In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score

import statsmodels.api as sm
import statsmodels.formula.api as smf

import scipy.stats as stats

import warnings
warnings.filterwarnings("ignore")



drop_cols = [
    "TWD","COG","TWA", "TimeUTC", "SecondsSince1970", "ISODateTimeUTC",
    "Lat", "LatBow", "LatCenter", "LatStern", "Lon", "LonBow", "LonCenter", "LonStern",
    "Leg", "Log", "LogAlongCourse", "MagneticVariation", "Rank", "TimeLocal",
    "DistanceToLeader", "interval_id", "boat_name", "interval_duration",
    "Heel", "Heel_Lwd", "Line_R", "Line_L", "BelowLineCalc", "VMC","XTE","VMG","gain_forward","gain_lateral", "gain_vmg","Total_lines", "LoadCell_1","LoadCell_2","LoadCell_3","LoadCell_4","LoadCell_5","L
]

def show_correlation_matrix(dataframe, taille_figure=(6, 4), cmap="coolwarm"):
    plt.figure(figsize=taille_figure)
    sns.heatmap(dataframe.corr(), cmap=cmap, center=0)
    plt.title("Correlation matrix")
    plt.show()

def show_target_correlation(df, variable="SOG", taille_figure=(6, 4)):
    corr_with_vmg = df.corr()[variable].sort_values(ascending=False)
    #print(f"Correlation with {variable} :")
    # print(corr_with_vmg)

    plt.figure(figsize=taille_figure)
    sns.heatmap(corr_with_vmg.to_frame(), annot=True, cmap="coolwarm")
    plt.title(f"Correlation with {variable}")
    plt.show()

def compute_anova(df, target="SOG"):
    # Make a copy of the DataFrame to avoid modifying the original
    df_copy = df.copy()
    cols = df_copy.columns.drop(target)
    df_no_missing = df_copy.dropna(subset=cols)
    if df_no_missing.empty:
        print("⚠️ Pas assez de données non-nulles pour faire une ANOVA.")
        return pd.DataFrame()  # retourne un DataFrame vide ou None
    formula_no_missing = f"{target} ~ " + " + ".join(
        list(df_no_missing.columns.drop(target)))
    model_no_missing = smf.ols(formula=formula_no_missing, data=df_no_missing).fit()
    anova_no_missing = sm.stats.anova_lm(model_no_missing, typ=2)
    return anova_no_missing.sort_values("F", ascending=False)

def linear_regression(df, target="SOG", degree=1, top_coefs=30):
    df_copy = df.copy()
    cols = df_copy.columns.drop(target)
```

```
    df_clean = df_copy.dropna(subset=cols)
    if df_clean[target].isna().any():
        df_clean = df_clean.dropna(subset=[target])

    X = df_clean[cols]
    y = df_clean[target]

    poly = PolynomialFeatures(degree=degree, include_bias=False)
    X_poly = poly.fit_transform(X)
    feature_names = poly.get_feature_names_out(X.columns)

    model = make_pipeline(
        StandardScaler(),
        LinearRegression()
    )

    model.fit(X_poly, y)
    y_pred = model.predict(X_poly)

    # Affichage de la performance
    print(f"R²: {r2_score(y, y_pred):.3f}")
    print(f"Used samples: {len(X)}")

    # Coefficients
    coefs = model.named_steps['linearregression'].coef_
    intercept = model.named_steps['linearregression'].intercept_

    coef_df = pd.DataFrame({
        'feature': feature_names,
        'coefficient': coefs
    }).sort_values('coefficient', key=abs, ascending=False)

    # Formule affichée
    print("\n Formula :")
    terms = [f"{coef:.3f} * {feat}" for feat, coef in zip(feature_names, coefs)]
    equation = " +\n  ".join(terms)
    print(f"{target} ≈ {intercept:.3f} +\n  {equation}")

    return coef_df.head(top_coefs)
"""from sklearn.linear_model import RidgeCV

def linear_regression(df, target="SOG", top_coefs=30):
    df_copy = df.copy()
    cols = df_copy.columns.drop(target)

    # Drop rows with missing values
    df_clean = df_copy.dropna(subset=cols)
    if df_clean[target].isna().any():
        df_clean = df_clean.dropna(subset=[target])

    X = df_clean[cols]
    y = df_clean[target]

    # Ridge regression with standard scaling
    model = make_pipeline(
        StandardScaler(),
        RidgeCV(alphas=[0.1, 1.0, 10.0])
    )

    model.fit(X, y)
    y_pred = model.predict(X)
```

```
    # Performance
    print(f"R²: {r2_score(y, y_pred):.3f}")
    print(f"Used samples: {len(X)}")

    # Extract coefficients
    coefs = model.named_steps['ridgecv'].coef_
    intercept = model.named_steps['ridgecv'].intercept_

    coef_df = pd.DataFrame({
        'feature': X.columns,
        'coefficient': coefs
    }).sort_values('coefficient', key=abs, ascending=False)

    # Display formula
    print("\nFormula:")
    terms = [f"{coef:.3f} * {feat}" for feat, coef in zip(X.columns, coefs)]
    equation = " +\n  ".join(terms)
    print(f"{target} ≈ {intercept:.3f} +\n  {equation}")

    return coef_df.head(top_coefs)"""




def t_test(df1, df2, target="SOG"):
    t_stat, p_value = stats.ttest_ind(df1[target].dropna(), df2[target].dropna())
    print(f"T-statistic: {t_stat:.3f}, p-value: {p_value:.15f}")

    # If p-value is less than 0.05, the difference is statistically significant
    if p_value < 0.05:
        print("The difference is statistically significant, keeping data split.")
    else:
        print("The difference is not statistically significant, keeping data combined.")

"""def full_analysis(df_numeric, target_variable="SOG"):
    # Display the correlation matrix
    # print("Correlation matrix:")
    #show_correlation_matrix(df_numeric)

    # Display correlation with the target variable
    print(f"\nCorrelation with {target_variable}:")
    show_target_correlation(df_numeric, variable=target_variable)

    # Compute and display ANOVA results
    print("\nANOVA:")
    anova_results = compute_anova(df_numeric)
    display(anova_results)

    # Apply and display polynomial regression results
    print("\nPolynomial fit:")
    regression_results = linear_regression(df_numeric)
    display(regression_results)"""

def full_analysis(df_numeric, target_variable="SOG"):
    # Drop 'Line_Side' column only if it's fully NaN
    if "Side_lines" in df_numeric.columns and df_numeric["Side_lines"].isna().all():
        df_numeric = df_numeric.drop(columns=["Side_lines"])
        print("Dropped column 'Side_lines' (all values were NaN).")

    # Display correlation with the target variable
    print(f"\nCorrelation with {target_variable}:")
    show_target_correlation(df_numeric, variable=target_variable)
```

```
    # Compute and display ANOVA results
    print("\nANOVA:")
    anova_results = compute_anova(df_numeric)
    display(anova_results)

    # Apply and display polynomial regression results
    print("\nPolynomial fit:")
    regression_results = linear_regression(df_numeric)
    display(regression_results)
```

In [2]:
```
df = pd.read_csv("all_data_enriched.csv")
df = df[df["boat_name"] == "SenseBoard"].copy()
load_cell_cols = ["LoadCell_1", "LoadCell_2", "LoadCell_3", "LoadCell_4", "LoadCell_5", "LoadCell_6"]
df = df.dropna(subset=load_cell_cols, how='all')

df.sample(10)
```

Out[2]:

	ISODateTimeUTC	SecondsSince1970	Heel_Abs	Heel_Lwd	Lat	LatBow	LatCenter	LatStern	Leg	Line_C	...	M_tot_X	M_tot_Y	M_front_X	M_front_Y	M_back_X	M_back_Y	P_front
36287	2025-06-07 13:09:58.248000+00:00	1.749302e+09	48.1	48.1	43.514553	43.514552	43.514557	43.514562	1.0	103.800	...	7560.819016	-1348.235683	5697.124723	957.552696	1863.694293	-111.651026	35.8257
34312	2025-06-07 13:03:37.555000+00:00	1.749301e+09	35.7	35.7	43.513202	43.513203	43.513198	43.513193	1.0	87.800	...	-1536.925144	412.234612	-843.050543	-472.941431	-693.874601	-306.079927	-20.8303
32316	2025-06-07 12:50:02.260000+00:00	1.749301e+09	53.5	53.5	43.513747	43.513748	43.513743	43.513738	1.0	6.200	...	-3545.401282	-1451.064160	-2395.969411	354.987107	-1149.431871	26.534840	12.3893
40430	2025-06-07 13:27:45.556000+00:00	1.749303e+09	52.2	52.2	43.515030	43.515029	43.515033	43.515038	NaN	98.200	...	6680.509003	-152.678581	4998.002285	770.359098	1682.506718	-427.271544	27.5057
28174	2025-06-07 12:27:07.660000+00:00	1.749299e+09	55.3	55.3	43.517807	43.517805	43.517810	43.517815	1.0	6.700	...	8978.596589	-1794.499264	7339.119557	1877.854535	1639.477032	-465.076404	50.8586
30433	2025-06-07 12:39:33.753000+00:00	1.749300e+09	52.2	52.2	43.514360	43.514362	43.514356	43.514351	1.0	6.900	...	148.987441	-1863.662142	-3.854552	-82.174104	152.841993	114.018203	-3.0455
31615	2025-06-07 12:47:16.857000+00:00	1.749300e+09	57.1	57.1	43.517535	43.517534	43.517538	43.517543	1.0	9.100	...	8204.781547	-466.777993	6382.367289	1190.516415	1822.414257	-471.434912	42.6757
28491	2025-06-07 12:27:23.555000+00:00	1.749299e+09	59.0	59.0	43.516595	43.516594	43.516598	43.516603	1.0	10.100	...	7161.413238	-950.740207	5967.706596	1391.997409	1193.706642	-313.792409	44.8516
14600	2025-06-06 14:49:36.858000+00:00	1.749221e+09	55.5	55.5	43.524970	43.524969	43.524973	43.524976	1.0	102.900	...	5517.857047	4850.981308	3666.496445	164.849989	1851.360602	-377.020210	11.0196
38072	2025-06-07 13:17:48.858000+00:00	1.749302e+09	53.8	53.8	43.516248	43.516247	43.516252	43.516257	1.0	116.322	...	6920.899903	555.143555	5153.350994	820.485072	1767.548909	-424.969781	31.2867

10 rows × 67 columns

I. All together

In [3]:
```
# Select numeric columns
df_numeric = df.select_dtypes(include=["float64", "int64"]).copy()

