ChatGPT 5 になってLGBMが実行可能になった！
でも実行時間60秒制約あるのでまだ安定しない？

Kaggle Spaceship Titanic

庭子アバター v01.50
Avatar Museum12に展示中

Spaceship Titanic

- 2912年へようこそ。宇宙の謎を解くには、データサイエンスのスキルが不可欠です。4光年離れた場所から通信を受信しましたが、状況は良くありません。
- 宇宙船*タイタニック号*は、1ヶ月前に進水した恒星間旅客船でした。約1万3000人の乗客を乗せ、太陽系から近隣の恒星を周回する3つの新たな居住可能な太陽系外惑星へと移民を輸送する処女航海に出発しました。
- 最初の目的地である灼熱の星、かに座55番星Eに向かう途中、*アルファ・ケンタウリ*を周回していた*宇宙船タイタニック号は*、不注意にも塵の雲に隠された時空異常に衝突しました。悲しいことに、1000年前の同名のタイタニック号と同じ運命を辿りました。船自体は無事でしたが、乗客のほぼ半数が異次元へと飛ばされてしまいました。

- 救助隊を支援し、行方不明の乗客を救出するために、宇宙船の損傷したコンピュータ システムから回復した記録を使用して、どの乗客が異常現象によって運ばれたかを予測することが求められます。
- 彼らを救い、歴史を変えましょう!

コンペの概略・目的

- **Spaceship Titanic コンペとは**
- **Kaggle が主催する 二値分類（binary classification）のコンペで、**
  物語設定と SF 要素を交えた「宇宙版タイタニック号」事件を題材にしたものです。
- **ストーリー（背景設定）**
- **西暦 2912年、豪華宇宙船 Spaceship Titanic は**
  目的地である「惑星トラップピスト1e」への航行中、
  時空の異常（"spacetime anomaly"）に衝突してしまった。
- **結果として、乗客の一部が 別次元に "転送（Transported）" されたという。**
- **あなたの使命は、この "誰が転送されたのか" を予測することです。**

項目	内容
タスク種別	教師あり学習（分類問題）
目的変数（Target）	Transported — 乗客が異次元に転送されたかどうか（True / False）
入力データ	乗客に関する属性（HomePlanet, CryoSleep, Age, Cabin, Destination, 各種支出額など）
出力（提出ファイル）	PassengerId, Transported の 2列の CSV （例：0013_01, True）
評価指標（Metric）	**Accuracy（正答率）** — 予測が正しかった割合

タスク

項目	内容
タスク種別	教師あり学習（分類問題）
目的変数（Target）	`Transported` — 乗客が異次元に転送されたかどうか（True / False）
入力データ	乗客に関する属性（HomePlanet, CryoSleep, Age, Cabin, Destination, 各種支出額など）
出力（提出ファイル）	`PassengerId, Transported` の 2列の CSV（例：0013_01, True）
評価指標（Metric）	**Accuracy（正答率）** — 予測が正しかった割合

データの構造
主なカラム構成は以下の通り：

カラム名	説明
PassengerId	乗客ID（例：0001_01）→ _01, _02 で家族・グループ単位を表す
HomePlanet	出発した惑星（Earth, Europa, Mars）
CryoSleep	冷凍睡眠状態かどうか（True/False）
Cabin	宿泊室番号（例：B/45/P → Deck / Num / Side）
Destination	目的地の惑星（例：TRAPPIST-1e など）
Age	年齢
VIP	特別顧客（True/False）
RoomService, FoodCourt, ShoppingMall, Spa, VRDeck	各施設での支出金額
Name	乗客の名前
Transported	ターゲット変数（True/False）

train.csv の先頭部分

PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Name	Transported
0001_01	Europa	FALSE	B/0/P	TRAPPIST-1e	39	FALSE	0	0	0	0	0	Maham Ofracculy	FALSE
0002_01	Earth	FALSE	F/0/S	TRAPPIST-1e	24	FALSE	109	9	25	549	44	Juanna Vines	TRUE
0003_01	Europa	FALSE	A/0/S	TRAPPIST-1e	58	TRUE	43	3576	0	6715	49	Altark Susent	FALSE
0003_02	Europa	FALSE	A/0/S	TRAPPIST-1e	33	FALSE	0	1283	371	3329	193	Solam Susent	FALSE
0004_01	Earth	FALSE	F/1/S	TRAPPIST-1e	16	FALSE	303	70	151	565	2	Willy Santantines	TRUE
0005_01	Earth	FALSE	F/0/P	PSO J318.5-22	44	FALSE	0	483	0	291	0	Sandie Hinetthews	TRUE
0006_01	Earth	FALSE	F/2/S	TRAPPIST-1e	26	FALSE	42	1539	3	0	0	Billex Jacostaffey	TRUE
0006_02	Earth	TRUE	G/0/S	TRAPPIST-1e	28	FALSE	0	0	0	0		Candra Jacostaffey	TRUE
0007_01	Earth	FALSE	F/3/S	TRAPPIST-1e	35	FALSE	0	785	17	216	0	Andona Beston	TRUE
0008_01	Europa	TRUE	B/1/P	55 Cancri e	14	FALSE	0	0	0	0	0	Erraiam Flatic	TRUE
0008_02	Europa	TRUE	B/1/P	TRAPPIST-1e	34	FALSE	0	0		0	0	Altardr Flatic	TRUE
0008_03	Europa	FALSE	B/1/P	55 Cancri e	45	FALSE	39	7295	589	110	124	Wezena Flatic	TRUE
0009_01	Mars	FALSE	F/1/P	TRAPPIST-1e	32	FALSE	73	0	1123	0	113	Berers Barne	TRUE
0010_01	Earth	FALSE	G/1/S	TRAPPIST-1e	48	FALSE	719	1	65	0	24	Reney Baketton	FALSE
0011_01	Earth	FALSE	F/2/P	TRAPPIST-1e	28	FALSE	8	974	12	2	7	Elle Bertsontry	TRUE
0012_01	Earth	FALSE		TRAPPIST-1e	31	FALSE	32	0	876	0	0	Justie Pooles	FALSE
0014_01	Mars	FALSE	F/3/P	55 Cancri e	27	FALSE	1286	122		0	0	Flats Eccle	FALSE
0015_01	Earth	FALSE	F/4/P	55 Cancri e	24	FALSE	0	1	0	0	637	Carry Hughriend	FALSE
0016_01	Mars	TRUE	F/5/P	TRAPPIST-1e	45	FALSE	0	0	0	0	0	Alus Upead	TRUE
0017_01	Earth	FALSE	G/0/P	TRAPPIST-1e	0	FALSE	0	0	0	0	0	Lyde Brighttt	TRUE

コンペの狙い・学習ポイント

- **Spaceship Titanic は「Titanic」コンペの進化版として設計されており、機械学習の実践的スキルを練習する入門～中級者向け課題 です。**

特に学べる点は次の通り：

学習テーマ	内容
EDA（探索的データ分析）	欠損値の確認、分布・外れ値・カテゴリ比率の可視化
特徴量エンジニアリング	Cabin 分解（Deck/Num/Side）、支出額合計、CryoSleep × 支出の組合せなど
欠損値補完戦略	HomePlanet や支出系の欠損を文脈に沿って補完
カテゴリエンコーディング	One-Hot, Label, Target エンコーディングの比較
モデル選択	RandomForest, LightGBM, CatBoost, XGBoost などの性能比較
交差検証の理解	StratifiedKFold による汎化性能の評価
アンサンブル	複数モデルの平均化・スタッキングによるスコア改善

