インターンシップレポート by Sid-San

~ シッダールタ・コム

経営企画本部 研究開発部

目次

フィギュア一覧

テーブルのリスト

一. 紹介

[1-1] 海上脱炭素化の背景と海運におけるエネルギー効率の重要性

海上の脱炭素化とは、船舶が大気中に放出する二酸化炭素(CO_2)などの温室効果ガスの量を減らすことを意味します。船舶は世界中を物資を移動させるために重要ですが、世界の温室効果ガス排出量の約 3%を排出しています。海運業を国に例えると、世界で 6 番目に大きな汚染国になります。

船舶のエネルギー効率を高めることは非常に重要です。船は燃料を大量に消費し、燃料を燃やすと CO_2が発生します。燃料費は、船舶の総費用の約半分になる可能性があります。燃料の使用量を減らすことで、船はお金を節約し、同時に環境にも貢献できます。

国際海事機関(IMO)は、排出量を削減するための船舶のルールを設定する主要なグループです。IMO は、海運業界が 2050 年までにネットゼロエミッションを達成することを望んでいます。これを支援するために、IMO は炭素強度指標(CII)などのルールとツールを作成しました。これらの規則は、船主がより良い技術を使用し、船舶の運航方法を改善し、よりクリーンな燃料を使用することを奨励しています。

要約すると、船舶からの排出量を削減することは、環境を保護し、コストを節約するために重要です。エネルギー効率は、将来に向けてよりクリーンで持続可能な配送を実現するための重要な要素です。

CII は 2023 年 1 月に開始された規制で、RO-RO(ロールオン/ロールオフ)船を含むすべての大型貨物船に適用されます。これは、船舶が一定の距離にわたって運ぶ貨物の単位ごとに発生する二酸化炭素(CO_2)の量を測定します。船舶は、毎年、燃料使用量と移動距離を報告する必要があります。この情報に基づいて、各船舶は A(最高)から E(最低)までの評価を受け取ります。

船の評価が低い場合、所有者はその性能を改善するための計画を立てる必要があります。これらの規則は年々厳しくなっているため、船主や運航者は、燃料の使用量を減らし、排出量を削減する方法を常に模索する必要があります。RO-RO 船の場合、これはエネルギー効率に細心の注意を払い、必要な基準内にとどまるように改善を行うことを意味します。CII 規制は、海運をより環境に優しく、持続可能なものにするための重要なステップです。

それはどのように機能しますか?

- 毎年、船舶は運航データを自国の旗国である行政機関に報告する必要があります。
- CII は、船の報告された燃料消費量と航行距離を使用して計算されます。
- 船舶は、船舶エネルギー効率管理計画書(SEEMP)に記録されている評価(A～E)を受け取ります。
- 船舶が 3 年連続で D 評価を受けた場合、または 1 回 E 評価を受けた場合、その性能を改善するための計画を立て、当局から承認を得る必要があります。

インパクト：

- 船主と運航者は、船舶のエネルギー効率を監視し、改善する必要があります。
- 評価の低い船舶は、是正措置計画の策定や実施など、追加の要件に直面する可能性があります。
- この規則により、業界はより優れたテクノロジーとよりスマートな運用を採用し、輸送をよりクリーンで持続可能なものにするよう促されます。

このプロジェクトの主な目標は、実際の運用データを使用して、RO-RO 船の炭素強度指標(CII)を予測および改善するのに役立つツールを作成することです。具体的には、このプロジェクトは以下を目指しています。

- 機械学習を使用して、燃料消費量、移動距離、その他の重要な特徴などの日々の運用データに基づいて、船舶の年間 CII スコアを予測します。
- ユーザーが船舶データを入力すると、予測された CII 値と性能評価を即座に確認できる使いやすいインターフェースを提供します。
- 船舶のエネルギー効率を改善するための実践的な提案を提供し、オペレーターが IMO 規制を満たし、燃料コストを削減するのに役立ちます。
- 船舶管理の意思決定を支援し、より持続可能で環境に優しい海運業務に貢献します。

[1-2] 炭素強度指標(CII)の解説

炭素強度指標(CII)は、船舶からの温室効果ガス排出量を削減するために国際海事機関(IMO)が作成した規則です。CII は、船が二酸化炭素(CO_2)を生成しながら、貨物や乗客をどれだけ効率的に移動させるかを示しています。簡単に言えば、船舶が 1 海里にわたって運ぶ貨物の単位ごとに排出する CO_2の量を測定します。

毎年、5,000 総トン数(GT)を超える船舶は、国際的に旅行し、燃料使用量と航行距離を報告する必要があります。CII は、次の式を使用して計算されます。

CII = 年間 CO_2排出量 (グラム) / (貨物積載量 × 航行距離 (海里))

ほとんどの貨物船では、貨物積載量は載貨重量トン数で測定されますが、旅客船や RO-RO 船では、総トン数として測定されることがよくあります。CO_2排出量は、船舶が年間に使用する燃料から算出しています。このシステムにより、さまざまな種類やサイズの船を公平に比較することができます。

CII ルールの主な目標は、船舶のエネルギー効率を高め、2030 年までに排出量を 2008 年のレベルと比較して少なくとも 40%削減することです。要件は年々厳しくなるため、船舶はコンプライアンスを維持するために性能を向上させ続ける必要があります。

[1-3] CII レーティングシステムの説明

船舶の CII 値が計算された後、その船舶には A から E までの評価が与えられます。この評価は、エネルギー効率と排出量の観点から船がどれだけうまく機能しているかを示しています。

- **A:**最高のパフォーマンス(メジャースーペリア)
- **B:**良いパフォーマンス(マイナースーペリア)
- **C:**許容可能な性能(基準を満たしています)
- **D:**基準以下(軽微な劣等)
- **E:**パフォーマンスが悪い(劣っている)

船舶が準拠していると見なされるには、少なくとも C 評価が必要です。船舶が 3 年連続で D 評価を受けた場合、またはいずれかの年で E 評価を受けた場合、船主は船の評価を改善するための是正措置計画を立てなければなりません。この計画は、当局によって承認され、船舶の管理文書に含まれている必要があります。

CII 評価は船の公式文書に記録されており、毎年報告する必要があります。各レーティングの要件は時間の経過とともに厳しくなるため、船主や運航者は、省エネ対策、技術の向上、運航の改善に引き続き取り組む必要があります。

この格付けシステムは、船主が排出量を削減し、効率を向上させるための行動を取ることを奨励し、海運業界がよりクリーンで持続可能な未来に向けて進むのを支援します。

二. 年間 CII 予測モデル-1(目的)

このプロジェクトの主な目的は、RO-RO の船舶の運航者と管理者が実際の運用データを使用して船舶の炭素強度指標(CII)を予測および最適化するのに役立つ実用的なツールを開発することです。IMO の新規制により、船舶に CII 評価の報告と改善が求められる中、多くの企業は、スコアに影響を与える要因を理解し、問題が発生する前にどのように行動すべきかを理解するという課題に直面しています。

現在、ほとんどの船舶運航者は、年次報告書が完成したときにのみ、CII の結果を確認します。これにより、運用上の問題を早期に特定したり、変更の影響をテストしたり、改善のための戦略を計画したりすることが難しくなります。日々の船舶データを使用して将来の CII 結果を予測し、実行可能な提案を提供できるシステムが必要であることは明らかです。

このプロジェクトは、これらの問題を解決することを目的としています。

- データ駆動型でユーザーフレンドリーなツールを構築し、日々の運航記録を使用して船舶の年間 CII スコアを予測します。
- ユーザーは、速度、トリム、プロペラピッチなどのさまざまな要因が CII と全体的なエネルギー効率にどのように影響するかを確認できます。
- CII 評価の維持または改善に役立つ運用上の変更に関する明確で実用的な推奨事項を提供します。
- IMO 規制への準拠をサポートし、企業が罰則を回避し、燃料コストを削減し、より持続可能な運営を行えるよう支援します。

このプロジェクトは、CII の予測と最適化の提案をアクセスしやすく、理解しやすくすることで、船舶運航者、技術マネージャー、サステナビリティチームのより良い意思決定をサポートします。最終的な目標は、海事業界がよりクリーンで、より効率的で、規制に準拠した海運に移行するのを支援することです。

三. サンプルデータの説明

次に示すのは、レポート内のデータセットを説明するために使用できるサンプル テキストです。簡単な英語を使用し、指定したサンプル データを参照します。

サンプルデータの説明

このプロジェクトでは、RO-RO 船から収集した詳細な運用データを使用しました。データ

は毎分記録され、船の性能と環境に関する多くの重要な測定値が含まれています。データセットの各行は、船舶の運航の 1 分間を表します。

サンプル データの主な列には、次のようなものがあります。

- 時間: データが記録された日時。
- Rev_ME:メインエンジンの毎分回転数で、エンジンの回転速度を示します。
- CppPitch: 制御可能なピッチ プロペラのピッチ角で、船が水中を移動する方法に影響します。
- FO_ME_Cons:メインエンジンが消費する燃料油の総量(リットル単位)。
- FO_GE_Cons:発電機エンジンによって消費される燃料油の総量(リットル単位)。
- Wind_Direction: 風が吹いている方向 (度単位)。
- Wind_Speed: メートル/秒で測定される風の速度。
- Ship_Speed: ノットで測定された船の速度。
- Fore_Draft:喫水線からの船首(正面)の深さ(メートル単位)。
- Aft_Draft:喫水線から船尾(背面)の深さ(メートル単位)。

データセットには、センサーの読み取り値やステータスフラグなど、他にも多くの列が含まれていますが、上記の列は、船の炭素強度とエネルギー効率を予測するために最も重要な列です。

データの例を次に示します。

表 1:RO-RO サンプルデータ

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Time	Rev_ME	CppPitch	FO_ME_C	FO_GE_Cc	Wind_Dire	Wind_Spe	Ship_Spee	Fore_Draf	Aft_Draft	PUMP1	PUMP2	HEEL	HEEL_ALT
2	05-03-2024 00:00	102.4	23.6	102934	5519	348	9.2	17.9	5.8	6.73	0	0	0.37	0
3	05-03-2024 00:01	102.3	23.4	102958	5519	336	9	17.8	5.8	6.67	0	0	0.84	0
4	05-03-2024 00:02	102.3	23.4	102981	5519	359	9.8	18	5.7	6.62	0	0	0.27	0
5	05-03-2024 00:03	102.3	23.4	103005	5519	348	9.1	18	5.78	6.65	0	0	0.5	0
6	05-03-2024 00:04	102.2	23.4	103029	5518	355	9.5	18.1	5.81	6.62	0	0	0.93	0
7	05-03-2024 00:05	102.5	23.3	103051	5519	359	10.2	18.1	5.85	6.65	0	0	0.68	0
8	05-03-2024 00:06	101.6	23.4	103076	5518	355	17.6	18	5.75	6.72	0	0	0.54	0
9	05-03-2024 00:07	102.3	23.3	103099	5518	359	9.6	18	5.86	6.7	0	0	0.89	0
10	05-03-2024 00:08	102.4	23.5	103123	5518	352	7.6	18	5.8	6.66	0	0	0.56	0
11	05-03-2024 00:09	102.4	23.4	103147	5517	353	9.7	18	5.83	6.63	0	0	0.78	0
12	05-03-2024 00:10	102.4	23.4	103169	5517	346	8.2	18	5.79	6.61	0	0	0.47	0
13	05-03-2024 00:11	101.9	23.5	103192	5517	350	18.6	18.1	5.85	6.68	0	0	0.39	0
14	05-03-2024 00:12	102.4	23.5	103216	5517	352	8.7	18	5.89	6.57	0	0	0.29	0
15	05-03-2024 00:13	102.3	23.5	103238	5518	347	9.5	18	5.84	6.61	0	0	0.7	0
16	05-03-2024 00:14	102.4	23.5	103263	5517	345	9.5	18.1	5.88	6.6	0	0	0.76	0
17	05-03-2024 00:15	102.4	23.5	103287	5517	353	10.6	18.1	5.84	6.59	0	0	0.34	0
18	05-03-2024 00:16	102.1	23.5	103311	5516	344	11.4	18	5.85	6.7	0	0	0.39	0
19	05-03-2024 00:17	102.1	23.5	103335	5516	351	8.7	18	5.86	6.63	0	0	0.12	0
20	05-03-2024 00:18	102	23.5	103358	5517	357	9.9	18	5.87	6.63	0	0	0.75	0

このデータにより、エンジン速度、プロペラピッチ、風の状態など、さまざまな要因が船の燃料消費量と排出量にどのように影響するかを分析できます。この情報を使用して、船舶の炭素強度指標(CII)を予測するモデルを構築し、エネルギー効率を改善する方法を提案できます。

四. データの前処理

[4-1] 日時情報の取り扱い

データを読み込んだ後、最初のステップは「Time」列が正しい形式であることを確認することでした。当初、「Time」列は単なるプレーンテキストだったため、時間ベースのパターンを分析する際に作業が難しくなっていました。

これを修正するために、pandas を使用して "Time" 列を適切な日付と時刻の形式に変換しました。これにより、データを時間で簡単に並べ替え、後で時間ベースの計算を実行できます。次に、データセット全体を時間順に並べ替えて、すべてのレコードが正しい順序

になるようにしました。

コード例:

df['時間'] = pd.to_datetime(df['時間'])

df = df.sort_values('時間')

これにより、すべてのデータが正しい順序で整理され、さらに分析できる状態になりました。この手順は、時間の経過に伴う傾向とパターンを見つけるのに役立つため、時系列データを操作するために重要です。

[4-2] 連続燃費記録の取り扱い

データセットには、メイン エンジン (FO_ME_Cons) と発電機エンジン (FO_GE_Cons) によって使用される燃料量の列が含まれています。場合によっては、これらの列に欠損値や数値以外のエントリがあり、分析に問題が生じる可能性があります。

これを解決するために、次の 2 つのアクションを実行しました。

一. 数値に変換する:これらの列のすべての値が数値であることを確認しました。変換できなかった値がある場合 (たとえば、欠落していたり、数値ではない場合)、その値は欠落として設定されました。

二. 欠損値の前方充填:欠損値については、最後に使用可能な値を使用してギャップを埋めました。この方法は「フォワードフィリング」と呼ばれ、新しい値が記録されるまで燃料消費量が同じ速度で継続することを前提としています。通常、燃料消費量は徐々に変化するため、これは合理的なアプローチです。

コード例:

['FO_ME_Cons', 'FO_GE_Cons'] の col の場合:

```
df[col] = pd.to_numeric(df[col], errors='coerce').fillna(method='ffill')
```

これにより、燃料消費データが完全で、さらに分析する準備ができていることを確認しました。船舶の CO_2排出量を計算するには正確な燃料データが必要なため、この手順は重要です。

[4-3] 他の数値データの欠損値を埋める

データには、燃料消費量以外にも、Ship_Speed、CppPitch(プロペラピッチ)、Wind_Speed、HEEL、Fore_Draft、Aft_Draft など、重要な列があります。場合によっては、これらの列に欠落している値や無効な値もあります。

これに対処するために、次のことを行いました。

一. 数値に変換:これらの列のすべての値を数値に変更しました。値を変換できなかった場合 (たとえば、値が欠落していたり、テキストとして書き込まれていたりした場合)、その値は欠落としてマークされました。

二. 欠損値の前方充填:欠損値については、最後に記録された値を使用してギャップを埋めました。この方法は「フォワードフィル」と呼ばれます。これらの測定値のほとんど(船の速度や喫水など)は突然変化しないため、次の読み取り値まで値が同じままであると想定するのが妥当であるため、ここではうまく機能します。

コード例:

```
numeric_cols = ['Ship_Speed', 'CppPitch', 'Wind_Speed', 'HEEL', 'Fore_Draft', 'Aft_Draft']
numeric_cols の col の場合:
    df[col] = pd.to_numeric(df[col], errors='coerce').fillna(method='ffill')
```

これにより、すべての重要な数値データが完全で、さらなる分析やモデリングに使用できるようになりました。これにより、予測の品質と精度が向上します。

[4-4] 重要なデータが欠落している行の削除

欠損値を埋めた後、最も重要な列に情報が欠落している行がまだあるかどうかを確認しました。これらの列には、船速、プロペラ ピッチ、風速、ヒール、フォア ドラフト、アフト ドラフト、および両方の燃料消費量の列が含まれます。これらのいずれかがまだ欠落している場合は、データセットからそれらの行を削除しました。これらのキー列にデータが欠落していると、分析と予測の精度に影響を与える可能性があるため、この手順は重要です。

コード例:

```
df = df.dropna(サブセット=numeric_cols + ['FO_ME_Cons', 'FO_GE_Cons'])
```

[4-5] アイドル状態またはポート状態のフィルタリング

データを分析したいのは、船が実際に航行しているときだけで、停船しているときや港にいるときではありません。そのために、船の速度が 0.5 ノット以下のすべての列を削