# Drop specified columns
df_numeric.drop(columns=[c for c in drop_cols if c in df_numeric.columns], inplace=True)

# Drop rows where 'SOG' is missing
df_numeric.dropna(subset=["SOG"], inplace=True)
```

```
# Print summary
print(f"Variables utilisées: {df_numeric.columns.tolist()}")
print(f"Number of rows after filtering: {len(df_numeric)}")
```

```
Variables utilisées: ['Heel_Abs', 'Line_C', 'ROT', 'Side_lines', 'Trim', 'TWA_Abs', 'TWS', 'SOG', 'boat_weight', 'F_front', 'F_back', 'M_tot_X', 'M_tot_Y', 'M_front_X', 'M_front_Y', 'M_back_X', 'M_back_
Y', 'P_front_X', 'P_front_Y', 'P_back_X', 'P_back_Y']
Number of rows after filtering: 9147
```

In [4]:
```
df_numeric.sample(20)
```

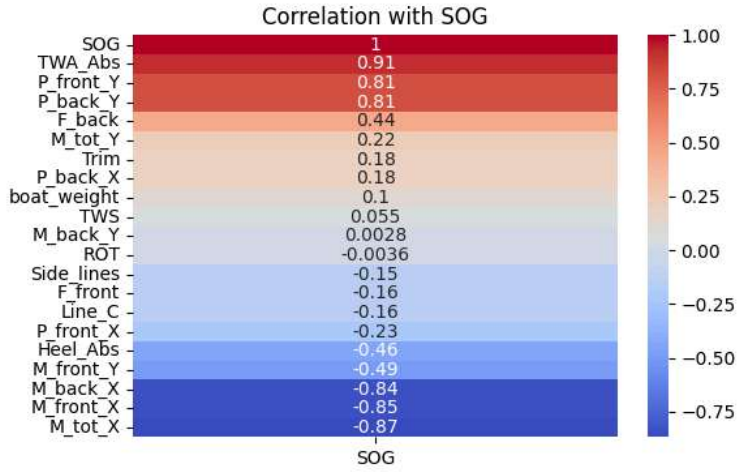
Out[4]:

	Heel_Abs	Line_C	ROT	Side_lines	Trim	TWA_Abs	TWS	SOG	boat_weight	F_front	...	M_tot_X	M_tot_Y	M_front_X	M_front_Y	M_back_X	M_back_Y	P_front_X	P_front_Y	P_b
16946	56.2	117.100	-4.040	14.670	10.5	44.400	8.800	22.1	109.09	17439.184350	...	7421.231875	132.454673	5622.428310	944.140764	1798.803565	-553.153350	54.139044	-322.402023	-30.4
19768	53.4	116.900	23.404	11.004	5.5	48.864	11.164	23.0	109.09	13050.070330	...	5335.897464	6463.259590	3626.365827	-322.369204	1709.531637	-504.220468	-24.702488	-277.880941	-14.5
29793	60.3	6.200	3.809	NaN	7.6	34.268	8.734	23.3	102.89	20225.222799	...	6168.999734	-436.555604	4809.681286	268.373786	1359.318447	-411.096254	13.269262	-237.806097	-19.0
13499	53.0	88.700	-6.316	8.616	14.5	152.451	9.949	26.7	109.09	10180.160673	...	-4088.213670	4726.990750	-3942.613555	-1055.928621	-145.600115	-348.227950	-103.724161	387.284021	-11.9
15830	59.0	80.900	11.765	16.200	14.4	153.546	9.637	28.8	109.09	16005.597222	...	-5874.565893	5804.096909	-4917.292846	-762.141799	-957.273046	-390.918777	-47.617205	307.223328	-10.4
15200	58.0	NaN	-17.273	14.682	7.2	43.028	10.400	22.7	109.09	11083.220112	...	4807.471264	4951.682948	3259.837357	-231.806184	1547.633907	-320.139321	-20.915057	-294.123668	-11.6
37923	51.4	128.100	-20.000	16.611	0.0	36.271	11.000	23.3	109.09	33469.794240	...	8930.267124	-2427.449631	6726.222621	1685.668338	2204.044502	-657.053093	50.363869	-200.963967	-24.9
14116	66.4	116.473	10.000	13.700	-0.6	44.897	11.100	21.7	109.09	17383.234772	...	5528.634488	6028.667789	3787.003504	242.976676	1741.630984	-298.199639	13.977645	-217.853786	-8.4
36957	41.6	84.800	-21.783	11.700	8.4	150.354	9.429	28.6	109.09	26490.907490	...	-4952.665392	764.635173	-2604.080280	403.424104	-2348.585112	-809.641050	15.228776	98.300909	-24.6
34450	45.7	70.500	-5.263	5.800	6.2	142.517	9.800	26.8	109.09	22087.062232	...	-1215.023725	42.502464	-1104.570348	-142.967215	-110.453377	-237.300780	-6.472894	50.009835	-9.2
37873	44.3	108.400	-0.961	13.300	8.8	35.686	10.986	22.7	109.09	30027.480434	...	7647.233635	-1070.262676	5661.423406	1097.902235	1985.810229	-562.491558	36.563249	-188.541407	-20.1
11729	66.0	120.800	-21.212	12.300	6.4	48.785	7.807	21.4	109.09	19773.202181	...	5566.458000	4289.803414	4289.990000	601.424158	1276.468000	-503.298176	30.416123	-216.959800	-15.3
37256	34.7	83.900	-20.000	13.342	8.4	140.307	11.500	27.8	109.09	28338.194325	...	-3538.835183	975.433943	-2169.290707	259.524181	-1369.544476	-511.115714	9.158106	76.550068	-14.8
38734	49.1	103.900	-4.255	15.127	4.2	140.725	10.085	29.9	109.09	22387.289725	...	-3670.439729	1393.441257	-2371.895545	362.689257	-1298.544183	-574.037706	16.200677	105.948311	-19.7
27510	47.8	85.000	4.000	NaN	7.0	153.024	9.500	24.6	102.89	23883.375471	...	-2428.424705	-256.097736	-1718.790379	238.567253	-709.634327	235.162045	9.988842	71.965974	10.2
18794	44.5	81.900	-7.000	11.300	9.2	147.529	9.786	28.1	109.09	16121.684318	...	-4989.210763	6506.474756	-3998.984001	-833.714811	-990.226762	-685.253798	-51.713878	248.050013	-16.7
12765	51.4	117.641	-10.891	8.072	6.0	49.800	8.900	22.0	109.09	14512.792485	...	4550.704097	4477.638267	3394.192176	-9.161429	1156.511921	-498.612567	-0.631266	-233.875884	-16.7
18168	45.4	108.200	-12.745	7.000	10.8	51.771	8.276	21.0	109.09	10965.310146	...	4659.960956	6049.556208	3202.842396	-455.287768	1457.118561	-416.373991	-41.520738	-292.088628	-13.2
15449	54.5	104.300	8.081	9.100	8.5	47.871	9.500	22.8	109.09	12478.754635	...	5014.674255	5081.714690	3232.877728	-276.284928	1781.796527	-293.604634	-22.140425	-259.070542	-10.0
36541	54.3	75.700	8.823	6.300	10.3	146.840	9.369	26.9	109.09	24721.372201	...	-1926.258419	-687.726508	-1012.149972	117.125170	-914.108447	-491.827001	4.737810	40.942305	-18.4

20 rows × 21 columns

In [5]:
```
full_analysis(df_numeric)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	3424.778775	1.0	4772.253669	0.000000e+00
TWA_Abs	517.306943	1.0	720.840708	1.839368e-150
TWS	478.615532	1.0	666.926210	6.753870e-140
M_tot_X	165.563582	1.0	230.704364	3.324978e-51
Trim	138.717969	1.0	193.296379	2.619091e-43
M_front_Y	94.342767	1.0	131.461810	3.889435e-30
M_back_Y	90.941252	1.0	126.721973	4.038607e-29
F_back	85.026643	1.0	118.480269	2.376568e-27
M_tot_Y	76.562295	1.0	106.685634	8.209234e-25
P_front_X	72.828826	1.0	101.483236	1.086863e-23
F_front	53.004331	1.0	73.858818	1.045350e-17
P_back_X	48.914515	1.0	68.159869	1.818916e-16
P_back_Y	46.027897	1.0	64.137515	1.370604e-15
Line_C	33.573402	1.0	46.782815	8.675374e-12
P_front_Y	24.490712	1.0	34.126552	5.421038e-09
M_back_X	14.204457	1.0	19.793182	8.776871e-06
M_front_X	9.003906	1.0	12.546481	3.998148e-04
ROT	1.135283	1.0	1.581959	2.085252e-01
Side_lines	0.555401	1.0	0.773922	3.790398e-01
Heel_Abs	0.265792	1.0	0.370367	5.428260e-01
Residual	4537.662432	6323.0	NaN	NaN

```
Polynomial fit:
R²: 0.904
Used samples: 6341

 Formula :
SOG ≈ 24.194 +
  0.008 * Heel_Abs +
  0.129 * Line_C +
  0.014 * ROT +
  -0.011 * Side_lines +
  -0.162 * Trim +
  1.497 * TWA_Abs +
  0.296 * TWS +
  3696238405.108 * boat_weight +
  2571836396689.119 * F_front +
  -1709453049098.549 * F_back +
  7014256530337.562 * M_tot_X +
  2536683781486.908 * M_tot_Y +
  -5444125146572.549 * M_front_X +
  -944758566663.617 * M_front_Y +
  -1725842801269.668 * M_back_X +
  -286750623056.243 * M_back_Y +
  -0.229 * P_front_X +
  0.182 * P_front_Y +
  0.442 * P_back_X +
  -0.465 * P_back_Y
```