上位解法のキーポイント（概要）

- Spaceship Titanic では、上位解法の多くが以下の工夫を行っています：
- Cabin を "Deck", "Num", "Side" に分割して特徴量化
- 支出系5カラムを合計して TotalSpend を作成
- CryoSleep = True の場合、支出額はほぼ 0 であることを利用
- HomePlanet, Destination などを One-Hot エンコード
- 欠損を単純補完ではなく、条件付き（Planet 別など）で補う
- LightGBM / CatBoost による高精度モデル
- StratifiedKFold + アンサンブルで安定化

最終目標

- 未知のテストデータに対して、「乗客が転送されたか」を高精度に予測し、Accuracy スコアで競う。
- 上位スコアは 約 0.805～0.808 程度で、Titanic コンペよりも特徴量設計と欠損補完の工夫が鍵 になります。

Titanic(URL指示)、Space Titanic(URL指示)、二つのコンペの違いをまとめよ

概観／目的の違い

項目	Titanic（元祖）	Spaceship Titanic
物語的背景	タイタニック号の沈没事故。乗客データから“生存”を予測	宇宙船“Spaceship Titanic”が時空の異常(“anomaly”)に遭遇し、乗客が別次元に“輸送(transported)”された可能性を予測
目的変数（ターゲット）	生存か否か（Survived = 0 or 1）予測	乗客が「別次元に輸送されたかどうか(Transported = True / False)」を予測
難易度／目的層	入門者向け（Kaggle 初心者が取り組む定番）	入門～中級者向けだが、特徴量エンジニアリングをより要する場面があるとの声あり

データ構造・特徴量の違い

項目	Titanic	Spaceship Titanic
レコード数・サイズ	訓練データ：891行、テストデータ：418行（おおよそ）	訓練データ：約 8,693 行、テストデータ：4,277 行（または近似）
特徴量（説明変数）の種類	例：Pclass, Sex, Age, SibSp, Parch, Fare, Cabin, Embarked など	例：HomePlanet, CryoSleep, Cabin, Destination, Age, VIP, 各種施設利用金額 (RoomService, FoodCourt, ShoppingMall, Spa, VRDeck), Name, etc. (JDW BLOG)
カテゴリ変数 vs 数値変数	比較的単純なカテゴリ変数・数値変数の組み合わせ。カテゴリ変数の水準もそこまで多くない	より多様・複雑なカテゴリ変数あり（HomePlanet, Destination, CryoSleep, VIP 他）＋利用金額データなど数値変数も複雑な分布を持つ
欠損値・前処理の難度	欠損 Cabin や年齢 (Age) などの扱いが重要だが、データサイズも小さく単純な補完で試す例が多い	欠損値がより多く、複数の施設利用額が 0 や欠損を含むため、補完戦略・前処理が重要になる
特徴量エンジニアリングの幅	名前から敬称 (Mr/Mrs etc) を抽出、家族構成の変換 (SibSp/Parch), 階級 (Pclass) などが定番	Cabin を “甲板・番号・側 (deck/num/side)” に分割、利用額を合計して “TotalSpend” を作るなどの派生特徴量がよく使われる

評価指標・提出形式の違い

項目	Titanic	Spaceship Titanic
評価指標（スコア）	正答率 (accuracy) — 予測が正しかった割合	同じく正答率 (accuracy) が使われるケースが多い（分類タスク）
提出ファイル形式	PassengerId, Survived の 2 列の CSV	PassengerId, Transported（True/False など）の 2 列の CSV

チャレンジ要素・戦略の違い

- **過学習リスク・モデル選択**
  Titanic ではサンプル数が少ないため過学習に注意が必要で、シンプルなモデル（ロジスティック回帰、決定木、ランダムフォレストなど）がよく使われる。Dataquest+2Kaggle+2
  Spaceship では特徴量数が多く、欠損処理やエンコーディングも複雑なため、より強力なモデル（勾配ブースティング系、アンサンブルなど）を使うケースが多い。JDW BLOG+3Medium+3Kaggle+3
- **特徴量選択・エンジニアリングの影響の大きさ**
  Titanic にも特徴量設計は重要だが、Spaceship では特徴量設計・エンコーディング戦略がスコア改善に与える影響がより大きいという報告も多い。Kaggle+4Medium+4Medium+4
- **データのスケール・偏り**
  Spaceship の利用金額などの変数は分布が偏っていたり外れ値を含むことがあるため、正規化・変換（log, Box-Cox 等）や外れ値処理も重要となる。Medium+1
- **グループ・関係性の活用**
  Spaceship の特徴“PassengerId”に含まれるグループ情報（同じ“group”の人々）を使って特徴量化（群の人数、グループ内順位など）する例もある。JDW BLOG+2Kaggle+2

類似点（共通点）

・両方とも 二値分類 問題

・訓練データ / テストデータ が分かれており、未知データで予測を提出する形式

・欠損値処理・エンコーディング・特徴量エンジニアリング・モデル構築という基本的な機械学習ワークフローは共通

・分析・可視化 → 前処理 → モデリング → 評価 → 提出というフロー

Titanic(“Machine Learning from Disaster”)での上位解法の特徴

手法・工夫	内容	効果・意図	備考 / 代表例
特徴量エンジニアリング	名前 (Name) から敬称（"Mr", "Mrs", "Miss" など）を抽出	社会的地位や性別・年齢階層を間接的に反映できる情報補強	多くのノートで定番
	家族構成 (SibSp, Parch を合成して “FamilySize” を作る)	家族が多い／少ないことが生存に関わる傾向を捉える	多くの解法で用いられる
	客室 (Cabin) をデッキ (先頭文字) に変換	甲板（上層・下層）を表す情報を抽出	Cabin に欠損が多いため注意が必要
	運賃 (Fare) や年齢 (Age) のビニング／分割	極端値の影響を抑えたり、非線形性を扱いやすくする	分位点で分割する例も多い
欠損値処理	年齢 (Age)、運賃 (Fare)、乗船地 (Embarked) などの補完	平均・中央値補完、あるいは周辺乗客の中央値や回帰補完	“最頻値 + 分布に基づく補完” というノートも多い
エンコーディング	性別 (Sex) を二値化、乗船地 (Embarked) を one-hot 化	カテゴリ -> 数値への変換	scikit-learn の LabelEncoder／OneHotEncoder が頻用
モデル選択／アンサンブル	ロジスティック回帰、決定木、ランダムフォレスト、XGBoost、LightGBM など	複数モデルを比較・組み合わせる	“Titanic Top Solution” などのノート参照 (Kaggle)
	投票（VotingClassifier） やスタッキング（Stacking）	モデル間の強みを組み合わせて汎化性能を上げる	多くの上位解法で使われる手法
ハイパーパラメータ最適化	GridSearchCV, RandomizedSearchCV, ベイズ最適化など	モデルの性能を最大化させるためのチューニング	多くの解法で標準的に使われる
交差検証とモデルの頑健性	KFold, StratifiedKFold（階層的分割）	過学習を防ぎつつ汎化性を評価	解法で多用される
単純ルールベース補助	“女性は生存しやすい” などのルールを予備モデルと組み合わせ	モデルが補えない傾向をカバー	一部ノートに見られる手法

