Shrey Shah, Robert Feliciano, and Ariela Litvin

Professor In Suk Jang

CS 559 WS

12 May 2023

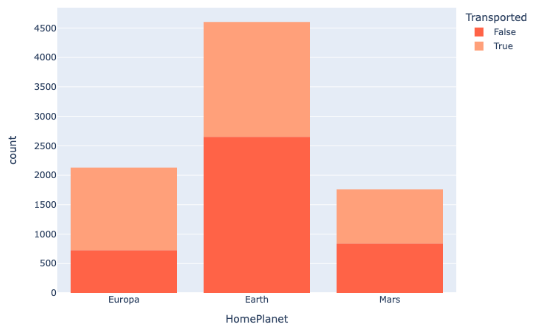
Group 18 Final Report

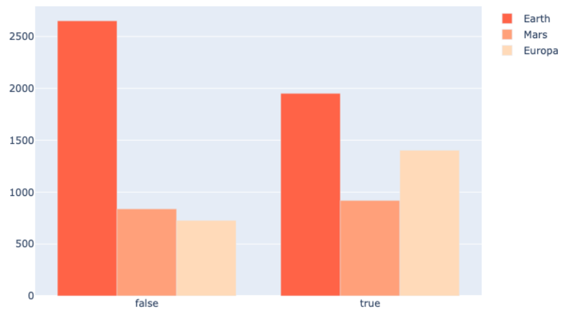
Introduction

This project was a Kaggle competition based on a fictitious science fiction scenario wherein a Spaceship Titanic crashed into a spacetime anomaly and caused many of the passengers to be transported to an alternate dimension. The goal of the project was to predict whether a passenger would be transported by the anomaly, represented by the transported boolean column in the data set, using a supervised ensemble machine learning algorithm. The provided data set included various information about the passengers and their onboard behavior and accommodations.

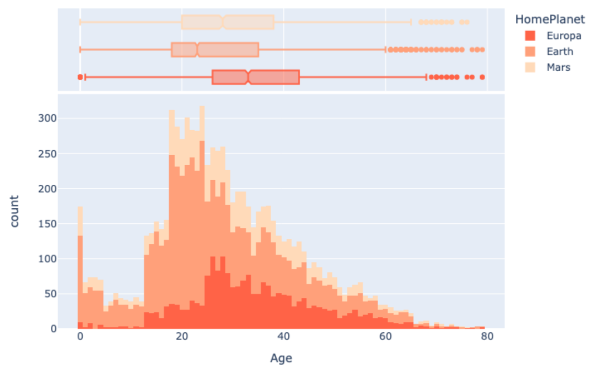
Methodology

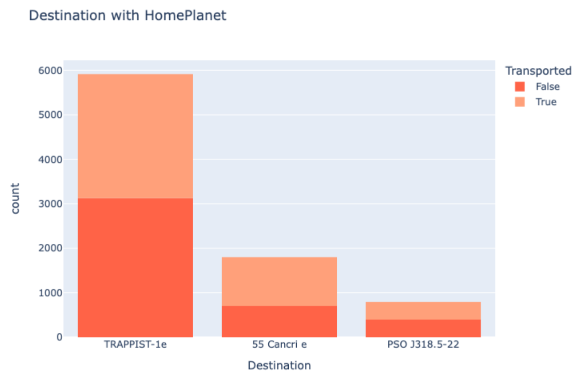
The project included various visualizations to become familiar with the data before working with it and preprocessing. For instance, below are various visualizations to see the relationship between the transported feature and the passenger's home planet. The graph shows that the most passengers came from Earth and the least came from Europa, but leaving from Europa meant that the passenger had a better chance of being transported than from Mars.

