Spaceship Titanic

Jason Baer, Richard Garcia, Devin Lane, Aaron Bruner

Overview of Dataset
Spaceship Titanic

In a very distant future set in 2912 the skills of data science are needed to solve a cosmic mystery. The *Spaceship Titanic* was an interstellar passenger liner launched a month ago. With almost 13,000 passengers on board, the vessel set out on its maiden voyage transporting emigrants from our solar system to three newly habitable exoplanets orbiting nearby stars.

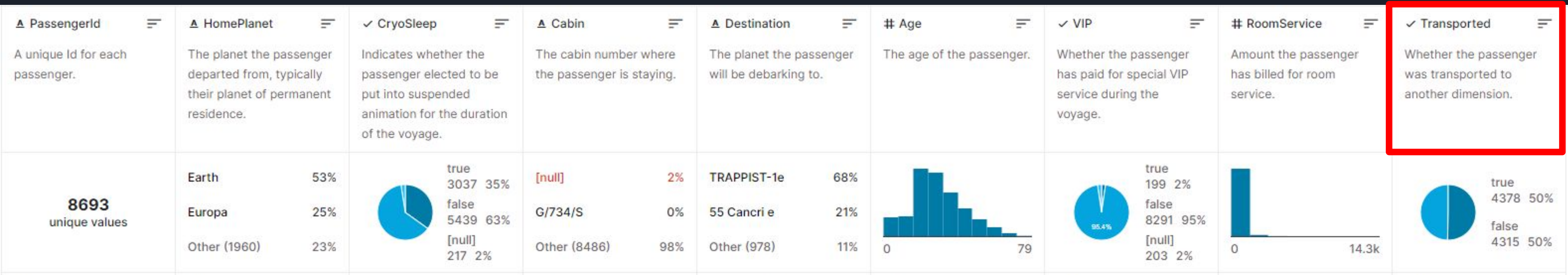
While rounding Alpha Centauri en route to its first destination—the torrid 55 Cancri E—the unwary *Spaceship Titanic* collided with a spacetime anomaly hidden within a dust cloud. Sadly, it met a similar fate as its namesake from 1000 years before. Though the ship stayed intact, almost half of the passengers were transported to an alternate dimension!

To help rescue crews and retrieve the lost passengers, the challenge is to predict which passengers were transported by the anomaly using records recovered from the spaceship's damaged computer system.

Objective of Competition

The competition has a total of 3 files included:

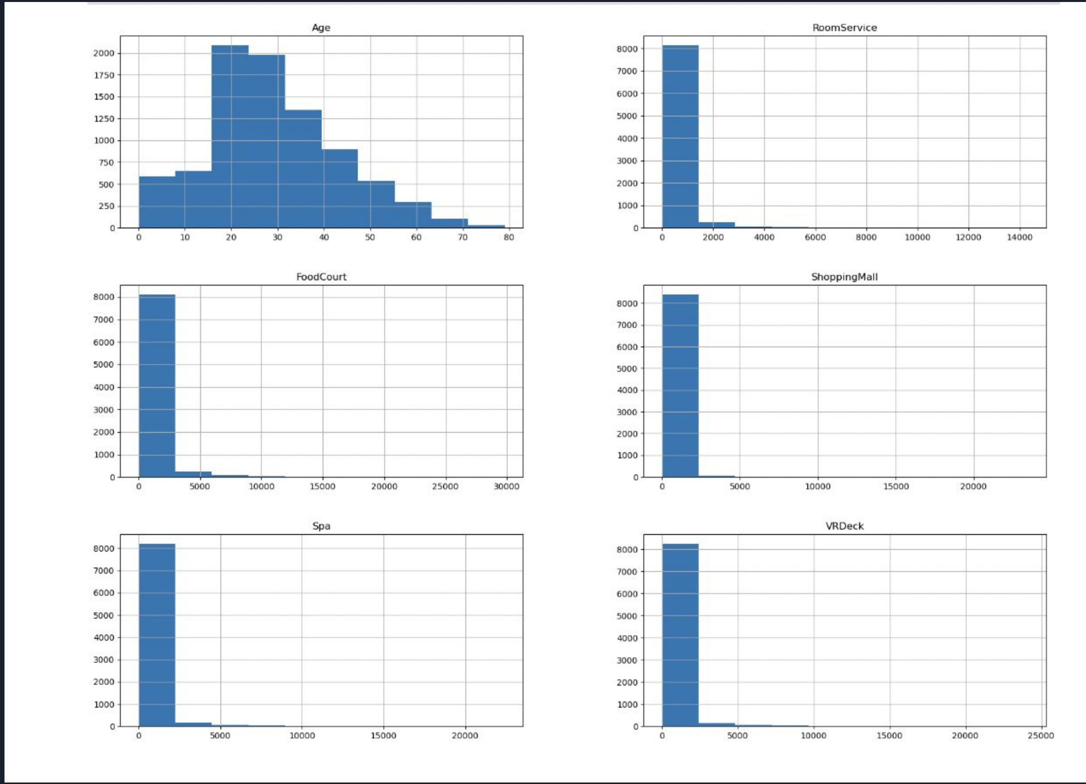
- Train.csv - Personal records for about two-thirds (~8700) of the passengers, to be used as training data
    - Columns: PassangerID, HomePlanet, CryoSleep, Cabin, Destination, Age, VIP, RoomService, FoodCourt, ShoppingMall, Spa, VRDeck, Name, Transported
- Test.csv - Personal records for the remaining one-third (~4300) of the passengers, to be used as test data.
    - **Task: Predict the column Transported for the passengers in this set**
- Sample_submission.csv - A submission file in the correct format
    - Columns: Contains PassengerID and Transported (True/False)