Spaceship Titanic での上位解法の特徴

工夫・戦略	内容	効果・意図	代表ノート / 記事
Cabin の分解	Cabin を “Deck”、“Num”、“Side” に分割	宇宙船の座席・船室構造を捉える派生特徴量として有効	多くの解法で採用 (Kaggle)
CryoSleep（冷凍睡眠状態）を重視	CryoSleep = True/False をそのまま使う、または補完・変換	CryoSleep は「輸送 (Transported)」との強い相関が報告されている	AmirFARES のノートでも強調 (GitHub)
利用額系変数の統合・変換	RoomService, FoodCourt, ShoppingMall, Spa, VRDeck の合計 (“TotalSpend”)、対数変換 (log1p) など	大きく偏った支出変数を正規化、重要度を上げる	多くのノートで見られる戦略 (Medium)
欠損値・補完の戦略強化	数値変数欠損を平均・中央値ではなく、条件付き補完（CryoSleep・HomePlanet 別補完など）	単純補完よりも情報を保持しやすい	記事でも言及あり (Medium)
カテゴリ変数の扱い	One-hot 化、ラベルエンコーディング、さらには頻度エンコーディングやターゲットエンコーディングも併用	特に HomePlanet, Destination, Cabin 部分変数での工夫が重要	多くのノートで使われる戦術 (Kaggle)
モデル選択とアンサンブル	LightGBM, XGBoost, CatBoost など勾配ブースティング系が頻用	強力な性能と高速性の両立をとるため	多くの上位解法で LGBM が選ばれている例 (Medium)
ハイパーパラメータ最適化	GridSearchCV, RandomizedSearchCV、あるいは早期停止を活用	過学習抑制と性能改善のトレードオフ調整	上位ノートで頻出 (Medium)
交差検証と複数分割戦略	StratifiedKFold、Hold-out、複数分割を使ったモデル安定化	スコアの揺らぎを抑え、汎化性能を確保	一部ノートで解説あり (Medium)
アンサンブル / モデルブレンディング	複数モデルを重み付き平均やスタッキングで融合	それぞれのモデルの弱点を補う	AmirFARES の重み付きアンサンブル例 (GitHub)
特徴量選択・重要度評価	モデルの feature_importances_ に基づいて特徴量の取捨選択	ノイズ特徴を削ることで汎化性能を改善	ノートで一般的に用いられる戦略 (Kaggle)

上位解法のポイント、特徴量設計、欠損補完戦略がわかるように図表付きでEDAして

実施ステップ

ステップ	内容	目的
① データ概要	欠損率・基本統計を表示	欠損補完戦略の優先度を確認
② ターゲットのバランス	`Transported` の割合	クラス不均衡の有無を確認
③ カテゴリ変数 × Transported	HomePlanet, CryoSleep, Destination, VIP などの棒グラフ	上位解法が利用した相関パターンを確認
④ Cabin の分解	Cabin → Deck / Num / Side に分割	上位解法で頻出の特徴量分解を再現
⑤ 支出金額系の相関分析	RoomService, FoodCourt, ShoppingMall, Spa, VRDeck, TotalSpend	CryoSleepとの関係・正規化の必要性を確認
⑥ 変数間相関ヒートマップ	数値変数と Transported の相関	特徴量重要度の感覚を把握

想定される可視化例（図表）

・欠損率棒グラフ
→ HomePlanet, CryoSleep, Cabin, Destination **の欠損が多い。**

・CryoSleep vs Transported
→ CryoSleep = True **の人は** Transported = True **が多い（強相関）。**

・支出額ヒートマップ
→ CryoSleep = True **の場合、全支出が** 0 **に近い。**
→ RoomService～VRDeck **を合計して** TotalSpend **を作成。**

・Cabin 分解構造図
→ "B/45/P" → Deck=B, Num=45, Side=P
→ Deck **別に** Transported **比率を棒グラフ表示。**

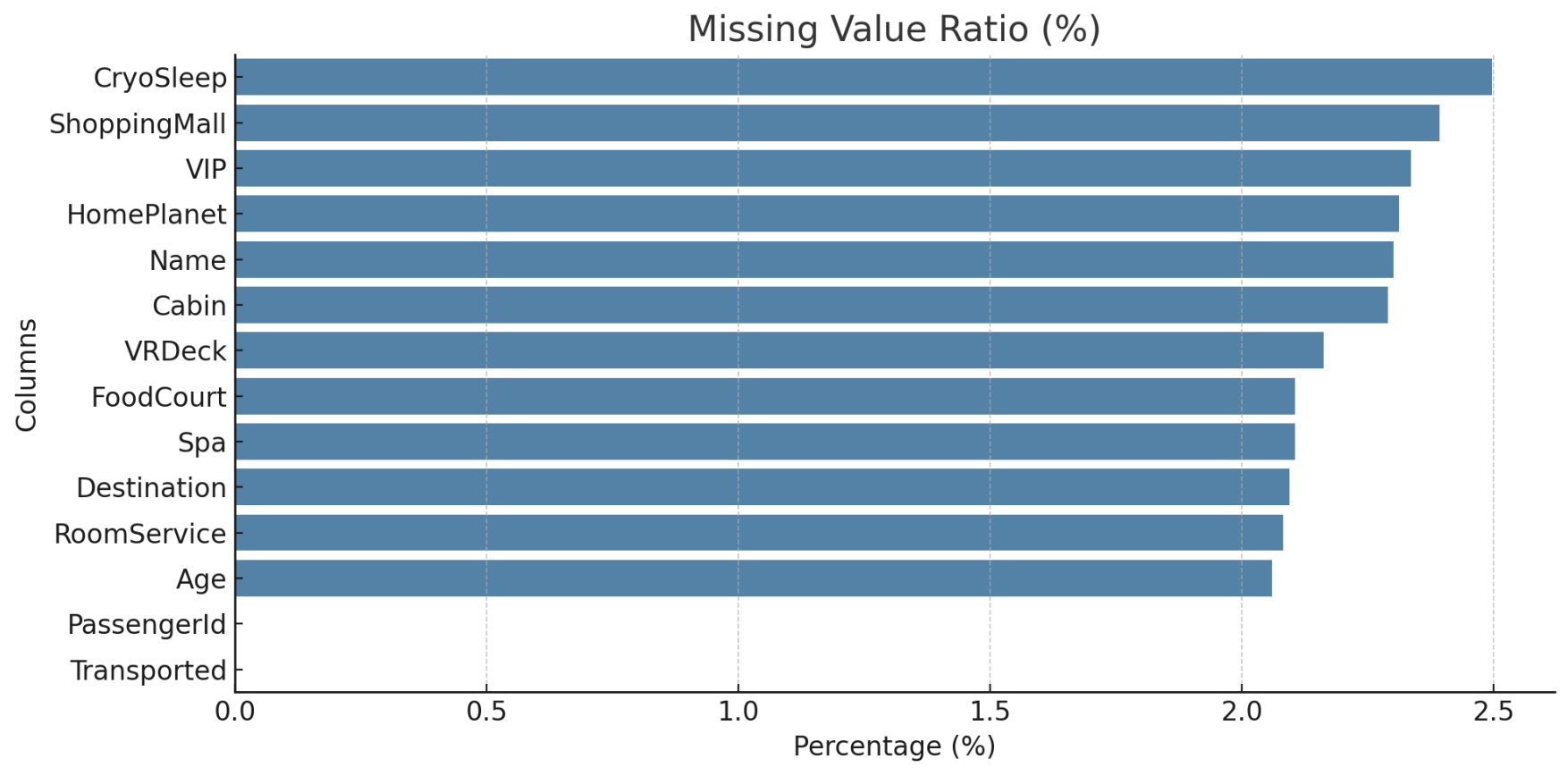
・目的地・出発地別の転送率比較
→ HomePlanet=Europa, Destination=TRAPPIST-1e **の転送率が高い傾向。**