	feature	coefficient
10	M_tot_X	7.014257e+12
12	M_front_X	-5.444125e+12
8	F_front	2.571836e+12
11	M_tot_Y	2.536684e+12
14	M_back_X	-1.725843e+12
9	F_back	-1.709453e+12
13	M_front_Y	-9.447586e+11
15	M_back_Y	-2.867506e+11
7	boat_weight	3.696238e+09
5	TWA_Abs	1.496994e+00
19	P_back_Y	-4.649710e-01
18	P_back_X	4.420223e-01
6	TWS	2.964625e-01
16	P_front_X	-2.285015e-01
17	P_front_Y	1.818830e-01
4	Trim	-1.622211e-01
1	Line_C	1.290753e-01
2	ROT	1.374633e-02
3	Side_lines	-1.073432e-02
0	Heel_Abs	7.721421e-03

II Upwind:

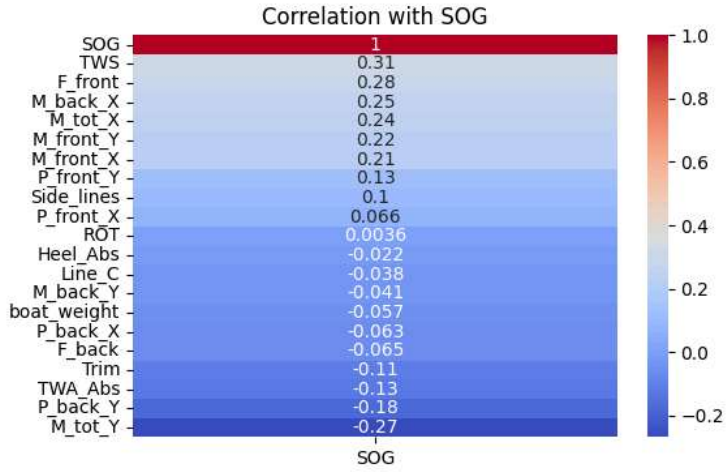
II.1. All upwind data

In [6]:
```
upwind_data = df[df['TWA'] >= 0]
df_numeric_upwind = upwind_data.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_upwind.drop(columns=[c for c in drop_cols if c in df_numeric_upwind.columns], inplace=True)
df_numeric_upwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_upwind)}")
```

```
Number of rows after filtering: 5895
```

In [7]:
```
full_analysis(df_numeric_upwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	1710.353668	1.0	3520.799492	0.000000e+00
TWS	205.591630	1.0	423.214753	1.744502e-89
M_back_X	56.199649	1.0	115.688175	1.278987e-26
M_front_X	43.804875	1.0	90.173270	3.634940e-21
Trim	34.073759	1.0	70.141559	7.538830e-17
Line_C	33.357976	1.0	68.668105	1.571179e-16
TWA_Abs	30.342300	1.0	62.460271	3.485133e-15
Side_lines	29.535584	1.0	60.799629	7.997483e-15
M_tot_X	29.384171	1.0	60.487943	9.347431e-15
F_front	25.199818	1.0	51.874363	7.031897e-13
M_front_Y	22.213919	1.0	45.727825	1.555128e-11
M_back_Y	21.833952	1.0	44.945655	2.308176e-11
F_back	16.615231	1.0	34.202808	5.361568e-09
M_tot_Y	14.276722	1.0	29.388936	6.269892e-08
P_front_X	12.173286	1.0	25.058969	5.799116e-07
P_back_X	10.454245	1.0	21.520286	3.612707e-06
P_back_Y	9.935318	1.0	20.452064	6.291009e-06
Heel_Abs	9.640824	1.0	19.845842	8.623319e-06
P_front_Y	6.094720	1.0	12.546111	4.015495e-04
ROT	0.299965	1.0	0.617485	4.320299e-01
Residual	1948.486012	4011.0	NaN	NaN

```
Polynomial fit:
R²: 0.355
Used samples: 4029

 Formula :
SOG ≈ 22.156 +
  -0.054 * Heel_Abs +
  0.105 * Line_C +
  -0.009 * ROT +
  -0.114 * Side_lines +
  -0.096 * Trim +
  -0.102 * TWA_Abs +
  0.263 * TWS +
  -15519038842.733 * boat_weight +
  51063358273.344 * F_front +
  -26090996948.877 * F_back +
  -2491176881593.895 * M_tot_X +
  49178986077.095 * M_tot_Y +
  2145341491930.120 * M_front_X +
  -15279185125.780 * M_front_Y +
  540527328151.450 * M_back_X +
  -5094602637.313 * M_back_Y +
  0.227 * P_front_X +
  -0.157 * P_front_Y +
  -0.308 * P_back_X +
  0.220 * P_back_Y
```

	feature	coefficient
10	M_tot_X	-2.491177e+12
12	M_front_X	2.145341e+12
14	M_back_X	5.405273e+11
8	F_front	5.106336e+10
11	M_tot_Y	4.917899e+10
9	F_back	-2.609100e+10
7	boat_weight	-1.551904e+10
13	M_front_Y	-1.527919e+10
15	M_back_Y	-5.094603e+09
18	P_back_X	-3.078049e-01
6	TWS	2.633981e-01
16	P_front_X	2.270347e-01
19	P_back_Y	2.200639e-01
17	P_front_Y	-1.566539e-01
3	Side_lines	-1.140350e-01
1	Line_C	1.049433e-01
5	TWA_Abs	-1.023790e-01
4	Trim	-9.611959e-02
0	Heel_Abs	-5.424112e-02
2	ROT	-8.537764e-03

II.2. Upwind: Gian vs Karl

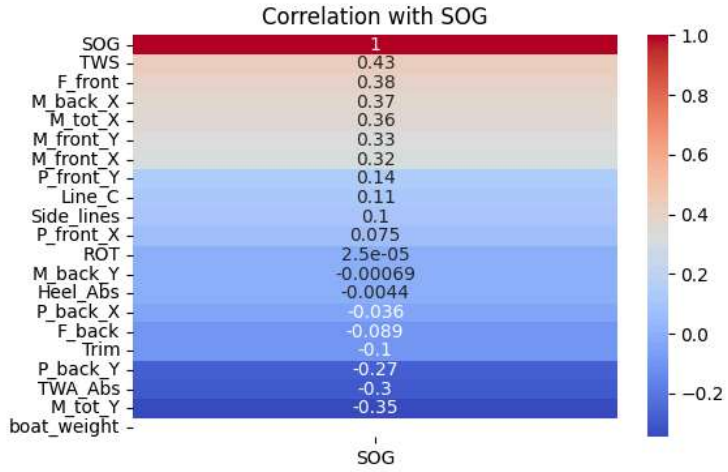
II.2.1. Upwind: Gian

In [8]:
```
gian_data_upwind = upwind_data[
    (upwind_data['boat_name'] == "Gian Stragiotti") |
    ((upwind_data['boat_name'] == "SenseBoard") & (upwind_data['opponent_name'] == "Karl Maeder"))
]
df_numeric_gian_upwind = gian_data_upwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_gian_upwind.drop(columns=[c for c in drop_cols if c in df_numeric_gian_upwind.columns], inplace=True)
df_numeric_gian_upwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_gian_upwind)}")
```