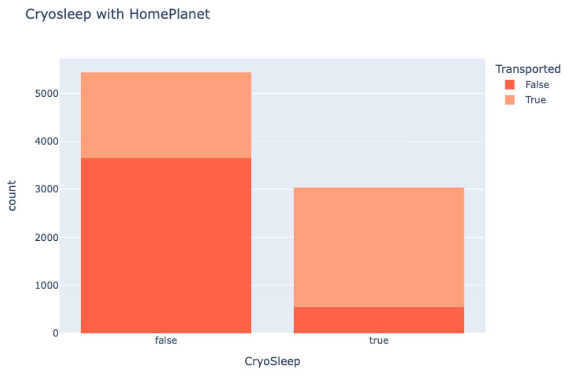


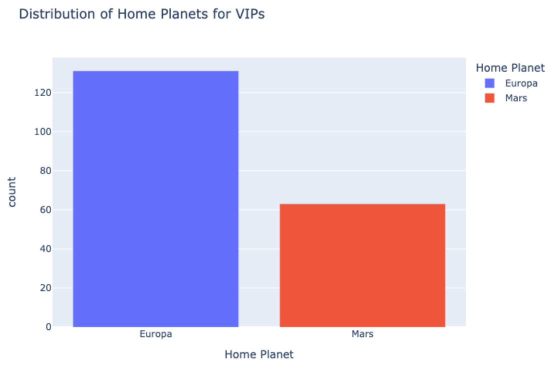


The data visualization step also included other features' relationship with the passengers' home planet since this was determined to be an informative and easy to visualize feature. This included seeing the age distribution for each planet using histograms and frequency graphs, which show the skew in the age range that may affect the training. We also created bar charts of these relationships, such as how the home planet affects the passenger's destination, whether they went into cryosleep, and whether they are a VIP. These graphs are depicted below.

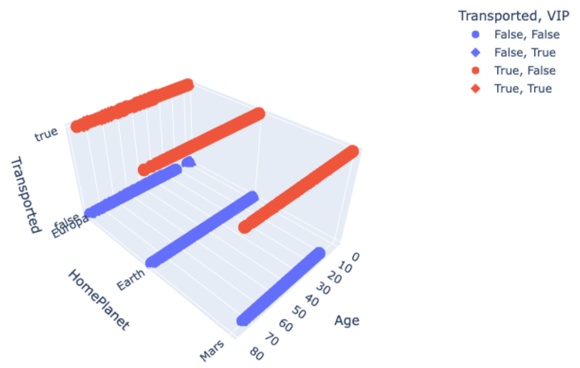




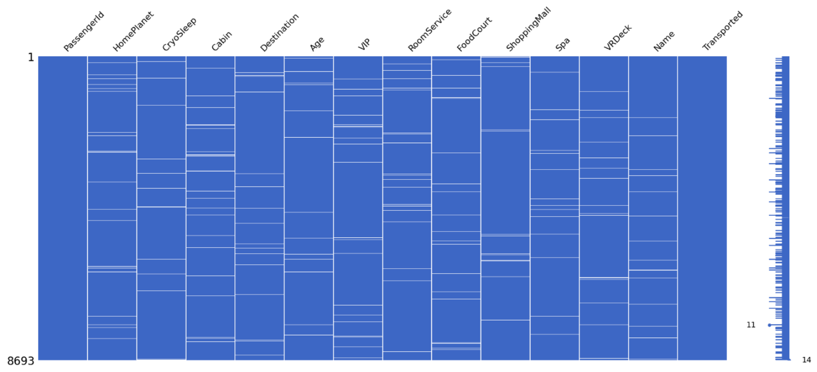




The project also had four-dimensional visualizations of relationships, such as the relationship between the age, home planet, transported, and VIP features. This helps us to identify relationships between features that ideally would be independent.

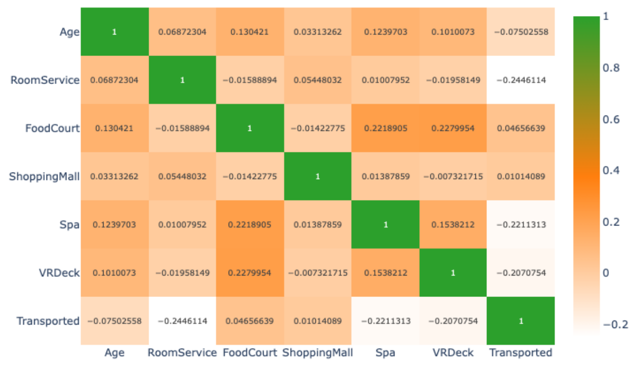


The next graph visualized the missing data with each white space representing a missing data value. Certain features, such as the cabin or room service labels, have many missing data entries, which would need to be addressed during preprocessing.



The preprocessing included filling empty values on text features, such as a passenger's home planet and destination, and boolean features, such as whether a passenger is VIP or in cryosleep, with the most frequent values for that feature. Numerical features, such as a passenger's age, were filled using the median for the home planet of the passenger.