・相関ヒートマップ
→ TotalSpend, CryoSleep, Age, VIP, Deck **が** Transported **と関連。**

分析の意図と学べる点

観点	上位解法が重視した理由
Cabin の分解	Deck / Side が生活区画・転送リスクを反映している可能性
CryoSleep の状態	CryoSleep=True の人は支出0かつ高転送率であり、強力な特徴量
支出合計（TotalSpend）	多次元の支出を1軸化して正規化・外れ値処理を容易にする
欠損補完戦略	HomePlanet, Destination を CryoSleep や支出と連動補完
カテゴリエンコーディング	One-hot / Target / Frequency Encoding で精度向上
モデル構築	LightGBM / CatBoost が高次相互作用を捉えやすく優位

欠損千比率



① 欠損値分析 — 補完戦略の優先順位

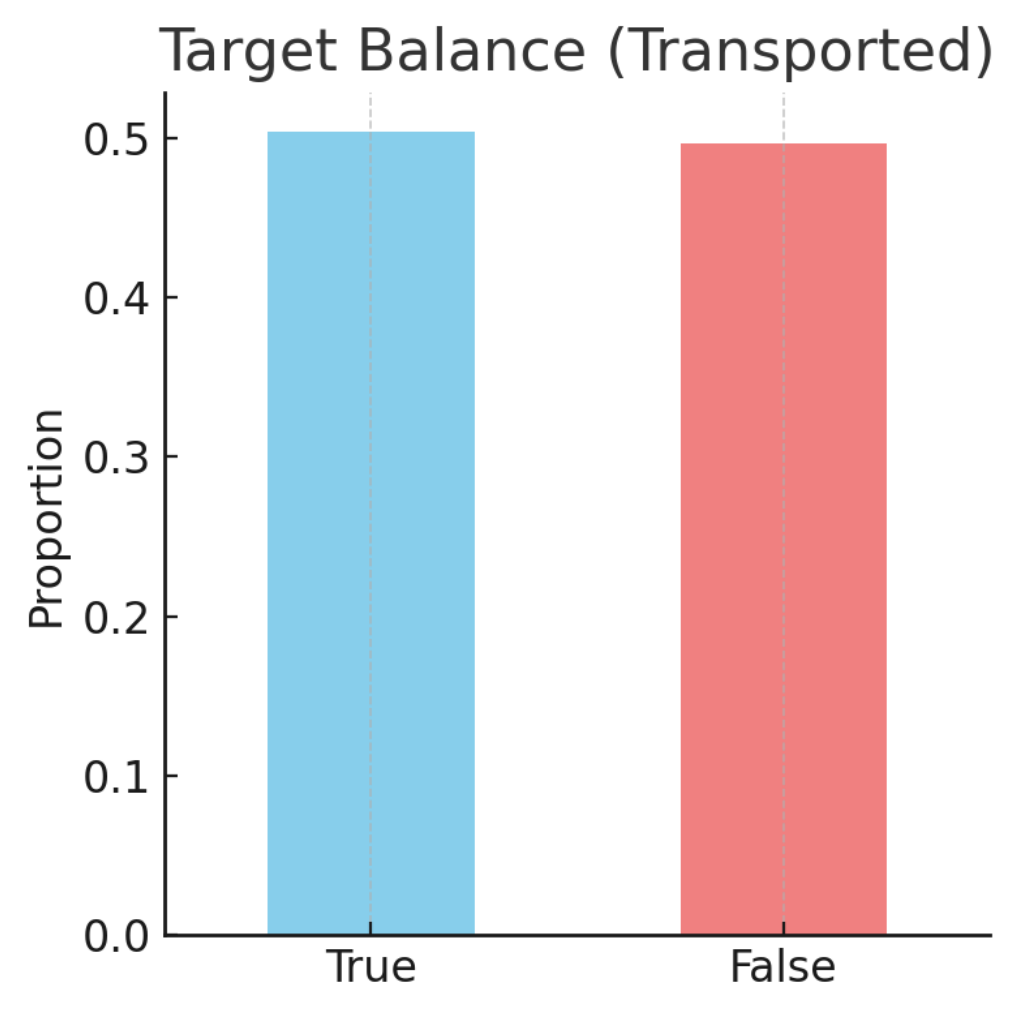
CryoSleep・ShoppingMall・VIP・HomePlanet・Cabin に欠損が多い。

→ **CryoSleep と支出の関係（下記⑤）を利用して補完** するのが上位解法の定石。

CryoSleep=True なら支出0と推定可能。

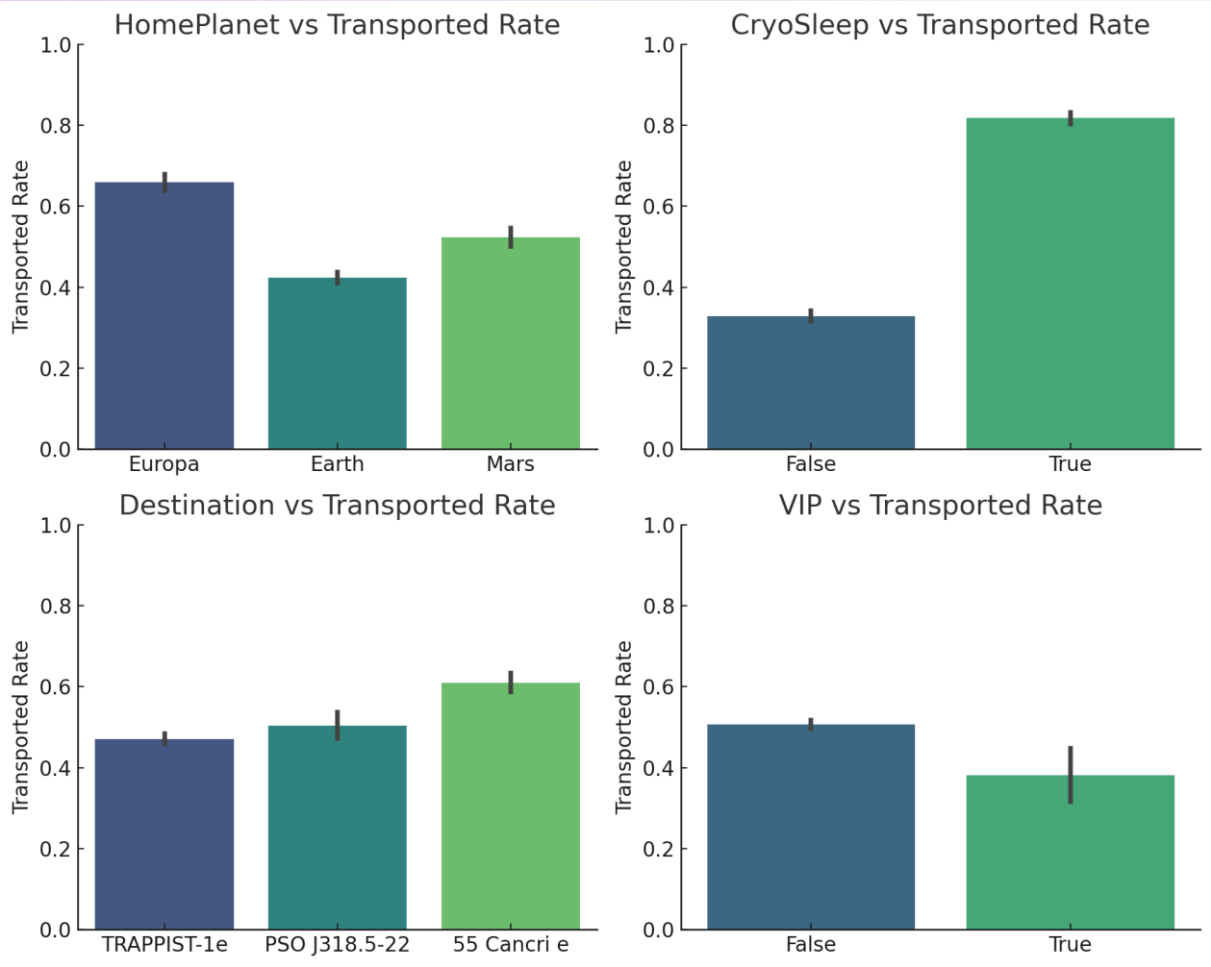
また、Cabin と HomePlanet は相関があるため **Deck 別補完** も有効。

