496	-7.199833	1.759386	31095.0	7929.0	-361.8	
2	0	6.112986	19	-7.316053	-7.217886	1.824263	32202.0	7929.0	-370.7	
3	0	5.921955	19	-7.334138	-7.228707	1.865396	32878.0	7929.0	-376.3	
4	0	2.228761	19	-7.332267	-7.227312	1.863127	32792.0	7929.0	-375.6	

5 rows × 103 columns

Dataset Information

In [280…
```
display(
    pd.DataFrame({
        'Number of rows': [len(df)],
        'Number of columns': [len(df.columns)]
    }).style.hide(axis='index')
)
```

Number of rows	Number of columns
24484	103

Filter columns

Filter marked columns.

In [280…
```
with open(os.path.join('..', 'filtered_columns.yml'), 'r') as file:
    columns_config = yaml.safe_load(file)

def get_columns_to_remove(config):
```

```
    columns_to_remove = []
    for key, value in config.items():
        if isinstance(value, dict):
            if 'kept' in value:
                if value['kept'] == False:
                    if (key.startswith('x_')):
                        columns_to_remove.append('c' + key[1:])
                        columns_to_remove.append('t' + key[1:])
                    else:
                        columns_to_remove.append(key)
            else:
                columns_to_remove.extend(get_columns_to_remove(value))
    return [col for col in columns_to_remove if col in df.columns]

columns_to_remove = get_columns_to_remove(columns_config['columns'])
df = df.drop(columns=columns_to_remove)

display(
    pd.DataFrame({
        'Number of columns': [len(df.columns)]
    }).style.hide(axis='index')
)
```

Number of columns
47

Columns Information

In [280…

```
with pd.option_context('display.max_rows', None, 'display.float_format', '{:,.2f}'.format):
    result = pd.DataFrame(
        {
        'Data Type': df.dtypes,
        'Missing Values (%)': df.isnull().sum() / len(df) * 100,
        'Unique Values (%)': df.nunique() / len(df) * 100,
        }
    ).sort_values(['Missing Values (%)', 'Unique Values (%)'], ascending=[False, False]
    ).join(df.describe().T[['mean', 'std', 'min', '25%', '50%', '75%', 'max']])
    display(result)
```

	Data Type	Missing Values (%)	Unique Values (%)	mean	std	min	25%	50%	75%	max
SSN	float64	5.43	0.56	20.04	25.04	0.00	0.00	13.00	28.00	172.00
F10	float64	5.43	0.39	77.48	14.01	66.00	69.00	72.00	79.00	182.00
F3M	float64	5.43	0.33	78.48	12.93	69.00	71.00	73.00	82.00	159.00
AP	float64	5.43	0.22	8.61	8.37	0.00	4.00	6.00	10.00	106.00
c_weighted_rms	float64	0.02	9.99	1.98	0.68	0.12	1.51	1.92	2.33	4.99
c_actual_od_span	float64	0.02	8.15	12.41	7.82	0.08	6.95	11.93	16.45	53.35
c_obs_available	float64	0.02	2.18	63.50	126.62	9.00	21.00	31.00	60.00	10,000.00
c_obs_used	float64	0.02	2.05	59.18	121.22	6.00	21.00	30.00	57.00	10,000.00
c_time_lastob_start	float64	0.02	0.01	41.74	74.88	1.00	1.00	1.00	2.00	180.00
c_time_lastob_end	float64	0.02	0.01	0.60	0.83	0.00	0.00	0.00	1.00	2.00
t_j2k_sma	float64	0.00	100.00	7,049.18	104.68	6,676.44	6,996.10	7,081.95	7,155.29	7,210.17
t_j2k_ecc	float64	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
t_j2k_inc	float64	0.00	100.00	95.53	4.07	87.24	92.04	97.90	98.47	98.84
c_j2k_sma	float64	0.00	100.00	7,193.06	1,226.26	6,652.86	7,003.57	7,080.82	7,148.56	29,835.01
c_j2k_ecc	float64	0.00	100.00	0.01	0.06	0.00	0.00	0.00	0.01	0.76
c_j2k_inc	float64	0.00	100.00	86.48	15.37	3.20	74.08	87.42	98.80	123.14
t_h_apo	float64	0.00	100.00	684.13	104.31	302.23	633.25	709.31	780.73	842.32
t_h_per	float64	0.00	100.00	657.97	105.82	275.66	603.69	682.63	757.17	821.78
c_h_apo	float64	0.00	100.00	994.35	2,464.36	308.25	653.12	734.24	814.46	46,221.55
c_h_per	float64	0.00	100.00	635.51	108.54	164.87	574.73	657.50	710.15	833.87
geocentric_latitude	float64	0.00	100.00	-1.15	67.28	-87.88	-72.70	-4.37	72.38	87.71
time_to_tca	float64	0.00	99.99	4.33	1.43	2.00	3.08	4.26	5.56	6.99
mahalanobis_distance	float64	0.00	99.96	114.75	254.32	0.00	16.53	51.11	131.53	19,076.92

	Data Type	Missing Values (%)	Unique Values (%)	mean	std	min	25%	50%	75%	max
relative_position_n	float64	0.00	96.23	-607.26	14,745.82	-50,774.50	-6,345.35	-135.95	5,821.20	50,527.00
relative_position_t	float64	0.00	94.38	129.19	13,353.24	-61,304.10	-4,152.00	58.75	4,729.32	61,287.50
miss_distance	float64	0.00	69.68	14,585.17	13,553.99	17.00	3,737.00	10,179.50	22,158.75	65,927.00
t_sedr	float64	0.00	67.92	0.00	0.00	-0.00	0.00	0.00	0.00	0.01
elevation	float64	0.00	67.07	0.06	2.19	-44.74	-0.14	-0.00	0.12	57.04
risk	float64	0.00	62.97	-16.55	9.85	-30.00	-30.00	-13.55	-7.42	-1.53
t_cd_area_over_mass	float64	0.00	52.97	0.01	0.01	-0.10	0.01	0.01	0.02	0.37
t_cr_area_over_mass	float64	0.00	51.81	0.01	0.01	0.00	0.01	0.01	0.01	0.12
relative_position_r	float64	0.00	51.80	-31.59	571.21	-2,586.80	-253.43	-28.55	185.83	2,406.90
c_sedr	float64	0.00	37.37	0.00	0.01	-0.00	0.00	0.00	0.00	0.52
c_cd_area_over_mass	float64	0.00	37.12	0.67	1.69	-4.58	0.19	0.44	0.68	118.75
c_cr_area_over_mass	float64	0.00	31.07	0.31	0.73	-0.33	0.06	0.19	0.30	15.69
azimuth	float64	0.00	25.22	1.10	45.67	-107.65	-32.07	4.16	35.07	119.34
relative_velocity_t	float64	0.00	19.38	-9,029.06	5,256.29	-15,979.00	-14,102.30	-10,479.90	-3,888.20	48.00
relative_velocity_n	float64	0.00	17.07	180.40	5,201.71	-9,832.40	-4,696.42	719.20	5,140.60	9,865.40
relative_velocity_r	float64	0.00	11.71	0.34	114.47	-1,826.40	-22.92	0.40	25.60	1,898.40
relative_speed	float64	0.00	9.06	10,837.46	4,335.10	58.00	7,673.00	12,566.00	14,562.00	16,956.00
event_id	int64	0.00	8.85	1,084.17	622.80	0.00	545.00	1,086.00	1,625.00	2,166.00
t_obs_used	int64	0.00	8.66	968.91	1,065.95	6.00	402.00	470.00	740.00	4,405.00
t_obs_available	int64	0.00	7.94	1,017.33	1,075.48	36.00	430.00	484.00	746.00	4,406.00
t_weighted_rms	float64	0.00	3.33	1.12	0.16	0.53	1.00	1.10	1.24	1.88
t_actual_od_span	float64	0.00	2.83	3.82	1.33	0.06	3.46	3.70	3.92	8.15
t_time_lastob_start	float64	0.00	0.01	1.00	0.02	1.00	1.00	1.00	1.00	2.00

	Data Type	Missing Values (%)	Unique Values (%)	mean	std	min	25%	50%	75%	max
t_time_lastob_end	float64	0.00	0.01	0.00	0.02	0.00	0.00	0.00	0.00	1.00

Column Vizualization and Preprocessing

Remove NaNs, -inf and +inf

NaNs are substituted by column mean. -inf and +inf are substituted by column min and max, respectively.

In [280…

```
for col in df.select_dtypes(include=np.number).columns:
    df[col] = np.nan_to_num(
        df[col],
        nan=df[col].mean(),
        posinf=df[col].max(),
        neginf=df[col].min()
    )
```

Outliers

Perform Yeo-Johnson transformation and capping (for values above and below the upper and lower thresholds).