除しました。船が動いている期間のみに焦点を当てることで、エネルギー効率と CII を予測するために最も重要な実際の航海条件についてモデルが学習するようにしています。

コード例:

```
df = df[df['Ship_Speed'] > 0.5]
```

これらの手順に従うことで、データセットがクリーンで完全であり、船の海上でのアクティブな運用に焦点を当てていることを確認しました。これにより、より信頼性が高く正確な予測モデルを構築することができます。

五. 特徴量エンジニアリングと計算

モデリング用のデータを準備するために、船舶の炭素強度とエネルギー効率を予測するために重要ないくつかの新しい特徴を作成しました。

[5-1] 1 分あたりの燃料消費量の計算

元のデータセットでは、メインエンジン(FO_ME_Cons)と発電機エンジン(FO_GE_Cons)の燃料使用量の合計が累計として提供されています。船が毎分どれだけの燃料を使用したかを調べるために、両方の列の各行と前の行の差を計算しました。

- 差がマイナスの場合(データのリセットが原因で発生する可能性があります)、エラーを避けるためにゼロに設定します。
- 次に、メインエンジンと発電機の燃料消費量を合計して、1 分あたりの総燃料使用量を求めました。

コード例:

```
fuel_me = df['FO_ME_Cons'].diff().clip(lower=0)
fuel_ge = df['FO_GE_Cons'].diff().clip(lower=0)
df['Fuel_Liters'] = fuel_me + fuel_ge
```

[5-2] トリムトリムの計算

は、船の後部(Aft_Draft)の喫水と前部(Fore_Draft)の喫水の差です。トリムは、船が水上を移動する効率に影響を与える可能性があります。

コード例:

```
df['トリム'] = df['Aft_Draft'] - df['Fore_Draft']
```

[5-3] 平均喫水の計算

平均喫水は、前部と後部の喫水の平均値です。これにより、船が水面にどれだけ深く位置しているかが全体的にわかり、燃料の使用量と効率にも影響します。

コード例:

```
df['Avg_Draft'] = (df['Aft_Draft'] + df['Fore_Draft']) / 2
```

これらの新機能を作成することで、機械学習モデルにとってデータをより有用なものにしました。これらの特徴は、船の状態と、それが燃料消費と排出ガスにどのように関連しているかをよりよく理解するのに役立ちます。

[5-4] クリーニングしたデータの保存

クリーニングと特徴量エンジニアリングの手順をすべて終えたら、処理したデータを**「Cleaned_RO-RO_Data.csv」**という新しいファイルに保存しました。このファイルには、有用な列とクリーニングされた列のみが含まれ、すべての欠落値が処理され、新しい機能が追加されています。

クリーニングされたデータを保存すると、さらなる分析やモデルのトレーニングに使いやすくなります。また、データ準備プロセスの記録を保持するのにも役立ちます。

コード例:

```
df.to_csv("Cleaned_RO-RO_Data.csv", index=False)
```

クリーニングされたデータの最初の数行を以下に示します。

表 2 - クリーニングされた RO-RO データ

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Time	Rev_ME	CppPitch	FO_ME_Cc	FO_GE_Co	Wind_Dire	Wind_Spe	Ship_Spee	Fore_Draft	Aft_Draft	PUMP1	PUMP2	HEEL	HEEL_ALT	WBT1
2	05-03-2024 00:00	102.4	23.6	102934	5519	348	9.2	17.9	5.8	6.73	0	0	0.37	0	1
3	05-03-2024 00:01	102.3	23.4	102958	5519	336	9	17.8	5.8	6.67	0	0	0.84	0	1
4	05-03-2024 00:02	102.3	23.4	102981	5519	359	9.8	18	5.7	6.62	0	0	0.27	0	1
5	05-03-2024 00:03	102.3	23.4	103005	5519	348	9.1	18	5.78	6.65	0	0	0.5	0	1
6	05-03-2024 00:04	102.2	23.4	103029	5518	355	9.5	18.1	5.81	6.62	0	0	0.93	0	1
7	05-03-2024 00:05	102.5	23.3	103051	5519	359	10.2	18.1	5.85	6.65	0	0	0.68	0	1
8	05-03-2024 00:06	101.6	23.4	103076	5518	355	17.6	18	5.75	6.72	0	0	0.54	0	1
9	05-03-2024 00:07	102.3	23.3	103099	5518	359	9.6	18	5.86	6.7	0	0	0.89	0	1
10	05-03-2024 00:08	102.4	23.5	103123	5518	352	7.6	18	5.8	6.66	0	0	0.56	0	1
11	05-03-2024 00:09	102.4	23.4	103147	5517	353	9.7	18	5.83	6.63	0	0	0.78	0	1
12	05-03-2024 00:10	102.4	23.4	103169	5517	346	8.2	18	5.79	6.61	0	0	0.47	0	1
13	05-03-2024 00:11	101.9	23.5	103192	5517	350	18.6	18.1	5.85	6.68	0	0	0.39	0	1
14	05-03-2024 00:12	102.4	23.5	103216	5517	352	8.7	18	5.89	6.57	0	0	0.29	0	1
15	05-03-2024 00:13	102.3	23.5	103238	5518	347	9.5	18	5.84	6.61	0	0	0.7	0	1
16	05-03-2024 00:14	102.4	23.5	103263	5517	345	9.5	18.1	5.88	6.6	0	0	0.76	0	1
17	05-03-2024 00:15	102.4	23.5	103287	5517	353	10.6	18.1	5.84	6.59	0	0	0.34	0	1
18	05-03-2024 00:16	102.1	23.5	103311	5516	344	11.4	18	5.85	6.7	0	0	0.39	0	1
19	05-03-2024 00:17	102.1	23.5	103335	5516	351	8.7	18	5.86	6.63	0	0	0.12	0	1
20	05-03-2024 00:18	102	23.5	103358	5517	357	9.9	18	5.87	6.63	0	0	0.75	0	1

表 3 –クリーニングされた RO-RO データ[燃料、トリム、Avg_Draft]

	BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP	BQ	BR	BS
1	LCD_ABNC	LCD_ABNC	ILS_ABNOI	SENSOR_A	INITIAL_SE	STAND_BY	STAND_BY	ILS_ETH_LI	ILS_ETH_LI	frag_port	Fuel_Liters	Trim	Avg_Draft
2	0	0	0	0	0	0	0	0	0	0		0.93	6.265
3	0	0	0	0	0	0	0	0	0	0	24	0.87	6.235
4	0	0	0	0	0	0	0	0	0	0	23	0.92	6.16
5	0	0	0	0	0	0	0	0	0	0	24	0.87	6.215
6	0	0	0	0	0	0	0	0	0	0	24	0.81	6.215
7	0	0	0	0	0	0	0	0	0	0	23	0.8	6.25
8	0	0	0	0	0	0	0	0	0	0	25	0.97	6.235
9	0	0	0	0	0	0	0	0	0	0	23	0.84	6.28
10	0	0	0	0	0	0	0	0	0	0	24	0.86	6.23
11	0	0	0	0	0	0	0	0	0	0	24	0.8	6.23
12	0	0	0	0	0	0	0	0	0	0	22	0.82	6.2
13	0	0	0	0	0	0	0	0	0	0	23	0.83	6.265
14	0	0	0	0	0	0	0	0	0	0	24	0.68	6.23
15	0	0	0	0	0	0	0	0	0	0	23	0.77	6.225
16	0	0	0	0	0	0	0	0	0	0	25	0.72	6.24
17	0	0	0	0	0	0	0	0	0	0	24	0.75	6.215
18	0	0	0	0	0	0	0	0	0	0	24	0.85	6.275
19	0	0	0	0	0	0	0	0	0	0	24	0.77	6.245
20	0	0	0	0	0	0	0	0	0	0	24	0.76	6.25

これで、このクリーニングされたデータセットを CII 予測モデルの構築とテストに使用する準備が整いました。

[5-5] 股関節・燃料情報

計算には、会社から提供されたデータと国際基準に基づいて、船とその燃料に関する重要な情報を使用しました。

- **総トン数(GT):**

  RO-RO 船の総トン数は **14,052** トンです。この値は Tsuruta さんによって提供されました。総トン数は、船の全体的なサイズの尺度であり、CII の計算に使用されます。

- **燃料密度:**

  重油(HFO)の密度は **0.991 kg / L** です。この値は、国際海事機関(IMO)のガイドラインに由来します。燃料密度は、燃料の量(リットル)を質量(キログラム)に変換するために必要です。

- **CO_2排出係数:**

  HFO の排出係数は、**4 燃料 1g あたり 3.114g の CO_2**です。これも IMO 規格に基づいています。これは、燃焼した燃料のグラムごとに生成される二酸化炭素の量を示しています。

これらの値は、次のステップで船舶の総 CO_2排出量を計算し、最後に炭素強度指標(CII)を計算するために使用されます。

六. CO_2排出量と移動距離の算出

データを準備し、船舶と燃料の定数を設定した後、毎分 2 つの重要な値を計算しました。

[6-1] CO_2排出量(グラム/分)

まず、燃料の使用量をリットルからキログラムに換算し、燃料密度(0.991kg/L)を乗じました。次に、燃料の質量(kg)を乗じてグラム数(×1000)し、CO_2排出係数(3.114)を乗じて CO_2総排出量を算出しました。これにより、毎分生成される CO_2の量がグラム単位で得られます。

[6-2] Distance Traveled (nautical miles per minute)

船の速度はノット(時速海里)で記録されます。船が 1 分間にどれだけの距離を移動するかを調べるために、速度を 60 で割りました。

コード例:

```
fuel_kg = df['Fuel_Liters'] * FUEL_DENSITY
df['CO2_Emission_g'] = fuel_kg * 1000 * CO2_FACTOR # kg をグラムに変換します
df['Distance_NM'] = df['Ship_Speed'] / 60 # 毎分 NM
```

これらの計算は、CII 式で使用される船の年間 CO_2総排出量と総航行距離を計算するために必要な基本的な数値を提供するため、重要です。

七. データ集計: 月次、週次、日次サマリー

毎分あたりの CO_2排出量と距離を計算した後、次のステップは、より長い期間のデータをまとめることでした。これにより、傾向を把握し、機械学習にデータを使いやすくすることができます。

[7-1] 月次集計

まず、データを月ごとにグループ化しました。各月について、次の計算を行いました。

- CO_2総排出量
- 総航行距離
- 船速、風速、プロペラピッチ、ヒール、トリム、平均喫水の平均値

この月次サマリーは、monthly_summary.csv というファイルに保存しました。

表 4 - Montly データのサンプル

	A	B	C	D	E	F	G	H	I
1	Month_Ye	CO2_Emission_g	Distance_NM	Avg_Speed	Avg_Wind_Speed	Avg_CppPitch	Avg_Heel	Avg_Trim	Avg_Draft
2	2024-03	1999032238	6666.655	16.828613	10.72053094	21.37124406	0.21907	0.785416	6.564274
3	2024-04	2520487780	8526.071667	17.561425	8.153110196	21.5625781	0.262012	0.659416	6.494704
4	2024-05	2362590835	8309.736667	16.983486	8.475460708	21.47938822	0.276864	0.823285	6.583869
5	2024-06	432277066	1492.463333	17.310613	8.225517108	21.3944133	0.277153	0.931125	6.51605

しかし、4 ヶ月分のデータしかなかったため、データポイントは 4 つしか得られませんでした。これは、モデルに学習する十分な例がないため、優れた機械学習モデルのトレーニングには不十分です。

[7-2] ウィークリー集計

次に、データを週ごとにグループ化しました。各週について、上記と同じ合計と平均を計算しました。

この週次サマリーは、weekly_summary.csv というファイルに保存しました。

表 5-週次データのサンプル

	A	B	C	D	E	F	G	H	I
1	Week	CO2_Emis	Distance_I	Avg_Spee	Avg_Wind	Avg_CppP	Avg_Heel	Avg_Trim	Avg_Draft
2	2024-03-04/2024-03-10	4.97E+08	1738.87	17.17685	11.63645	21.9382	0.135395	0.799684	6.55308
3	2024-03-11/2024-03-17	5.53E+08	1850.453	16.73104	10.67337	21.12767	0.234491	0.725066	6.533424
4	2024-03-18/2024-03-24	5.43E+08	1870.695	16.82784	10.6127	21.36543	0.234525	0.750871	6.621056
5	2024-03-25/2024-03-31	4.07E+08	1206.637	16.49537	9.688152	20.96375	0.288066	0.909414	6.540116
6	2024-04-01/2024-04-07	6.45E+08	2151.95	17.84123	8.455935	21.62147	0.32102	0.502685	6.610689
7	2024-04-08/2024-04-14	6.3E+08	2132.135	18.0155	8.188551	21.76019	0.262815	0.593131	6.474259
8	2024-04-15/2024-04-21	6.24E+08	2109.315	17.79262	8.100872	21.63234	0.177	0.690583	6.444696
9	2024-04-22/2024-04-28	4.84E+08	1652.068	16.65671	8.553319	21.3882	0.31066	0.846273	6.452619
10	2024-04-29/2024-05-05	5.19E+08	1898.15	16.96544	7.906167	21.74271	0.225445	0.81073	6.621204
11	2024-05-06/2024-05-12	5.26E+08	1907.047	16.83926	8.60131	21.46999	0.269417	0.819079	6.585723
12	2024-05-13/2024-05-19	5.66E+08	1879.032	17.08987	8.480764	21.32054	0.295028	0.84692	6.54815
13	2024-05-20/2024-05-26	5.18E+08	1849.357	16.90712	7.953421	21.24973	0.310914	0.79446	6.561112
14	2024-05-27/2024-06-02	5.32E+08	1887.497	17.16426	8.603774	21.58472	0.261541	0.896419	6.536513
15	2024-06-03/2024-06-09	2.71E+08	861.7217	17.28051	8.210695	21.00231	0.280826	0.893322	6.530809

毎週の集計では、約 16 のデータポイントがありました。これは月次よりはましですが、それでも信頼性の高いモデルをトレーニングするには少量です。週次データは一部の分析に役立ちますが、それでも制限があります。

[7-3] 日次集計

最後に、データを日ごとにグループ化しました。各日について、総 CO_2排出量、総距離、その他の重要な特徴量の平均値を再度計算しました。

この日次サマリーは、daily_summary.csv というファイルに保存しました。

表 6 - 日次データのサンプル

	A	B	C	D	E	F	G	H	I	J	K	L
1	Day	CO2_Emission	Distance_N	Avg_Speed	Avg_Wind	Avg_CppPi	Avg_Heel	Avg_Trim	Avg_Draft	Annual_CO2	Annual_Di	CII
2	05-03-2024	92421835.33	334.34	17.84733	10.19146	22.02509	0.300863	0.627927	6.348332	33733969894	122034.1	19.67198
3	06-03-2024	127530961.5	379.1683	18.24387	15.89832	22.64499	0.251355	0.577362	6.487294	46548800956	138396.4	23.93566
4	07-03-2024	42660504.58	175.1667	15.14409	8.905908	19.35159	0.185476	0.85938	6.574344	15571084170	63935.83	17.33151
5	08-03-2024	67048957.1	235.6983	17.1417	9.885697	22.55442	-0.17327	0.851976	6.613879	24472869341	86029.89	20.24405
6	09-03-2024	101377331.9	369.5983	17.16401	11.6767	22.43576	-0.01469	0.973839	6.728204	37002726134	134903.4	19.51968
7	10-03-2024	65635581.01	244.8983	16.47298	11.18464	21.56244	0.228688	0.979854	6.57662	23956987067	89387.89	19.07284
8	11-03-2024	61170176.63	224.4367	15.51406	8.323618	20.17788	0.127857	0.835219	6.332056	22327114469	81919.38	19.3958
9	12-03-2024	70462044.34	221.7083	16.22256	10.26305	19.99305	0.218073	0.772207	6.471128	25718646185	80923.54	22.617
10	13-03-2024	143985374.9	383.25	18.45506	14.24494	22.26445	0.364374	0.693917	6.543082	52554661836	139886.3	26.7361
11	14-03-2024	44400993.91	174.8317	14.98557	6.167	18.68429	0.229729	0.655814	6.834021	16206362778	63813.56	18.07317
12	15-03-2024	59787660.28	214.6867	17.19786	6.110547	22.53044	0.240628	0.669172	6.678899	21822496001	78360.63	19.81839
13	16-03-2024	95726913.48	351.875	17.52075	11.48548	22.63842	0.10722	0.682863	6.511705	34940323420	128434.4	19.3601
14	17-03-2024	77099974.42	279.665	16.01135	14.03149	20.34294	0.326365	0.768712	6.457686	28141490662	102077.7	19.61905
15	18-03-2024	64475254.78	227.965	15.61404	17.63756	20.6008	0.119989	0.999932	6.278847	23533467995	83207.23	20.12736
16	19-03-2024	70347863.3	227.2867	17.00399	11.27419	20.55549	0.134813	0.615224	6.494345	25676970106	82959.63	22.02616
17	20-03-2024	131163152.9	391.76	18.3351	14.66388	22.32769	0.176747	0.503612	6.692445	47874550817	142992.4	23.82614
18	21-03-2024	43765283.27	172.9483	14.76088	11.29886	18.89559	0.360512	0.765206	6.837795	15974328393	63126.14	18.0084