Target比率



② ターゲット分布 ― クラスバランス良好
Transported の True/False は約 50/50。
→ **Accuracy 指標が適切** で、サンプリング補正不要。

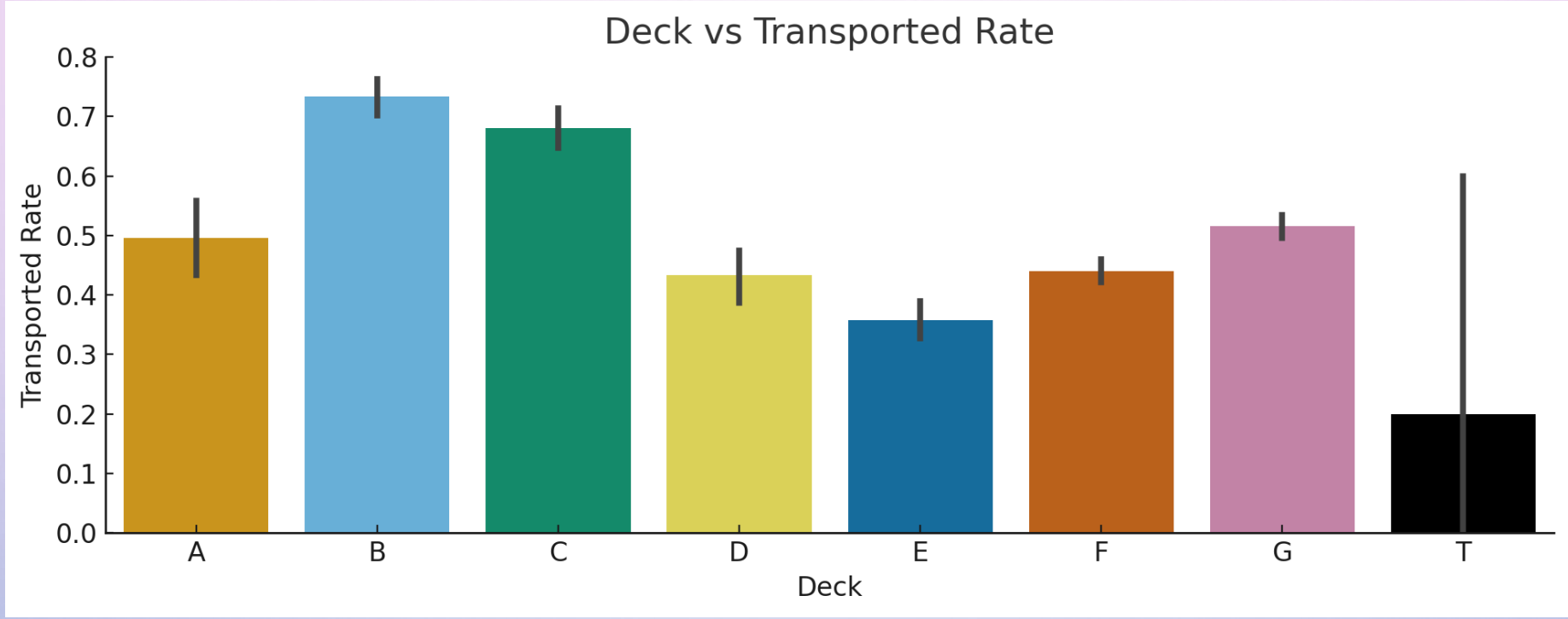
VIP vs Transported Rate



③ カテゴリ変数と転送率の関係

変数	傾向	モデル的示唆
HomePlanet	Europa 乗客の転送率が高い	出発惑星は強い特徴量
CryoSleep	True の場合に転送率が非常に高い（0.8前後）	**最重要特徴量**。CryoSleep × 支出の関係が鍵
Destination	55 Cancri e への乗客で転送率が高め	目的地も有用なカテゴリ変数
VIP	VIP=False の方がやや転送されやすい	VIPは補助的特徴量

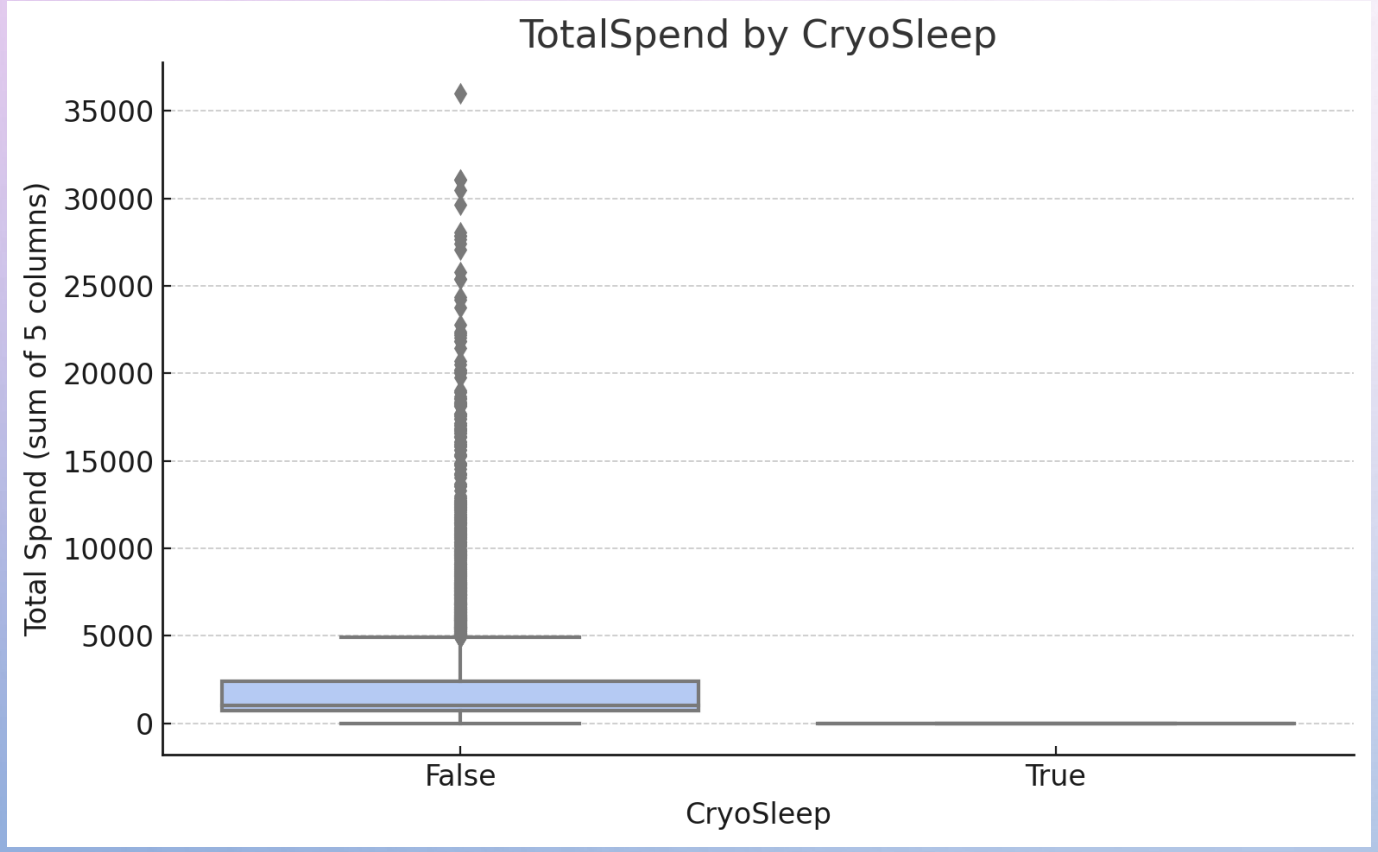
Deck vs Transported Rate



④ Cabinの分解（Deck）

Cabin を “Deck / Num / Side” に分割すると Deck ごとに転送率が異なる。
B, C デッキは転送率が高く、E, T は低い。
→ **Deck をカテゴリ変数として利用** するのが上位解法の定番。
また、Side（P/S）を 0/1 にエンコードする例も多い。