The below graph shows the visualization of correlation between features. Many features were fairly independent. Certain features had more impactful correlation, such as the amount spent on the food court having a correlation to the amount spent on the spa and the amount spent on the virtual reality deck. This graph helped to inform our feature engineering.



During the project’s feature engineering, the passenger id and name columns were dropped due to them only being used as sample identification, rather than as an informative training feature. This is in order to ensure that the training algorithm will not make any false correlations based on these features. The cabin column was replaced with deck, room number, and ship side columns. The amount spent on room service, the food court, the shopping mall, and the virtual reality deck were combined in a total spent column. This was due to the high correlation between these features in order to keep the features as independent as possible. This also minimizes the number of features which in turn lessens the training time. The categorical text was then one-hot encoded. For instance, boolean features, such as whether a passenger is a VIP, were given a zero or a one to represent true or false. The text features home planet, destination, and room deck were split into separate categories for each unique value and given a one for when that value was true for that passenger sample. This categorical encoding makes it easier for the training algorithms to learn from the data set and utilize the features in their calculations. The final step of the data processing was to separate the transported column to be used as the target values for the model.

The data set was used to train a supervised ensemble machine learning model. Seven classifiers were used, with a decision tree classifier, a random forest classifier, a gradient boosting classifier, extreme gradient boosting classifier, light gradient boosting classifier, and categorical boosting classifier, as the base classifier and a voting classifier which votes by average to reach the final decision. Decision tree classification uses CART decision trees, or classification and regression trees, as base learners. Each leaf node scores whether an instance belongs to a group. When the tree reaches max depth, the decision is converted into categories based on certain thresholds. This type of classification serves as a base for the other types of ensemble algorithms used for this project. Random forest classification creates a forest by building each tree from a sample randomly drawn without replacement from the training set. Each tree has a low correlation. They then vote either by majority or average to get the final result of each sample to help avoid overfitting and make more accurate predictions than one large individual tree would.

Gradient boosting tries to improve its predecessors by reducing the errors. After the first stage, it fits on the residual errors made by the previous predictors, rather than the training data. This means that unlike random forest classification, the decision trees are built additively onto each other rather than independent of each other. Since the trees are learning from and boosting each other, it can give better accuracy than random forest classification. Extreme gradient boosting, or XGBoost, uses more advanced regularization than gradient boosting, which improves model generalization capabilities to avoid overfitting and gives faster performance. Light gradient boosting classification, or LGBM, grows trees vertically, or leaf-wise, unlike gradient boosting or XGBoost, by choosing the leaf with the maximum delta loss to grow which allows it to often reduce loss quicker than other algorithms. It is more sensitive to overfitting than some other ensemble algorithms, which is why it was combined with algorithms that avoid overfitting, such as random forest classification and XGBoost. Categorical boosting, or CatBoost, builds symmetrical balanced decision trees, unlike XGBoost and LGBM. In every step, leaves from the previous tree are split by selecting the lowest loss to split the level's nodes. This helps the model's time efficiency while controlling overfitting by serving as regularization. CatBoost also uses ordered boosting, a permutation-driven approach to train the model on a sample of data while calculating residuals on another sample, which helps to prevent overfitting, similar to random forest classification. All the base classifiers had their hyperparameters tuned using GridSearchCV. The options provided for each tuning were maximum tree depths of 3, 4, or 5 and 100, 200, or 300 estimators where applicable.

Results

Our team ranked 524 with a score of 0.802.

524	Group-18	0.80243	2	7d

Before hyperparameter tuning our logistic regression model resulted in an accuracy of 0.8000. Hyperparameter tuning optimized the decision tree classifier to a maximum tree depth of 5 which resulted in an accuracy of 0.7860. The random forest classifier was tuned to a maximum depth of 5 for each tree and 200 total trees to achieve an accuracy of 0.7776 on the testing data set. The gradient boosting, extreme gradient boosting, and light gradient boosting classifiers were found to work best at maximum depths of 4 and with 100 boosting stages each. They received scores of 0.8083, 0.8021, and 0.8106, respectively. Ensembling all these sub models using a voting classifier provided a final prediction accuracy of 0.8100 when scored on the testing data set.

Roles

Shrey did most of the programming for this assignment. All group members contributed to the presentation, with Ariela and Robert doing most of the powerpoint slides. Robert edited the presentation video. Ariela did most of the report.

I pledge my honor that I have abided by the Stevens Honor System.