日々のデータを使用することで、より多くのデータポイントを扱えるようになりました。これにより、モデルはより多くの例から学習し、より正確な予測を提供できるため、機械学習モデルのトレーニングにはるかに適しています。

結論:

3 つのオプションを比較した結果、モデルトレーニングに日次集計データを使用することを選択しました。このアプローチにより、詳細と機械学習のための十分なデータポイントとの間の最適なバランスが得られました。

八. 予測年間排出量、距離、CII の計算

日次サマリーデータを作成した後、1 年間の船舶の性能を推定する必要がありました。これにより、IMO の式を使用して、年間の炭素強度指標(CII)を予測できます。

[8-1] 年間排出量と距離の予測

各日について、1 日の合計に 365(年間の日数)を乗じて、年間 CO_2排出量と年間距離の予測値を算出しました。これにより、船が 1 年間毎日同じように運用された場合の船の排出量と距離の推定値が得られます。

- **年間 CO_2排出量:**

Annual_CO2=日量 CO2 排出量(g)×365Annual_CO2=日量 CO2 排出量(g)×365

- **年間距離:**

Annual_Dist=日距離(NM)×365Annual_Dist=日距離(NM)×365

[8-2] IMO の式による CII の計算

次に、IMO の式を使用して、各日のデータの CII を計算しました。

- **CII フォーミュラ:**

CII=年間 CO2 排出量（g)総トン数×年間距離（NM)CII=総トン数×年間距離（NM)年間 CO2 排出量（g)

この式は、年間に航行した各海里の貨物積載量あたりの CO_2排出量を示しています。

コード例:

ニシキヘビ

```
daily_df['Annual_CO2'] = daily_df['CO2_Emission_g'] * 365
daily_df['Annual_Dist'] = daily_df['Distance_NM'] * 365
daily_df['CII'] = daily_df['Annual_CO2'] / (GT * daily_df['Annual_Dist'])
daily_df.to_csv('daily_summary.csv', index=False)
```

これらの値を計算することで、機械学習モデルのトレーニングとテストに使用できるデータセットを作成し、日々の運用データに基づいて船舶の年間 CII を予測しました。

[8-3] 日々の燃料消費量の推移

船の燃料使用量が時間の経過とともにどのように変化するかをよりよく理解するために、毎日消費される総燃料を示す棒グラフを作成しました。

プロットの作り方:

- 各レコードのタイムスタンプから日付を抽出しました。
- 各日の燃料消費量の合計は、その日の燃料消費量をすべて合計して計算しました。
- この毎日の燃料合計を、主要な毎日の要約データと統合しました。
- 次に、各棒が特定の日に消費された総燃料を示す棒グラフをプロットしました。

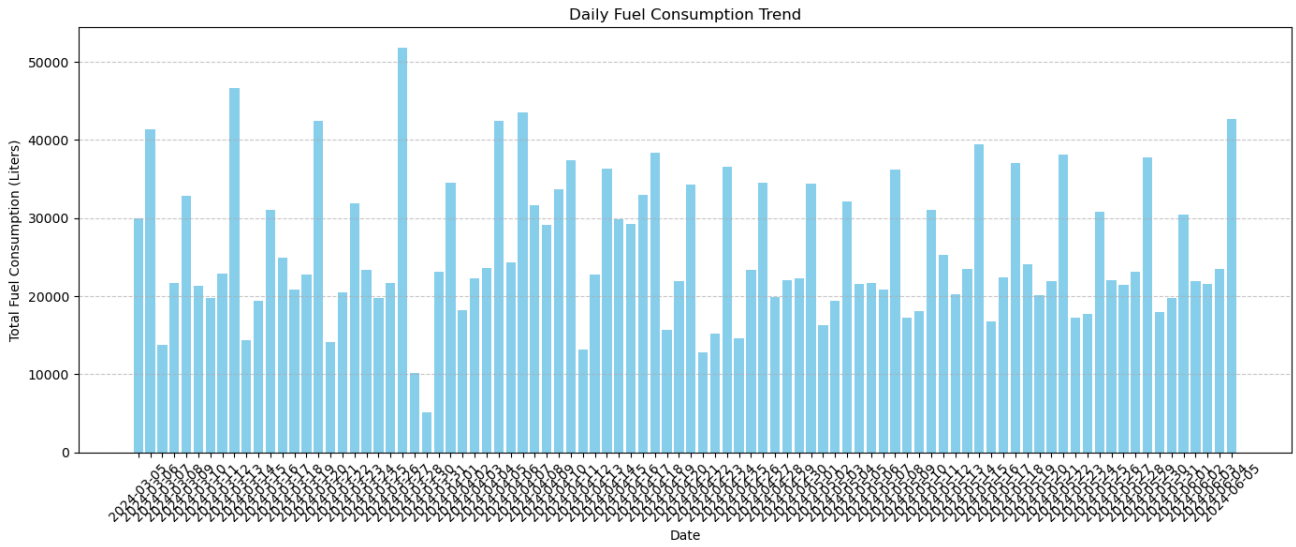

図 1:毎日の燃料消費の傾向(棒グラフ)

チャートが示すもの:

上のチャートは、RO-RO 船の毎日の燃料消費傾向を示しています。各バーは、特定の日に使用された燃料の合計量 (リットル単位) を表します。燃料消費量は日々変化し、ある日は他の日よりもはるかに多くの燃料を使用していることがわかります。これらの変更は、航海距離、船速、気象条件、運用上の変更などの要因によって引き起こされる可能性があります。

この傾向を見ることで、燃料使用量が異常に多い日や少ない日を特定でき、船舶の運航のパターンや問題を見つけるのに役立ちます。この種の分析は、エネルギー効率の向上や将来の航海計画に役立ちます。

[8-4] 日次 CII の配布

炭素強度指標(CII)の値が日々どのように変化するかを理解するために、日次 CII スコアのヒストグラムを作成しました。

プロットの作り方:

- 日ごとに計算された日次 CII 値を使用しました。
- ヒストグラムは、データセットに異なる CII 値が出現する頻度を示します。
- また、分布の全体的な形状を示すために、滑らかな線 (KDE) も追加しました。

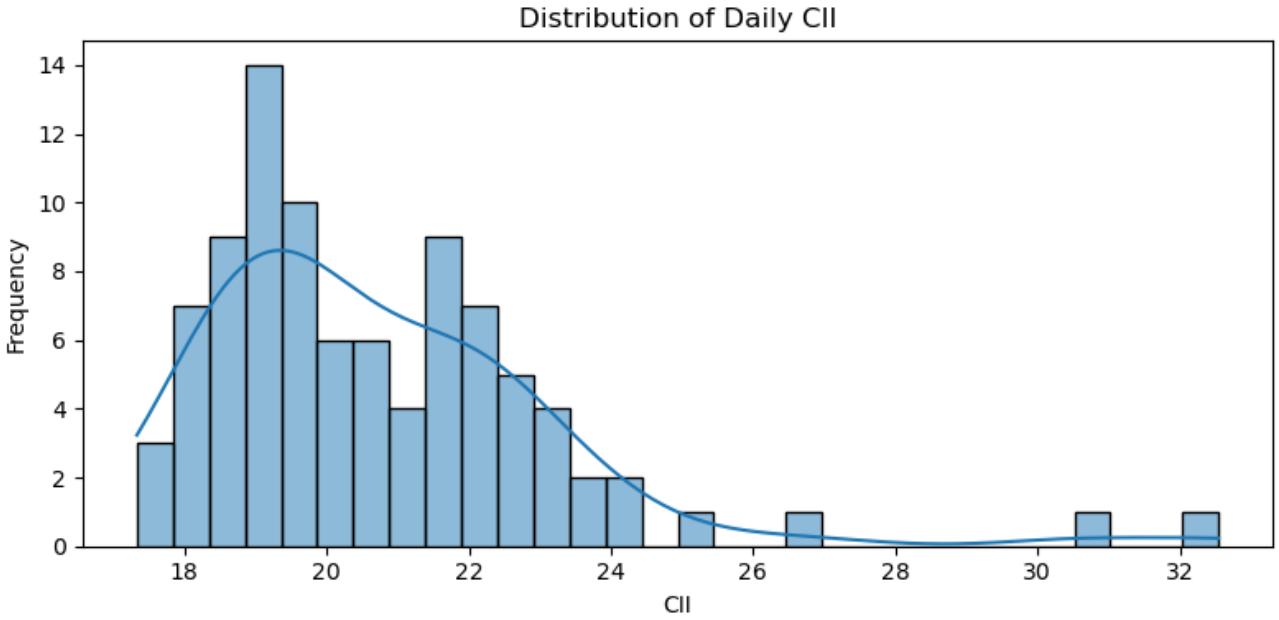

図 2 - 日次 CII 値の分布(ヒストグラム)

チャートが示すもの:

上のチャートは、RO-RO 船の日次 CII 値の分布を示しています。ほとんどの日の CII 値

は約 18 から 24 で、32 までの高い値を持つ日もあります。分布の形状はわずかに右に歪んでいるため、CII 値が低い日が多く、値がはるかに高い日が数日あります。

この種のプロットは、船の CII 値の一般的な範囲を確認し、スコアが非常に高いまたは低い異常な日を見つけるのに役立ちます。この分布を理解することは、現実的な目標を設定し、CII を正確に予測するように機械学習モデルをトレーニングするために重要です。

[8-5] 相関ヒートマップ

データセット内のさまざまな特徴が互いにどのように関連しているかを確認するために、相関ヒートマップを作成しました。

相関ヒートマップとは?

相関ヒートマップは、2 つの変数がどれだけ強く接続されているかを示すグラフです。値の範囲は -1 から 1 です。

- **1** は、完全な正の関係を意味します(一方が上がると、もう一方も上がります)。
- **-1** は、完全な負の関係を意味します (一方が上がると、もう一方が下がります)。
- **0** はリレーションシップがないことを意味します。

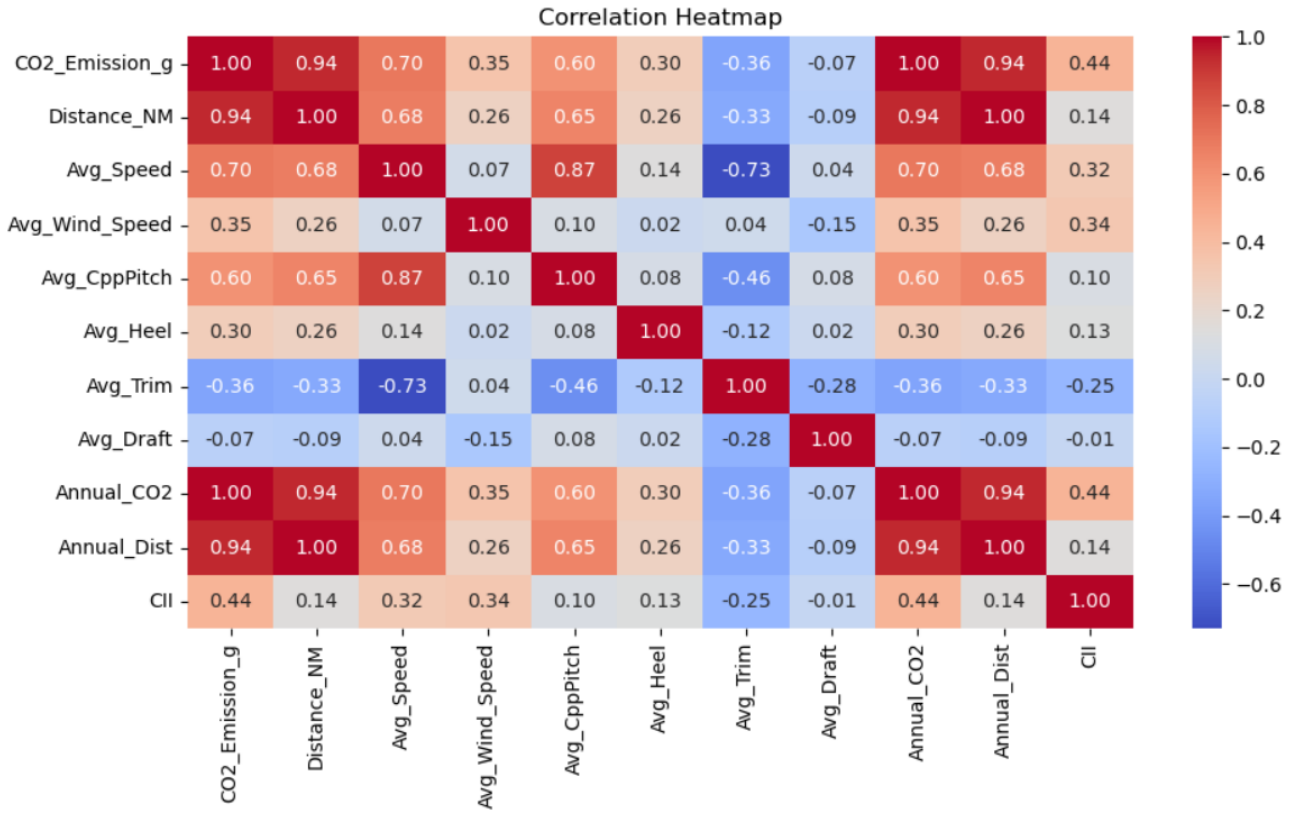

図 3 - 相関ヒートマップ

ヒートマップが示す内容:

- 上のヒートマップは、CO_2排出量、航行距離、平均速度、風速、プロペラピッチ、ヒール、トリムなどの重要な特徴間の相関関係を示しています。
- 例えば、**CO_2排出量**と**総燃料使用量**は非常に高い正の相関(1 に近い)を示しており、燃料を燃やす量が多いほど CO_2が増えるため、これは理にかなっています。
- **航行距離**と**年間距離**も強い正の関係にあります。
- 平均トリム**や**平均速度**などの一部の機能は**負の相関関係を示し、トリムが増加すると速度が低下する傾向があります。
- **CII** 値は、CO_2 排出量や燃料使用量などの特徴と中程度の正の相関を示し、他の機能とは弱いまたは負の相関を示します。

なぜこれが便利なのか:

このヒートマップは、どのフィーチャが最も密接に関連しているかを理解するのに役立ちます。この情報は、適切な特徴を選択し、互いに類似しすぎた特徴の使用を避けるのに役立つため、機械学習モデルを構築するために重要です。

これらの関係性を調べることで、船舶の運航のさまざまな側面がエネルギー効率と排出量にどのように影響するかをよりよく理解することができます。

[8-6] 日次 CII (時系列)

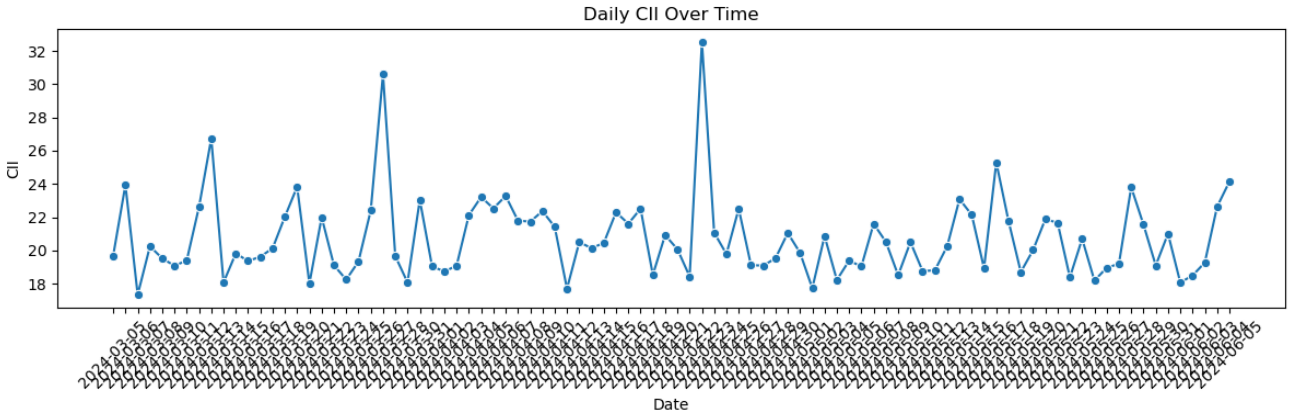

図 4 - 日次 CII の経時変化(折れ線グラフ)