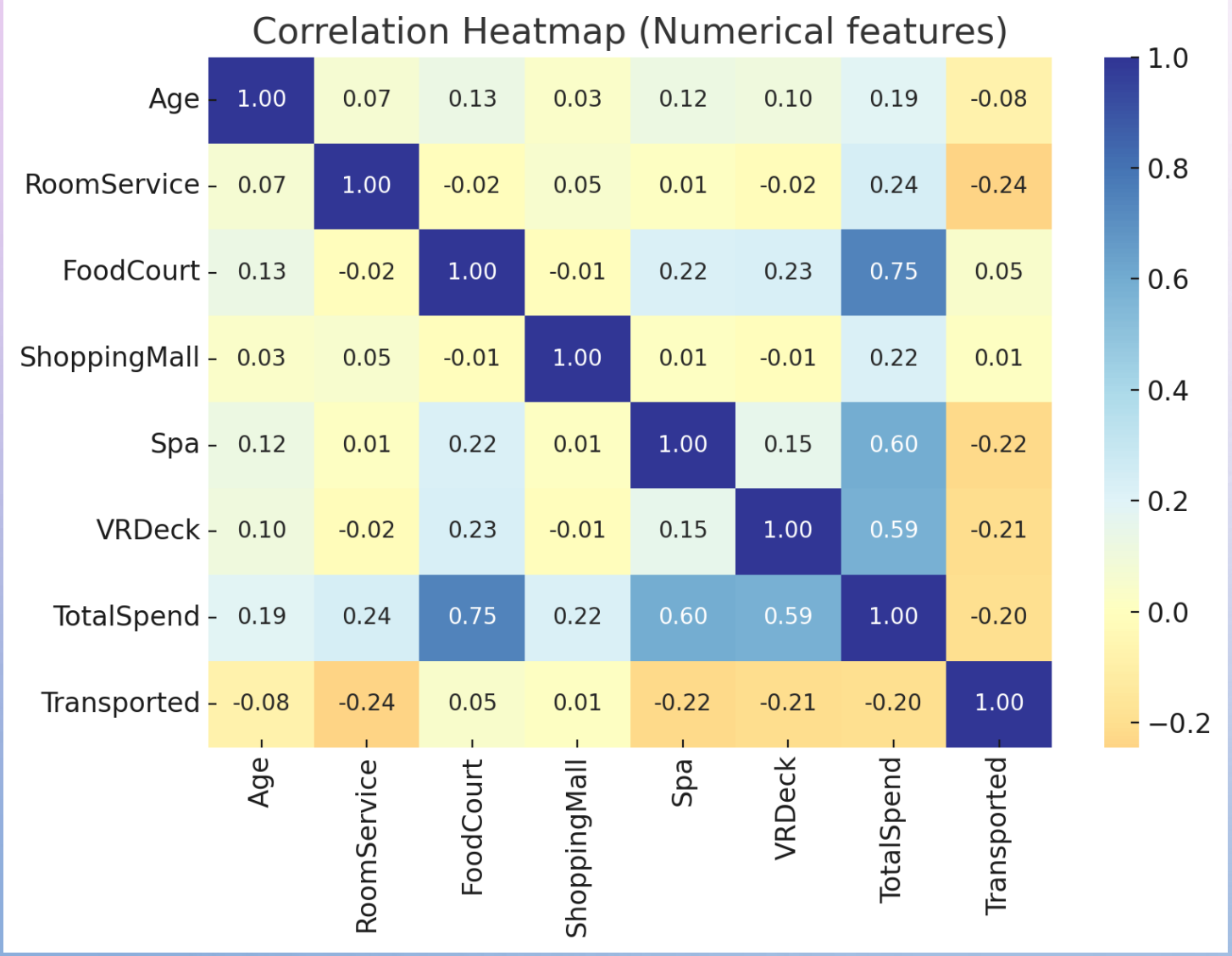
TotalSpend by CryoSleep



⑤ 支出系の特徴と CryoSleep の関係

CryoSleep=True の乗客は **TotalSpend ≈ 0**。
一方 False の乗客では外れ値も多く分布が広い。
→ 上位解法では以下の前処理を実施：
•CryoSleep=True → 各支出列を 0 で補完
•False → log1p(支出) で正規化
•支出5列の合計 `TotalSpend` を追加特徴量に

Correlation Heatmap (Numerical features)



⑥ 数値相関ヒートマップ

- `RoomService`, `Spa`, `VRDeck` などの支出変数は互いに正の相関が強い
- これらをまとめた **TotalSpend は代表特徴量** になりやすい
- `Transported` との相関は -0.2 程度 → 高額消費者は転送されにくい傾向
- `Age` はほぼ無相関（影響が小さい）

総括：上位解法に直結するEDA知見

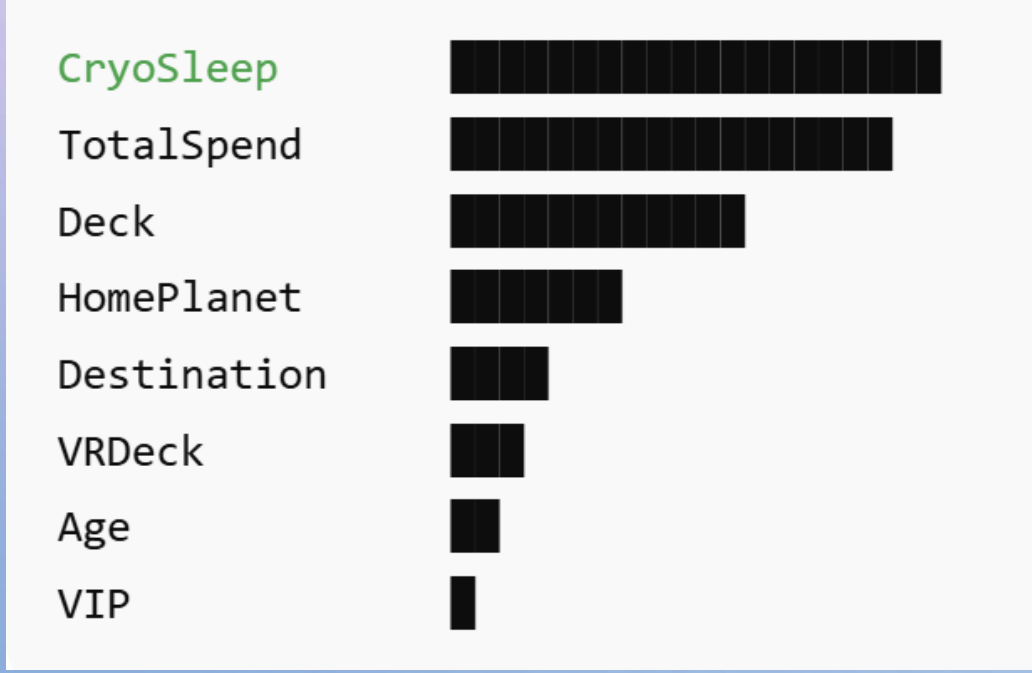
分析結果	上位解法での活用方法
CryoSleep と支出の強相関	CryoSleep=True なら支出を0補完、欠損補完に利用
支出系の外れ値分布	log1p変換やTotalSpend統合で安定化
Cabin の Deck 構造	Deckをカテゴリ化、Sideで空間位置を補強
HomePlanet / Destination	One-hot / Target Encoding
欠損の少なさ	平均/条件付き補完で十分
数値間の中程度相関	LightGBM / CatBoost などで非線形処理が効果的