```
Number of rows after filtering: 4045
```

In [9]:
```
full_analysis(df_numeric_gian_upwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	1710.353668	1.0	3520.799492	0.000000e+00
TWS	205.591630	1.0	423.214753	1.744502e-89
M_back_X	56.199649	1.0	115.688175	1.278987e-26
M_front_X	43.804875	1.0	90.173270	3.634940e-21
Trim	34.073759	1.0	70.141559	7.538830e-17
Line_C	33.357976	1.0	68.668105	1.571179e-16
TWA_Abs	30.342300	1.0	62.460271	3.485133e-15
Side_lines	29.535584	1.0	60.799629	7.997483e-15
M_tot_X	29.384171	1.0	60.487943	9.347431e-15
F_front	25.199818	1.0	51.874363	7.031897e-13
M_front_Y	22.213919	1.0	45.727825	1.555128e-11
M_back_Y	21.833952	1.0	44.945655	2.308176e-11
F_back	16.615231	1.0	34.202808	5.361568e-09
M_tot_Y	14.276722	1.0	29.388936	6.269892e-08
P_front_X	12.173286	1.0	25.058969	5.799116e-07
P_back_X	10.454245	1.0	21.520286	3.612707e-06
P_back_Y	9.935318	1.0	20.452064	6.291009e-06
Heel_Abs	9.640824	1.0	19.845842	8.623319e-06
P_front_Y	6.094720	1.0	12.546111	4.015495e-04
ROT	0.299965	1.0	0.617485	4.320299e-01
Residual	1948.486012	4011.0	NaN	NaN

```
Polynomial fit:
R²: 0.355
Used samples: 4029

 Formula :
SOG ≈ 22.156 +
  -0.054 * Heel_Abs +
  0.105 * Line_C +
  -0.009 * ROT +
  -0.114 * Side_lines +
  -0.096 * Trim +
  -0.102 * TWA_Abs +
  0.263 * TWS +
  -15519038842.733 * boat_weight +
  51063358273.344 * F_front +
  -26090996948.877 * F_back +
  -2491176881593.895 * M_tot_X +
  49178986077.095 * M_tot_Y +
  2145341491930.120 * M_front_X +
  -15279185125.780 * M_front_Y +
  540527328151.450 * M_back_X +
  -5094602637.313 * M_back_Y +
  0.227 * P_front_X +
  -0.157 * P_front_Y +
  -0.308 * P_back_X +
  0.220 * P_back_Y
```

	feature	coefficient
10	M_tot_X	-2.491177e+12
12	M_front_X	2.145341e+12
14	M_back_X	5.405273e+11
8	F_front	5.106336e+10
11	M_tot_Y	4.917899e+10
9	F_back	-2.609100e+10
7	boat_weight	-1.551904e+10
13	M_front_Y	-1.527919e+10
15	M_back_Y	-5.094603e+09
18	P_back_X	-3.078049e-01
6	TWS	2.633981e-01
16	P_front_X	2.270347e-01
19	P_back_Y	2.200639e-01
17	P_front_Y	-1.566539e-01
3	Side_lines	-1.140350e-01
1	Line_C	1.049433e-01
5	TWA_Abs	-1.023790e-01
4	Trim	-9.611959e-02
0	Heel_Abs	-5.424112e-02
2	ROT	-8.537764e-03

II.2.2. Upwind: Karl

In [10]:
```
karl_data_upwind = upwind_data[((upwind_data['boat_name'] == "SenseBoard") & (upwind_data['opponent_name'] == "Gian Stragiotti"))
]
df_numeric_karl_upwind = karl_data_upwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_karl_upwind.drop(columns=[c for c in drop_cols if c in df_numeric_karl_upwind.columns], inplace=True)
df_numeric_karl_upwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_karl_upwind)}")
print(df_numeric_karl_upwind.shape)
print(df_numeric_karl_upwind.columns)
```

```
Number of rows after filtering: 1850
(1850, 21)
Index(['Heel_Abs', 'Line_C', 'ROT', 'Side_lines', 'Trim', 'TWA_Abs', 'TWS',
       'SOG', 'boat_weight', 'F_front', 'F_back', 'M_tot_X', 'M_tot_Y',
       'M_front_X', 'M_front_Y', 'M_back_X', 'M_back_Y', 'P_front_X',
       'P_front_Y', 'P_back_X', 'P_back_Y'],
      dtype='object')
```

In [11]:
```
df_numeric_karl_upwind.head(10)
```

Out[11]:

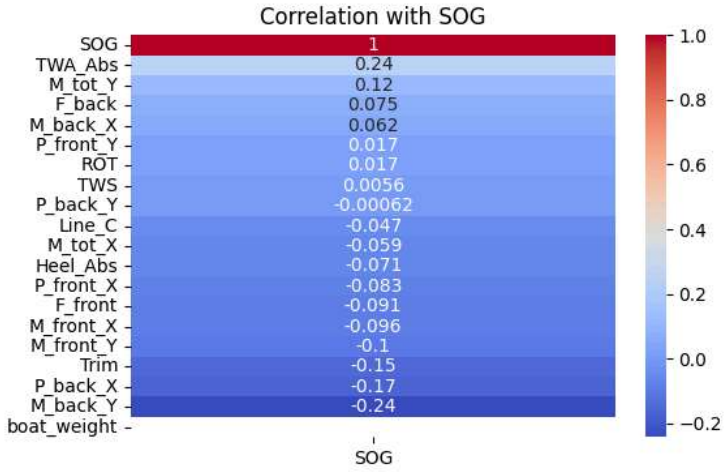
	Heel_Abs	Line_C	ROT	Side_lines	Trim	TWA_Abs	TWS	SOG	boat_weight	F_front	...	M_tot_X	M_tot_Y	M_front_X	M_front_Y	M_back_X	M_back_Y	P_front_X	P_front_Y	P_back_X
25553	59.5	93.3	11.000	NaN	7.5	41.174	9.352	21.4	102.89	27567.598622	...	6530.224065	-713.048129	5638.902083	1219.704616	891.321982	-322.846790	44.244137	-204.548178	-13.091291
25556	57.4	88.1	-14.141	NaN	12.3	42.564	9.372	21.4	102.89	30282.257552	...	7305.571614	-1763.759238	6091.881706	1425.550444	1213.689908	-429.662418	47.075435	-201.169998	-17.514047
25557	59.8	98.4	1.020	NaN	8.7	42.454	9.391	21.3	102.89	30629.872010	...	6811.715689	-1398.451796	5764.919843	1233.056963	1046.795846	-415.817770	40.256680	-188.212339	-15.668410
25560	57.8	108.1	15.000	NaN	1.5	40.944	9.411	21.4	102.89	29534.445709	...	6924.413623	-1221.335306	5731.189744	1214.808948	1193.223878	-368.446330	41.131937	-194.051035	-14.352249
25562	53.0	111.4	-7.619	NaN	4.6	41.734	9.432	21.4	102.89	31969.060202	...	7732.919196	-1202.778863	6309.024879	1556.310267	1423.894317	-377.563733	48.681765	-197.347837	-13.942098
25563	54.8	111.3	-10.309	NaN	7.0	42.724	9.452	21.4	102.89	32674.719191	...	8570.026241	-1558.310970	6720.991498	1772.416496	1849.034744	-457.739613	54.244276	-205.693933	-17.299894
25565	54.6	110.7	11.224	NaN	7.0	41.614	9.471	21.3	102.89	34743.378192	...	8671.218286	-1997.018077	7073.531115	1910.525524	1597.687170	-431.450793	54.989630	-203.593648	-16.053770
25571	51.7	104.6	1.000	NaN	8.5	42.784	9.516	21.4	102.89	30655.266332	...	7395.790468	-1079.527292	6394.101992	1592.460392	1001.688476	-461.493984	51.947368	-208.580866	-17.578087
25573	60.9	110.8	21.782	NaN	10.3	40.574	9.526	21.3	102.89	35339.840155	...	8713.352894	-2557.710299	7160.962384	1994.616379	1552.390511	-700.983149	56.441013	-202.631431	-25.963291
25575	65.1	123.4	-12.000	NaN	7.7	41.764	9.536	21.2	102.89	35994.499228	...	8971.350435	-2722.784403	7263.796996	2023.220608	1707.553439	-701.947032	56.209161	-201.802974	-25.841774