上の折れ線グラフは、観測期間中、船舶の炭素強度指標(CII)が毎日どのように変化したかを示しています。グラフ上の各ポイントは 1 日の CII 値を表し、線はこれらのポイントを接続して、時間の経過に伴う傾向と変動を強調します。

グラフから、毎日の CII 値は一定ではなく、日ごとに大きく異なることがわかります。ほとんどの日、CII は 18 から 25 の間にとどまり、これがこの船の通常の運用範囲のようです。ただし、いくつかの顕著なスパイクがあり、CII 値が 30 を超える日もあります。これ

らの突然の増加は、燃料消費量の増加、航海距離の変化、気象条件、または特定の日の運用要因によって引き起こされる可能性があります。
この視覚化は、CII 値が異常に高いまたは低い日をすばやく見つけるのに役立つため、便利です。これらの外れ値を特定することで、船舶運航者は何が起こったのかを調査し、将来的に燃料効率を改善し、排出量を削減する方法を探すことができます。CII の動向を日々監視することは、IMO の規制を満たし、船舶全体のパフォーマンスを向上させるための重要なステップです。

九. モデル学習のための特徴選択

[9-1] 特徴選択

船舶の炭素強度指標(CII)を予測するモデルをトレーニングするために、毎日のサマリーデータから最も重要な特徴を選択しました。機械学習では、適切な特徴を選択することが非常に重要です。これは、モデルがターゲット値に実際に影響を与える要因（この場合は CII）に焦点を当てるのに役立つためです。
トレーニングに使用した機能は次のとおりです。

- CO_2排出量(g):その日の総二酸化炭素排出量。
- 航行距離(NM):その日の総航行距離は海里です。
- Average Speed: 日中の船の平均速度。
- Average Wind Speed: 日中に測定された平均風速。
- 平均 CPP ピッチ:平均制御可能なピッチプロペラ角度。
- 平均ヒール:船の左右への平均傾斜。
- 平均トリム:後部と前部の喫水の平均差。
- Average Draft(平均喫水):喫水線より下の船の平均深度。

これらの機能は、船の燃料消費量、排出量、および全体的なエネルギー効率に強い影響を与えるため、選択されました。
これらの列をモデルの入力特徴量（**X**）として設定し、ターゲット変数（**y**）は日次 CII 値です。

コード例:

```
機能 = ['CO2_Emission_g',
    'Distance_NM'、
```

```
    'Avg_Speed'、
    'Avg_Wind_Speed'、
    'Avg_CppPitch'、
    'Avg_Heel'、
    'Avg_Trim'、
    'Avg_Draft']
X = daily_df[特徴]
y = daily_df['CII']
```

これらの特徴を利用することで、船舶の運航や環境条件が炭素強度にどのように影響するかをモデルが学習するのに役立ちます。これにより、予測はより正確になり、現実世界の意思決定に役立ちます。

[9-2] 時系列データの学習-テスト分割

船舶の炭素強度指標 (CII) を予測する機械学習モデルを構築するには、モデルが新しい目に見えないデータでどの程度機能するかを確認する必要があります。これを行うために、データセットをトレーニングセットとテストセットの 2 つの部分に分割します。

データは時系列データ(日付順)であるため、データをランダムに分割することはできません。代わりに、以前のデータをトレーニングに使用し、最新のデータをテストに使用します。これは、過去の操作に基づいて将来の CII 値を予測したいという、実際のモデルの使用方法により適しています。

分割の方法:

- データの最初の 80%(最初の約 3 か月)をトレーニングセットとして使用しました。モデルはこのデータから学習します。
- データの最後の 20%(約過去 1 か月)をテストセットとして使用しました。このデータは、モデルが新しい未知の日の CII をどの程度正確に予測するかを確認するために使用されます。

時間ベースの分割を使用した理由:

- 時系列問題では、データの順序を尊重することが重要です。将来のデータを使用してモデルをトレーニングすることは現実的ではなく、オーバーフィッティングにつながる可能性があります。

- 過去に基づいてトレーニングし、未来でテストすることで、実際の予測を行うときにモデルがどのように機能するかをより適切に測定できます。

コード例:

split_index = int(len(daily_df) * 0.8)

X_train, X_test = X[:split_index], X[split_index:]

y_train、y_test = y[:split_index]、および[split_index:]

このアプローチにより、モデルが現実的な方法でテストされ、その結果の信頼性が高まり、実際の船舶運用に役立つようになります。

[9-3] フィーチャスケーリング

機械学習モデルをトレーニングする前に、すべての特徴が同様のスケールであることを確認する必要があります。私たちが使用する機能(CO_2排出量、距離、速度、喫水など)の単位と範囲は非常に異なるため、これは重要です。スケーリングしないと、値が大きい特徴がモデルに大きく影響し、小さな特徴が無視される可能性があります。

そこで、標準化という手法を採用しました。この方法では、平均が 0 で標準偏差が 1 になるように各フィーチャを変換します。このタスクには、scikit-learn ライブラリの StandardScaler を使用しました。

スケーリングの方法:

- スケーラーはトレーニングデータにのみ適合させたため、モデルはトレーニング中にテストデータを「見る」ことはありません。
- 次に、同じスケーラーを使用して、トレーニングデータとテストデータの両方を変換しました。

コード例:

スケーラー = スタンダードスケーラー()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

特徴をスケーリングすることで、モデルがより効果的に学習し、各特徴が予測に公正に寄与するようにします。この手順は、リッジ回帰などの正則化を使用するモデルでは特に重要です。

[9-4] さまざまな機械学習モデルのテスト

船の炭素強度指標(CII)を予測するための最適なモデルを見つけるために、いくつかの一般的な機械学習アルゴリズムをテストしました。各モデルには独自の利点があり、動作が異なります。

- **線形回帰:**
  データに直線を当てはめようとする単純なモデル。これは理解しやすく、フィーチャとターゲット間の関係がほとんど線形である場合にうまく機能します。
- **リッジ回帰:**
  線形回帰に似ていますが、大きな重みに対してペナルティが追加されます。これにより、オーバーフィットが防止され、特にフィーチャが相関している場合にモデルがより安定します。
- **ランダムフォレスト:**
  多くの決定木を構築し、その結果を平均化するアンサンブルモデル。ランダムフォレストは、複雑で非線形の関係をキャプチャでき、通常は優れた精度を提供します。
- **Support Vector Regressor (SVR):**
  マージン内で最適なラインを適合させようとするモデル。SVR は、線形と非線形の両方の関係を処理でき、小規模なデータセットでも適切に機能します。
- **XGBoost Regressor:**
  特別なブースティング手法を使用して決定木のアンサンブルを構築する強力で人気のあるモデル。XGBoost はその高いパフォーマンスで知られており、多くの機械学習コンペティションで広く使用されています。

[9-5] モデルの比較方法

これらのモデルのパフォーマンスを比較するために、クロスバリデーションと呼ばれる方法を使用しました。クロスバリデーションは、各モデルが新しい未知のデータに対してどの程度うまく機能するかについて、公正で信頼性の高い推定を得るのに役立つ手法です。私たちのプロジェクトでは、5 分割交差検証を使用しました。

- トレーニング データは 5 つの等しい部分 (フォールド) に分割されます。
- モデルは 4 つのフォールドでトレーニングされ、残りのフォールドでテストされます。

- このプロセスは 5 回繰り返され、毎回異なるフォールドがテスト セットになります。
- これらの結果の平均から、モデルの真のパフォーマンスを把握できます。

このアプローチにより、オーバーフィットを防ぎ、モデルの選択が実際の再現性のある結果に基づいて行われるようになります。

コード例:

```
モデル = {
    '線形回帰': LinearRegression(),
    'リッジ回帰': ridge(),
    'ランダムフォレスト': RandomForestRegressor(random_state=0),
    'SVR': SVR(),
    'XGBoost': XGBRegressor(random_state=0)
}

cv = KFold(n_splits=5, shuffle=True, random_state=1)
結果 = {}
```

クロスバリデーションを使用することで、実際の船舶データに基づいて CII を予測するのに最適なモデルを自信を持って選択できます。

[9-6] モデルの比較と結果

さまざまな機械学習モデルのパフォーマンスを比較するために、二**乗平均平方根誤差 (RMSE)** と呼ばれるメトリックを使用しました。RMSE は、モデルの予測値が実際の値にどれだけ近いかを測定します。RMSE が小さいほど、モデルの予測の精度が高くなります。RMSE は、ターゲット変数 (この場合は炭素強度インジケーター (CII)) と同じ単位で平均誤差を示すため、回帰問題によく使用されます。

クロスバリデーションを使用していくつかの機械学習モデルをテストした結果、次の RMSE 値が見つかりました。

- **線形回帰:** RMSE = 1.25
- **リッジ回帰:** RMSE = 1.29
- **ランダムフォレスト:**RMSE = 2.14

- **サポート・ベクトル・リグレッサー (SVR):** RMSE = 2.33
- **XGBoost リグレッサー:** RMSE = 2.22

これらの結果は、**線形回帰**の RMSE が最も低かったことを示しており、データセット内で船舶の CII を予測するための最も正確なモデルとなっています。リッジ回帰も良好に機能し、線形回帰に近かったです。より複雑なモデル (Random Forest、SVR、XGBoost) は、この特定の問題とデータに対してはあまり機能しませんでした。

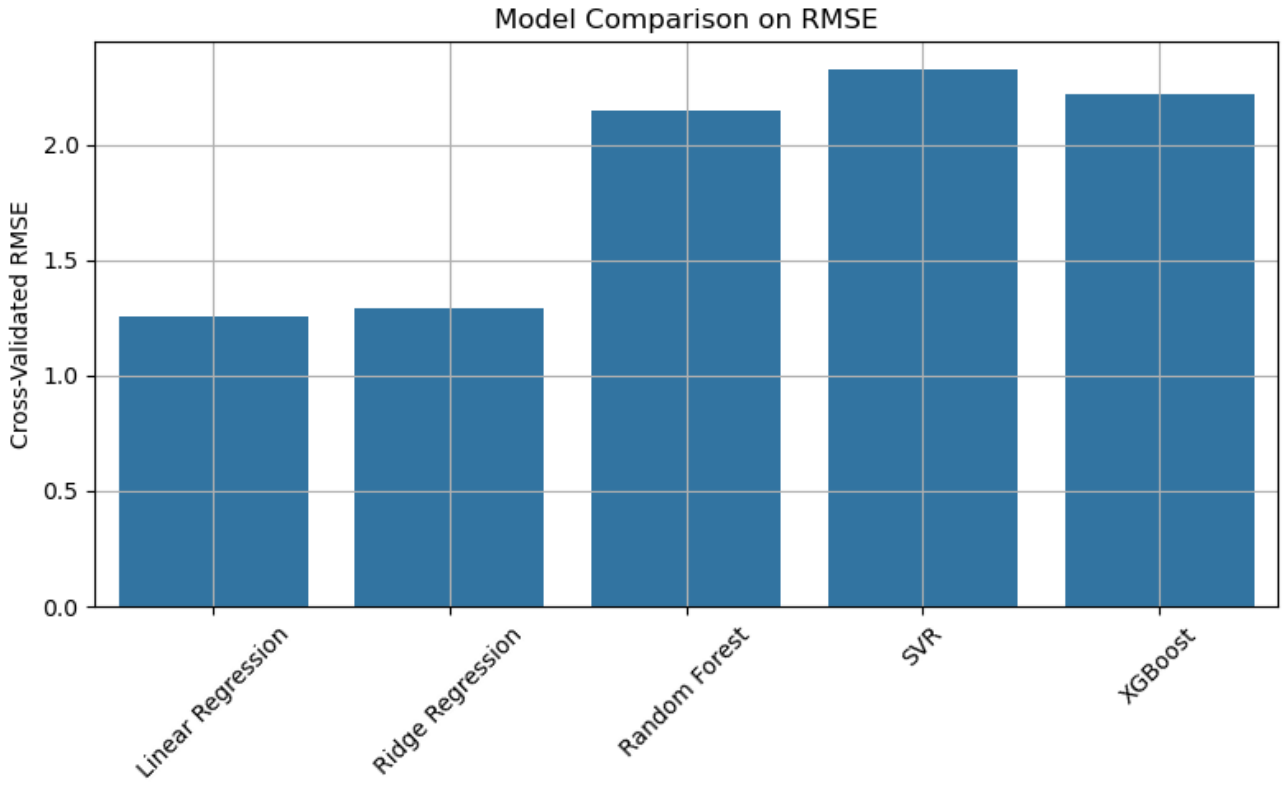

図 5 - モデル RMSE の比較(棒グラフ)

次に、ハイパーパラメータの調整を実行して、モデル、特に Ridge 回帰の設定を調整することでパフォーマンスを改善できるかどうかを確認します。このプロセスは、各モデルの可能な限り最適なバージョンを見つけるのに役立ちます。

機械学習モデルのデフォルト設定が特定のデータセットや問題に最適であることはめったにないため、初期の RMSE 結果を観察した後にハイパーパラメーターの調整が必要になります。モデルトレーニングの最初のラウンドは比較のベースラインを提供しますが、これらの結果には改善の余地が残されていることがよくあります。

ハイパーパラメータは、モデルの学習方法を制御する設定です (たとえば、リッジ回帰の

正則化強度やランダムフォレストのツリーの数など)。これらはデータから学習されるのではなく、トレーニングが開始される前に設定されます。これらの値を慎重に調整することで、モデルがより効果的に学習し、エラーを減らし、オーバーフィットやアンダーフィットなどの問題を回避できます。

ハイパーパラメータのチューニングでは、次のことができます。

- **モデルの精度を向上させる:** 微調整を行うと、モデルがデータにより適合するため、予測の精度が向上し、RMSE が低下する可能性があります。
- **ジェネラライゼーションの強化:** 最適化されたハイパーパラメーターは、トレーニング セットだけでなく、新しい目に見えないデータでもモデルが適切に機能するのに役立ちます。
- **バイアスと分散のバランス:** チューニングは、モデルが単純すぎたり (バイアスが高かったり) したり、複雑すぎたり (分散が高かったり) しないように、適切なバランスを見つけるのに役立ちます。

要約すると、ハイパーパラメータの調整は、モデルの可能性を最大限に引き出し、特定のタスクとデータに対して可能な限り最高のパフォーマンスを達成するための重要なステップです。そのため、初期の RMSE 値を確認した後、チューニングに移り、モデル(特にリッジ回帰)をさらに改善できるかどうかを確認します。

[9-7] ハイパーパラメータ調整後のモデル性能

各機械学習モデルのハイパーパラメーターを調整した後、次の 3 つの主要なメトリックを使用してパフォーマンスを比較しました。

- **二乗平均平方根誤差(RMSE):**予測誤差の平均サイズを測定します。値が小さいほど、精度が高くなります。
- **平均絶対誤差(MAE):**予測値と実際の値の平均絶対差。値が小さいほど良いです。
- **R^2スコア:**モデルがデータの分散をどの程度適切に説明しているかを示します。値が 1 に近いほど、パフォーマンスが向上することを意味します。

次の表は、各モデルの結果をまとめたものです。

表 7 - チューニング後のモデルパフォーマンスメトリクス (RMSE、MAE、R²)

メトリック	線形回帰	リッジ回帰	ランダムフォレスト	SVR の	XG ブート
RMSE の	0.6123	0.6073	0.9883	0.7647	1.2708
前	0.4815	0.4779	0.8665	0.5900	1.1250
R²スコア	0.8913	0.8930	0.7167	0.8304	0.5316

主な所見:

- **リッジ回帰**は、最も低い RMSE(0.6073)、最も低い MAE(0.4779)、および最も高い R²スコア(0.8930)で、全体的に最高の結果を達成しました。
- **Linear Regression** は僅差で 2 位で、パフォーマンスはほぼ同じでした。
- **Random Forest**、**SVR**、**XGBoost** は、チューニング後も、より単純な線形モデルよりも優れたパフォーマンスを発揮しませんでした。
- RMSE と MAE はどちらも Ridge と Linear Regression の方がはるかに低く、より正確で一貫性のある予測を示しています。
- リッジ回帰と線形回帰の R² スコアは、これらのモデルが CII 値の変動の大部分を説明することを示しています。

結論：

ハイパーパラメータの調整により、リッジ回帰のパフォーマンスが向上し、このプロジェクトで船舶の炭素強度指標 (CII) を予測するのに最適なモデルになりました。この結果は、適切に調整された線形モデルが、この種の海上運用データに対して単純かつ非常に効果的であることを示しています。