Yeojohnson Transformation

In [280…

```
with open('../configs.yml', 'r') as f:
    configs = yaml.safe_load(f)

def cap_outliers(df, outlier_summary, iqr_factor):
    for col in df.select_dtypes(include=np.number).columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
```

```
        lower_bound = Q1 - (iqr_factor * IQR)
        upper_bound = Q3 + (iqr_factor * IQR)

        outliers_mask = (df[col] < lower_bound) | (df[col] > upper_bound)
        outliers_count_before = df[outliers_mask].shape[0]

        percent_outliers = (outliers_count_before / len(df)) * 100
        if percent_outliers > 0:
            outlier_summary[col] = {
                'Count': outliers_count_before,
                '%': percent_outliers
            }
            df[col], _ = yeojohnson(df[col])

iqr_factor = configs['iqr_factor']
outlier_summary = {}
cap_outliers(df, outlier_summary, iqr_factor)

summary_df = pd.DataFrame.from_dict(outlier_summary, orient='index')
summary_df = summary_df.sort_values(by='%', ascending=False)

with pd.option_context('display.max_rows', None, 'display.float_format', '{:,.3f}'.format):
    display(summary_df)
```

	Count	%
t_actual_od_span	5680	23.199
c_time_lastob_start	5557	22.696
t_obs_available	5226	21.345
t_obs_used	5098	20.822
t_sedr	4191	17.117
relative_position_t	4029	16.456
elevation	3454	14.107
relative_position_r	3092	12.629
c_obs_available	3034	12.392
c_obs_used	2987	12.200
F10	2825	11.538
relative_position_n	2734	11.166
c_sedr	2598	10.611
mahalanobis_distance	2081	8.499
relative_velocity_r	2040	8.332
c_j2k_ecc	2039	8.328
F3M	1847	7.544
AP	1827	7.462
c_cr_area_over_mass	1764	7.205
SSN	1560	6.372
c_cd_area_over_mass	1489	6.082
t_cr_area_over_mass	1270	5.187
c_h_apo	1235	5.044
c_actual_od_span	753	3.075

	Count	%
c_j2k_sma	753	3.075
c_weighted_rms	727	2.969
t_cd_area_over_mass	518	2.116
miss_distance	487	1.989
c_j2k_inc	383	1.564
c_h_per	351	1.434
t_h_per	136	0.555
t_j2k_sma	136	0.555
t_h_apo	136	0.555
t_j2k_ecc	118	0.482
t_weighted_rms	31	0.127
t_time_lastob_end	9	0.037
t_time_lastob_start	9	0.037

Capping

In [280…

```
with open('../configs.yml', 'r') as f:
    configs = yaml.safe_load(f)

lower_bound_outliers_capping = configs['lower_bound_outliers_capping']
upper_bound_outliers_capping = configs['upper_bound_outliers_capping']

def cap_outliers(df, outlier_summary, iqr_factor):
    for col in df.select_dtypes(include=np.number).columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - (iqr_factor * IQR)
```

```
        upper_bound = Q3 + (iqr_factor * IQR)

        outliers_mask = (df[col] < lower_bound) | (df[col] > upper_bound)
        outliers_count_before = df[outliers_mask].shape[0]

        percent_outliers = (outliers_count_before / len(df)) * 100
        if percent_outliers > lower_bound_outliers_capping and percent_outliers <= upper_bound_outliers_capping:
            outlier_summary[col] = {
                'Count': outliers_count_before,
                '%': percent_outliers
            }
            df[col] = df[col].clip(lower=lower_bound, upper=upper_bound)

iqr_factor = configs['iqr_factor']
outlier_summary = {}
cap_outliers(df, outlier_summary, iqr_factor)

summary_df = pd.DataFrame.from_dict(outlier_summary, orient='index')
summary_df = summary_df.sort_values(by='%', ascending=False)

with pd.option_context('display.max_rows', None, 'display.float_format', '{:,.3f}'.format):
    display(summary_df)
```

	Count	%
c_sedr	2386	9.745
relative_velocity_r	2027	8.279
c_h_apo	1908	7.793
c_j2k_ecc	1215	4.962
c_cd_area_over_mass	1204	4.917
t_cd_area_over_mass	474	1.936
c_actual_od_span	349	1.425
c_weighted_rms	319	1.303
AP	297	1.213
c_j2k_inc	93	0.380
mahalanobis_distance	74	0.302
t_weighted_rms	32	0.131
t_time_lastob_start	9	0.037
t_time_lastob_end	9	0.037
c_obs_used	7	0.029
c_cr_area_over_mass	1	0.004

Variance

Drop columns with low variance

In [280…

```
with open("../configs.yml", "r") as f:
    configs = yaml.safe_load(f)

VAR_THRESHOLD = configs["low_variance_percent_threshold"]
```

```
ranges = df.select_dtypes(include=[np.number]).max() - df.select_dtypes(include=[np.number]).min()
var_values = df.var()
thresholds = ranges * VAR_THRESHOLD

cols_to_drop = set()
removed_cols = []

for col in var_values.index:
    if col in thresholds.index and var_values[col] <= thresholds[col]:
        cols_to_drop.add(col)
        removed_cols.append({
                'removed': col,
                'variance': var_values[col],
                'threshold': thresholds[col]
            })

should_keep = set()
def update_yaml(config, removed_cols):
    for key, value in config.items():
        for col in removed_cols:
            if key[2:] == col['removed'][2:]:
                if isinstance(value, dict):
                    if value.get('kept', False) == True:
                        should_keep.add(key)
                        continue
                    else:
                        config[key]['kept'] = False
                        config[key]['reason'] = f'low variance ({col["variance"]:.2f}) <= {col["threshold"]:.2f}'
                break
    return config

with open('../filtered_columns.yml', 'r') as f:
    columns_config = yaml.safe_load(f)
    columns_config['columns'] = update_yaml(columns_config['columns'], removed_cols)

with open('../filtered_columns.yml', 'w+') as f:
    yaml.dump(columns_config, f)

df = df.drop(columns=list(cols_to_drop - should_keep))
```

Correlation

Remove columns with high correlation and display the heat map.

In [280…

```
corr_matrix = df.corr().abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))

with open('../configs.yml', 'r') as f:
    config = yaml.safe_load(f)

THRESHOLD = config['high_correlation_threshold']

cols_to_drop = set()
removed_cols = []

for i in range(len(upper.columns)):
    for j in range(i + 1, len(upper.columns)):
        col_a = upper.columns[i]
        col_b = upper.columns[j]
        corr_value = upper.iloc[i, j]
        if corr_value >= THRESHOLD:
            cols_to_drop.add(col_b)
            removed_cols.append({
                'removed': col_b,
                'correlated': col_a,
                'correlation': corr_value
            })

should_keep = set()
new_removed_cols = []
def update_yaml(config, removed_cols):
    for key, value in config.items():
        found = False
        for col in removed_cols:
            if key[2:] == col['removed'][2:]:
                if isinstance(value, dict):
                    if value.get('kept', False) == True:
                        should_keep.add(key)
                        if isinstance(config.get(col['correlated'], {}), dict):
```

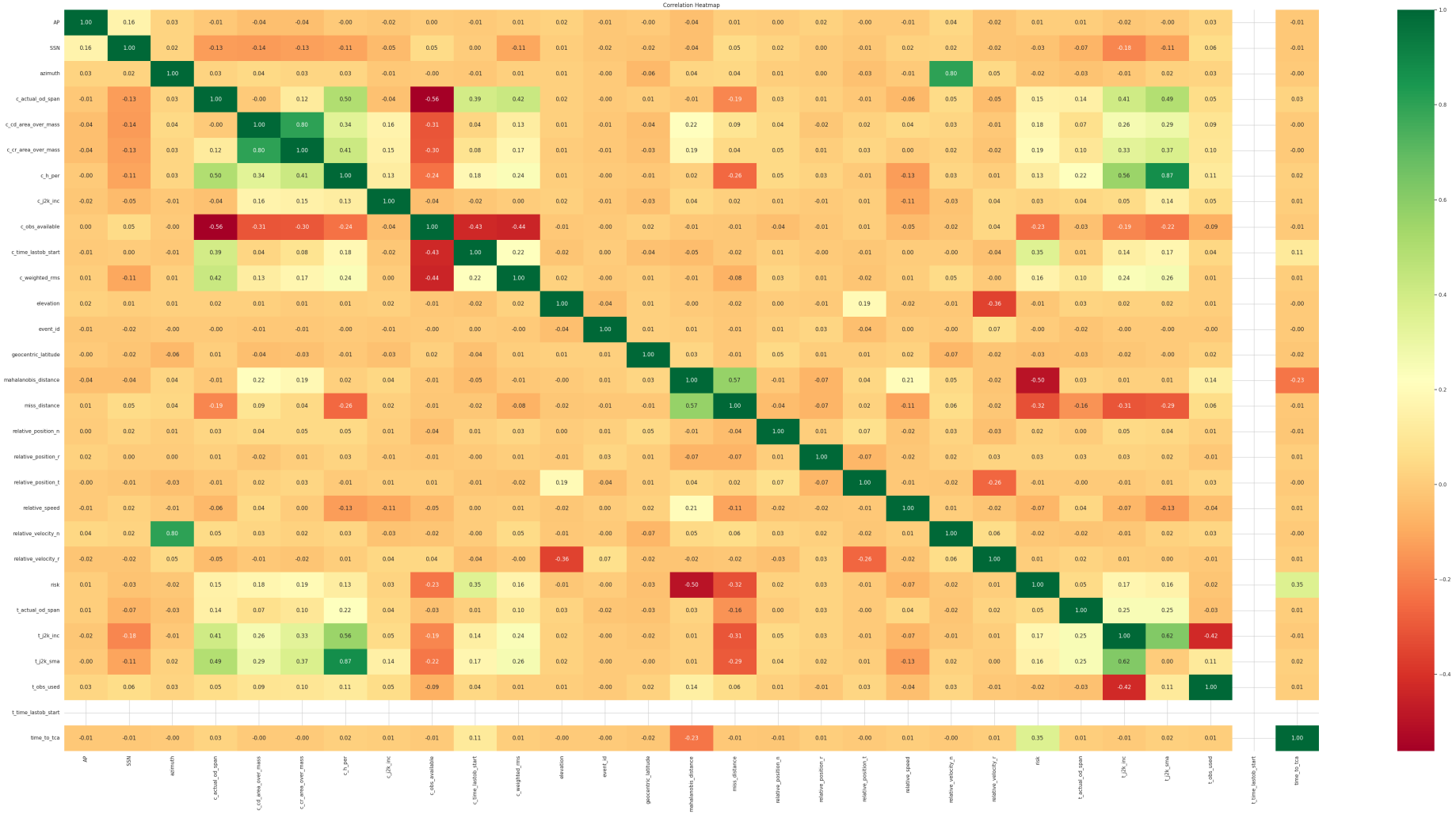
```
                            if config.get(col['correlated'], {}).get('kept', False) == True:
                                should_keep.add(col['correlated'])
                                continue
                            else:
                                new_removed_cols.append({
                                    'removed': col['correlated'],
                                    'correlated': col['removed'],
                                    'correlation': col['correlation']
                                })
                                break
                    else:
                        config[key]['kept'] = False
                        config[key]['reason'] = f'multicoliniarity with column {col["correlated"]} ({col["correlati
                        found = True
                        break

        if not found:
            if (isinstance(value, dict)):
                config[key] = update_yaml(value, removed_cols)

    return config

with open('../filtered_columns.yml', 'r') as f:
    columns_config = yaml.safe_load(f)
    columns_config['columns'] = update_yaml(columns_config['columns'], removed_cols)
    columns_config['columns'] = update_yaml(columns_config['columns'], new_removed_cols)

with open('../filtered_columns.yml', 'w+') as f:
    yaml.dump(columns_config, f)

df = df.drop(columns=list(cols_to_drop - should_keep))
```

