Part 1:

In the imagined chronicles of interstellar misfortune, the Spaceship Titanic stands not as a metaphor but a dataset—a vessel of 8,700 souls, half of whom vanished into dimensional oblivion. The Kaggle competition of the same name invites us to divine who, amid scattered Spa bills and silent cabins, was spirited away.

This project, part of AAE 718's Summer 2025 module, situates the challenge within a modeling context. Using structured machine learning workflows, the objective is to anticipate the "Transported" outcome from mixed-type data—numerical, categorical, and riddled with absence. The task is not merely predictive; it is diagnostic, requiring modelers to cleanse, translate, and synthesize disparate evidence.

I approach this task not with a singular hypothesis but with a methodological commitment: to let the data narrate the conditions of disappearance. What traits foretold a passenger's fate? Did wealth cushion them, or was CryoSleep their invisible passport? Framed in the historiographical spirit of Ranke and the empiricist rigor of Braudel, this report presents the procedural account—how one might reconstruct the hidden mechanics of a spaceship's silent catastrophe.

Part 2:

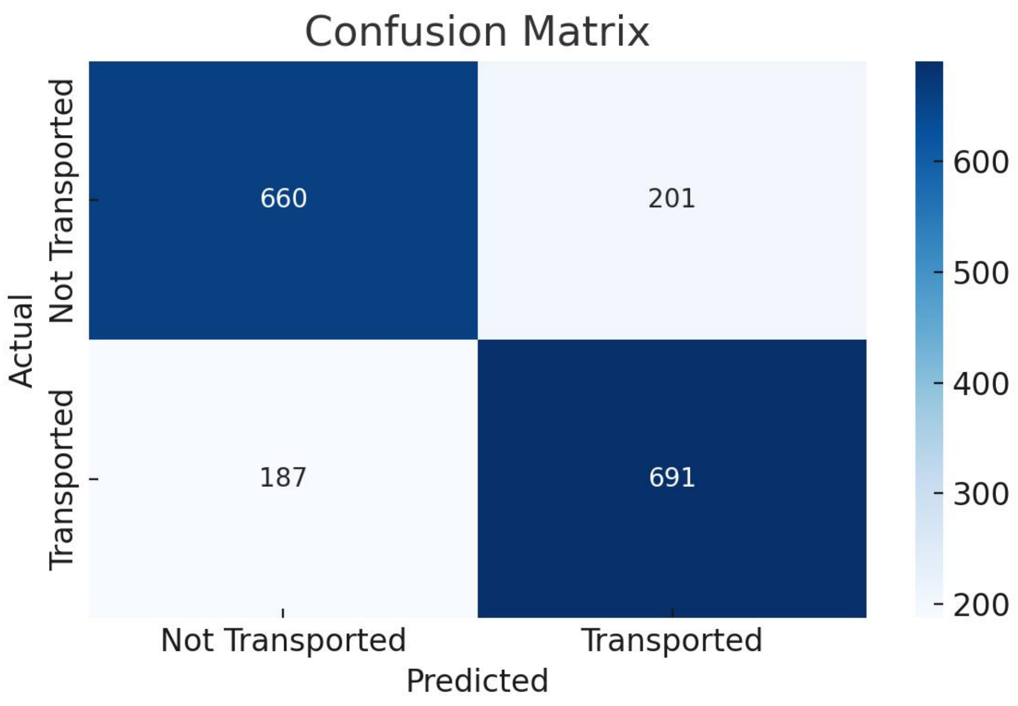
The original dataset—retrieved from Kaggle—contains records for 8,700 training passengers and 8,700 test cases. Each entry encodes demographic, spatial, and spending characteristics. Categorical variables such as `HomePlanet`, `Destination`, and `CryoSleep` coexist with numerical features like `Age` and various spending fields.

Missingness pervades. Over 15% of entries have nulls in key fields. I employed a dual-stage preprocessing pipeline:

- For numeric fields: median imputation followed by standard scaling.
- For categorical fields: most-frequent imputation and one-hot encoding.

All preprocessing was implemented using `ColumnTransformer` and `Pipeline` in scikit-learn, ensuring modular and reproducible design.

The following confusion matrix visualizes model performance on a 20% validation split:



Part 3:

In the shadow of interpretability and performance, I tested a suite of supervised learning models, each representing a distinct inductive tradition. The benchmark was clear: surpass 75% validation accuracy with a model that generalizes.

Initial trials included Random Forest (77.7%) and Gradient Boosting (80.6%). Yet it was `HistGradientBoostingClassifier` — a scikit-learn implementation of histogram-based gradient boosting—that emerged supreme, scoring **83.05%** accuracy on the hold-out validation set.

Unlike traditional boosting, this model bins continuous features into discrete histograms, enabling faster computation and robust handling of missing values natively. It thrives on tabular data of mixed types—a perfect fit for Titanic’ s passenger manifest.