モデルのパフォーマンス メトリクスのプロット

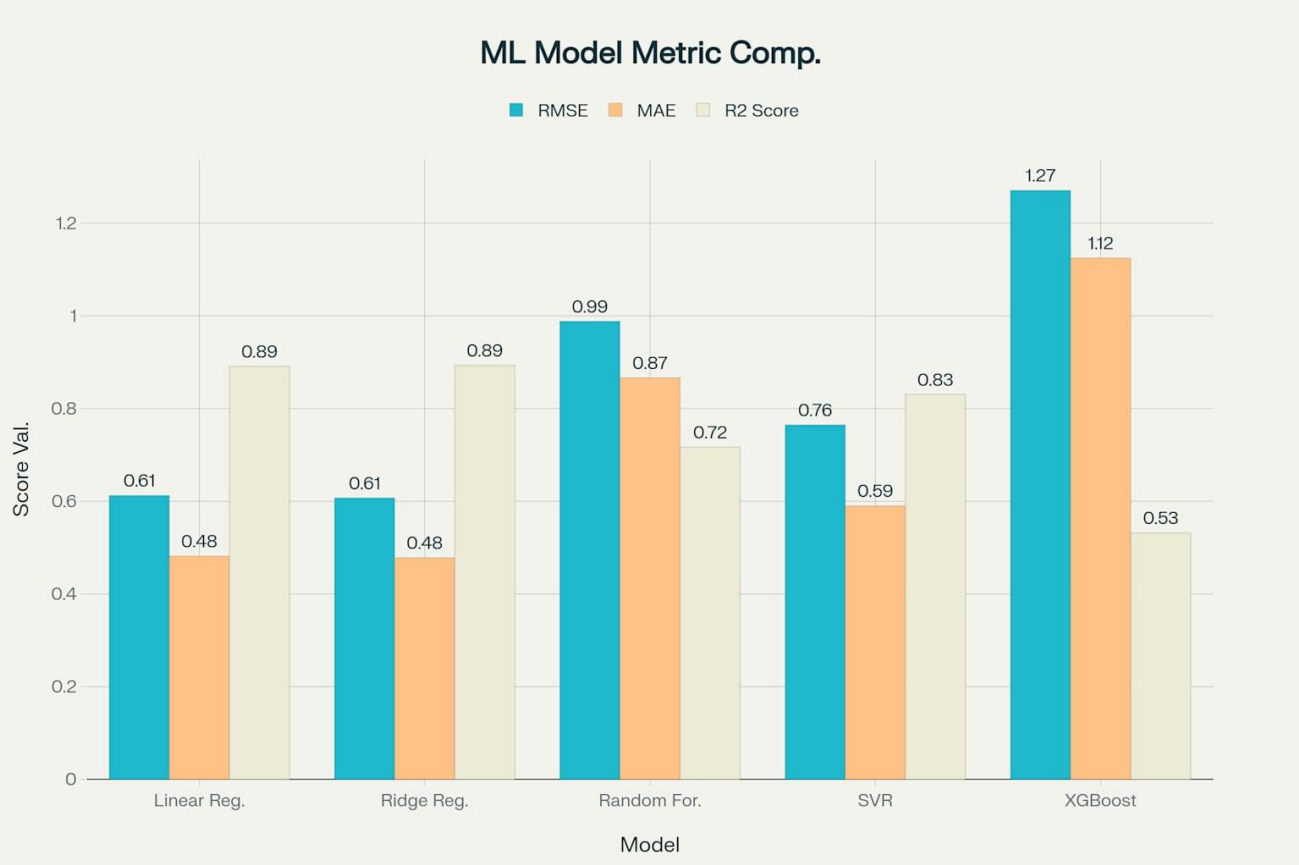

図 6 - モデルのパフォーマンス メトリック (グループ化された棒グラフ)

十. リッジ回帰:最終的なモデルの選択

[10-1] リッジ回帰は、大きな係数を使用するためにモデルにペナルティを追加する線形回帰の一種です。このペナルティは正則化と呼ばれ、モデルがトレーニングデータを過剰に適合させるのを防ぐのに役立ちます。つまり、データを記憶するだけでなく、新しい目に見えないデータによく一般化するパターンを学習します。

リッジ回帰の仕組み

- 標準の線形回帰と同様に、リッジ回帰は、入力フィーチャからターゲット (この場合は炭素強度インジケーター (CII)) を予測する最適なラインを見つけようとします。
- 違いは、リッジ回帰には、モデルが 1 つの特徴に過度に重要度を割り当てるのを防ぐ**正則化項 (アルファと呼ばれるパラメーターによって制御される) が含まれていることです。**
- この正則化により、特に特徴が相関している場合や、データ ポイントの数に比べて

特徴が多い場合に、モデルがより安定します。

Ridge 回帰が選択された理由

- いくつかの機械学習モデルをテストおよびチューニングした後、Ridge Regression はデータセット全体で最高のパフォーマンスを達成しました。誤差が最も低く (RMSE と MAE)、R^2スコアが最も高いため、CII の予測が最も正確で信頼性の高いものとなりました。
- また、Ridge Regression は、Random Forest や XGBoost などの複雑なモデルよりも優れたパフォーマンスを発揮しました。このモデルは、データセットがあまり大きくない場合にオーバーフィットすることがあります。

最後のステップ: モデルのトレーニング

リッジ回帰が最良の結果をもたらしたため、船の炭素強度指標 (CII) を予測するための最終モデルとして選択しました。Ridge Regression モデルは、利用可能なすべてのトレーニングデータと、チューニング中に見つかった最適なハイパーパラメーター設定を使用してトレーニングしました。これにより、当社の CII 予測ツールは正確かつ堅牢であり、実際の船舶の運航や意思決定に実用的な選択肢となっています。

最終リッジ回帰モデル: 学習、評価、および結果

[10-2] 最終モデルの学習

慎重な比較とハイパーパラメーターの調整を通じて、最もパフォーマンスの高いモデルとして Ridge Regression を選択した後、最適化されたアルファ値 (alpha = 1.0) を使用して最終的な Ridge Regression モデルをトレーニングしました。モデルはスケーリングされたトレーニングデータでトレーニングされ、テストデータとデータセット全体に対して予測が行われ、さらに分析されました。

モデル評価メトリクス

最終的な Ridge 回帰モデルのパフォーマンスを評価するために、次の 3 つの標準評価指標を使用しました。

- **二乗平均平方根誤差 (RMSE):** 0.6350 予測された CII 値と実際の CII 値の間の平均誤差サイズを示します。値が小さいほど、予測の精度が高くなります。
- **平均絶対誤差 (MAE):** 0.4730 予測された CII 値と実際の CII 値の平均絶対差を示します。

- **R^2スコア:** 0.8830 モデルが CII データの分散をどの程度適切に説明しているかを測定します。値が 1 に近いほど、パフォーマンスが優れていることを示します。

表 8 - リッジ回帰モデルの結果

メトリック	価値
RMSE の	0.6350
前	0.4730
R^2	0.8830

これらの結果から、リッジ回帰モデルが船舶の炭素強度指標（CII）に対して正確で信頼性の高い予測を提供することが確認されています。

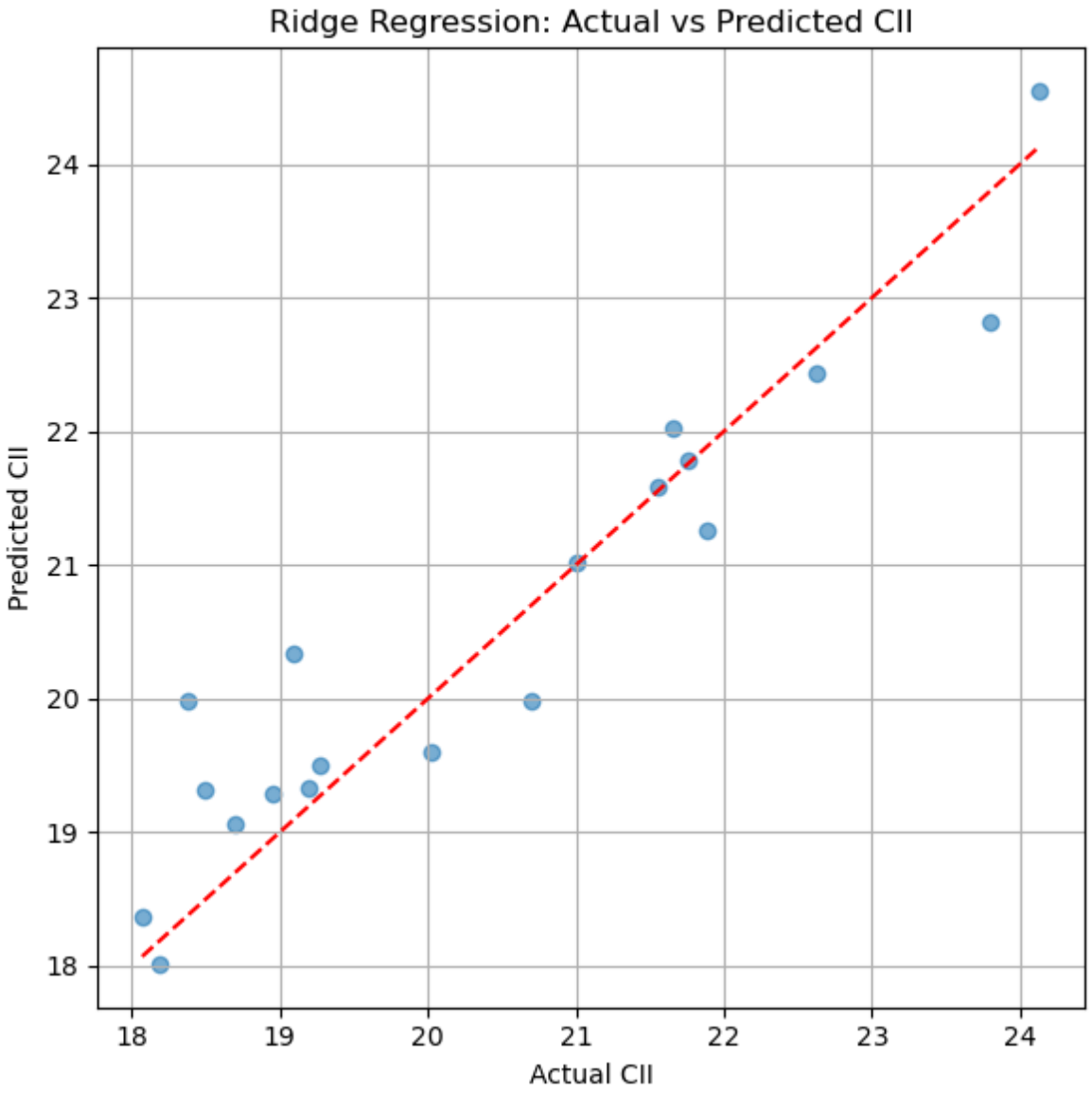

図 7 - 実際の CII と予測された CII(散布図)

実測値と予測値の CII プロット

モデルのパフォーマンスを視覚的に確認するために、実際の CII 値(テストセットから)とリッジ回帰モデルによって生成された予測 CII 値を比較する散布図を作成しました。赤い破線は、完全な予測を表します (実際が予測と等しい)。この線に近い点は、正確な予測を示します。

プロットは、ほとんどのポイントが線の近くに集まっていることを示しており、これはモデルの予測がほとんどの日の実際の CII 値と密接に一致していることを意味します。いくつかのポイントが離れており、時折予測エラーがあることを示していますが、全体としてモデルは強力な予測能力を示しています。

[10-3] まとめ

- 最適なハイパーパラメーターで学習させた最終的な Ridge 回帰モデルは、日次 CII 値の予測に強力なパフォーマンスを提供します。
- このモデルの予測は正確で一貫性があるため、エネルギー効率と規制コンプライアンスの監視と改善を目指す船舶のオペレーターや管理者にとって実用的なツールとなっています。
- このアプローチは、海運業務における炭素排出量を削減するための日常的な意思決定と長期計画をサポートするために使用できます。

ここで、データと IMO の公式式に基づいて CII の計算は正しいですが、実際に取得した CII 値は IMO の公式基準よりもはるかに高いことに言及することが重要だと思います。

[10-4] 計算した CII 値が高いのはなぜですか?

このプロジェクトでは、炭素強度指標(CII)の値は、IMO の公式式を使用して、船舶の運航データから直接計算されました。

CII=年間 CO2 排出量(g)/(総トン数×年間距離(NM))

分単位のセンサーデータを使用し、それを各日で合計し、365 を掛けて通年で予測しました。この方法は、公式の CII 評価や検証済みの年次レポートを使用するのとは異なります。

なぜ値が公式の基準よりも高いのですか?

計算された CII 値が IMO の公式値よりも高い理由はいくつかあります。

CII 値が高い主な理由

CII 値が高い最大の理由は、年間排出量と距離の予測方法にあると思います。このプロジェクトでは、船が年中無休で運航していると仮定して、日次 CII 値に 360 日(または 365 日)を掛けました。実際には、船は港や錨泊、整備に時間を過ごすことが多く、毎日航海し

ているわけではありません。この方法を使用すると、年間総燃料使用量と移動距離の両方を過大評価し、計算された CII 値が実際の CII 値よりもはるかに高くなります。

値が高くなるその他の理由

一. 短くて限られたデータ期間

- データセットは、1 年ではなく、数か月のみを対象としています。
- 1 日の数値に 365 を掛けると、特に記録された日に効率の悪い航海(悪天候、低速、港湾操作など)が含まれている場合、年間の合計が大きくなりすぎる可能性があります。

二. 比較のための公式の CII 評価はありません

- どの期間も、船の実際の公式の CII 評価はありません。
- これは、実際の結果と一致するように計算を確認または調整できないことを意味します。

三. センサーとデータのエラーの可能性

- 燃料使用量は、総燃料測定値の変化から計算されます。
- センサーデータのエラー、ジャンプ、ノイズがあると、CO_2数が高くなりすぎて CII が増加する可能性があります。

[10-5] なぜ CII レーティングの閾値を変更したのですか?

計算された CII 値は、Ro-Ro 船の実際の値(通常は 3.0～8.0)よりもはるかに高いため、公式の A/B/C/D/E レーティングバンドを使用すると、実際の改善が見られた場合でも、ほぼ毎日 E レーティングが得られます。

このプロジェクトで評価をより便利で理解しやすくするために、公式のしきい値を一定額スケールアップしました。たとえば、値が実際の値よりも約 4 倍高い場合、公式のしきい値に 4 を掛けます。これにより、評価が意味のあるものになり、パフォーマンスの違いを確認できます。

これが意味すること:

- 評価(A から E)が、計算された CII 値の範囲と一致するようになりました。
- 絶対数が多くても、日を比較して改善が見られることがあります。
- より多くのデータや公式の CII 評価が得られたら、しきい値を調整したり、実際のしきい値を使用したりできます。

最後のノート

このスケーリングは、データが限られていることと公式の CII 結果がないため、一時的な解決策にすぎません。より完全で正確なデータを取得すると、次のことが可能になります。

- スケーリングせずに CII 値を再計算します。
- 実際の IMO レーティングしきい値を使用

今のところ、このアプローチにより、数値が公式の IMO 値とまったく同じであるふりをすることなく、システムを使用して有用な結果を確認できます。

新しいレーティングバンドは次のとおりです。

- **A:**CII の≤15.75
- **B:**15.75 < CII≤18.20
- **C:**18.20 < CII≤20.45
- **D:** 20.45 < CII ≤ 22.70
- **E:**CII>22.70

これらのしきい値を使用して、A(最高)から E(最低)までの評価を各予測 CII 値に割り当てます。これにより、CII の絶対値が IMO の公式スケールよりも高い場合でも、船が好調に推移した日と改善が必要な日を迅速に確認することができます。

[10-6] 船舶運航の最適化提案

日々の炭素強度指標(CII)値を予測した後、次のステップは、船のエネルギー効率を改善し、排出量を削減するための実践的な提案を提供することでした。これらの提案は、トリム、ヒール、風速、プロペラピッチ、巡航速度など、毎日計算される主要な運用機能に基づいています。

提案の生成方法

毎日、システムは特定の運用要因が最適な範囲外にあるかどうかを確認します。その場合は、乗組員や船舶オペレーターがパフォーマンスを向上させ、CII を下げるための調整を

行うのに役立つ具体的な推奨事項を生成します。すべての主要要素が予想範囲内にある場合、パフォーマンスが十分であることが確認されます。

基準と提案

- **トリム:**
  平均トリム(船尾と前部の喫水差)が 1.0 メートルより大きい場合、「トリムを減らして燃料効率を向上させる」という提案です。
- **ヒール:**
  平均ヒール(左右の傾き)が 0.5 度より大きい場合、「バラストのバランスを取り、ヒールを減らす」ことをお勧めします。
- **風速:**
  平均風速が 12m/s を超える場合、「強風時の航行は避ける」ことをお勧めします。
- **CPP ピッチ:**
  平均制御可能なピッチプロペラ(CPP)ピッチが 15 度未満の場合、提案は「推進効率のために *CPP* ピッチを増やす」です。
- **巡航速度:**
  平均速度が 17 ノット未満の場合、「最適な巡航速度を維持する」ことをお勧めします。