10 rows × 21 columns

In [12]:
```
full_analysis(df_numeric_karl_upwind)
```

```
Dropped column 'Side_lines' (all values were NaN).

Correlation with SOG:
```




```
ANOVA:
```

	sum_sq	df	F	PR(>F)
boat_weight	351.499504	1.0	840.239706	2.134337e-152
TWA_Abs	33.632840	1.0	80.397404	7.367066e-19
Trim	23.360502	1.0	55.841959	1.209428e-13
M_back_X	11.266167	1.0	26.931136	2.342117e-07
M_front_X	10.700326	1.0	25.578526	4.671802e-07
Line_C	9.054082	1.0	21.643271	3.518732e-06
Heel_Abs	7.519618	1.0	17.975223	2.349368e-05
TWS	4.767615	1.0	11.396714	7.511020e-04
M_tot_X	3.511536	1.0	8.394129	3.809026e-03
P_back_Y	2.835445	1.0	6.777972	9.303451e-03
P_front_X	2.392165	1.0	5.718335	1.688895e-02
F_back	2.077321	1.0	4.965719	2.597545e-02
M_tot_Y	1.594115	1.0	3.810643	5.107997e-02
P_front_Y	1.296915	1.0	3.100202	7.844934e-02
F_front	0.974339	1.0	2.329101	1.271477e-01
ROT	0.431169	1.0	1.030686	3.101315e-01
M_back_Y	0.277384	1.0	0.663071	4.155835e-01
P_back_X	0.158945	1.0	0.379950	5.377065e-01
M_front_Y	0.006165	1.0	0.014738	9.033882e-01
Residual	766.803312	1833.0	NaN	NaN

```
Polynomial fit:
R²: 0.222
Used samples: 1850

 Formula :
SOG ≈ 22.365 +
  -0.071 * Heel_Abs +
  -0.101 * Line_C +
  0.015 * ROT +
  -0.118 * Trim +
  0.150 * TWA_Abs +
  0.063 * TWS +
  904513583.684 * boat_weight +
  650178429795.927 * F_front +
  -389467763336.420 * F_back +
  -589705111094.172 * M_tot_X +
  394550345526.404 * M_tot_Y +
  474165279524.497 * M_front_X +
  -266483994319.965 * M_front_Y +
  169411773944.681 * M_back_X +
  -53128692959.483 * M_back_Y +
  0.153 * P_front_X +
  -0.119 * P_front_Y +
  -0.065 * P_back_X +
  0.255 * P_back_Y
```

	feature	coefficient
7	F_front	6.501784e+11
9	M_tot_X	-5.897051e+11
11	M_front_X	4.741653e+11
10	M_tot_Y	3.945503e+11
8	F_back	-3.894678e+11
12	M_front_Y	-2.664840e+11
13	M_back_X	1.694118e+11
14	M_back_Y	-5.312869e+10
6	boat_weight	9.045136e+08
18	P_back_Y	2.548537e-01
15	P_front_X	1.531433e-01
4	TWA_Abs	1.495566e-01
16	P_front_Y	-1.185640e-01
3	Trim	-1.182396e-01
1	Line_C	-1.011653e-01
0	Heel_Abs	-7.082766e-02
17	P_back_X	-6.483301e-02
5	TWS	6.343217e-02
2	ROT	1.539630e-02

II.2.3. Upwind: Karl vs Gian t_test

In [13]:
```
t_test(df_numeric_gian_upwind,df_numeric_karl_upwind)
```

```
T-statistic: -4.351, p-value: 0.000013771261761
The difference is statistically significant, keeping data split.
```

II.3. Upwind: Master vs Slave

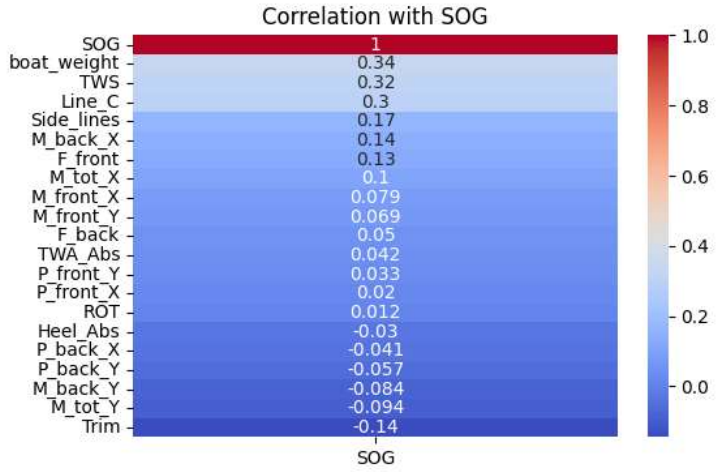
II.3.1. Master

In [14]:
```
master_data_upwind = upwind_data[upwind_data['boat_role'] == "master"]
df_numeric_master_upwind = master_data_upwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_master_upwind.drop(columns=[c for c in drop_cols if c in df_numeric_master_upwind.columns], inplace=True)
df_numeric_master_upwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_master_upwind)}")
```

```
Number of rows after filtering: 2867
```

In [15]:
```
full_analysis(df_numeric_master_upwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	827.717837	1.0	2469.483328	0.000000e+00
TWS	82.212130	1.0	245.278615	5.179438e-52
TWA_Abs	55.846578	1.0	166.617400	1.454441e-36
P_back_Y	12.605613	1.0	37.608651	1.054259e-09
P_back_X	8.733098	1.0	26.055062	3.658625e-07
M_front_Y	8.174837	1.0	24.389499	8.576741e-07
M_tot_X	7.994822	1.0	23.852428	1.129492e-06
Line_C	7.277872	1.0	21.713417	3.391434e-06
Trim	6.085619	1.0	18.156351	2.137081e-05
M_back_X	5.467605	1.0	16.312515	5.590028e-05
M_back_Y	5.274517	1.0	15.736439	7.558122e-05
P_front_X	5.068143	1.0	15.120727	1.044082e-04
F_front	2.389140	1.0	7.127961	7.655430e-03
ROT	2.256939	1.0	6.733542	9.536346e-03
Heel_Abs	1.845449	1.0	5.505870	1.905765e-02
F_back	1.745831	1.0	5.208659	2.258761e-02
M_front_X	0.823555	1.0	2.457064	1.171681e-01
Side_lines	0.493363	1.0	1.471940	2.251945e-01
M_tot_Y	0.427413	1.0	1.275180	2.589434e-01
P_front_Y	0.106127	1.0	0.316628	5.737092e-01
Residual	620.750671	1852.0	NaN	NaN

```
Polynomial fit:
R²: 0.414
Used samples: 1870

 Formula :
SOG ≈ 22.831 +
  -0.033 * Heel_Abs +
  0.072 * Line_C +
  -0.037 * ROT +
  -0.019 * Side_lines +
  -0.060 * Trim +
  -0.243 * TWA_Abs +
  0.274 * TWS +
  -14406456185.389 * boat_weight +
  8542153544345.415 * F_front +
  -4200014554577.834 * F_back +
  1457661942801.732 * M_tot_X +
  8250878354495.330 * M_tot_Y +
  -1281496463077.676 * M_front_X +
  -2463868947631.610 * M_front_Y +
  -293165738183.639 * M_back_X +
  -824187412469.346 * M_back_Y +
  0.305 * P_front_X +
  -0.031 * P_front_Y +
  -0.540 * P_back_X +
  0.391 * P_back_Y
```

	feature	coefficient
8	F_front	8.542154e+12
11	M_tot_Y	8.250878e+12
9	F_back	-4.200015e+12
13	M_front_Y	-2.463869e+12
10	M_tot_X	1.457662e+12
12	M_front_X	-1.281496e+12
15	M_back_Y	-8.241874e+11
14	M_back_X	-2.931657e+11
7	boat_weight	-1.440646e+10
18	P_back_X	-5.404717e-01
19	P_back_Y	3.906849e-01
16	P_front_X	3.051720e-01
6	TWS	2.736300e-01
5	TWA_Abs	-2.432498e-01
1	Line_C	7.244363e-02
4	Trim	-5.956685e-02
2	ROT	-3.719524e-02
0	Heel_Abs	-3.296162e-02
17	P_front_Y	-3.109957e-02
3	Side_lines	-1.947924e-02

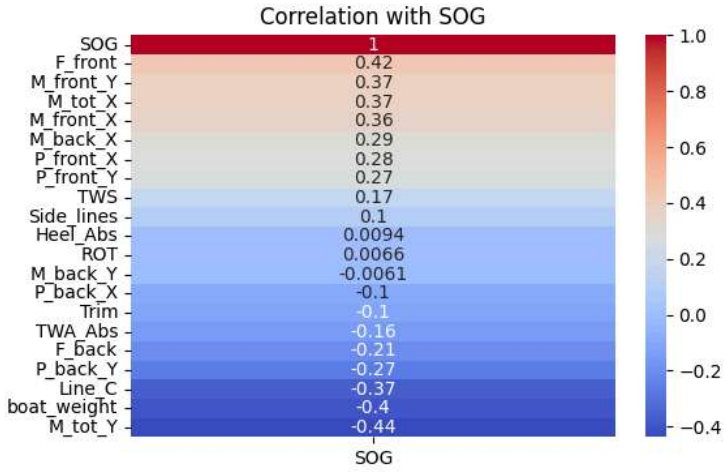
II.3.2 Slave

In [16]:
```
slave_data_upwind = upwind_data[upwind_data['boat_role'] == "slave"]
df_numeric_slave_upwind = slave_data_upwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_slave_upwind.drop(columns=[c for c in drop_cols if c in df_numeric_slave_upwind.columns], inplace=True)
df_numeric_slave_upwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_slave_upwind)}")
```