Correlation Heatmap

In [281…

```
df = df.sort_index(axis=1)
plt.figure(figsize=(50, 25))
sns.heatmap(df.corr(), annot=True, fmt='.2f', cmap='RdYlGn')
plt.title('Correlation Heatmap')
plt.tight_layout()
```

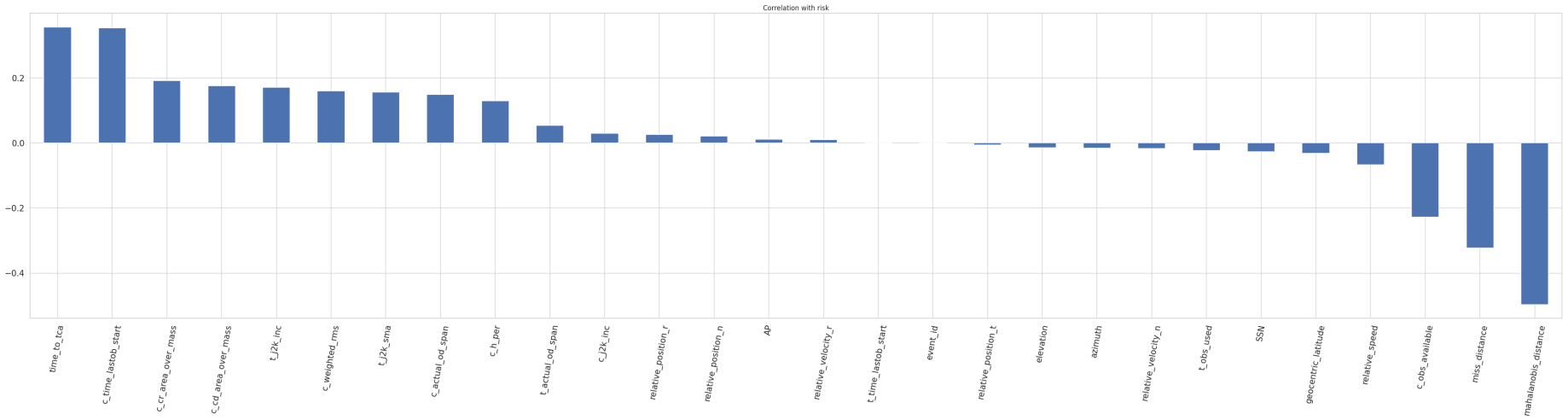
```
plt.show()
```



Correlation with Risk (Target Variable)

In [281…
```
corr_with_risk = df.drop(columns=['risk']).corrwith(df['risk']).sort_values(ascending=False)
corr_with_risk.plot.bar(figsize=(50,10),title="Correlation with risk",fontsize=15,rot=80,grid=True)
```

Out[281… `<Axes: title={'center': 'Correlation with risk'}>`



Remove Low Correlated Columns

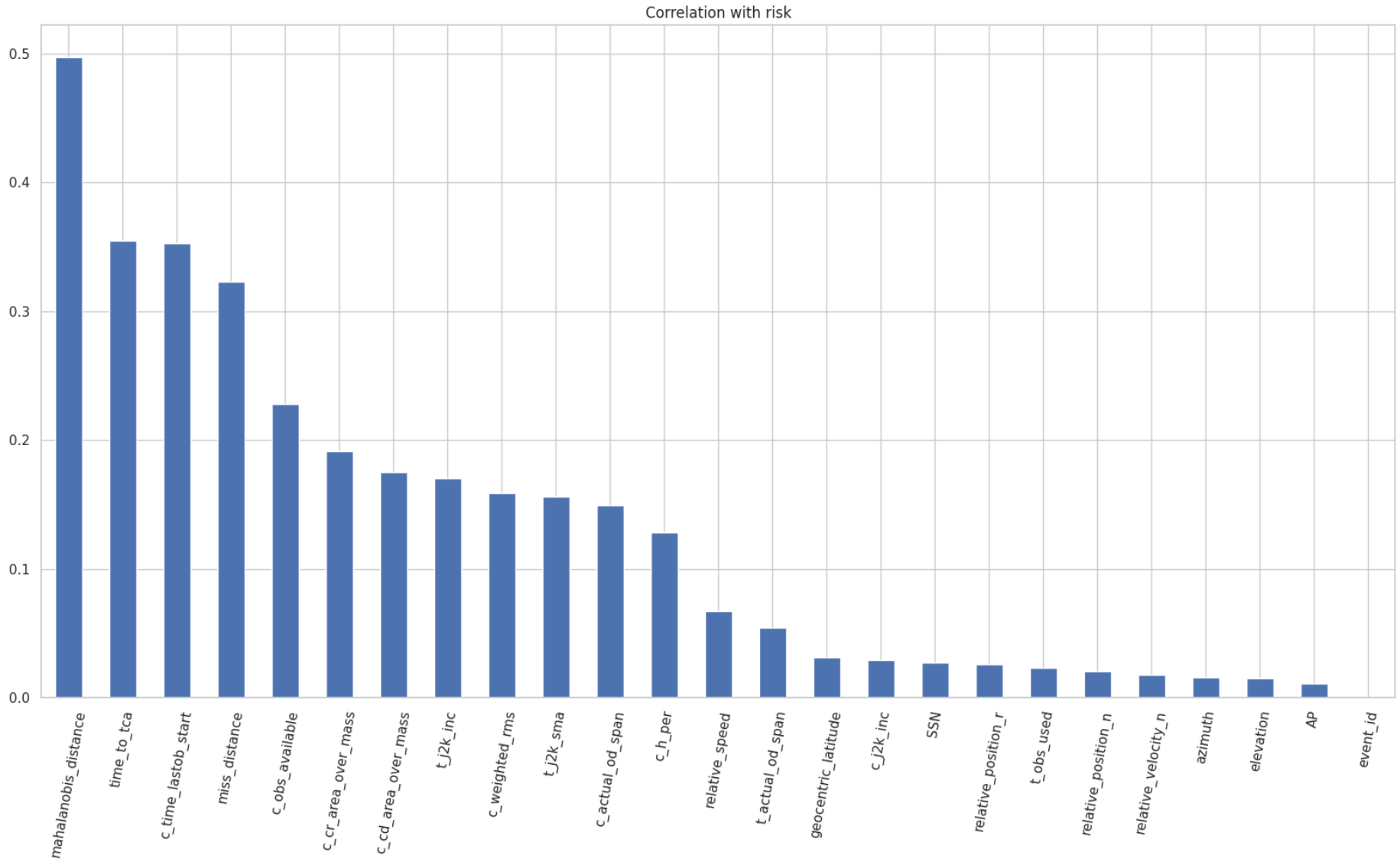
Remove columns with low correlation with risk

In [281…

```
corr_with_risk = df.drop(columns=['risk']).corrwith(df['risk']).abs().sort_values(ascending=False)

with open('../configs.yml', 'r') as f:
    config = yaml.safe_load(f)

THRESHOLD = config['low_risk_correlation_threshold']

cols_to_drop = set()
removed_cols = []

def remove_low_correlation_cols(corr_with_risk):
    for col_a, corr_a in corr_with_risk.items():
        if corr_a <= THRESHOLD:
            cols_to_drop.add(col_a)
            removed_cols.append({
                'removed': col_a,
                'removed_corr': corr_a,
            })

    return cols_to_drop
```

```
cols_to_drop = remove_low_correlation_cols(corr_with_risk)

should_keep = set()
def update_yaml(config, removed_cols):
    for key, value in config.items():
        found = False
        for col in removed_cols:
            if (isinstance(value, dict)):
                if value.get('kept', False) == True:
                    should_keep.add(key)
                    continue
            if key[2:] == col['removed'][2:]:
                config[key]['kept'] = False
                config[key]['reason'] = f'low correlation with risk ({col["removed_corr"]})'
                found = True
                break

        if not found:
            if (isinstance(value, dict)):
                config[key] = update_yaml(value, removed_cols)

    return config

with open('../filtered_columns.yml', 'r') as f:
    columns_config = yaml.safe_load(f)
    columns_config['columns'] = update_yaml(columns_config['columns'], removed_cols)

with open('../filtered_columns.yml', 'w+') as f:
    yaml.dump(columns_config, f)

df = df.drop(columns=list(cols_to_drop - should_keep))

corr_with_risk = df.drop(columns=['risk']).corrwith(df['risk']).abs().sort_values(ascending=False)
corr_with_risk.plot.bar(figsize=(20,10),title="Correlation with risk",rot=80,grid=True)
```