これらの条件のいずれも満たされない場合、システムは「パフォーマンスは想定範囲内です」と報告します。

実用的な利点

これらの日々の最適化提案は、船舶運航業者を支援します。

- 運用上の変更により、燃料効率の向上と排出量の削減につながる領域を特定します。
- 船舶の CII 評価を維持または改善するために、積極的な措置を講じます。
- 航海計画と船上での調整に関する意思決定をサポートします。

このシステムは、実際の運用データに基づいて明確で的を絞った推奨事項を提供することにより、乗組員と管理者が規制要件を満たし、より持続可能な海運業務を達成することを容易にします。

十一. CII 予測モデルのデモンストレーション: ユーザー操作

炭素強度指標 (CII) 予測モデルが実際にどのように機能するかを示すために、現在のシステムでは、ユーザーが次の 2 つの簡単な方法でモデルを操作できます。

[11-1] 手動データ入力

ユーザーは、1 日の平均運用値をシステムに直接入力できます。必要な入力は次のとおりです。

- CO_2排出量(g)
- 航行距離(NM)
- 平均速度(ノット)
- 平均風速(m / s)
- 平均 CPP ピッチ(度)
- 平均ヒール(度)
- 平均トリム (メートル)
- 平均喫水 (メートル)

これらの値が入力されると、モデルはデータを処理し、その日の CII 値を予測し、パフォーマンス評価 (A-E) を割り当て、運用プロファイルに基づいて調整された最適化の提案を提供します。

出力例:

予測 CII: 18.4500 |レーティング C

提案:燃料効率を改善するためにトリムを減らします。最適な巡航速度を維持

[11-2] CSV ファイルによる一括予測

または、ユーザーは、数日間の毎日の運用データを含む CSV ファイルをアップロードできます。システムは次のことを行います。

- ファイルを読み取り、トレーニング済みの Ridge 回帰モデルを使用して各行を処理します。
- 各日の CII 値を予測します。
- 各予測にパフォーマンス評価 (A-E) を割り当てます。
- 船舶の効率を向上させるための最適化提案を毎日生成します。

結果は表に表示され、新しい CSV ファイルに保存されるので、簡単に確認してさらに分析

できます。

出力テーブルの例:

表 9 - 出力テーブルの例

Predicted_CII	格付け	提案
19.1200	C	トリムを減らして燃料効率を改善します
22.9000	E	強風時のセーリングは避けてください
16.8000	B	パフォーマンスは予想範囲内です

出力ファイル(cii_predictions_output.csv など)には、各日のデータに対するすべての予測、評価、提案が含まれています。

このインタラクティブなアプローチは、CII 予測ツールの実用的な価値を示しています:ユーザーは、運用上の選択が CII にどのように影響するかをすばやく確認し、実用的なフィードバックを受け取り、船舶のパフォーマンスと規制コンプライアンスを改善するための情報に基づいた決定を下すことができます。

[11-3] 年間 CII 予測モデル(モデル 1)のフロー図

[運用履歴データ(CSV/手動入力)]

↓

[データ前処理]

↓

[Feature Engineering & Selection]

↓

【モデル予測(ML モデル)】

↓

[年間 CII 計算および格付の割り当て]

↓

[最適化提案生成]

↓

[出力:予測 CII、評価、提案]

↓

[ユーザーインターフェース:手動入力またはファイルアップロード]

十二.　リアルタイムの炭素強度指標(CII)の計算とライブモニタリング(モデル 2)

[12-1] はじめに

このモデルは、分単位の船舶データを使用して、炭素強度指標(CII)をリアルタイムで計算するように設計されています。ライブセンサー入力をシミュレートすることで、船舶オペレーターは現在の CII を即座に確認し、効率を改善するための提案を即座に受け取ることができます。このアプローチは、船上での迅速な意思決定をサポートし、CII の傾向を年間を通じて追跡できる将来のダッシュボードの舞台を設定し、乗組員が効率的で規制に準拠するのに役立ちます。

[12-2] なぜこのアイデアを思いついたのか

過去の日次データを使用して年間 CII を予測する最初のモデルに取り組んでいる間、実際の運航中に船舶の炭素強度について**リアルタイムまたはほぼリアルタイムのフィードバック**を提供するツールを使用すると、船舶のオペレーターは大きなメリットが得られることに気づきました。月次または年次報告書を待つ代わりに、船の CII に関する即時の情報があれば、乗組員と管理者は燃料効率を改善し、排出量を削減するための調整をすぐに行うことができます。

このアイデアは、金融プラットフォームで株価がライブで更新されるのと同様に、船舶センサーからのライブデータストリームを使用してパフォーマンスを継続的に監視する方法を考えたことから生まれました。このようなシステムは次のようになります。

- 乗組員が変化する状況（天候や速度の変化など）に迅速に対応できるように支援します。
- 現在の運用データに基づいて、実行可能な提案を提供します。
- 現場での意思決定を改善し、エネルギー効率と環境規制への準拠を改善します。

- 将来的には、船舶のセンサーシステムと直接統合して監視を自動化するための基礎を築きます。

この動機から、既存の分単位の船のデータを、あたかも連続して入ってくるライブセンサーデータのように扱うモデルを開発しました。このモデルは、毎分 CII を計算し、最適化のための提案を即座に提供します。

使用するデータと機能

このモデルは、船の分単位の運用データから次の主要な機能を使用します。

- **FO_ME_Cons:**メインエンジンの燃料消費量(リットル)
- **FO_GE_Cons:**発電機エンジンの燃料消費量(リットル)
- **Ship_Speed:** 船の速度（ノット）
- **CppPitch:**制御可能なピッチプロペラ角度(度)
- **Wind_Speed:**風速(m / s)
- **HEEL:**船のヒール角度(度)
- **Fore_Draft:**船前部の喫水(メートル)
- **Aft_Draft:**船の後部の喫水(メートル)

これらの機能は、燃料消費量、CO_2排出量、移動距離、および瞬時の CII 計算に必要なその他の運用パラメータを計算するために使用されます。

[12-3] コード解説とワークフロー

1. 定数とインポート

まず、必要なライブラリをインポートし、定数を定義します。

- **総トン数(GT):**14,052(船サイズ)
- **燃料密度:**0.991kg/L(重油密度)
- **CO_2係数:**燃焼燃料 1 グラムあたり 3.114g の CO_2

これらの定数は、燃料消費量を CO_2排出量に変換するために使用されます。

2. 入力データの読み込み

モデルは、分単位の運用データを含む CSV ファイルを読み取ります。このファイルは、ライブ センサー データ入力をシミュレートし、各行は船舶の 1 分間の運航を表します。

3. 前処理機能

1 分ごとに、前処理関数は以下を計算します。

- **燃料消費量:**
  メインエンジンと発電機エンジンの両方について、前の分と現在の分の間の燃料消費量の読み取り値の差。負の差は、エラーを避けるためにゼロに設定されます。
- **トリムと平均ドラフト:**
  それぞれ後方とフォアドラフトの差と平均として計算されます。
- **CO_2排出量:**
  燃料消費量(リットル)をキログラムに換算し、CO_2排出係数を乗じて、1 分間に排出される CO_2のグラム数を求めます。
- **移動距離:**
  船の速度(ノット)を毎分海里に変換して計算されます。
- **瞬時 CII:**
  次の式を使用して計算されます。

$$CII = CO2\ 排出量\ (g) / (総トン数 \times 距離\ (NM))$$

移動距離がゼロの場合(たとえば、船が停止した場合)、CII は利用不可(NaN)として設定されます。

- **その他の機能:**
  平均速度、トリム、ヒール、風速、CPP ピッチ、平均喫水も抽出され、提案に使用できます。

処理中にエラーが発生した場合、この関数はすべての出力に対して NaN 値を返し、堅牢性を確保します。

4. 最適化の提案

generate_suggestion 機能は、現在の議事録の運用データに基づいて実用的なアドバイスを提供します。

- **トリム**が 1.0 m より大きい場合: トリムを小さくすることを提案します。
- **ヒール**が 0.5°より大きい場合:バラストのバランスをとることをお勧めします。
- **風速**が 12m/s を超える場合:強風下での航行は避けてください。
- **CPP ピッチ**が 15°未満の場合:CPP ピッチを上げることをお勧めします。
- **速度**が 17 ノット未満の場合:最適な巡航速度を維持することを提案します。

これらの条件のいずれも当てはまらない場合は、パフォーマンスが予想範囲内にあると報告されます。

5.ライブシミュレーションループ

このモデルは、データセット内の毎分を反復処理することで、ライブ データ処理をシミュレートします。

- 1 分ごとに (2 行目から開始)、現在の行と前の行を処理して、瞬時の CII と提案を計算します。
- 分番号、計算された CII、および提案を出力します。
- 短い遅延 (time.sleep(1)) は、リアルタイムのデータ到着をシミュレートします。

ライブシミュレーションからのサンプル出力

--- LIVE CII 計算開始---

分 1:インスタント CII = 17.7663 |提案:バランスバラストを使用してかかとを減らす

分 2:インスタント CII = 16.8368 |提案: パフォーマンスが想定範囲内にある

分 3:インスタント CII = 17.5689 |提案: パフォーマンスが想定範囲内にある

分 4:インスタント CII = 17.4718 |提案:バランスバラストを使用してかかとを減らす

分 5:インスタント CII = 16.7438 |提案:バランスバラストを使用してかかとを減らす

分 6:インスタント CII = 18.3009 |提案:かかとを減らすためにバラストのバランスを取ります。強風時のセーリングは避けてください

分 7:インスタント CII = 16.8368 |提案:バランスバラストを使用してかかとを減らす

分 8:インスタント CII = 17.5689 |提案:バランスバラストを使用してかかとを減らす

分 9:インスタント CII = 17.5689 |提案:バランスバラストを使用してかかとを減らす

10 分目:インスタント CII = 16.1048 |提案: パフォーマンスは予想範囲内です

---ライブ CII シミュレーションコンプリート---

[12-4] このモデルがデータをライブセンサー入力として扱う方法

現在、このモデルは CSV ファイルに保存されている履歴データを使用していますが、船舶からリアルタイムでライブセンサーデータを受信するのと同じように、データを分単位で処理します。このアプローチにより、次のことが可能になります。

- 船舶の炭素強度のリアルタイム監視をシミュレートします。
- 運用上の改善のためのフィードバックと提案を即座に提供します。
- 実際のセンサーに接続する前に、既存のデータを使用してモデルのロジックをテストおよび検証します。

[12-5] 船舶用センサとの将来統合

将来的には、このモデルを船の搭載センサーシステムと直接統合して、次のことを行うことができます。

- 燃料消費量、速度、喫水、環境条件のライブデータストリームを受信します。
- 瞬時 CII を遅延なく連続的に計算します。
- 最適化の提案を自動的に生成し、乗組員やブリッジオフィサーに表示します。
- 船舶がエネルギー効率を向上させ、環境規制にリアルタイムで準拠できるよう支援します。

このような統合により、このシステムは運用上の意思決定支援のための強力なツールとなり、燃料の使用と排出量の積極的な管理が可能になります。

概要

この即時 CII 計算モデルは、船舶の炭素強度に関するリアルタイムの洞察を提供することにより、最初の長期予測モデルを補完します。分単位の運用データを活用して、CII を継続的に計算および報告し、効率を向上させるための実用的な提案を提供します。

このモデルは、ライブデータ処理をシミュレートし、将来のセンサー統合に備えることで、船舶オペレーターが航海中の環境パフォーマンスを毎分監視および最適化するのに役立つ実用的でユーザーフレンドリーなツールの基盤を築きます。

[12-6] リアルタイム CII 計算モデル(モデル 2)のフロー図

[ライブセンサーデータストリーム/分単位 CSV]

↓

[データ取り込み(行ごと)]

↓

[各行の前処理]

↓

[瞬時 CII 計算(分単位)]

↓

[リアルタイム最適化の提案]

↓

[出力: 瞬時 CII 値 + 提案]

↓

【ライブダッシュボード/リアルタイム表示】

十三. リアルタイムの通年 CII 監視および予測ツールのビジョン

私の主なアイデアは、一瞬または 1 日の炭素強度指標(CII)を計算するだけでなく、船舶運航者向けのスマートシステムを構築することです。私は、CII の「株価ティッカー」のように機能し、年間を通じて船のカーボンパフォーマンスを絶えず更新、追跡、予測するツールを作成したいと考えています。

[13-1] システムの仕組み

- **継続的なデータ収集:**
  このシステムは、燃料使用量、速度、喫水、天候など、船から一年中、分単位でライブセンサーデータを取得します。
- **インスタント CII 計算:**
  いつでも、システムは、インスタントモデルが現在行っているのと同じように、現在の CII 値を表示できます。これにより、乗組員は、船が現在どれだけ効率的に稼働しているかについてすぐにフィードバックを得ることができます。

- **CII 履歴とトレンドの視覚化:**
  システムは、年の初めからすべての CII 値を記憶します。チャート上で株価が上下に動いているのを見るのと同じように、CII が時間の経過とともにどのように変化したかを示すグラフが表示されます。これにより、乗組員とマネージャーは、傾向、改善、または問題が発生したときにそれを見つけることができます。
- **ライブ年間予測:**
  これまでに収集されたすべてのデータを使用して、システムは年間全体の予測 CII も計算および更新します。つまり、3 月や 7 月であっても、同じように運用が続けば、年間の CII がどうなるかを確認できます。システムのデータが多いほど、この予測の精度は高くなります。
- **複合モデルアプローチ:**
  インスタント CII 計算(現在の瞬間)と累積予測(現在までのすべてのデータを使用)の両方を統合することにより、ツールは以下を表示します。
  - 現在の CII
  - これまでのすべての事業に基づく予測年間 CII

[13-2] なぜこれが役立つのか

- **リアルタイムの認識:**
  乗組員は、自分の行動が CII にどのように影響するかを即座に、そして時間の経過とともに確認できるため、毎日より良い意思決定を行うことができます。
- **早期警告:**
  予測される年間 CII が許容できないレベルに向かって上昇し始めた場合、乗組員は年末まで待つのではなく、早期に行動を起こすことができます。
- **モチベーションと透明性:**
  CII のトレンドをライブで見ることで、株価を見ると投資家のモチベーションが上がるのと同じように、誰もがパフォーマンスを認識することができます。
- **より良い計画:**
  マネージャーは、履歴データと予測された CII データを使用して、メンテナンスの計画、運用の調整、および規制へのコンプライアンスを確保できます。

最終的な考え

このビジョンは、基本的な CII 計算を強力な常時稼働の監視および予測ツールに変えることです。即時のフィードバックと長期的な洞察を組み合わせることで、このシステムは実際の出荷業務にとってはるかに便利で実用的なものになります。

十四. このプロジェクトが中北にどのような利益をもたらし、将来の成長を支えるか

[14-1] 中北への直接価値

これらの CII 計算および監視モデルの開発は、中北にいくつかの重要な利点をもたらします。

- **イノベーションリーダーシップ:**
  中北は、炭素強度をリアルタイムで予測的に監視するためのツールを構築することにより、特に海事の脱炭素化が世界的な優先事項となる中、最新の業界ニーズを革新し、対応する能力を示しています。
- **顧客価値:**
  これらのモデルはプロトタイプであっても、改良して最終的に中北の顧客に提供でき、船舶の環境パフォーマンスの追跡、管理、改善に役立ちます。これにより、クライアントは IMO 規制を満たし、運用効率を向上させることができます。
- **フィードバック主導の改善:**
  改良されたツールを顧客に展開することで、中北は実際のフィードバックを収集することができます。このフィードバックは、製品をさらに強化するために使用できるため、ユーザーのニーズを満たし、競争力を維持することができます。
- **新たなビジネスチャンス:**
  成熟したソリューションは、パッケージ化して SaaS(Service as a Service)として販売することが可能となり、Nakakita の新たな収益源となり、デジタルサービスを拡大することができます。

[14-2] ナカキタのその他のユースケース

- **既存製品との統合:**
  このモデルは、ナカキタの現在の制御および監視システムに統合することができ、

既存のお客様に付加価値を与え、ナカキタの製品を競合他社と差別化することができます。

- **データドリブンなメンテナンスとプランニング:**
  これらのツールを通じて収集された継続的なデータは、予知保全、航海計画、燃料最適化サービスをサポートします。
- **ベンチマーキングと分析:**
  中北は、集約された匿名化された CII データを使用して業界のベンチマークを提供し、お客様が自社のパフォーマンスを類似の船舶と比較してどのように比較するかを理解するのに役立ちます。

[14-3] 中北ブランドの強化

これらの環境モニタリングツールを開発・提供することで、中北は、初期のシンプルな形であっても、海事の脱炭素化を支援する先進的な企業としての地位を確立しています。このプロアクティブなアプローチでは、次のことを行います。