```
Number of rows after filtering: 3028
```

In [17]:
```
full_analysis(df_numeric_slave_upwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	769.811067	1.0	2069.984447	8.185373e-317
TWA_Abs	66.297860	1.0	178.271715	4.050752e-39
M_front_Y	28.090869	1.0	75.534977	7.002827e-18
M_back_Y	24.569579	1.0	66.066401	7.293222e-16
F_front	17.565194	1.0	47.231950	8.240824e-12
Side_lines	16.943416	1.0	45.560019	1.899676e-11
P_back_X	15.964026	1.0	42.926488	7.097502e-11
Line_C	12.523694	1.0	33.675603	7.480383e-09
M_tot_Y	12.389230	1.0	33.314036	8.982827e-09
F_back	11.915082	1.0	32.039076	1.713951e-08
Trim	10.364000	1.0	27.868292	1.429856e-07
M_tot_X	8.430399	1.0	22.668933	2.053312e-06
M_back_X	5.461816	1.0	14.686556	1.306076e-04
P_back_Y	4.975173	1.0	13.377997	2.607396e-04
Heel_Abs	4.646168	1.0	12.493319	4.170749e-04
M_front_X	2.137343	1.0	5.747211	1.660007e-02
ROT	1.706606	1.0	4.588982	3.229063e-02
P_front_Y	0.237626	1.0	0.638964	4.241748e-01
P_front_X	0.203965	1.0	0.548453	4.590316e-01
TWS	0.000173	1.0	0.000464	9.828139e-01
Residual	796.221198	2141.0	NaN	NaN

```
Polynomial fit:
R²: 0.334
Used samples: 2159

 Formula :
SOG ≈ 21.844 +
  -0.052 * Heel_Abs +
  0.090 * Line_C +
  -0.029 * ROT +
  -0.119 * Side_lines +
  -0.073 * Trim +
  -0.230 * TWA_Abs +
  0.001 * TWS +
  6252314647.196 * boat_weight +
  -1236532627322.172 * F_front +
  651105329258.594 * F_back +
  -1638431354159.697 * M_tot_X +
  -1189545019768.684 * M_tot_Y +
  1392617986104.779 * M_front_X +
  382700569691.020 * M_front_Y +
  339821939719.798 * M_back_X +
  127268764188.695 * M_back_Y +
  0.023 * P_front_X +
  -0.033 * P_front_Y +
  -0.565 * P_back_X +
  0.217 * P_back_Y
```

	feature	coefficient
10	M_tot_X	-1.638431e+12
12	M_front_X	1.392618e+12
8	F_front	-1.236533e+12
11	M_tot_Y	-1.189545e+12
9	F_back	6.511053e+11
13	M_front_Y	3.827006e+11
14	M_back_X	3.398219e+11
15	M_back_Y	1.272688e+11
7	boat_weight	6.252315e+09
18	P_back_X	-5.647383e-01
5	TWA_Abs	-2.296049e-01
19	P_back_Y	2.167725e-01
3	Side_lines	-1.191296e-01
1	Line_C	8.982249e-02
4	Trim	-7.253475e-02
0	Heel_Abs	-5.160850e-02
17	P_front_Y	-3.282507e-02
2	ROT	-2.892375e-02
16	P_front_X	2.337784e-02
6	TWS	1.274263e-03

III.3.3. Upwind: Master vs Slave t_test

In [18]:
```
t_test(df_numeric_master_upwind,df_numeric_slave_upwind)
```

```
T-statistic: 23.713, p-value: 0.000000000000000
The difference is statistically significant, keeping data split.
```

III Downwind

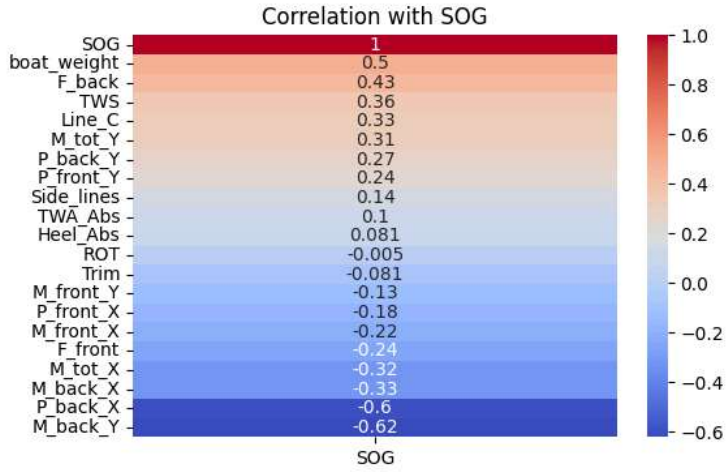
III.1. All downwind data

In [19]:
```
downwind_data = df[df['TWA'] < 0]
df_numeric_downwind = downwind_data.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_downwind.drop(columns=[c for c in drop_cols if c in df_numeric_downwind.columns], inplace=True)
df_numeric_downwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_downwind)}")
```

```
Number of rows after filtering: 3252
```

In [20]:
```
full_analysis(df_numeric_downwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	455.523144	1.0	637.346342	2.672430e-124
TWS	68.166485	1.0	95.375308	4.217503e-22
Trim	61.329638	1.0	85.809516	4.414967e-20
Heel_Abs	40.981245	1.0	57.339011	5.282257e-14
P_back_X	33.721702	1.0	47.181804	8.302549e-12
M_tot_X	21.374323	1.0	29.905938	5.027348e-08
P_front_Y	17.550853	1.0	24.556319	7.744030e-07
P_back_Y	16.553172	1.0	23.160412	1.587267e-06
F_front	12.792202	1.0	17.898241	2.421934e-05
M_tot_Y	10.928665	1.0	15.290869	9.483659e-05
Line_C	10.591773	1.0	14.819506	1.215410e-04
F_back	10.369975	1.0	14.509176	1.431423e-04
M_back_Y	7.221591	1.0	10.104107	1.499098e-03
M_back_X	6.204488	1.0	8.681025	3.247803e-03
P_front_X	5.123127	1.0	7.168036	7.474273e-03
M_front_Y	5.038947	1.0	7.050255	7.980212e-03
TWA_Abs	4.636008	1.0	6.486482	1.093459e-02
ROT	2.863077	1.0	4.005881	4.545938e-02
Side_lines	2.517192	1.0	3.521935	6.068855e-02
M_front_X	0.198025	1.0	0.277067	5.986811e-01
Residual	1639.563959	2294.0	NaN	NaN

```
Polynomial fit:
R²: 0.407
Used samples: 2312

 Formula :
SOG ≈ 27.661 +
  0.182 * Heel_Abs +
  0.098 * Line_C +
  -0.039 * ROT +
  0.045 * Side_lines +
  -0.229 * Trim +
  0.058 * TWA_Abs +
  0.195 * TWS +
  -26199319074.434 * boat_weight +
  1318085753336.969 * F_front +
  -959307000101.745 * F_back +
  10201902875148.852 * M_tot_X +
  1263392102270.947 * M_tot_Y +
  -9719703612078.463 * M_front_X +
  -377123962278.674 * M_front_Y +
  -5442639892397.313 * M_back_X +
  -178752900294.475 * M_back_Y +
  -0.227 * P_front_X +
  -0.437 * P_front_Y +
  -1.083 * P_back_X +
  -0.704 * P_back_Y
```

	feature	coefficient
10	M_tot_X	1.020190e+13
12	M_front_X	-9.719704e+12
14	M_back_X	-5.442640e+12
8	F_front	1.318086e+12
11	M_tot_Y	1.263392e+12
9	F_back	-9.593070e+11
13	M_front_Y	-3.771240e+11
15	M_back_Y	-1.787529e+11
7	boat_weight	-2.619932e+10
18	P_back_X	-1.083399e+00
19	P_back_Y	-7.038276e-01
17	P_front_Y	-4.368462e-01
4	Trim	-2.294101e-01
16	P_front_X	-2.273544e-01
6	TWS	1.951251e-01
0	Heel_Abs	1.819538e-01
1	Line_C	9.826150e-02
5	TWA_Abs	5.809749e-02
3	Side_lines	4.509489e-02
2	ROT	-3.938481e-02

III.2. Downwind: Gian vs Karl

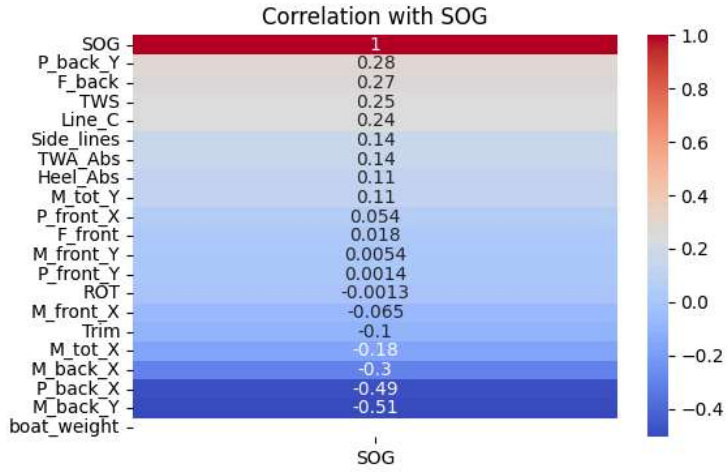
III.2.1. Downwind: Gian

In [21]:
```
gian_data_downwind = downwind_data[
    (downwind_data['boat_name'] == "Gian Stragiotti") |
    ((downwind_data['boat_name'] == "SenseBoard") & (downwind_data['opponent_name'] == "Karl Maeder"))
]
df_numeric_gian_downwind = gian_data_downwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_gian_downwind.drop(columns=[c for c in drop_cols if c in df_numeric_gian_downwind.columns], inplace=True)
df_numeric_gian_downwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_gian_downwind)}")
```