Out[281… <Axes: title={'center': 'Correlation with risk'}>



Remove Columns with Similar Correlation with Risk

In [281…

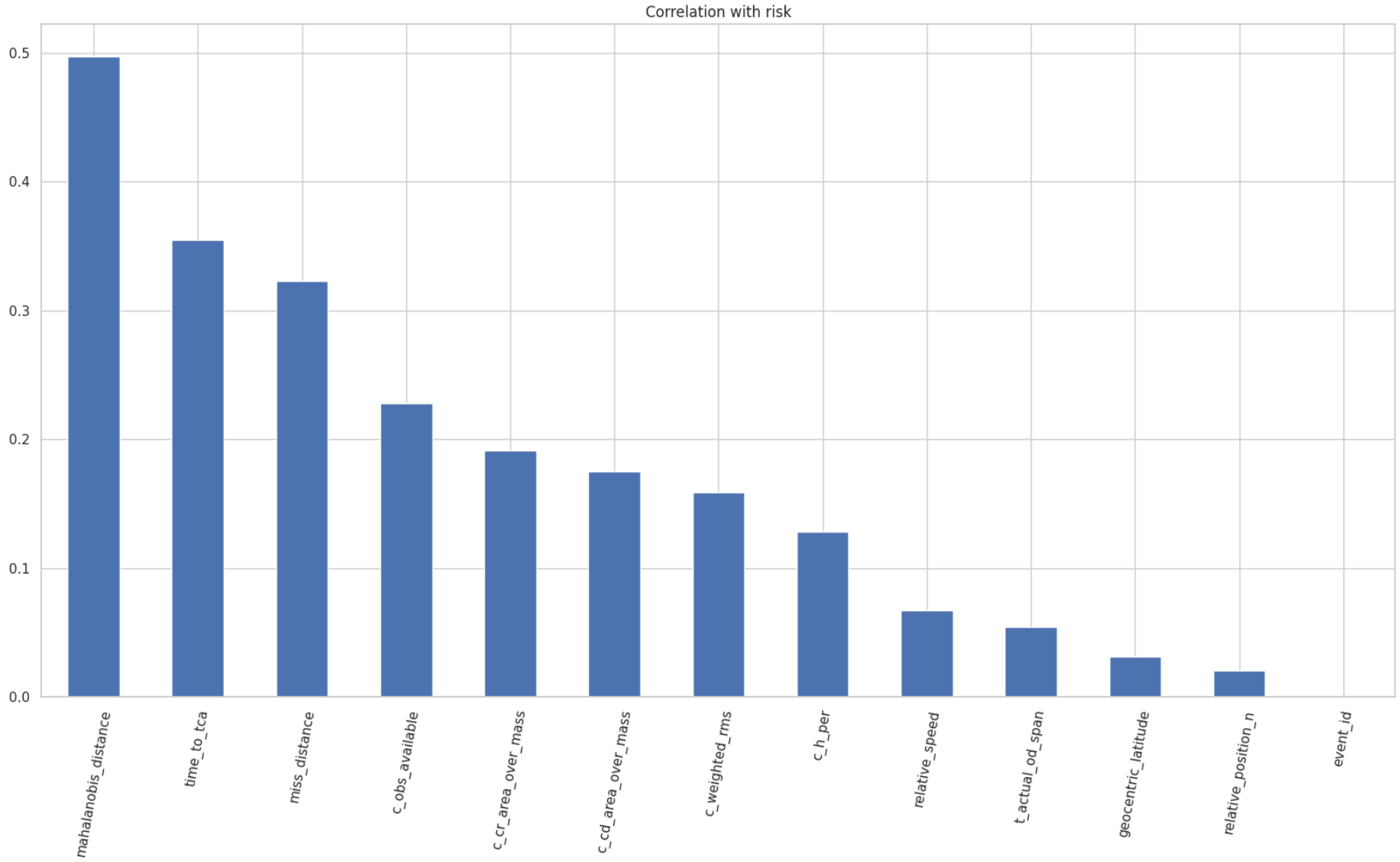
```
corr_with_risk = df.drop(columns=['risk']).corrwith(df['risk']).abs().sort_values(ascending=False)

with open('../configs.yml', 'r') as f:
```

```
    config = yaml.safe_load(f)

TOLERANCE_THRESHOLD = config['similar_risk_correlation_threshold']

cols_to_drop = set()
removed_cols = []

def remove_redundant_cols(corr_with_risk):
    for col_a, corr_a in corr_with_risk.items():
        for col_b, corr_b in corr_with_risk.items():
            if col_a == col_b:
                continue
            if abs(corr_a - corr_b) <= TOLERANCE_THRESHOLD:
                if col_a in cols_to_drop or col_b in cols_to_drop:
                    continue
                if corr_a >= corr_b:
                    cols_to_drop.add(col_b)
                    removed_cols.append({
                        'removed': col_b,
                        'removed_corr': corr_b,
                        'correlated': col_a,
                        'correlated_corr': corr_a
                    })
                else:
                    cols_to_drop.add(col_a)
                    removed_cols.append({
                        'removed': col_b,
                        'removed_corr': corr_b,
                        'correlated': col_a,
                        'correlated_corr': corr_a
                    })

    return cols_to_drop

cols_to_drop = remove_redundant_cols(corr_with_risk)

should_keep = set()
new_removed_cols = []

def update_yaml(config, removed_cols):
    for key, value in config.items():
```

```
        found = False
        for col in removed_cols:
            if key[2:] == col['removed'][2:]:
                if isinstance(value, dict):
                    if value.get('kept', False) == True:
                        should_keep.add(key)
                        if isinstance(config.get(col['correlated'], {}), dict):
                            if config.get(col['correlated'], {}).get('kept', False) == True:
                                should_keep.add(col['correlated'])
                                continue
                            else:
                                new_removed_cols.append({
                                    'removed': col['correlated'],
                                    'removed_corr': col['correlated_corr'],
                                    'correlated': col['removed'],
                                    'correlated_corr': col['removed_corr']
                                })
                                break
                    else:
                        config[key]['kept'] = False
                        config[key]['reason'] = f'similar risk correlation ({col['removed_corr']:.2f}) with column
                        found = True
                        break

        if not found:
            if (isinstance(value, dict)):
                config[key] = update_yaml(value, removed_cols)

    return config

with open('../filtered_columns.yml', 'r') as f:
    columns_config = yaml.safe_load(f)
    columns_config['columns'] = update_yaml(columns_config['columns'], removed_cols)
    columns_config['columns'] = update_yaml(columns_config['columns'], new_removed_cols)

with open('../filtered_columns.yml', 'w+') as f:
    yaml.dump(columns_config, f)

df = df.drop(columns=list(cols_to_drop - should_keep))

corr_with_risk = df.drop(columns=['risk']).corrwith(df['risk']).abs().sort_values(ascending=False)
```

```
corr_with_risk.plot.bar(figsize=(20,10),title="Correlation with risk",rot=80,grid=True)
```

Out[281… <Axes: title={'center': 'Correlation with risk'}>



Updated Columns

Updated Columns Information

In [281…

```
with pd.option_context('display.max_rows', None, 'display.float_format', '{:,.2f}'.format):
    result = pd.DataFrame(
        {
        'Data Type': df.dtypes,
        'Missing Values (%)': df.isnull().sum() / len(df) * 100,
        'Unique Values (%)': df.nunique() / len(df) * 100,
        }
    ).sort_values(['Missing Values (%)', 'Unique Values (%)'], ascending=[False, False]
    ).join(df.describe().T[['mean', 'std', 'min', '25%', '50%', '75%', 'max']])
    display(result)
```

	Data Type	Missing Values (%)	Unique Values (%)	mean	std	min	25%	50%	75%	max
c_h_per	float64	0.00	100.00	2,088,085.63	760,984.81	81,106.22	1,570,374.16	2,162,374.83	2,596,999.45	3,804,953.43
geocentric_latitude	float64	0.00	100.00	-1.15	67.28	-87.88	-72.70	-4.37	72.38	87.71
time_to_tca	float64	0.00	99.99	4.33	1.43	2.00	3.08	4.26	5.56	6.99
mahalanobis_distance	float64	0.00	99.66	4.70	2.04	0.00	3.26	4.74	6.12	10.41
relative_position_n	float64	0.00	96.23	-320.09	14,721.13	-49,168.04	-6,186.48	-134.20	5,969.07	52,177.58
miss_distance	float64	0.00	69.68	45.77	16.37	4.51	33.70	46.15	58.62	81.55
risk	float64	0.00	62.97	-16.55	9.85	-30.00	-30.00	-13.55	-7.42	-1.53
c_cd_area_over_mass	float64	0.00	35.04	0.38	0.27	-0.35	0.17	0.37	0.52	1.05
c_cr_area_over_mass	float64	0.00	31.07	0.13	0.09	-0.13	0.06	0.13	0.18	0.34
c_weighted_rms	float64	0.00	9.63	0.92	0.16	0.50	0.81	0.92	1.02	1.33
relative_speed	float64	0.00	9.06	10,837.46	4,335.10	58.00	7,673.00	12,566.00	14,562.00	16,956.00
event_id	int64	0.00	8.85	1,084.17	622.80	0.00	545.00	1,086.00	1,625.00	2,166.00
t_actual_od_span	float64	0.00	2.83	2.48	0.66	0.06	2.33	2.46	2.57	4.37
c_obs_available	float64	0.00	2.18	1.39	0.07	1.20	1.34	1.39	1.44	1.55