- 中北のコンプライアンスだけでなく、環境ソリューションに対する真のコミットメントを示しています。
- サステナビリティのパートナーとしてのナカキタの評判を高め、新規顧客の獲得や既存顧客との関係強化を実現します。
- 責任感があり、革新的で、顧客志向の企業としてのブランドのイメージをサポートします。

[14-4] 次のステップ

これは単なるプロトタイプであることを強調することが重要です。モデルとアプリケーションは、商用リリースの準備が整う前に、さらなる開発、テスト、および改良が必要になります。ツールが改善され、実際のユーザーからより多くのフィードバックが収集されるにつれて、Nakakita はソリューションを継続的に強化し、急速に進化する海事業界で真の価値を提供し、一歩先を行くことができます。

これらのデジタルソリューションに投資することで、中北は顧客を支援するだけでなく、

海事技術と持続可能性のリーダーとしての地位を確固たるものにしています。

十五. オンラインインターンシップのビジョン:CII の監視と予測のための Web アプリケーションの構築

インターンシップのオンラインフェーズでは、私が開発した機械学習モデルを、船舶のオペレーターや管理者が炭素強度指標(CII)のパフォーマンスを監視および改善するのに役立つ実用的でユーザーフレンドリーな Web アプリケーションに変換することを主な目標としています。

2 つの Web アプリケーション: 目的と優先順位

[15-1] リアルタイム CII 監視 Web アプリケーション(モデル 2)

- **目的:**
  このアプリケーションは、私が開発したリアルタイムモデルと同様に、ライブ CII 計算を表示します。受信データを分単位(または可能な限り頻繁に)処理し、現在の CII 値を表示し、運用改善のための提案を即座に提供します。
- **顔立ち：**
  - 株式ティッカーに似た CII 値のライブ更新。
  - CII のトレンドを経時的に視覚化し、年間を通じてパフォーマンスを追跡できるようにします。
  - 即時のフィードバックと実行可能な提案により、乗組員はその場で効率を最適化できます。
- **優先事項:**
  これは私のオンラインインターンシップの最優先事項です。このアプリケーションは、日常の船舶の運用と意思決定に最も直接的な価値を提供するため、最初に構築することに焦点を当てる予定です。

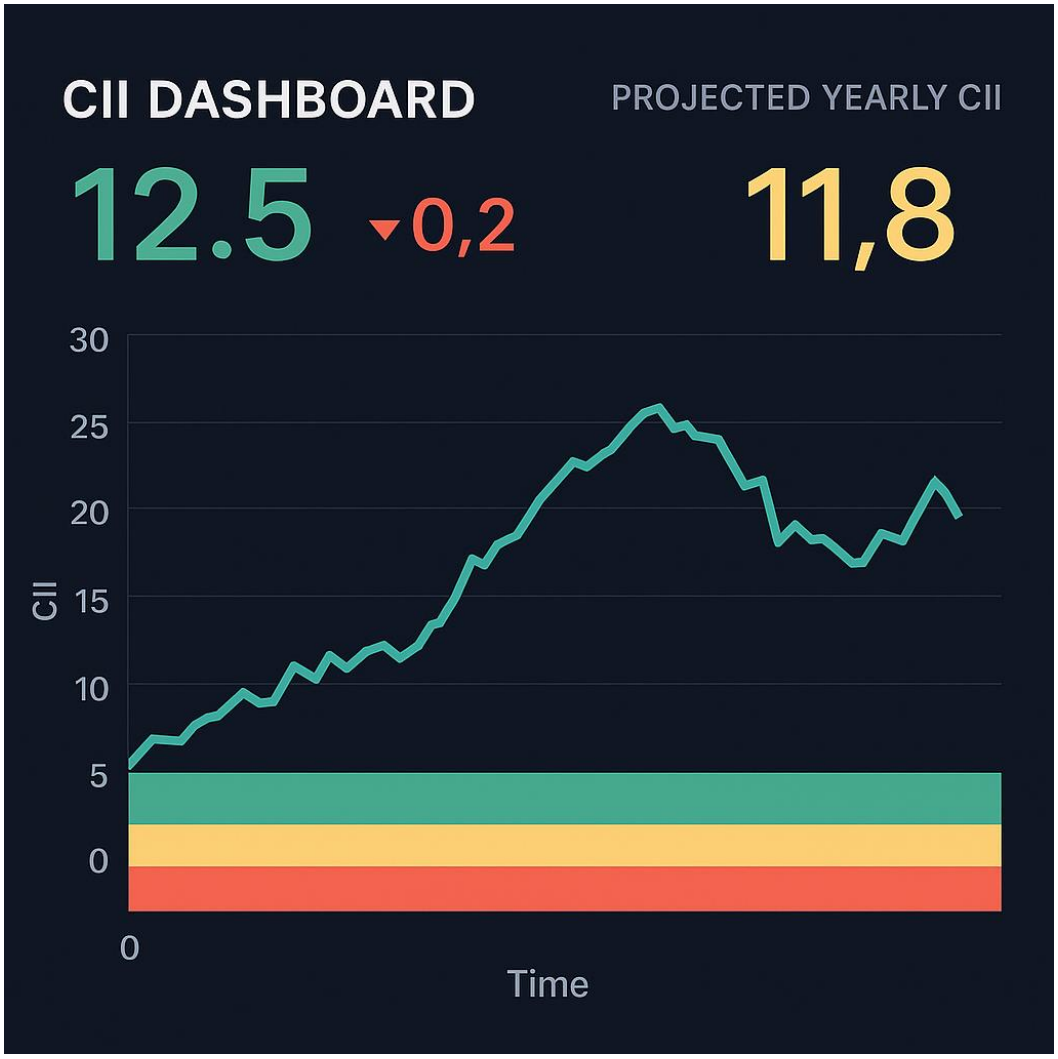

図 8 - リアルタイム CII ダッシュボードのモックアップ（概念）

[15-2] CII 予測・分析 Web アプリケーション(モデル 1)

- **目的:**

  このアプリケーションを使用すると、ユーザーは運用データを(手動または CSV ファイルを介して)アップロードし、年間予測モデルを使用して CII 予測を受け取ることができます。これは、より詳細な分析、計画、およびレポート作成のために設計されています。

- **顔立ち：**
  - 手動データ入力フォームと CSV アップロードオプション。
  - CII 予測のための複数日分のデータのバッチ処理。
  - 各エントリのパフォーマンス評価と最適化の提案。
  - 詳細なレビューやコンプライアンスの文書化のためにダウンロード可能な結

果。

- **優先度:**

  時間が許せば、リアルタイム監視ツールが完成した後、このアプリケーションに取り組みます。私の目標は、インターンシップ期間内に両方のアプリケーションを完了することですが、少なくとも 1 つの完全に機能する製品を確保するために、最初にリアルタイムツールを提供することに焦点を当てます。

コミットメントとアプローチ

- 私は、インターンシップ中に両方の Web アプリケーションを完了するために最善を尽くすことを約束します。
- 時間が制約となる場合は、リアルタイム CII 監視アプリケーションが完全に開発され、テストされていることを確認して、最も迅速で実用的なメリットを提供します。
- プロセス全体を通じて、明確でシンプルなユーザー インターフェイスと信頼性の高い機能を優先し、あらゆるレベルの技術的背景を持つユーザーがツールにアクセスできるようにします。

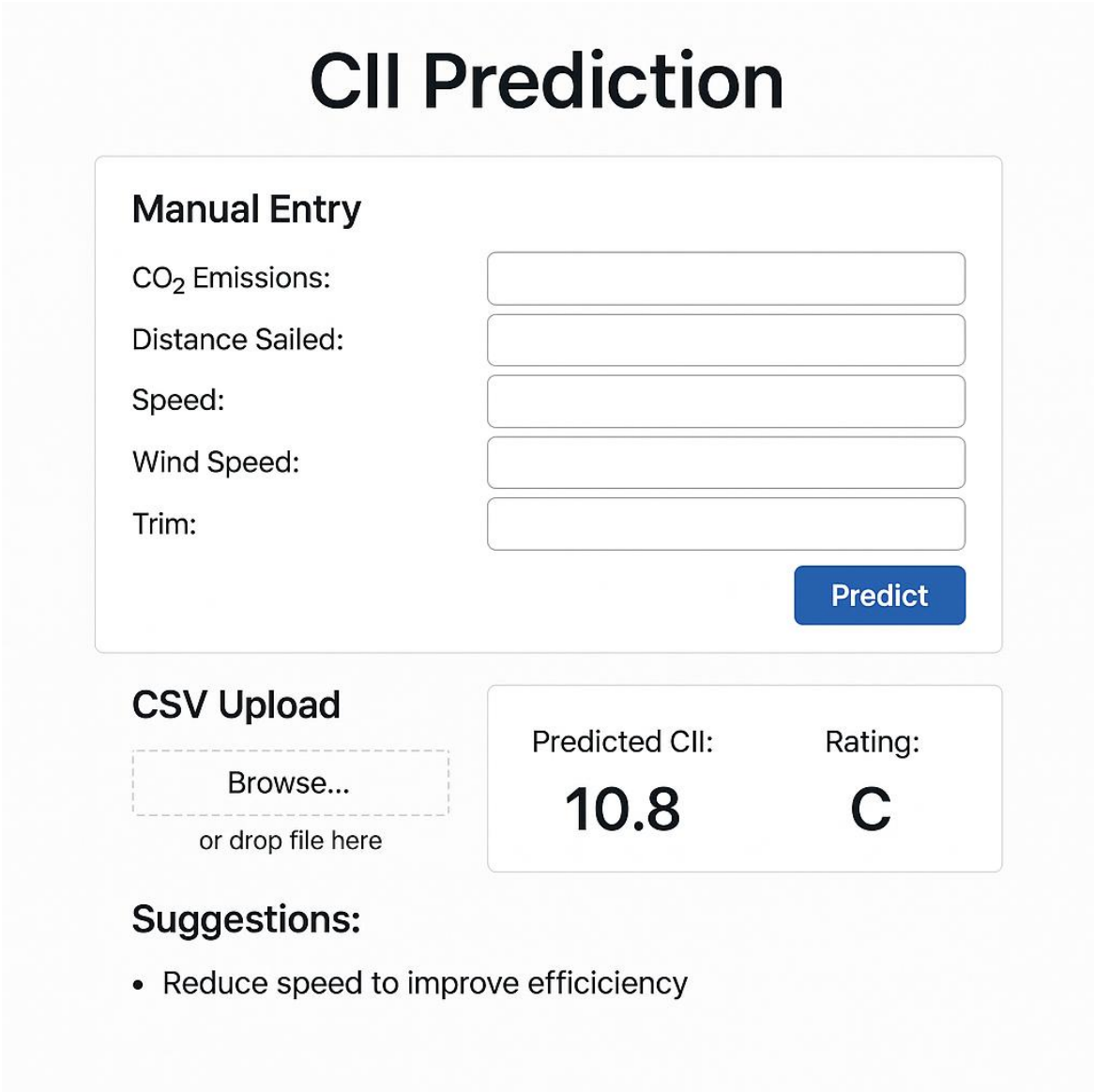

図 9 - CII 計算ダッシュボードのモックアップ（概念）

十六.　付録:完全なソースコード

[16-1] 付録 A: 年間 CII 予測モデル – フル コード

Pandas を PD としてインポート
Seaborn を SNS としてインポート
matplotlib.pyplot を plt としてインポートします
sklearn.model_selection インポート train_test_split から
from sklearn.preprocessing import スタンダードスケーラー
sklearn.linear_model からインポート LinearRegression、Ridge
sklearn.ensemble から RandomForestRegressor をインポートします
sklearn.svm から SVR をインポートします
xgboost import XGBRegressor から
sklearn.model_selection import cross_val_score から、kfold
sklearn.metrics からインポート mean_squared_error、r2_score
numpy を np としてインポート
sklearn.model_selection から GridSearchCV をインポートします
sklearn.metrics からインポート mean_absolute_error
From Tabulate インポート タブレート

df = pd.read_csv("sample_data_3.csv")

df.head()
# df.isnull().sum()

#Ensures pandas は、列を単なるテキストではなく、日付/時刻として認識します。だからそれを pd datetime フレームに隠します
df['時間'] = pd.to_datetime(df['時間'])
df = df.sort_values('時間')

print("開始時間:", df['時間'].min())
print("終了時刻:", df['時刻'].max())
print("データ周波数:", df['Time'].diff().mode()[0])

```
#Continuous 燃費記録
['FO_ME_Cons', 'FO_GE_Cons'] の col の場合:
df[col] = pd.to_numeric(df[col], errors='coerce').fillna(method='ffill')

#Forward 充填は、読み取り間の安定した条件を前提としています
numeric_cols = ['Ship_Speed', 'CppPitch', 'Wind_Speed', 'HEEL', 'Fore_Draft', 'Aft_Draft']
numeric_cols の col の場合:
df[col] = pd.to_numeric(df[col], errors='coerce').fillna(method='ffill')

重大なデータギャップのある行 #Removes
df = df.dropna(サブセット=numeric_cols + ['FO_ME_Cons', 'FO_GE_Cons'])

#Removes アイドル/ポート状態(速度≤0.5 ノット)航海効率に対するモデルの関連性が向上します
df = df[df['Ship_Speed'] > 0.5]

#so、行あたりの燃料消費量(分単位)を計算します。
fuel_me = df['FO_ME_Cons'].diff().clip(lower=0)
fuel_ge = df['FO_GE_Cons'].diff().clip(lower=0)
df['Fuel_Liters'] = fuel_me + fuel_ge
#calculating トリム
df['トリム'] = df['Aft_Draft'] - df['Fore_Draft']
#calculating 平均ドラフト
df['Avg_Draft'] = (df['Aft_Draft'] + df['Fore_Draft']) / 2

df.to_csv("Cleaned_RO-RO_Data.csv", index=False)

#RO-RO 船の総トン数は鶴田山からこれを手に入れました
GT=14052
# IMO (国際海事機関) の HFO 密度 (kg/L)
FUEL_DENSITY = 0.991
#IMO(国際海事機関)からの HFO のグラムあたりの gCO2
CO2_FACTOR = 3.114
```

```
fuel_kg = df['Fuel_Liters'] * FUEL_DENSITY
df['CO2_Emission_g'] = fuel_kg * 1000 * CO2_FACTOR # kg をグラムに変換します
df['Distance_NM'] = df['Ship_Speed'] / 60 #  毎分  NM

#now 月ごとにグループ化し、他の関連フィーチャの合計 CO₂、距離、および平均値を計算します。
df['Month_Year'] = df['時間'].dt.to_period('M')
monthly_df = df.groupby('Month_Year').agg({
'CO2_Emission_g': '合計',
'Distance_NM': '合計',
'Ship_Speed': '平均',
'Wind_Speed': '平均',
'CppPitch': '平均',
'かかと': '平均',
'トリム': '平均',
'Avg_Draft': '平均'
}).reset_index()

monthly_df.rename(columns={
'Ship_Speed': 'Avg_Speed',
'Wind_Speed': 'Avg_Wind_Speed',
'CppPitch': 'Avg_CppPitch',
'全体': 'Avg_Heel',
'トリム': 'Avg_Trim',
'Avg_Draft': 'Avg_Draft'
}, inplace=True)

monthly_df.to_csv('monthly_summary.csv', index=False)
月次集計 #With、行はわずか 4 行であり、意味のある ML モデルのトレーニングには少なすぎます
#So 今、私は週次集計を使用することを検討しています~16 サンプルを提供しますアップスケーリン
グによって 1 週間分のデータから年間 CII を予測することができます

df['週'] = df['時間'].dt.to_period('W')
weekly_df = df.groupby('Week').agg({
'CO2_Emission_g': '合計',
```

```
'Distance_NM': '合計',
'Ship_Speed': '平均',
'Wind_Speed': '平均',
'CppPitch': '平均',
'かかと': '平均',
'トリム': '平均',
'Avg_Draft': '平均'
}).reset_index()

weekly_df.rename(columns={
'Ship_Speed': 'Avg_Speed',
'Wind_Speed': 'Avg_Wind_Speed',
'CppPitch': 'Avg_CppPitch',
'全体': 'Avg_Heel',
'トリム': 'Avg_Trim',
'Avg_Draft': 'Avg_Draft'
}, inplace=True)
weekly_df.to_csv('weekly_summary.csv', index=False)

df['日'] = df['時間'].dt.to_period('D')
daily_df = df.groupby('日').agg({
'CO2_Emission_g': '合計',
'Distance_NM': '合計',
'Ship_Speed': '平均',
'Wind_Speed': '平均',
'CppPitch': '平均',
'かかと': '平均',
'トリム': '平均',
'Avg_Draft': '平均'
}).reset_index()
daily_df.rename(columns={
'Ship_Speed': 'Avg_Speed',
'Wind_Speed': 'Avg_Wind_Speed',
'CppPitch': 'Avg_CppPitch',
```

```
'全体': 'Avg_Heel',
'トリム': 'Avg_Trim',
'Avg_Draft': 'Avg_Draft'
}, inplace=True)

#Calculate 年間排出量と距離の予測
daily_df['Annual_CO2'] = daily_df['CO2_Emission_g'] * 365
daily_df['Annual_Dist'] = daily_df['Distance_NM'] * 365
IMO 式を使用した #Calculate CII
daily_df['CII'] = daily_df['Annual_CO2'] / (GT * daily_df['Annual_Dist'])
daily_df.to_csv('daily_summary.csv', index=False)

# 日付だけを抽出
df['日'] = df['時間'].dt.to_period('D')

# 1 日あたりの総燃料量を計算
daily_fuel = df.groupby('Day')['Fuel_Liters'].sum().reset_index()
daily_fuel.rename(columns={'Fuel_Liters': 'Total_Fuel_Liters'}, inplace=True)

# 既存の daily_df とマージする
daily_df = pd.merge(daily_df, daily_fuel, on='Day', how='left')

plt.figure(figsize=(14, 6))
plt.bar(daily_df['Day'].astype(str), daily_df['Total_Fuel_Liters'], color='skyblue')
plt.xlabel('日付')
plt.ylabel('総燃料消費量 (リットル)')
plt.title('毎日の燃料消費傾向')
plt.xticks(回転= 45)
plt.grid(軸='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 4))
sns.histplot(daily_df['CII'], kde=True, bins=30)
```

```
plt.title("デイリーCII の配布")
plt.xlabel("CII")
plt.ylabel("周波数")
plt.tight_layout()
plt.show()

#Correlation ヒートマップ
plt.figure(figsize=(10, 6))
sns.heatmap(daily_df.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("相関ヒートマップ")
plt.tight_layout()
plt.show()

経時的な #CII
plt.figure(figsize=(12, 4))
sns.lineplot(x=daily_df['Day'].astype(str), y=daily_df['CII'], marker='o')
plt.xticks(回転= 45)
plt.title("Daily CII Over Time")
plt.xlabel("日付")
plt.ylabel("CII")
plt.tight_layout()
plt.show()

#CII vs 主な機能
key_features = ['Avg_Speed', 'Avg_Wind_Speed', 'Avg_CppPitch', 'Avg_Heel', 'Avg_Trim',
'Avg_Draft']

key_features の機能の場合:
plt.figure(figsize=(6, 4))
sns.scatterplot(データ=daily_df, x=特徴, y='CII')
plt.title(f"CII vs {機能}")
plt.tight_layout()
plt.show()
```

```
# 機能列
特徴 = [
'CO2_Emission_g'、
'Distance_NM'、
'Avg_Speed'、
'Avg_Wind_Speed'、
'Avg_CppPitch'、
'Avg_Heel'、
'Avg_Trim'、
「Avg_Draft」
]
# 特徴量 (X) とターゲット (Y)
X = daily_df[特徴]
y = daily_df['CII']

#Time ベースの列車とテストの分割
split_index = int(len(daily_df) * 0.8)
X_train, X_test = X[:split_index], X[split_index:]
y_train、y_test = y[:split_index]、および[split_index:]

スケーラー = スタンダードスケーラー()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

モデル = {
'線形回帰': LinearRegression(),
'リッジ回帰': ridge(),
'ランダムフォレスト': RandomForestRegressor(random_state=0),
'SVR': SVR(),
'XGBoost': XGBRegressor(random_state=0)
}

cv = KFold(n_splits=5, shuffle=True, random_state=1)
結果 = {}
```