```
Number of rows after filtering: 2327
```

In [22]:
```
full_analysis(df_numeric_gian_downwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	455.523144	1.0	637.346342	2.672430e-124
TWS	68.166485	1.0	95.375308	4.217503e-22
Trim	61.329638	1.0	85.809516	4.414967e-20
Heel_Abs	40.981245	1.0	57.339011	5.282257e-14
P_back_X	33.721702	1.0	47.181804	8.302549e-12
M_tot_X	21.374323	1.0	29.905938	5.027348e-08
P_front_Y	17.550853	1.0	24.556319	7.744030e-07
P_back_Y	16.553172	1.0	23.160412	1.587267e-06
F_front	12.792202	1.0	17.898241	2.421934e-05
M_tot_Y	10.928665	1.0	15.290869	9.483659e-05
Line_C	10.591773	1.0	14.819506	1.215410e-04
F_back	10.369975	1.0	14.509176	1.431423e-04
M_back_Y	7.221591	1.0	10.104107	1.499098e-03
M_back_X	6.204488	1.0	8.681025	3.247803e-03
P_front_X	5.123127	1.0	7.168036	7.474273e-03
M_front_Y	5.038947	1.0	7.050255	7.980212e-03
TWA_Abs	4.636008	1.0	6.486482	1.093459e-02
ROT	2.863077	1.0	4.005881	4.545938e-02
Side_lines	2.517192	1.0	3.521935	6.068855e-02
M_front_X	0.198025	1.0	0.277067	5.986811e-01
Residual	1639.563959	2294.0	NaN	NaN

```
Polynomial fit:
R²: 0.407
Used samples: 2312

 Formula :
SOG ≈ 27.661 +
  0.182 * Heel_Abs +
  0.098 * Line_C +
  -0.039 * ROT +
  0.045 * Side_lines +
  -0.229 * Trim +
  0.058 * TWA_Abs +
  0.195 * TWS +
  -26199319074.434 * boat_weight +
  1318085753336.969 * F_front +
  -959307000101.745 * F_back +
  10201902875148.852 * M_tot_X +
  1263392102270.947 * M_tot_Y +
  -9719703612078.463 * M_front_X +
  -377123962278.674 * M_front_Y +
  -5442639892397.313 * M_back_X +
  -178752900294.475 * M_back_Y +
  -0.227 * P_front_X +
  -0.437 * P_front_Y +
  -1.083 * P_back_X +
  -0.704 * P_back_Y
```

	feature	coefficient
10	M_tot_X	1.020190e+13
12	M_front_X	-9.719704e+12
14	M_back_X	-5.442640e+12
8	F_front	1.318086e+12
11	M_tot_Y	1.263392e+12
9	F_back	-9.593070e+11
13	M_front_Y	-3.771240e+11
15	M_back_Y	-1.787529e+11
7	boat_weight	-2.619932e+10
18	P_back_X	-1.083399e+00
19	P_back_Y	-7.038276e-01
17	P_front_Y	-4.368462e-01
4	Trim	-2.294101e-01
16	P_front_X	-2.273544e-01
6	TWS	1.951251e-01
0	Heel_Abs	1.819538e-01
1	Line_C	9.826150e-02
5	TWA_Abs	5.809749e-02
3	Side_lines	4.509489e-02
2	ROT	-3.938481e-02

III.2.2. Downwind: Karl

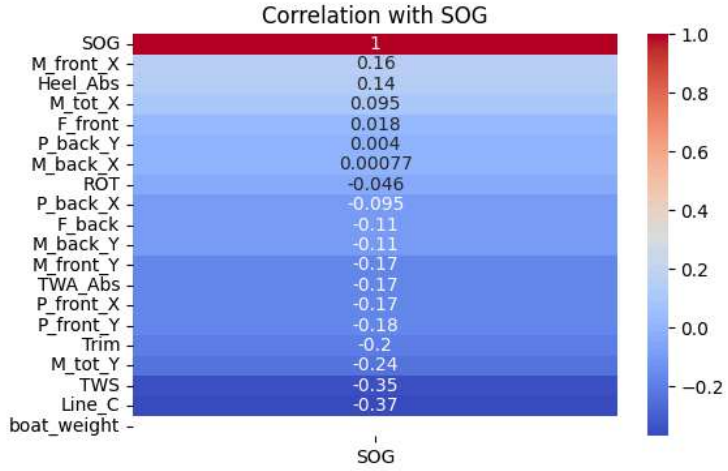
In [23]:
```
karl_data_downwind = downwind_data[
    (downwind_data['boat_name'] == "Karl Maeder") |
    ((downwind_data['boat_name'] == "SenseBoard") & (downwind_data['opponent_name'] == "Gian Stragiotti"))
]
df_numeric_karl_downwind = karl_data_downwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_karl_downwind.drop(columns=[c for c in drop_cols if c in df_numeric_karl_downwind.columns], inplace=True)
df_numeric_karl_downwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_karl_downwind)}")
```

```
Number of rows after filtering: 925
```

In [24]:
```
full_analysis(df_numeric_karl_downwind)
```

```
Dropped column 'Side_lines' (all values were NaN).

Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	82.007094	1.0	148.035617	1.174499e-31
Trim	68.781953	1.0	124.162171	4.077421e-27
Line_C	65.293152	1.0	117.864339	6.710535e-26
Heel_Abs	65.150486	1.0	117.606805	7.527790e-26
TWA_Abs	61.013096	1.0	110.138170	2.137886e-24
ROT	8.819297	1.0	15.920208	7.139871e-05
M_tot_X	3.771901	1.0	6.808871	9.220016e-03
F_front	3.124369	1.0	5.639974	1.776259e-02
F_back	2.634372	1.0	4.755453	2.946193e-02
M_front_X	2.544853	1.0	4.593858	3.235215e-02
P_front_Y	2.541802	1.0	4.588349	3.245574e-02
P_back_X	1.631871	1.0	2.945782	8.644346e-02
M_tot_Y	1.558315	1.0	2.813003	9.384777e-02
M_back_X	0.765337	1.0	1.381552	2.401453e-01
P_front_X	0.608648	1.0	1.098704	2.948299e-01
TWS	0.382422	1.0	0.690332	4.062703e-01
M_back_Y	0.091620	1.0	0.165388	6.843394e-01
M_front_Y	0.069841	1.0	0.126073	7.226215e-01
P_back_Y	0.061145	1.0	0.110376	7.397930e-01
Residual	502.449585	907.0	NaN	NaN

```
Polynomial fit:
R²: 0.361
Used samples: 924

 Formula :
SOG ≈ 26.201 +
  0.337 * Heel_Abs +
  -0.564 * Line_C +
  -0.108 * ROT +
  -0.339 * Trim +
  0.436 * TWA_Abs +
  -0.030 * TWS +
  8608000697.145 * boat_weight +
  805056405607.145 * F_front +
  -823191232573.262 * F_back +
  -3360270974926.748 * M_tot_X +
  636074340686.390 * M_tot_Y +
  2025620348114.531 * M_front_X +
  -202799685917.443 * M_front_Y +
  1705860956818.280 * M_back_X +
  -184640479173.961 * M_back_Y +
  -0.333 * P_front_X +
  0.656 * P_front_Y +
  -0.244 * P_back_X +
  0.058 * P_back_Y
```

	feature	coefficient
9	M_tot_X	-3.360271e+12
11	M_front_X	2.025620e+12
13	M_back_X	1.705861e+12
8	F_back	-8.231912e+11
7	F_front	8.050564e+11
10	M_tot_Y	6.360743e+11
12	M_front_Y	-2.027997e+11
14	M_back_Y	-1.846405e+11
6	boat_weight	8.608001e+09
16	P_front_Y	6.555735e-01
1	Line_C	-5.644459e-01
4	TWA_Abs	4.363541e-01
3	Trim	-3.391051e-01
0	Heel_Abs	3.373206e-01
15	P_front_X	-3.327255e-01
17	P_back_X	-2.437588e-01
2	ROT	-1.079920e-01
18	P_back_Y	5.758171e-02
5	TWS	-2.996049e-02

III.2.3. Downwind: Karl vs Gian t_test

In [25]:
```
t_test(df_numeric_karl_downwind,df_numeric_gian_downwind)
```