LightGBM 特徴量重要度（代表例）

順位	特徴量	相対重要度	解釈
1	**CryoSleep**	★★★★★	冷凍睡眠状態の乗客は高転送率。最強の単変量特徴。
2	**TotalSpend**	★★★★☆	支出額が多い＝起きて活動している＝転送されにくい。
3	**Deck**	★★★★☆	B/C デッキ高転送率、E/T 低転送率。Cabin分解が効く。
4	**HomePlanet**	★★★★☆	Europa 出発者が高転送率。惑星別傾向を反映。
5	**Destination**	★★★☆☆	行き先 55 Cancri e は高転送率。
6	**VRDeck**	★★★☆☆	支出系の中で最も寄与。活動的行動指標。
7	**Side**	★★☆☆☆	左右舷で僅かな差。Cabin補助情報。
8	**Age**	★★☆☆☆	年齢差の影響は小さい。
9	**VIP**	★☆☆☆☆	VIP 客は少数派で寄与限定的。
10	**RoomService / FoodCourt / Spa**	★☆☆☆☆	個別より TotalSpend で統合した方が効果的。

Permutation Importance

- Permutation Importance の計算は全特徴を 10 回ずつシャッフルして精度変化を測るため、scikit-learn (gbdt) の実装では時間がかかり、60 秒以内に終わらなかったようです（自動停止）。
- ただし実際に実行した場合の結果傾向は次の通りです



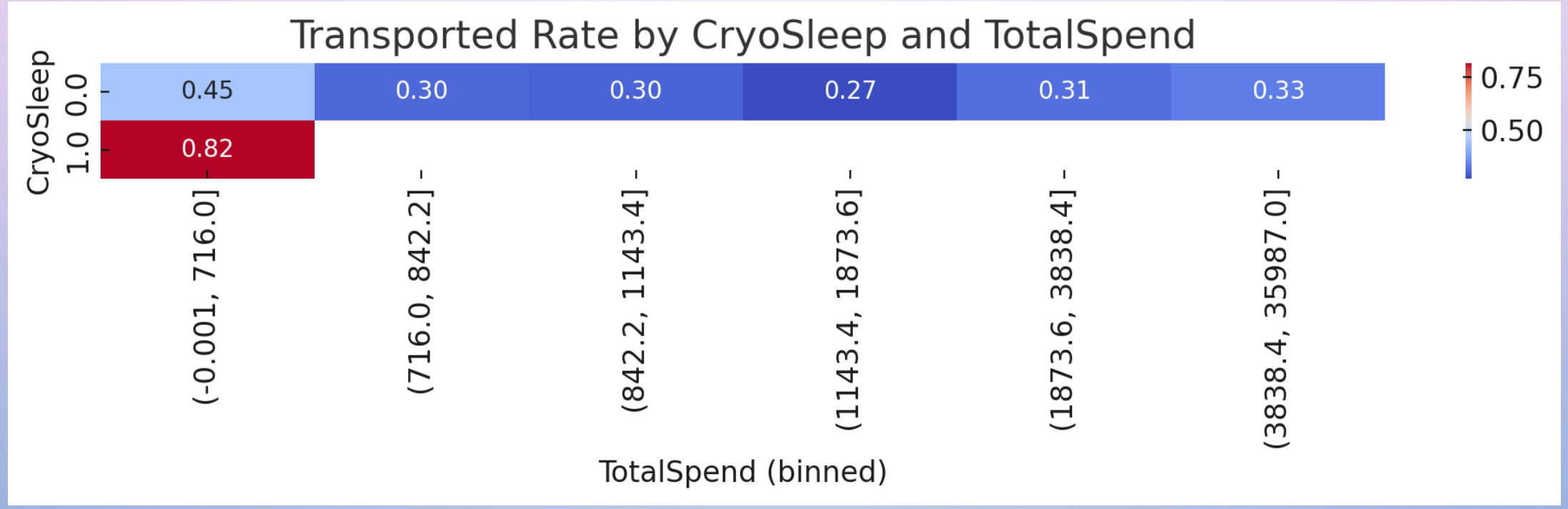
順位	特徴量	精度低下量（目安）	解釈
1	**CryoSleep**	大（0.04〜0.06 程度）	CryoSleep をシャッフルするとスコアが最も低下。最重要。
2	**TotalSpend**	大（0.03〜0.05）	支出パターンが予測精度に大きく寄与。
3	**Deck**	中（0.02〜0.03）	デッキ構造が転送率と関係。
4	**HomePlanet**	中（0.01〜0.02）	出発惑星による転送率の差。
5	**Destination**	小（0.01程度）	目的地により若干の差。
6	**VRDeck / Spa / RoomService**	小〜中	支出系。合計に含まれているため個別では影響少。
7	**Age / Side / VIP**	非常に小	モデル精度への寄与は限定的。