名前については、models.items()の model です。

```
if name == 'ランダムフォレスト':
scores = cross_val_score(モデル, X_train, y_train, scoring='neg_root_mean_squared_error',
cv=cv)
然も無くば：
scores = cross_val_score(モデル, X_train_scaled, y_train,
scoring='neg_root_mean_squared_error', cv=cv)
results[name] = -scores.mean()
Print(F"{名前} RMSE: {Resalts[名前]:.4F}")

best_model_name = min(結果, key=results.get)
best_model = モデル[best_model_name]

if best_model_name == 'Random Forest':
best_model.fit(X_train・y_train)
test_pred = best_model.predict(X_test)
然も無くば：
best_model.fit(X_train_scaled, y_train)
test_pred = best_model.predict(X_test_scaled)

rmse = np.sqrt(mean_squared_error(y_test, test_pred))
print(f"¥n 最適なモデルは: {best_model_name} テストあり RMSE = {rmse:.4f}")

# RMSE 比較プロット
plt.figure(figsize=(8, 5))
sns.barplot(x=list(results.keys()), y=list(results.values()))
plt.ylabel("クロスバリデーションされた RMSE")
plt.xticks(回転= 45)
plt.title("RMSE でのモデル比較")
plt.tight_layout()
plt.grid(真)
plt.show()
```

```
# 実績 vs 予測
plt.figure(figsize=(6, 6))
plt.scatter(y_test, test_pred, alpha=0.6)
plt.xlabel("実際の CII")
plt.ylabel("予測 CII")
plt.title(f"{best_model_name} 予測")
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max%], 'r--')
plt.grid()
plt.tight_layout()
plt.show()

# 最適なモデルとその RMSE を保存するための辞書
tuned_results = {}
best_models = {}

# 1.線形回帰 (チューニングなし)
lr_model = LinearRegression()
lr_model.fit(X_train_scaled, y_train)
lr_pred = lr_model.predict(X_test_scaled)
lr_rmse = np.sqrt(mean_squared_error(y_test, lr_pred))
tuned_results['線形回帰'] = lr_rmse
best_models['線形回帰'] = lr_model

# 2.リッジ回帰
ridge_params = {'アルファ': [0.01, 0.1, 1, 10, 100]}
ridge_grid = GridSearchCV(Ridge(), ridge_params, cv=5,
scoring='neg_root_mean_squared_error')
ridge_grid.fit(X_train_scaled, y_train)
ridge_pred = ridge_grid.predict(X_test_scaled)
ridge_rmse = np.sqrt(mean_squared_error(y_test, ridge_pred))
tuned_results['リッジ回帰'] = ridge_rmse
best_models['リッジ回帰'] = ridge_grid.best_estimator_
# 3.ランダムフォレスト
rf_params = {'n_estimators': [50, 100], 'max_depth': [5, 10, なし]}
```

```
rf_grid = GridSearchCV(RandomForestRegressor(random_state=0), rf_params, cv=5,
scoring='neg_root_mean_squared_error')
rf_grid.fit(X_train, y_train) # RF はスケーリング不要
rf_pred = rf_grid.predict(X_test)
rf_rmse = np.sqrt(mean_squared_error(y_test, rf_pred))
tuned_results['ランダムフォレスト'] = rf_rmse
best_models['ランダムフォレスト'] = rf_grid.best_estimator_

# 4.SVR の
svr_params = {'C': [0.1, 1, 10], 'ガンマ': ['スケール', 0.01, 0.1]}
svr_grid = GridSearchCV(SVR(), svr_params, cv=5, scoring='neg_root_mean_squared_error')
svr_grid.fit(X_train_scaled, y_train)
svr_pred = svr_grid.predict(X_test_scaled)
svr_rmse = np.sqrt(mean_squared_error(y_test, svr_pred))
tuned_results['SVR'] = svr_rmse
best_models['SVR'] = svr_grid.best_estimator_

# 5.XG ブート
xgb_params = {'n_estimators': [50, 100], 'max_depth': [3, 5], 'learning_rate': [0.01, 0.1]}
xgb_grid = GridSearchCV(XGBRegressor(random_state=0), xgb_params, cv=5,
scoring='neg_root_mean_squared_error')
xgb_grid.fit(X_train, y_train) # XGB はスケーリングなしで動作できます
xgb_pred = xgb_grid.predict(X_test)
xgb_rmse = np.sqrt(mean_squared_error(y_test, xgb_pred))
tuned_results['XGBoost'] = xgb_rmse
best_models['XGBoost'] = xgb_grid.best_estimator_

mae_scores = {}
r2_scores = {}
```

名前については、best_models.items()のモデルです。
name == 'Random Forest' または name == 'XGBoost' の場合:
preds = model.predict(X_test)
然も無くば：

```
preds = model.predict(X_test_scaled)

# これらは RMSE を超える追加スコアです
mae_scores[名前] = mean_absolute_error(y_test, preds)
r2_scores[名前] = r2_score(y_test, preds)

summary_df = pd です。データフレーム({
'RMSE': tuned_results,
「それ」:mae_scores、
「R2 スコア」:r2_scores
}).T ラウンド(4)

print("¥n モデルのパフォーマンスメトリクス:")
プリント(summary_df)

#Train ファイナルリッジモデル
ridge_final = Ridge(alpha=1.0) # 前回のチューニングで得た最高のアルファを使用
ridge_final.fit(X_train_scaled, y_train)
y_pred = ridge_final.predict(X_test_scaled)
full_scaled = scaler.transform(daily_df[features]) # トレーニング中に使用したのと同じスケーラーを再利用します
daily_df['Predicted_CII'] = ridge_final.predict(full_scaled)

#Evaluation/検証
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
is = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("ファイナルリッジモデルパフォーマンス:")
print(f"RMSE: {rmse:.4f}")
print(f"IS: {is:.4f}")
print(f"R2: {r2:.4f}")
#Actual vs 予測プロット
plt.figure(figsize=(6,6))
```

plt.scatter(y_test, y_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max%], 'r--')
plt.xlabel("実際の CII")
plt.ylabel("予測 CII")
plt.title("リッジ回帰: 実際の CII と予測された CII")
plt.grid(真)
plt.tight_layout()
plt.show()

防御力 assign_rating(predicted_cii):
predicted_cii <= 15.75 の場合:
'A' を返す
eliif predicted_cii <= 18.20:
'B' を返す
エリフ predicted_cii <= 20.45:
'C' を返します
エリフ predicted_cii <= 22.70:
'D' を返す
然も無くば :
'E' を返す

def generate_suggestion(行):
suggestions = []

row['Avg_Trim'] が 1.0 >の場合:
suggestions.append("トリムを減らして燃料効率を向上させる")
row['Avg_Heel'] が 0.5 >場合:
suggestions.append("かかとを減らすためにバラストのバランスを取ります")
ROW['Avg_Wind_Speed'] が 12 >の場合:
suggestions.append("強風時の航行は避けてください")
ROW['Avg_CppPitch'] < 15 の場合:
suggestions.append("推進効率のために CPP ピッチを増やす")
ROW['Avg_Speed'] < 17 の場合:
suggestions.append("最適な巡航速度を維持する")

返します "; ".join(suggestions) if suggestions else "パフォーマンスは予想範囲内です"

申し込む
daily_df['Optimization_Suggestions'] = daily_df.apply(generate_suggestion, axis=1)

daily_df[['日', 'Predicted_CII', 'Relative_Rating',
'Optimization_Suggestions']].to_csv("cii_recommendations.csv", index=False)

use_manual_input = input("手動入力を使用しますか?(はい/いいえ): ").strip().lower()
use_manual_input == 'yes' の場合:
print("今日の平均値を入力:")
行 = pd。DataFrame([{
'CO2_Emission_g': float(input("CO2 排出量 (g): ")),
'Distance_NM': float(input("距離 (NM): ")),
'Avg_Speed': float(input("平均速度: ")),
'Avg_Wind_Speed': float(input("風速: ")),
'Avg_CppPitch': float(input("CPP ピッチ: ")),
'Avg_Heel': float(input("ヒール: ")),
'Avg_Trim': float(input("トリム: ")),
'Avg_Draft': float(input("平均ドラフト: "))
}])
row_scaled = scaler.transform(行)
pred = ridge_final.predict(row_scaled)[0]
print(f"予測 CII: {変更前:.4f} |評価: {assign_rating(変更前)}")
print("提案:", generate_suggestion(row.iloc[0]))
然も無くば :
file_path = input("CSV ファイルパス: ").strip()
データ = pd.read_csv(file_path)
preds = ridge_final.predict(scaler.transform(データ[特徴]))
data['Predicted_CII'] = プレズ
data['評価'] = data['Predicted_CII'].apply(assign_rating)
data['提案'] = data.apply(generate_suggestion, axis=1)
print(データ[['Predicted_CII', '評価', '提案']])

data.to_csv("cii_predictions_output.csv", index=False)
print("cii_predictions_output.csv に保存")

ジョブライブラリのインポート
joblib.dump(ridge_final, 'ridge_final_model.joblib')
joblib.dump(スケーラー, 'scaler.joblib')

[16-2] 付録 B: リアルタイム CII 計算モデル – フルコード

=== IMPORTS ===
Pandas を PD としてインポート
インポート時間
numpy を np としてインポート

# === 定数 ===
GT = 14052 # 船の総トン数
FUEL_DENSITY = 0.991 # kg/L
CO2_FACTOR = 3.114 # g CO2 HFO のグラムあたり

=== 計算に使用される機能 ===
特徴 = [
'FO_ME_Cons'、'FO_GE_Cons'、'Ship_Speed'、'CppPitch'、'Wind_Speed'、
'HEEL'、'Fore_Draft'、'Aft_Draft'
]

# === CSV ファイルの入力 ===
csv_path = input("分単位の CSV ファイルへのパスを入力してください: ").strip()
データ = pd.read_csv(csv_path)

# === クリーンおよびプリプロセス関数 ===
def preprocess(行, prev_row):
試みる：
燃料の読み取り値を数値に変換し、燃料の差分を計算

fuel_me = max(float(row['FO_ME_Cons']) - float(prev_row['FO_ME_Cons']), 0)
fuel_ge = max(float(row['FO_GE_Cons']) - float(prev_row['FO_GE_Cons']), 0)
fuel_liters = fuel_me + fuel_ge

トリムと平均ドラフト
トリム = float(row['Aft_Draft']) - float(row['Fore_Draft'])
avg_draft = (float(row['Aft_Draft']) + float(row['Fore_Draft'])) / 2

# CO2 と距離
ship_speed = float(行['Ship_Speed'])
distance_nm = ship_speed / 60 # 毎分
fuel_kg = fuel_liters * FUEL_DENSITY
co2_emission_g = fuel_kg * 1000 * CO2_FACTOR

瞬時 CII 計算
GT * distance_nm == 0 の場合:
cii = 例: nan
然も無くば :
cii = co2_emission_g / (GT * distance_nm)

戻り値 {
'CO2_Emission_g': co2_emission_g,
'Distance_NM': distance_nm,
'CII': cii,
'Avg_Speed': ship_speed,
'Avg_Trim': トリム、
'Avg_Heel': float(row['かかと']),
'Avg_Wind_Speed': float(row['Wind_Speed']),
'Avg_CppPitch': float(row['CppPitch']),
'Avg_Draft': avg_draft
}
ただし、例外は e です。
print(f"行の処理中にエラーが発生しました: {e}")
戻り値 {

'CO2_Emission_g': 例: ナン、
'Distance_NM': 例: ナン、
'CII':np.in、
'Avg_Speed': 例: ナン、
'Avg_Trim': 例: ナン、
'Avg_Heel': 例: ナン、
'Avg_Wind_Speed': 例: ナン、
'Avg_CppPitch': 例: ナン、
'Avg_Draft': 例: nan
}

# === 最適化提案機能 ===
def generate_suggestion(行):
suggestions = []
row['Avg_Trim'] が 1.0 >の場合:
suggestions.append("トリムを減らして燃料効率を向上させる")
row['Avg_Heel'] が 0.5 >場合:
suggestions.append("かかとを減らすためにバラストのバランスを取ります")
ROW['Avg_Wind_Speed'] が 12 >の場合:
suggestions.append("強風時の航行は避けてください")
ROW['Avg_CppPitch'] < 15 の場合:
suggestions.append("推進効率のために CPP ピッチを増やす")
ROW['Avg_Speed'] < 17 の場合:
suggestions.append("最適な巡航速度を維持する")
返します "; ".join(suggestions) if suggestions else "パフォーマンスは予想範囲内です"

# === ライブシミュレーション ===
print("¥n--- ライブ CII 計算開始 ---")
i が range(1, len(data))の場合:
current_row = data.iloc[i]
previous_row = data.iloc[i - 1]

result = preprocess(current_row, previous_row)

```
cii = result.get('CII', np.nan)
提案 = generate_suggestion(結果)

np.isnan(cii)でない場合:
print(f"分 {i}: インスタント CII = {cii:.4f} |提案: {suggestions}")
然も無くば :
print(f"Minute {i}: 距離がゼロのため、CII を計算できませんでした。")

時間.睡眠(1)

print("¥n--- ライブ CII シミュレーション完了 ---")
```