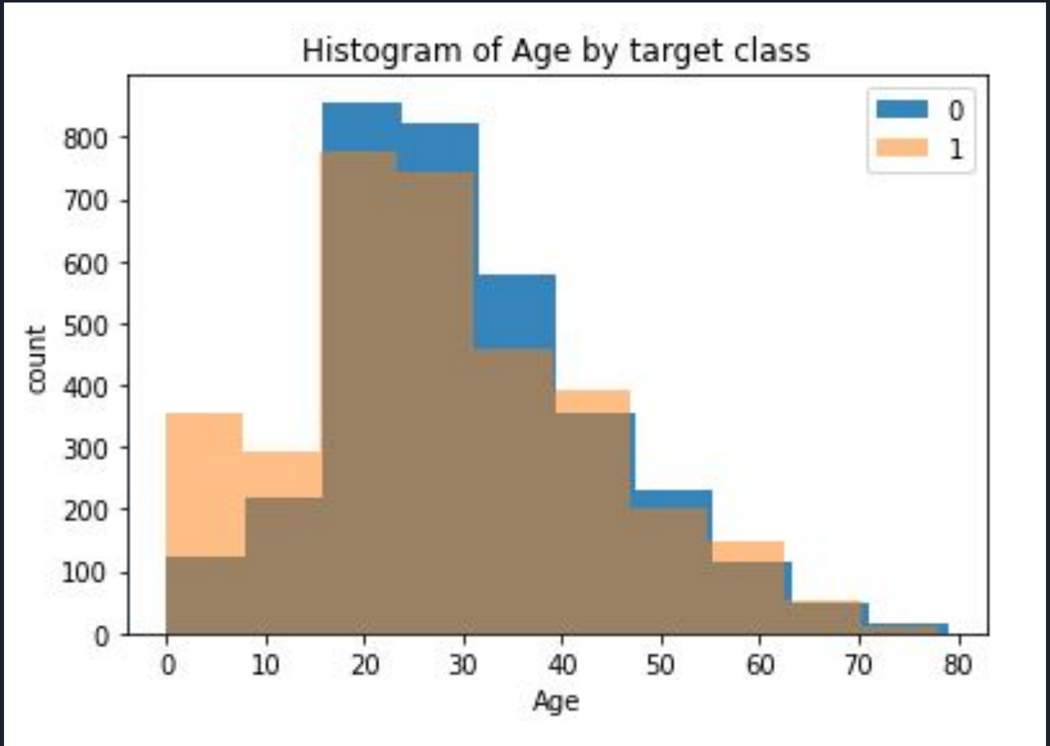
Exploratory Data Analysis - NaN

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   PassengerId   8693 non-null   object 
 1   HomePlanet    8492 non-null   object 
 2   CryoSleep     8476 non-null   object 
 3   Cabin         8494 non-null   object 
 4   Destination   8511 non-null   object 
 5   Age           8514 non-null   float64
 6   VIP           8490 non-null   object 
 7   RoomService   8512 non-null   float64
 8   FoodCourt     8510 non-null   float64
 9   ShoppingMall  8485 non-null   float64
 10  Spa           8510 non-null   float64
 11  VRDeck        8505 non-null   float64
 12  Name          8493 non-null   object 
 13  Transported   8693 non-null   bool   
dtypes: bool(1), float64(6), object(7)
memory usage: 891.5+ KB
```

Exploratory Data Analysis - Distributions



Exploratory Data Analysis - Age Distribution



Preprocessing/Feature Engineering - Column Transformer

Feature	Transformation
HomePlanet	imputation, One-Hot Encoding
Cabin (or cabin-1/2/3)	imputation, One-Hot Encoding
Destination	imputation, One-Hot Encoding
Age	imputation, scaling
RoomService	imputation, scaling
FoodCourt	imputation, scaling
ShoppingMall	imputation, scaling
Spa	imputation, scaling
VRDeck	imputation, scaling
CryoSleep	imputation, One-Hot Encoding
VIP	imputation, One-Hot Encoding
name	drop
Transported	target

In [7]:
```
numeric_features = ['Age', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']
categorical_features = ['HomePlanet', 'CryoSleep', 'Cabin', 'Destination', 'VIP']
drop_features = ['PassengerId', 'Name']
target = 'Transported'