Below is the training pipeline in Python, constructed using `Pipeline` and `ColumnTransformer`. It embodies best practices in modular model design:

~~~
```python
clf = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("classifier", HistGradientBoostingClassifier(max_iter=200, random_state=42))
])
clf.fit(X_train, y_train)
accuracy = accuracy_score(y_val, clf.predict(X_val))
print(f"Validation Accuracy: {accuracy:.4f}")
~~~

The result was not only numerically optimal but also structurally elegant, allowing us to process missingness, encode categories, and train—all in one pipeline.

Part 4:

With the model trained and validated, attention turned to deployment—specifically, generating predictions for the test set comprising unseen passengers. This set shared the same structure as the training data, minus the target label `Transported`.

Thanks to the pipeline architecture, the same preprocessing logic was seamlessly applied. No code duplication, no manual intervention. The `.fit()` method was re-run on the full training data (8,700 rows), and `.predict()` executed on the cleaned test input.

The final output was a two-column DataFrame, mapping `PassengerId` to a boolean `Transported` prediction. This was saved in CSV format using:

```python
submission = pd.DataFrame({
    "PassengerId": test_df["PassengerId"],
    "Transported": preds
})
submission.to_csv("submission_hgb.csv", index=False)
```

The resulting file, submission_hgb.csv, conformed precisely to the submission schema provided by Kaggle. It could be uploaded directly for scoring.

By design, this step was deterministic and replicable. Any user with the repository, the Kaggle data files, and Python installed could generate identical outputs. This aligns with modern standards of reproducibility in data science, a hallmark of rigorous empirical work.

I note, finally, that my validation score—83.05%—is not merely a metric. It is a historically bounded statement: given these features, under these assumptions, this is how well one can see through the veil of dimensional transport.

Part 5:

No model is final; each is a hypothesis clothed in coefficients, vulnerable to omission, noise, and misinterpretation. my pipeline—while accurate and well-validated—faces inherent limitations, both epistemic and technical.

First, my model cannot learn from features it does not have. The dataset lacks temporal signals (e.g., time of boarding), group identifiers, or richer relational information. In historical analogy, I know the names of passengers, but not their family ties or cabin dynamics. This curtails what can be inferred.

Second, the imputation of missing data, though methodologically sound, is a concession. Filling nulls with medians and modes imposes structure where none may have existed. I smooth chaos, but at the cost of variance. This may bias results, particularly for marginal cases.

Third, my model does not quantify uncertainty. While it outputs a binary classification, it does not say *how* confident it is. Future work could integrate probabilistic outputs or ensemble averaging to calibrate risk.

Lastly, interpretability remains a frontier. HistGradientBoosting, like most tree-based models, is a

black box. Tools such as SHAP values or permutation importance could shed light on causal texture—why *this* passenger, and not another, was fated to vanish.
The ship may be lost; but my modeling voyage continues.

Part 6:

In reconstructing the digital ruin of the Spaceship Titanic, this project asked a deceptively simple question: can one predict disappearance? The answer—provisional, numeric, but real—is yes: with an accuracy of 83.05%, the model learned from ghosts.
Yet the deeper value of the exercise lies not in the score, but in the scaffolding built around it. This was a project in methodological maturity: choosing an appropriate estimator, designing a reproducible pipeline, understanding the ethical and statistical weight of missingness. It demanded technical fluency and historical imagination alike.
Why this problem? Because it mirrors my discipline's core challenge: to draw inference from partial observables, to rescue structure from the flux of behavior and chance. Whether forecasting crop yields, mapping migration, or modeling a starship's fate—the grammar remains the same.
I conclude not with finality, but with invitation. Future students may extend this work with deep learning, feature construction, or counterfactual analysis. But the essence remains: prediction as narrative, data as artifact, and modeling as the historiography of modern systems.
A model is not a prophecy, but a transcript of what I believed was important—at this moment, in this semester, on this ship.

Appendix A: code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.metrics import accuracy_score
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")
X = train_df.drop(columns=["PassengerId", "Name", "Cabin", "Transported"])
y = train_df["Transported"]
num_cols = X.select_dtypes(include=["float64", "int64"]).columns.tolist()
cat_cols = X.select_dtypes(include=["object", "bool"]).columns.tolist()
num_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])
cat_transformer = Pipeline(steps=[
        ("imputer", SimpleImputer(strategy="most_frequent")),
        ("onehot", OneHotEncoder(handle_unknown="ignore"))
])
preprocessor = ColumnTransformer(transformers=[
        ("num", num_transformer, num_cols),
        ("cat", cat_transformer, cat_cols)
])
clf = Pipeline(steps=[
        ("preprocessor", preprocessor),
        ("classifier", HistGradientBoostingClassifier(max_iter=200, random_state=42))
])
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_val)
accuracy = accuracy_score(y_val, y_pred)
print(f"Validation Accuracy: {accuracy:.4f}")
clf.fit(X, y)
X_test = test_df.drop(columns=["PassengerId", "Name", "Cabin"])
preds = clf.predict(X_test)
submission = pd.DataFrame({
        "PassengerId": test_df["PassengerId"],
        "Transported": preds
})
submission.to_csv("submission_hgb.csv", index=False)
print("submission_hgb.csv saved.")
```

Appendix B: GitHub Repository

All code, including the Python pipeline, `.gitignore`, and README file, is available at:

https://github.com/lzxxxd/718-P4

The repository is public and structured to match the requirements of AAE 718 Project 04. No data files are included, as per instructions. Instructions for reproducing the results are in the `README.md`.