Updated Outliers

In [281…

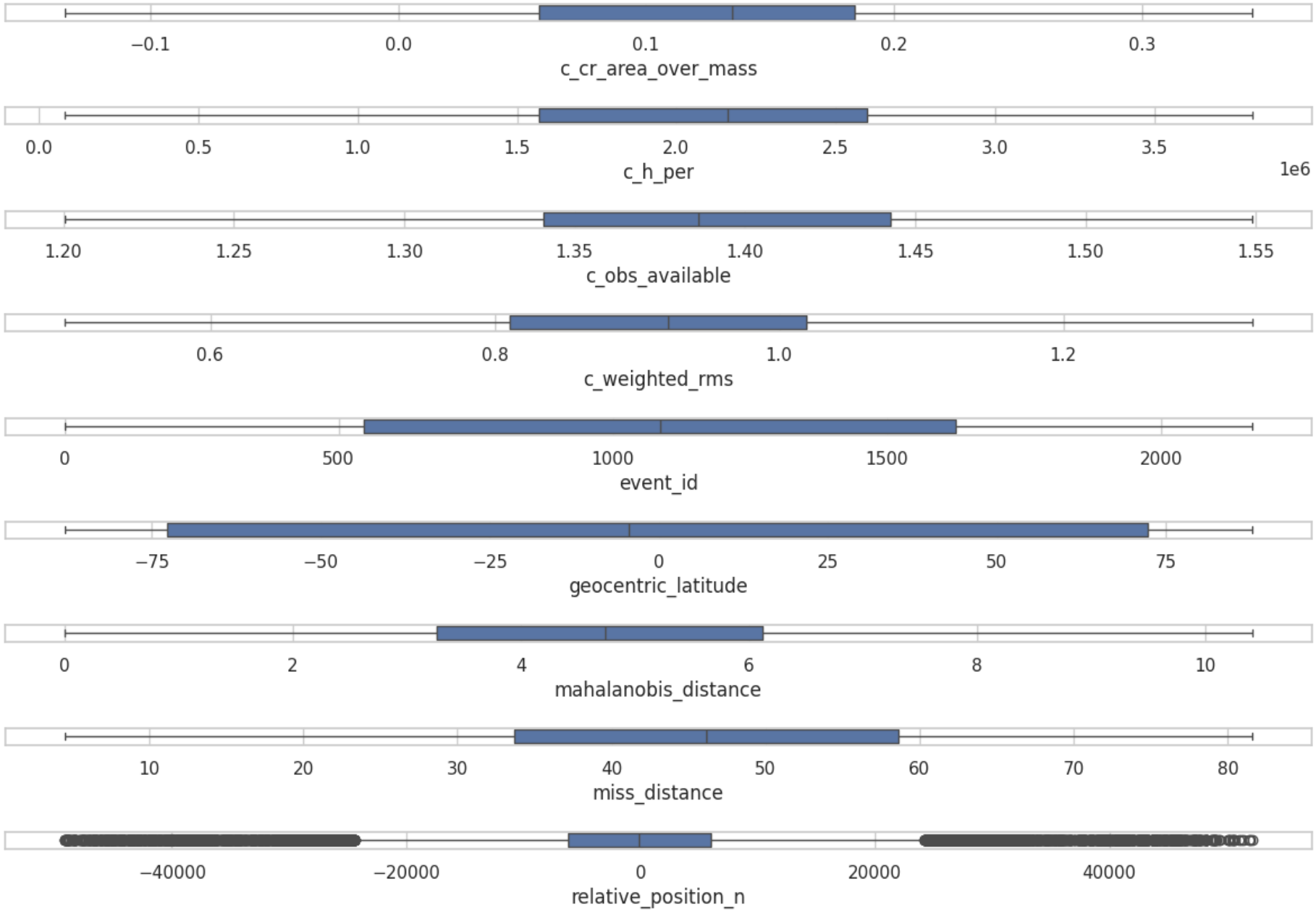
```
with open('../configs.yml', 'r') as f:
    configs = yaml.safe_load(f)

def cap_outliers(df, outlier_summary, iqr_factor):
    for col in df.select_dtypes(include=np.number).columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
```

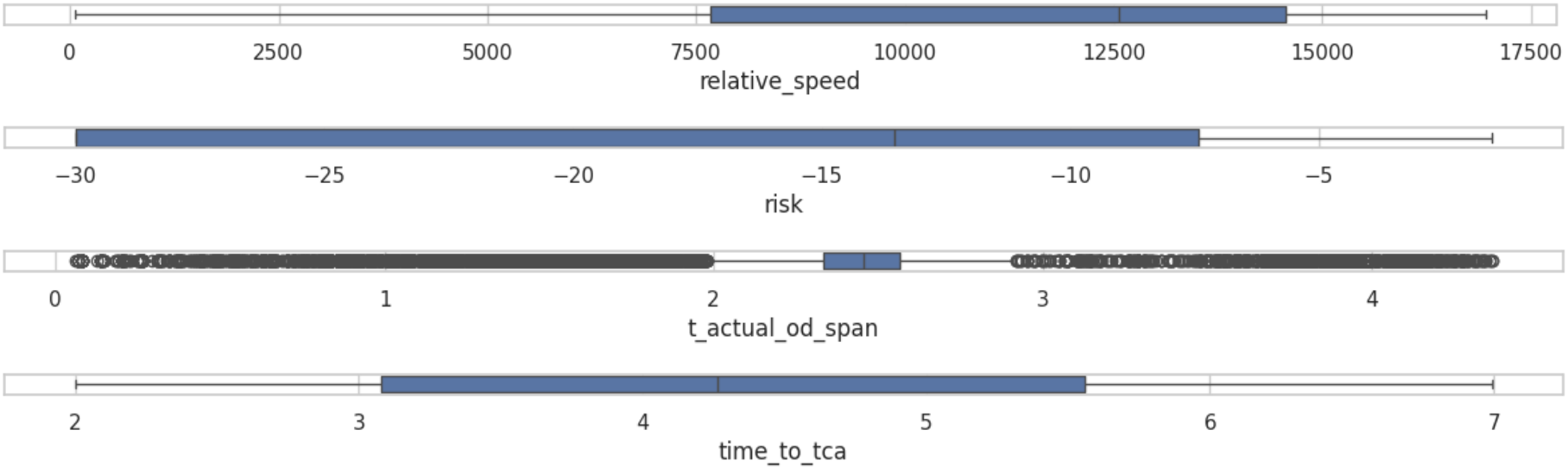
```
        lower_bound = Q1 - (iqr_factor * IQR)
        upper_bound = Q3 + (iqr_factor * IQR)

        outliers_mask = (df[col] < lower_bound) | (df[col] > upper_bound)
        outliers_count_before = df[outliers_mask].shape[0]

        percent_outliers = (outliers_count_before / len(df)) * 100
        if percent_outliers > 0:
            outlier_summary[col] = {
                'Count': outliers_count_before,
                '%': percent_outliers
            }

iqr_factor = configs['iqr_factor']
outlier_summary = {}
cap_outliers(df, outlier_summary, iqr_factor)

summary_df = pd.DataFrame.from_dict(outlier_summary, orient='index')
summary_df = summary_df.sort_values(by='%', ascending=False)

with pd.option_context('display.max_rows', None, 'display.float_format', '{:,.3f}'.format):
    display(summary_df)
```

	Count	%
t_actual_od_span	5704	23.297
relative_position_n	2766	11.297

In [281…

```
for col in df.select_dtypes(include=np.number).columns:
    plt.figure(figsize=(15, 0.2))
    sns.boxplot(x=df[col])
    plt.show()
```