```
T-statistic: -33.288, p-value: 0.000000000000000
The difference is statistically significant, keeping data split.
```

III.3. Downwind: Master vs Slave

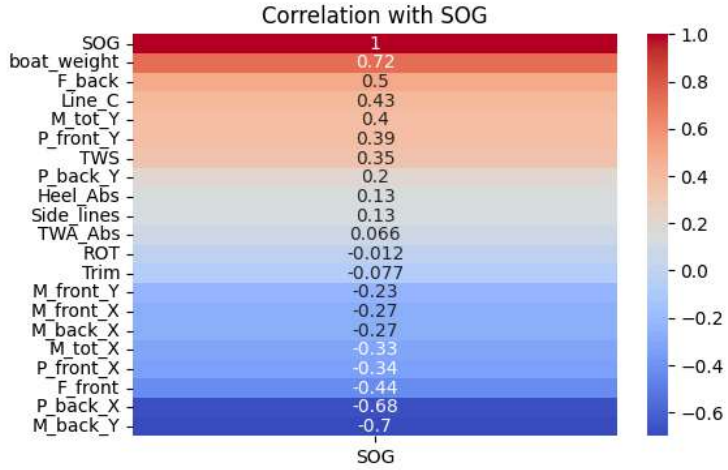
III.3.1 Downwind Master

In [26]:
```
master_data_downwind = downwind_data[downwind_data['boat_role'] == "master"]
df_numeric_master_downwind = master_data_downwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_master_downwind.drop(columns=[c for c in drop_cols if c in df_numeric_master_downwind.columns], inplace=True)
df_numeric_master_downwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_master_downwind)}")
```

```
Number of rows after filtering: 1673
```

In [27]:
```
full_analysis(df_numeric_master_downwind)
```

```
Correlation with SOG:
```




```
ANOVA:
```

	sum_sq	df	F	PR(>F)
boat_weight	116.657192	1.0	192.759904	2.814579e-40
Line_C	19.012061	1.0	31.414805	2.679794e-08
Trim	15.515553	1.0	25.637309	4.885090e-07
P_back_X	14.758800	1.0	24.386880	9.199841e-07
F_front	12.947486	1.0	21.393933	4.218857e-06
TWS	12.896209	1.0	21.309204	4.405493e-06
M_tot_Y	11.106153	1.0	18.351384	2.010116e-05
Heel_Abs	10.874884	1.0	17.969245	2.448163e-05
M_back_Y	10.725206	1.0	17.721922	2.781720e-05
F_back	10.491618	1.0	17.335950	3.396103e-05
M_front_Y	10.066495	1.0	16.633494	4.886798e-05
TWA_Abs	4.725917	1.0	7.808925	5.296130e-03
M_tot_X	4.251992	1.0	7.025830	8.158408e-03
P_back_Y	3.856650	1.0	6.372581	1.174018e-02
Side_lines	1.292217	1.0	2.135211	1.442580e-01
ROT	1.166214	1.0	1.927008	1.653886e-01
P_front_Y	1.161855	1.0	1.919804	1.661805e-01
M_back_X	0.776009	1.0	1.282247	2.577472e-01
P_front_X	0.118401	1.0	0.195641	6.583560e-01
M_front_X	0.028930	1.0	0.047803	8.269758e-01
Residual	618.508556	1022.0	NaN	NaN

```
Polynomial fit:
R²: 0.244
Used samples: 1040

 Formula :
SOG ≈ 27.810 +
  0.142 * Heel_Abs +
  0.219 * Line_C +
  -0.035 * ROT +
  -0.047 * Side_lines +
  -0.172 * Trim +
  0.088 * TWA_Abs +
  0.132 * TWS +
  -37348374046.557 * boat_weight +
  -1544178649946.608 * F_front +
  1327432276053.808 * F_back +
  6584518420634.194 * M_tot_X +
  -1604187849040.032 * M_tot_Y +
  -5977707821102.457 * M_front_X +
  495998265173.466 * M_front_Y +
  -3727900378852.000 * M_back_X +
  164071269967.318 * M_back_Y +
  -0.056 * P_front_X +
  -0.151 * P_front_Y +
  -0.884 * P_back_X +
  -0.520 * P_back_Y
```

	feature	coefficient
10	M_tot_X	6.584518e+12
12	M_front_X	-5.977708e+12
14	M_back_X	-3.727900e+12
11	M_tot_Y	-1.604188e+12
8	F_front	-1.544179e+12
9	F_back	1.327432e+12
13	M_front_Y	4.959983e+11
15	M_back_Y	1.640713e+11
7	boat_weight	-3.734837e+10
18	P_back_X	-8.840787e-01
19	P_back_Y	-5.203043e-01
1	Line_C	2.187205e-01
4	Trim	-1.724257e-01
17	P_front_Y	-1.510318e-01
0	Heel_Abs	1.415127e-01
6	TWS	1.318144e-01
5	TWA_Abs	8.752035e-02
16	P_front_X	-5.604109e-02
3	Side_lines	-4.669204e-02
2	ROT	-3.512108e-02

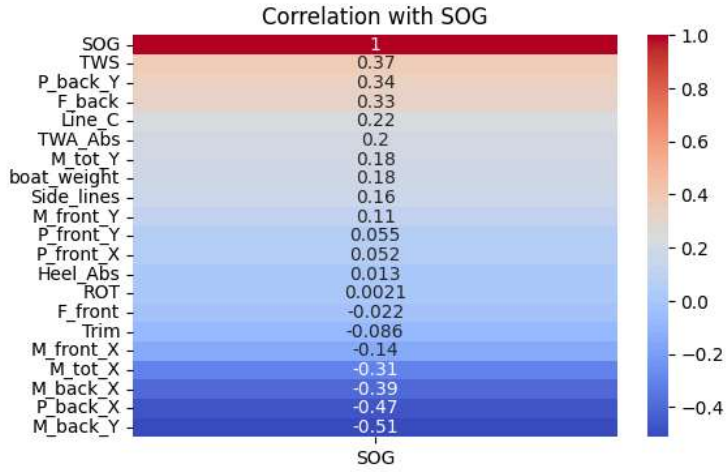
III.3.2 Downwind Slave

In [28]:
```
slave_data_downwind = downwind_data[downwind_data['boat_role'] == "slave"]
df_numeric_slave_downwind = slave_data_downwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_slave_downwind.drop(columns=[c for c in drop_cols if c in df_numeric_slave_downwind.columns], inplace=True)
df_numeric_slave_downwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_slave_downwind)}")
```

```
Number of rows after filtering: 1579
```

In [29]:
```
full_analysis(df_numeric_slave_downwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	183.591788	1.0	290.137687	1.098204e-58
TWS	59.447749	1.0	93.947734	1.795355e-21
Trim	33.475988	1.0	52.903487	6.153368e-13
P_front_Y	31.035907	1.0	49.047325	4.064087e-12
TWA_Abs	23.422396	1.0	37.015380	1.554257e-09
Heel_Abs	17.890282	1.0	28.272752	1.246442e-07
P_back_X	17.178718	1.0	27.148238	2.201187e-07
P_front_X	9.155276	1.0	14.468462	1.493883e-04
Side_lines	7.126972	1.0	11.263049	8.143217e-04
M_tot_X	5.869559	1.0	9.275907	2.370325e-03
F_back	5.551519	1.0	8.773295	3.114302e-03
F_front	5.460452	1.0	8.629378	3.368290e-03
M_tot_Y	5.386116	1.0	8.511901	3.591189e-03
M_front_X	2.952052	1.0	4.665250	3.096797e-02
ROT	2.846112	1.0	4.497828	3.413376e-02
P_back_Y	2.391100	1.0	3.778754	5.213072e-02
Line_C	1.222609	1.0	1.932139	1.647713e-01
M_back_Y	1.201419	1.0	1.898653	1.684750e-01
M_front_Y	0.124170	1.0	0.196231	6.578566e-01
M_back_X	0.058984	1.0	0.093215	7.601798e-01
Residual	793.499476	1254.0	NaN	NaN

```
Polynomial fit:
R²: 0.579
Used samples: 1272

 Formula :
SOG ≈ 27.423 +
  0.159 * Heel_Abs +
  -0.046 * Line_C +
  -0.051 * ROT +
  0.097 * Side_lines +
  -0.240 * Trim +
  0.183 * TWA_Abs +
  0.239 * TWS +
  -9351451424.464 * boat_weight +
  5915144993048.043 * F_front +
  -3713631417304.865 * F_back +
  10295158339960.328 * M_tot_X +
  5344176339449.651 * M_tot_Y +
  -10059447831470.029 * M_front_X +
  -1506149551478.628 * M_front_Y +
  -5327874174441.482 * M_back_X +
  -854903402547.643 * M_back_Y +
  -0.413 * P_front_X +
  -0.921 * P_front_Y +
  -1.206 * P_back_X +
  -0.386 * P_back_Y
```

	feature	coefficient
10	M_tot_X	1.029516e+13
12	M_front_X	-1.005945e+13
8	F_front	5.915145e+12
11	M_tot_Y	5.344176e+12
14	M_back_X	-5.327874e+12
9	F_back	-3.713631e+12
13	M_front_Y	-1.506150e+12
15	M_back_Y	-8.549034e+11
7	boat_weight	-9.351451e+09
18	P_back_X	-1.206312e+00
17	P_front_Y	-9.205201e-01
16	P_front_X	-4.131763e-01
19	P_back_Y	-3.858977e-01
4	Trim	-2.396024e-01
6	TWS	2.388335e-01
5	TWA_Abs	1.826137e-01
0	Heel_Abs	1.585404e-01
3	Side_lines	9.692972e-02
2	ROT	-5.093865e-02
1	Line_C	-4.613166e-02

III.3.3. Downwind: Master vs Slave t_test

In [30]:
```
t_test(df_numeric_master_downwind,df_numeric_slave_downwind)
```

```
T-statistic: -6.001, p-value: 0.000000002171240
The difference is statistically significant, keeping data split.
```