```

In [8]:
```
# Split our data
X_train, y_train = train_df.drop(columns=[target]), train_df[[target]]
X_test, y_test = test_df.drop(columns=[target]), test_df[[target]]
```

In [9]:
```
# Create our preprocessor
preprocessor = make_column_transformer(
    # Apply standard scaling and imputation to our numeric features
    (make_pipeline(SimpleImputer(strategy='mean'), StandardScaler()), numeric_features),
    # Apply one-hot encoding and imputation to categorical features
    (make_pipeline(SimpleImputer(strategy='most_frequent'), OneHotEncoder(handle_unknown='ignore')), categorical_features),
    # Drop our bad features
    ('drop', drop_features)
)
```

In [10]:
```
# Let's observe our tranformation on the data
X_train_transformed = preprocessor.fit_transform(X_train)
X_train_transformed.shape # Notice how many columns we have!
```

Out[10]:
```
(6954, 5482)
```

Models

Models used and their test scores (from cross-validation):

	DummyClassifier	**K-Nearest Neighbors (n=23)**	**Logistic Regression**	**Random Forest**	**XGBOOST**	**CatBoost**
fit_time	0.050 (+/-) 0.002	0.0453	0.050 (+/-) 0.002	0.545 (+/-) 0.013	0.242 (+/-) 0.007	2.667 (+/-) 0.036
score_time	0.021 (+/-) 0.000	0.6689	.004	0.039 (+/-) 0.001	0.026 (+/-) 0.023	0.011 (+/-) 0.000
test_score	0.510 (+/-) 0.000	0.782	.807 (+/-) 0.02	0.820 (+/-) 0.017	0.813 (+/-) 0.013	0.829 (+/-) 0.039
train_score	0.502 (+/-) 0.000	0.7966	.79	.853(+/-) 0.004	0.838 (+/-) 0.006	0.881 (+/-) 0.007

Ensembles

Prediction Score

Random Forests®

- Tuned 6 hyperparameters
- ‘Mean’ and ‘Median’ for scalars
- Increased by .6%