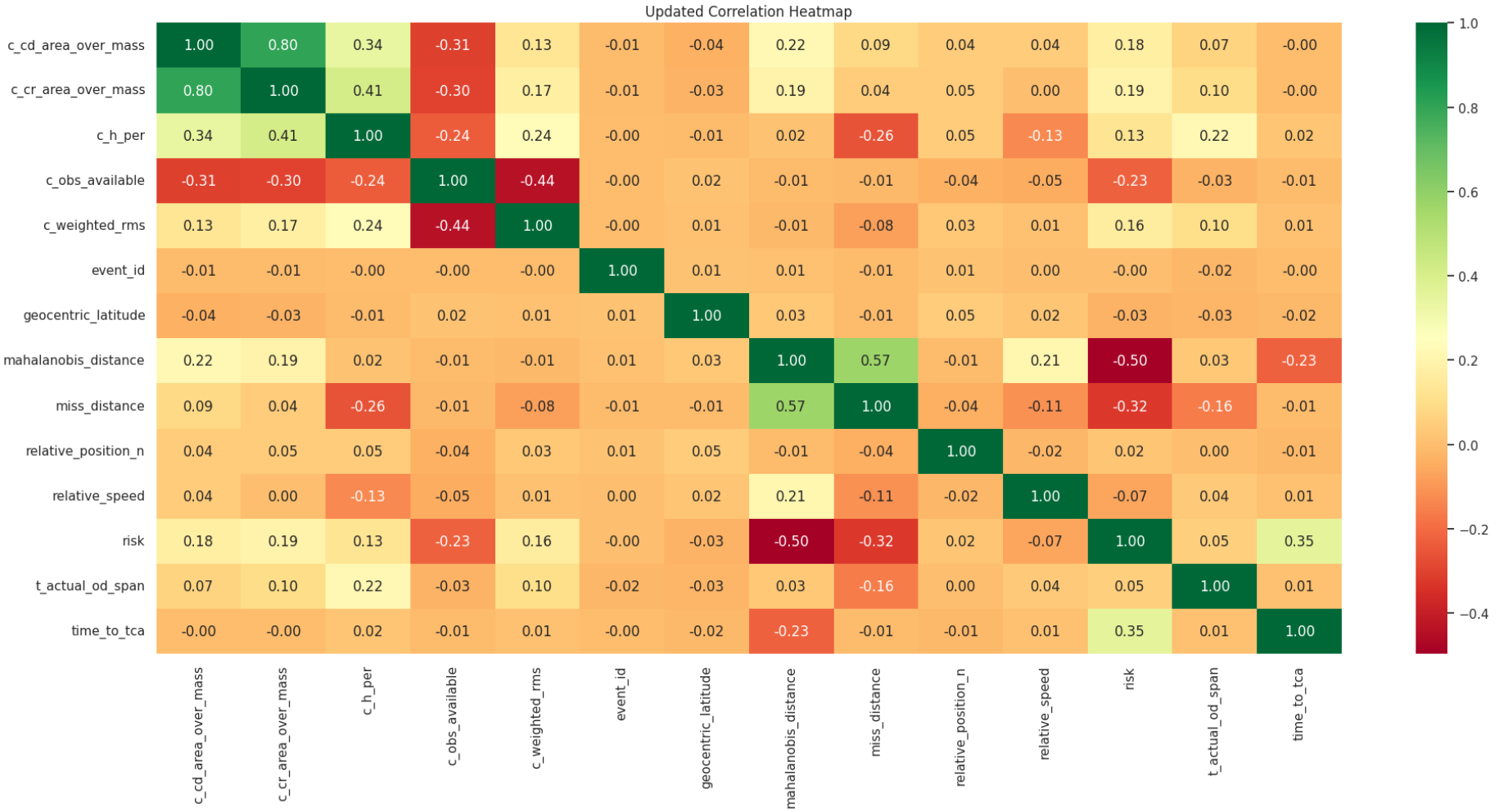

−0.1
0.0
0.1
0.2
0.3
c_cr_area_over_mass
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
c_h_per
1e6
1.20
1.25
1.30
1.35
1.40
1.45
1.50
1.55
c_obs_available
0.6
0.8
1.0
1.2
c_weighted_rms
0
500
1000
1500
2000
event_id
−75
−50
−25
0
25
50
75
geocentric_latitude
0
2
4
6
8
10
mahalanobis_distance
10
20
30
40
50
60
70
80
miss_distance
−40000
−20000
0
20000
40000
relative_position_n



Updated Correlation Heatmap

In [281…

```
df = df.sort_index(axis=1)
plt.figure(figsize=(20, 10))
sns.heatmap(df.corr(), annot=True, fmt='.2f', cmap='RdYlGn')
plt.title('Updated Correlation Heatmap')
plt.tight_layout()
plt.show()
```



Timeseries Visualizations

Select events with a minimum number of entries (CDMs)

In [281…

```
with open('../configs.yml', 'r') as f:
    configs = yaml.safe_load(f)

N_EVENTS_TO_PLOT = configs['n_events_to_plot']
```

```
MIN_ENTRIES_THRESHOLD = configs['min_entries_threshold']
FIRST_EVENT_TO_PLOT = configs['first_event_to_plot']

qualified_events = df['event_id'].value_counts()[df['event_id'].value_counts() >= MIN_ENTRIES_THRESHOLD].index
if len(qualified_events) > N_EVENTS_TO_PLOT:
    sample_ids = qualified_events[FIRST_EVENT_TO_PLOT:FIRST_EVENT_TO_PLOT + N_EVENTS_TO_PLOT]
else:
    sample_ids = qualified_events

df_plot = df[df['event_id'].isin(sample_ids)].copy()
```

Remove columns that do not make sense to be plotted

In [281…

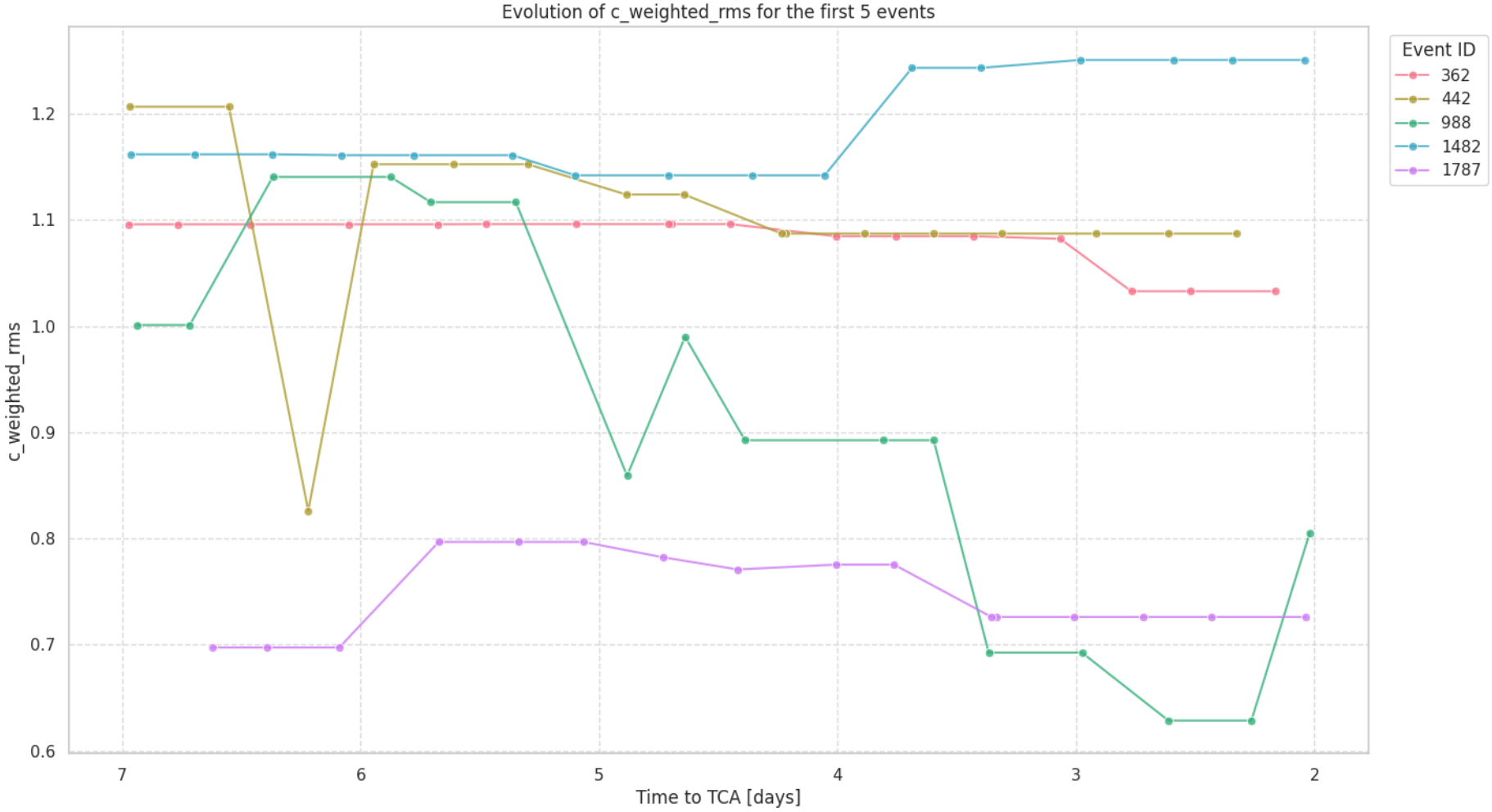
```
df_plot = df_plot.drop(
    columns=[
        'c_cd_area_over_mass',
        'c_cr_area_over_mass',
        'c_h_per',
        'c_obs_available',
        'geocentric_latitude',
        'relative_speed'
    ])
```

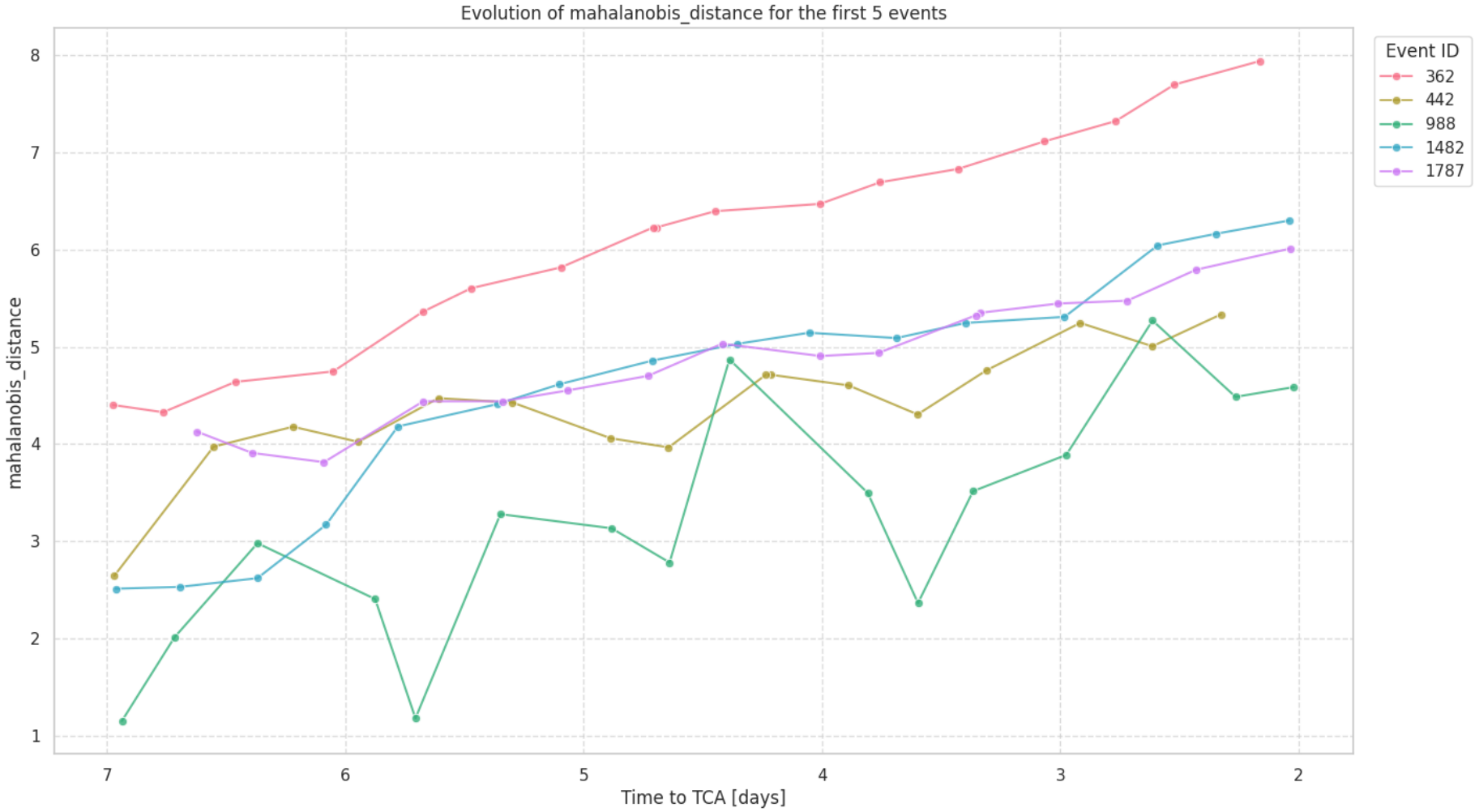
Plots the selected events for the relevant columns (not removed)