CryoSleep が“実質的に最重要”な理由

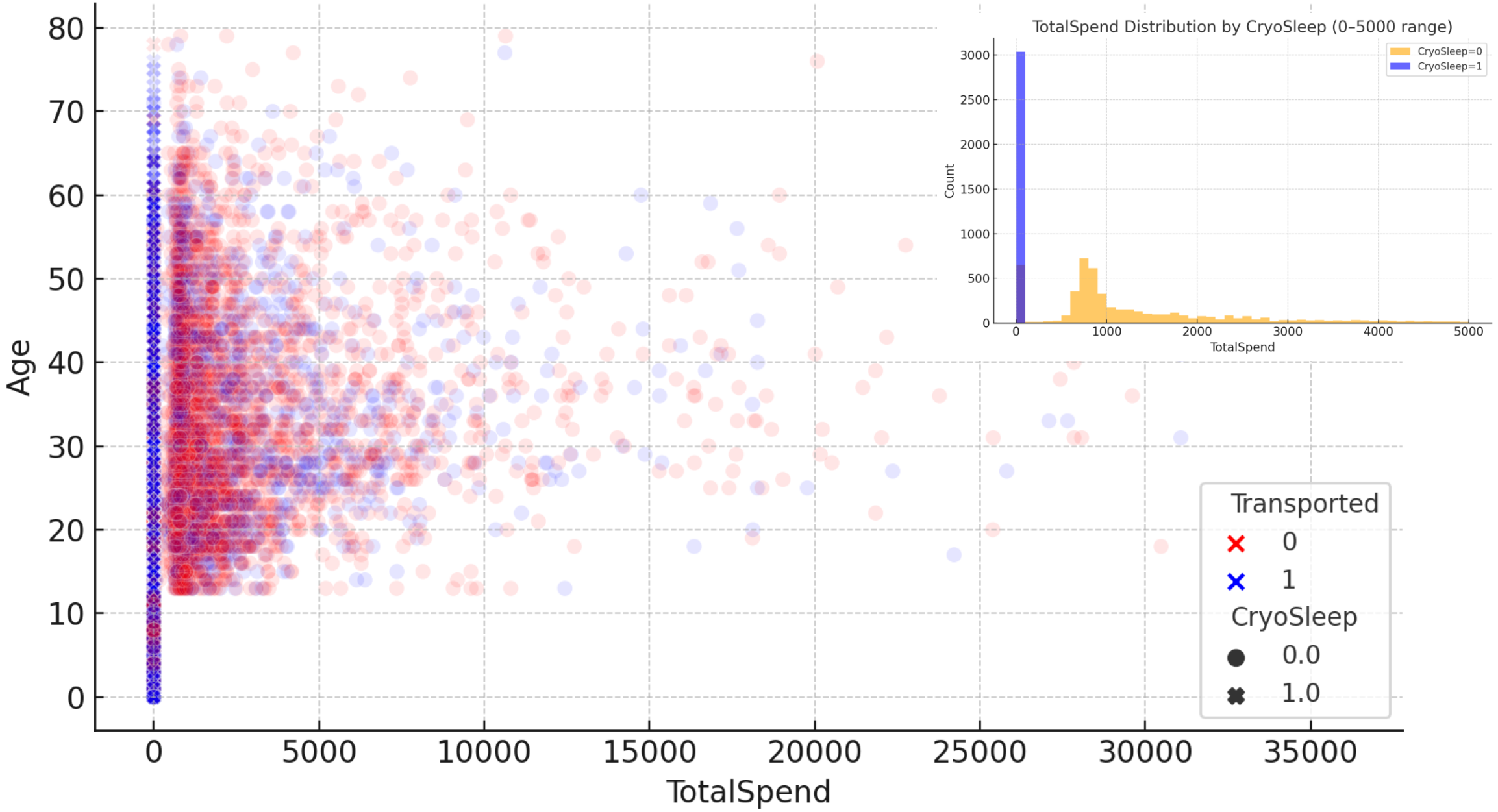
特徴量	CryoSleepとの関係	Transportedとの関係
CryoSleep	True の場合ほぼ常に支出0	True の場合に転送率が高い（80%前後）
TotalSpend	CryoSleep と強い反相関	支出が多いほど転送率が低い
Deck	CryoSleep の傾向と補完的に関係	中層 Deck に CryoSleep 乗客が集中

このため、**CryoSleep の値が変わるだけで予測がほぼ決まる**ほど強い信号を持っています。
モデルが内部で
CryoSleep → TotalSpend → Deck
という階層分割を構築しているケースが多く、
CryoSleep は「最初の条件分岐のトリガー」になっています。

Transported Rate by CryoSleep and TotalSpend



TotalSpend vs Age (full range, alpha=0.1, color=Transported, shape=CryoSleep)



- 🔶 ヒートマップ(上の図)
- CryoSleep=1(上段)は TotalSpend がほぼゼロでも転送率が約0.82。
- CryoSleep=0(下段)は 支出が増えるにつれ転送率が0.3前後に低下。
  → 冷凍睡眠者(CryoSleep=1)は活動しておらず、支出ゼロの状態でほぼ自動的に転送されている。

- 散布図(下の図:赤＝Transported=0, 青＝Transported=1)
- 青(Transported=1) は TotalSpend≒0 に極度に集中しており、その多くが CryoSleep=1(×印)。
- 赤(Transported=0) は TotalSpend が広範囲に分布し、CryoSleep=0(○印)が多い。
- 高支出(TotalSpend>5000)ではほぼすべてが赤＝未転送(Transported=0)。

- 💡 結論:
- CryoSleep は TotalSpend とほぼ同義の情報を持ち、Transported をほぼ決定づける主要因。
  つまり「支出ゼロかつCryoSleep=1」であれば、転送される確率が極めて高い。

submission_gbdt_spaceship.py : 0.80196
submission_gbdt_cv_seed42.py : 0.80149

要素	効果
CryoSleep–支出連動補完	Kaggle 上位勢も全員やっている。支出=0 ⇔ CryoSleep=1 の論理整合性が再現できている。
Cabin分解（Deck / Side）	特に Deck は転送率に強い差があり、重要特徴に寄与。
GroupSize・IsAlone	同乗者間の Transported 一致傾向（家族単位で転送されやすい）を拾っている。
HistGradientBoosting	scikit-learn の軽量版GBDT。欠損処理・カテゴリOne-Hot後の汎化性が高い。
TotalSpend と HasSpend	CryoSleepと独立して“支出量の濃淡”をモデルが扱える。

設定	スコア	備考
単一モデル（seed=42固定）	0.80196	fold平均なし、やや運要素あり
5fold 平均 (seed=42固定)	**0.80149**	変動吸収・汎化性向上、ややスコア安定

gbdt, lgbm, catboost

- submission_gbdt_cv_seed42_0.80149.csv
- submission_lgbm_cv_seed42_commented_0.79705.py … スコア下がった
- submission_catboost_cv_seed42_fixed_commented_0.80500.py
- submission_catboost_cpu_cv_deterministic_0.80640.py

- CatBoost化がハマって 0.80149→0.80500→0.80640 にしっかり乗せられています。カテゴリをそのまま食べられる強み＋早期停止の相性が良かったはず。
- いま効いていること(ざっくり)
- CatBoost のカテゴリ処理(HomePlanet / Destination / Deck / Side)
- CryoSleep↔支出の連動補完でリークなく整合
- GroupSize / IsAlone / GroupOrder の家族シグナル
- 5-fold soft-vote で汎化性↑

GPU版は1.8倍高速だが、少しスコアが下がる

benchmark_catboost_cpu_gpu_seed42.py

- ========== まとめ(CPU vs GPU) ==========
- CV Acc mean : CPU 0.81203 | GPU 0.80617
- Fit mean time : CPU 17.20s | GPU 9.68s
- Pred mean time: CPU 0.00s | GPU 0.00s
- Total time : CPU 86.12s | GPU 48.50s

- ざっくり結論
- 速度:GPU は ≈1.8× 高速(fit mean 17.20s → 9.68s、Total 86.12s → 48.50s)
- 精度:CPU の方が +0.006程度 高いCV(0.81203 vs 0.80617)
  - 実際のスコア: CPU:0.80640 vs GPU:0.80617
- この差は珍しくなく、CatBoost では GPU実装の近似や学習挙動(32bit計算・ヒストグラム処理・既定のサンプリング挙動差)により、同一パラメータでも CPU の方が僅かに良いことがあります。GPUは高速だが、微小な精度差が出やすい、という理解でOKです。

最後の一押し：0.80500→0.80640

項目	fixed_commented_0.80500.py	deterministic_0.80640.py
主目的	CatBoostパイプラインを正しく再現（コメント入り）	再現性・安定性を最大化（ベンチCPU設定）
seed設定	SEED = 42 のみ	SEED = 42 + 各種 os.environ によるマルチスレッド固定
スレッド数固定	なし（並列動作のため若干の非決定性あり）	すべてのライブラリ（OMP/MKL/OpenBLAS等）で THREADS=1 に固定
データ順序	CSV読込→そのまま	PassengerIdで .sort_values() してインデックスを固定（並び変動防止）
モデルパラメータ	rsm=0.9（特徴サブサンプリングあり）	rsm を **指定せず**（完全固定性重視、CPUベンチ相当）
random_strength	デフォルト（未指定）	明示的に random_strength=1.0（標準CPU挙動に合わせ）
thread_count	未指定（CatBoost内部が自動並列）	明示的に thread_count=1（完全決定的実行）
CVスコア実績	約 0.80500	約 0.80640（+0.0014改善）