```
{'randomforestclassifier__n_estimators': 130,
 'randomforestclassifier__min_samples_split': 9,
 'randomforestclassifier__min_samples_leaf': 4,
 'randomforestclassifier__max_features': 'log2',
 'randomforestclassifier__max_depth': 11,
 'randomforestclassifier__criterion': 'gini'}
```

dmlc XGBoost

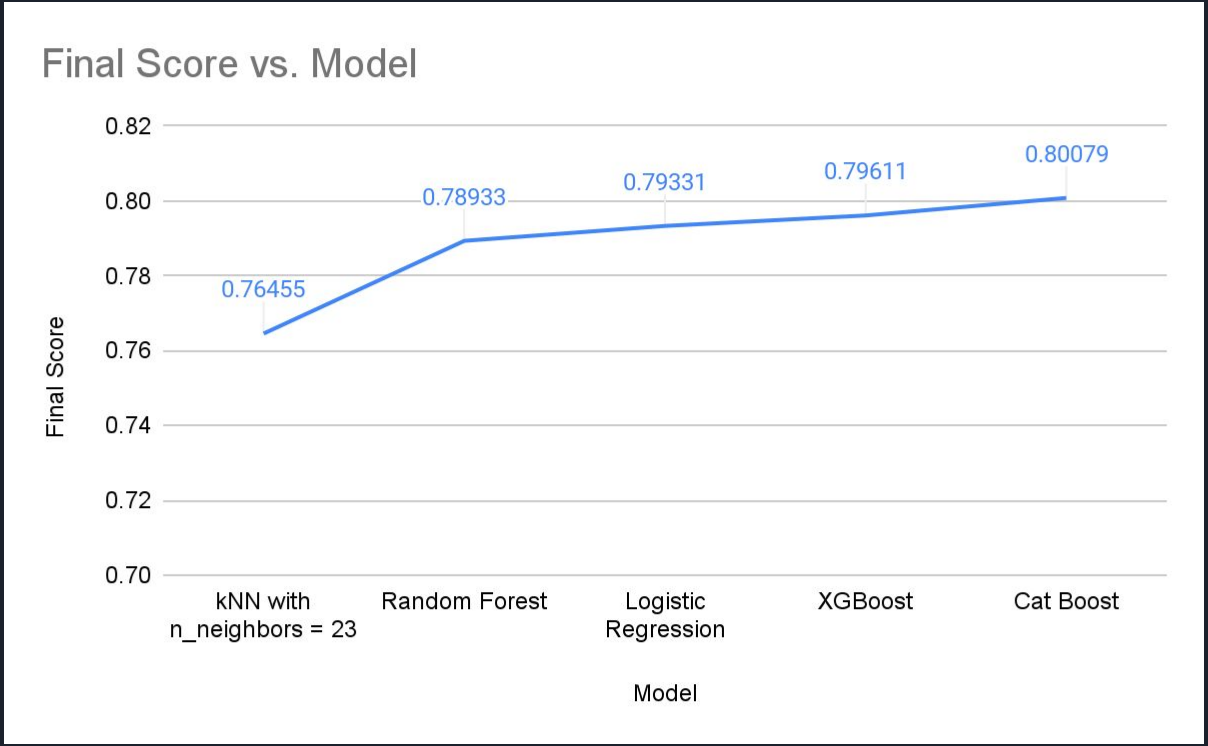
- Tuned 5 hyperparameters
- ‘Mean’ and ‘Median’ for scalars
- Increased by .5%

```
{'xgbclassifier__min_child_weight': 3,
 'xgbclassifier__max_depth': 3,
 'xgbclassifier__learning_rate': 0.25,
 'xgbclassifier__gamma': 0.4,
 'xgbclassifier__colsample_bytree': 0.5}
```

CatBoost

- Untuned
    - Very high possibility for a greater score
- Tried with and without one-hot encoding

Evaluation of Submission Scores



Conclusion and Future Work

- As the full project is not due yet, we will continue creating more models and doing more tuning
  - We will try new models that were not discussed in class
  - More hyperparameter tuning is possible for CatBoost
- Future Work:
  - Complete at least one more non-dummy submission.
  - Potentially learn and develop applications of Deep Learning models for the Spaceship Titanic dataset.
  - Ex: Multilayer Perceptrons (Classic Neural Networks): a neural network consisting of more than 2 layers
    - Best for tabular data and classification problems.