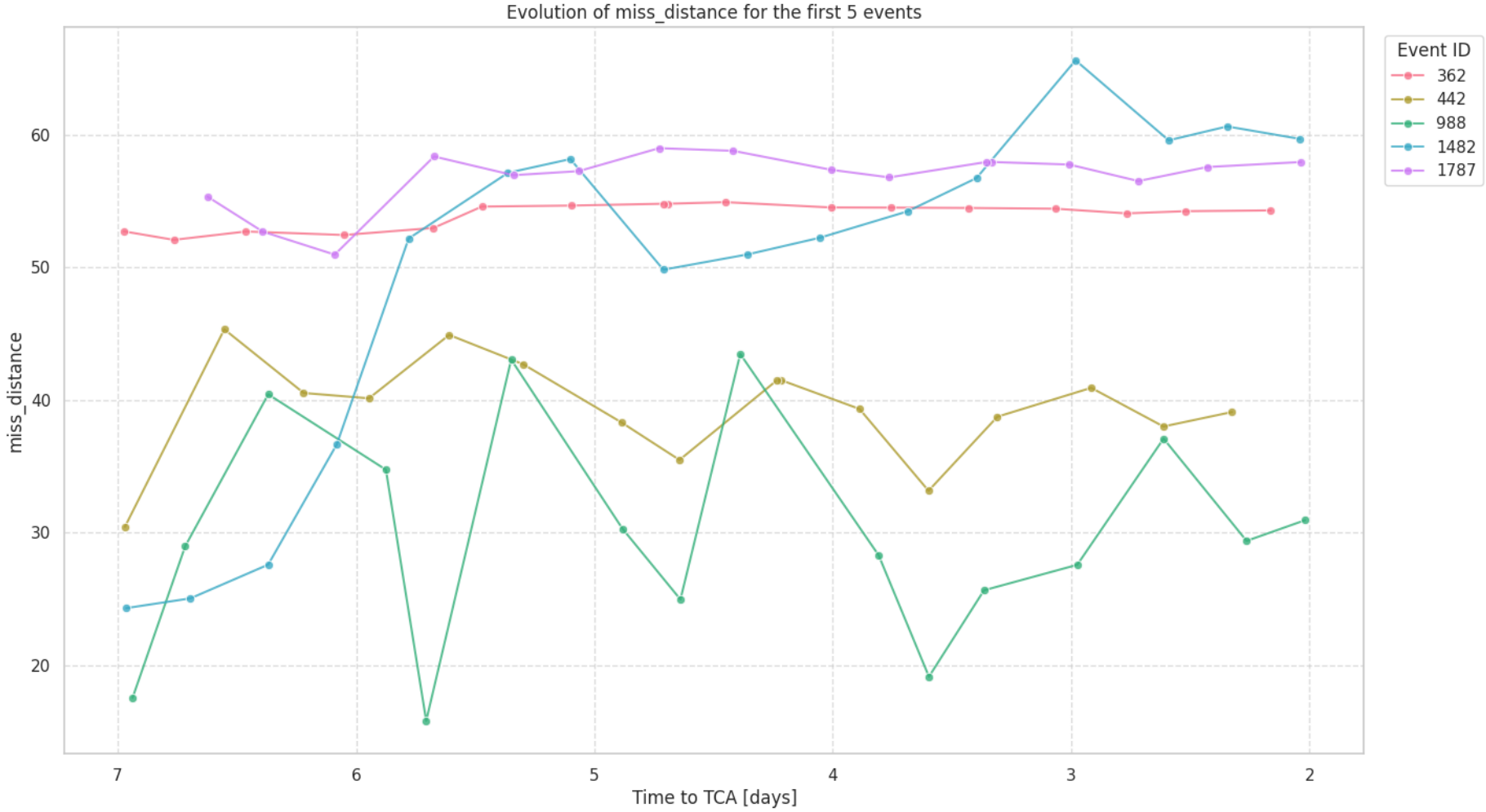
In [282…

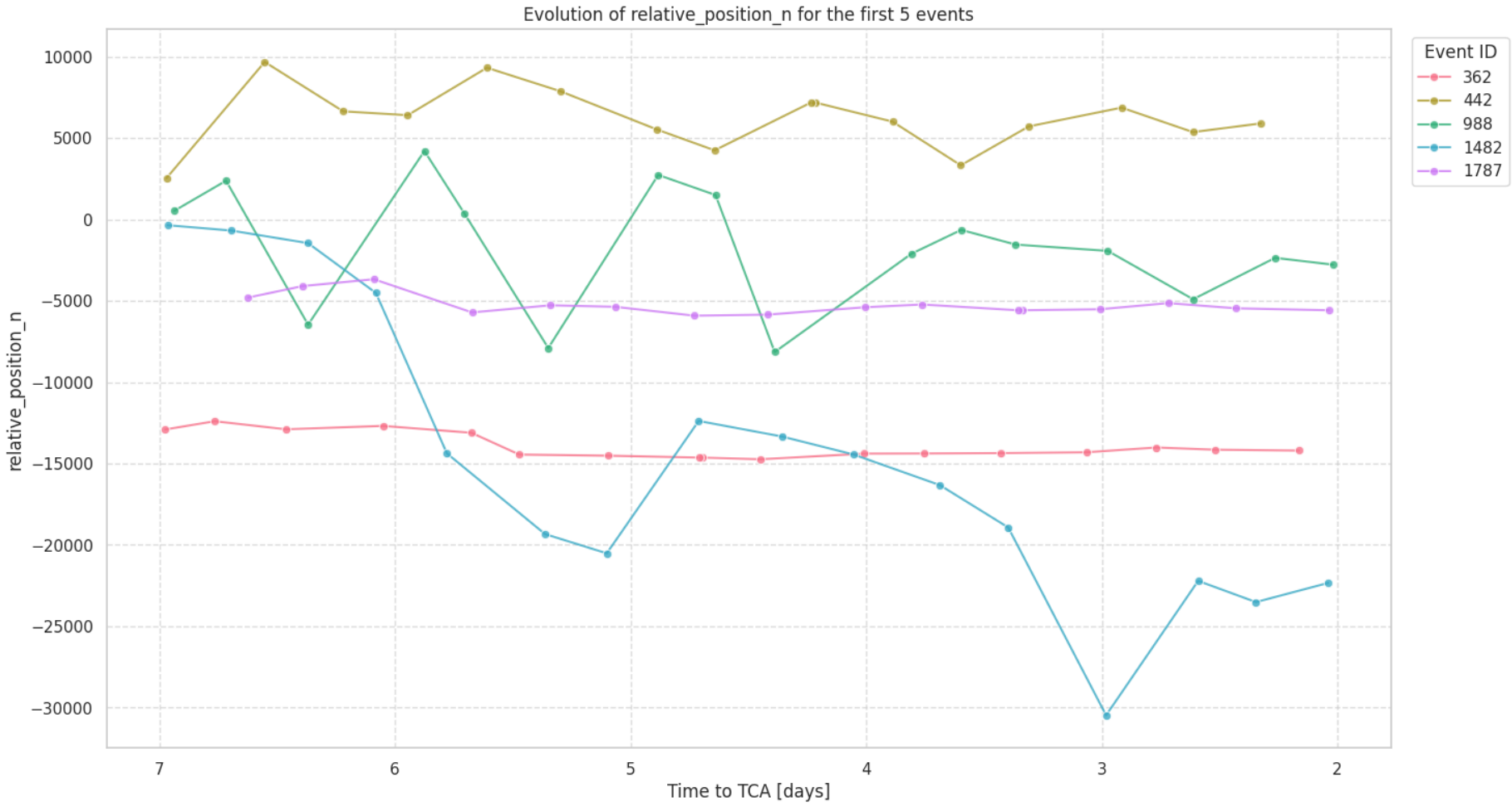
```
for y_var in df_plot.drop(columns=['event_id', 'time_to_tca']).columns:
    plt.figure(figsize=(16, 9))
    sns.lineplot(
        data=df_plot,
        x='time_to_tca',
        y=y_var,
        hue='event_id',
        marker='o',
        alpha=0.8,
        palette=sns.color_palette('husl', N_EVENTS_TO_PLOT)
    )

    plt.gca().invert_xaxis()
    plt.xlabel("Time to TCA [days]")
```

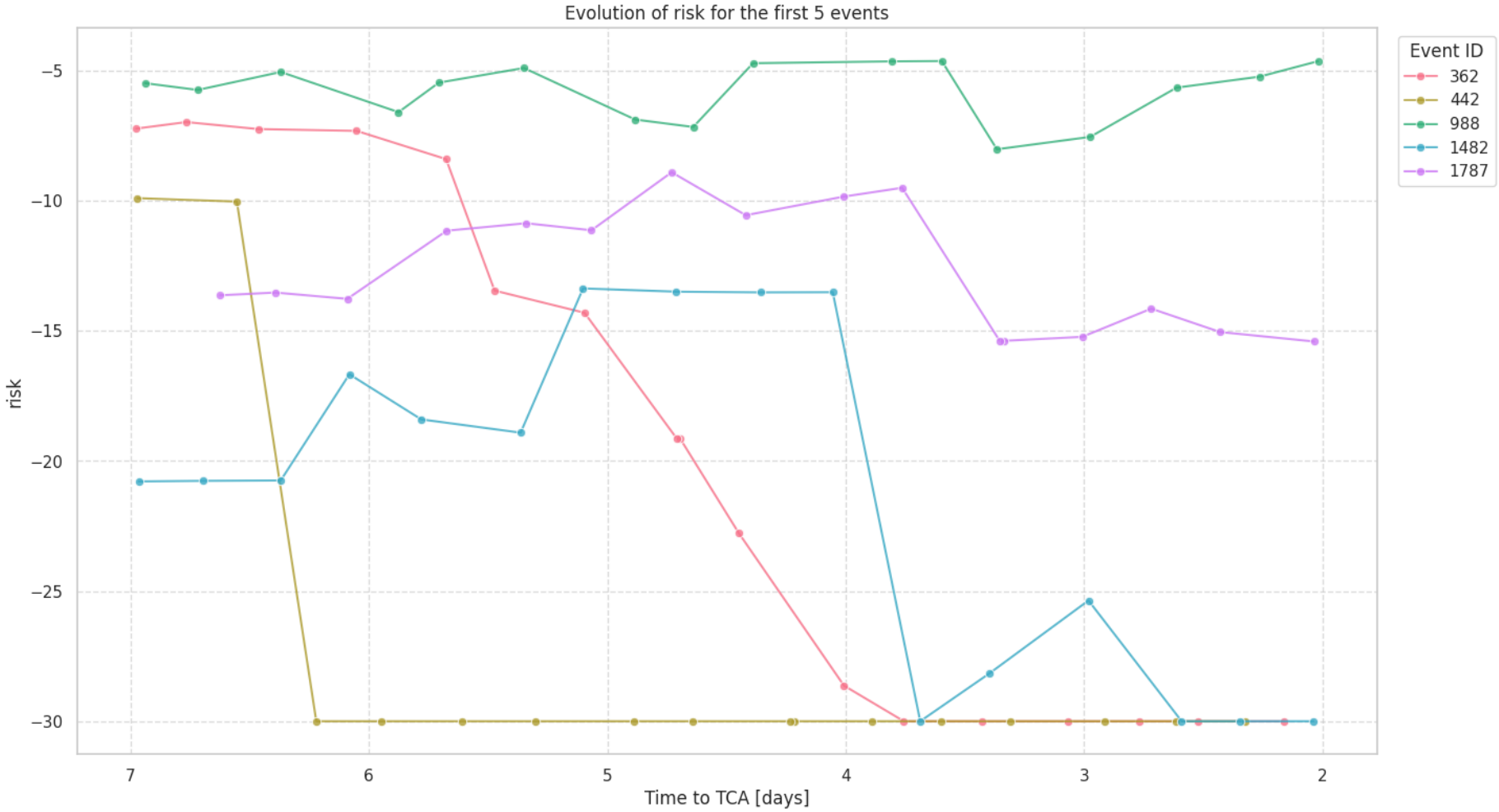
```
    plt.ylabel(y_var)
    plt.title(f"Evolution of {y_var} for the first {N_EVENTS_TO_PLOT} events")
    plt.grid(True, linestyle='--', alpha=0.7)

    plt.legend(title='Event ID', bbox_to_anchor=(1.01, 1), loc='upper left')
    plt.show()
```

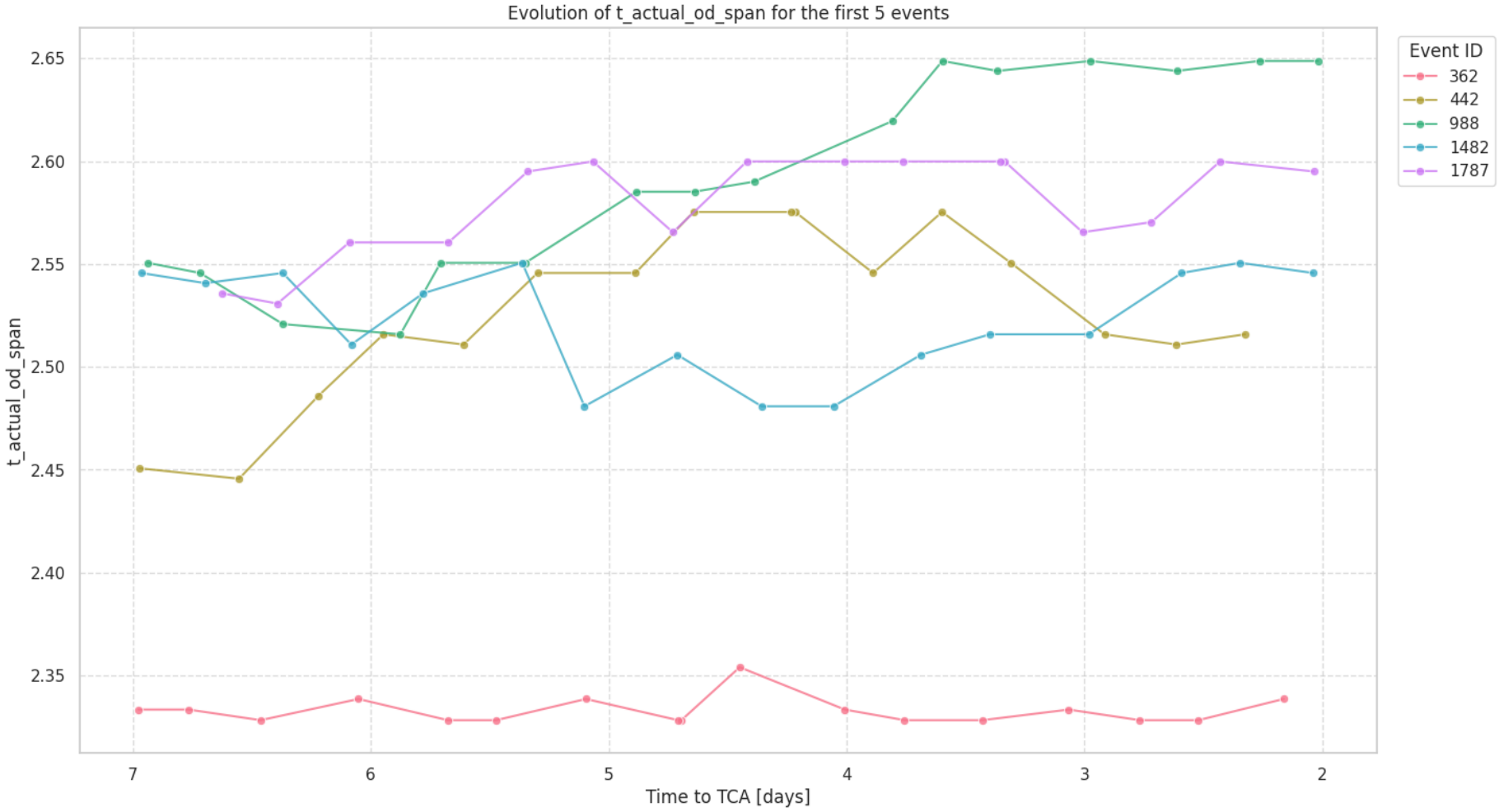



Evolution of mahalanobis_distance for the first 5 events
mahalanobis_distance
Time to TCA [days]
Event ID
362
442
988
1482
1787
8
7
6
5
4
3
2
1
7
6
5
4
3
2


Evolution of miss_distance for the first 5 events
Event ID
362
442
988
1482
1787
miss_distance
60
50
40
30
20
7
6
5
4
3
2
Time to TCA [days]


Evolution of relative_position_n for the first 5 events
relative_position_n
10000
5000
0
−5000
−10000
−15000
−20000
−25000
−30000
7
6
5
4
3
2
Time to TCA [days]
Event ID
362
442
988
1482
1787


Evolution of risk for the first 5 events
risk
−5
−10
−15
−20
−25
−30
7
6
5
4
3
2
Time to TCA [days]
Event ID
362
442
988
1482
1787



Save Parsed Dataset

In [282…

```
!rm ../data/processed_data.csv
df.to_csv('../data/processed_data.csv', index=False)
!tar -czvf ../data/processed_data.tar.gz ../data/processed_data.csv
```

```
tar: Removing leading `../' from member names
../data/processed_data.csv
```