Spaceship Titanic

Testing stacking methods in a Kaggle competition

By Jordan Boswell

Introduction

Spaceship Titanic Competition

kaggle

- *“Predict which passengers are transported to an alternate dimension”*
  - A traveling spaceship collides with a spacetime anomaly, transporting some passengers to an alternate dimension
  - Predict which passengers were transported
- Ongoing Kaggle competition set up by the Kaggle team, with synthetic data
- Standard supervised learning task to predict a binary outcome using a tabular dataset
- Data is split (by Kaggle) into a training set and a test set
- Predictions made on the test set are uploaded and autograded
- A leaderboard tracks the performance of your model relative to peers

Goals

- Feature Engineering – Create meaningful features from the data
- Data Imputation – Impute missing data optimally
- Modeling – Build and test the performance of multiple classification models
- Stacking – Utilize different stacking methods, and compare their performance with each other and with singular-model methods

Dataset

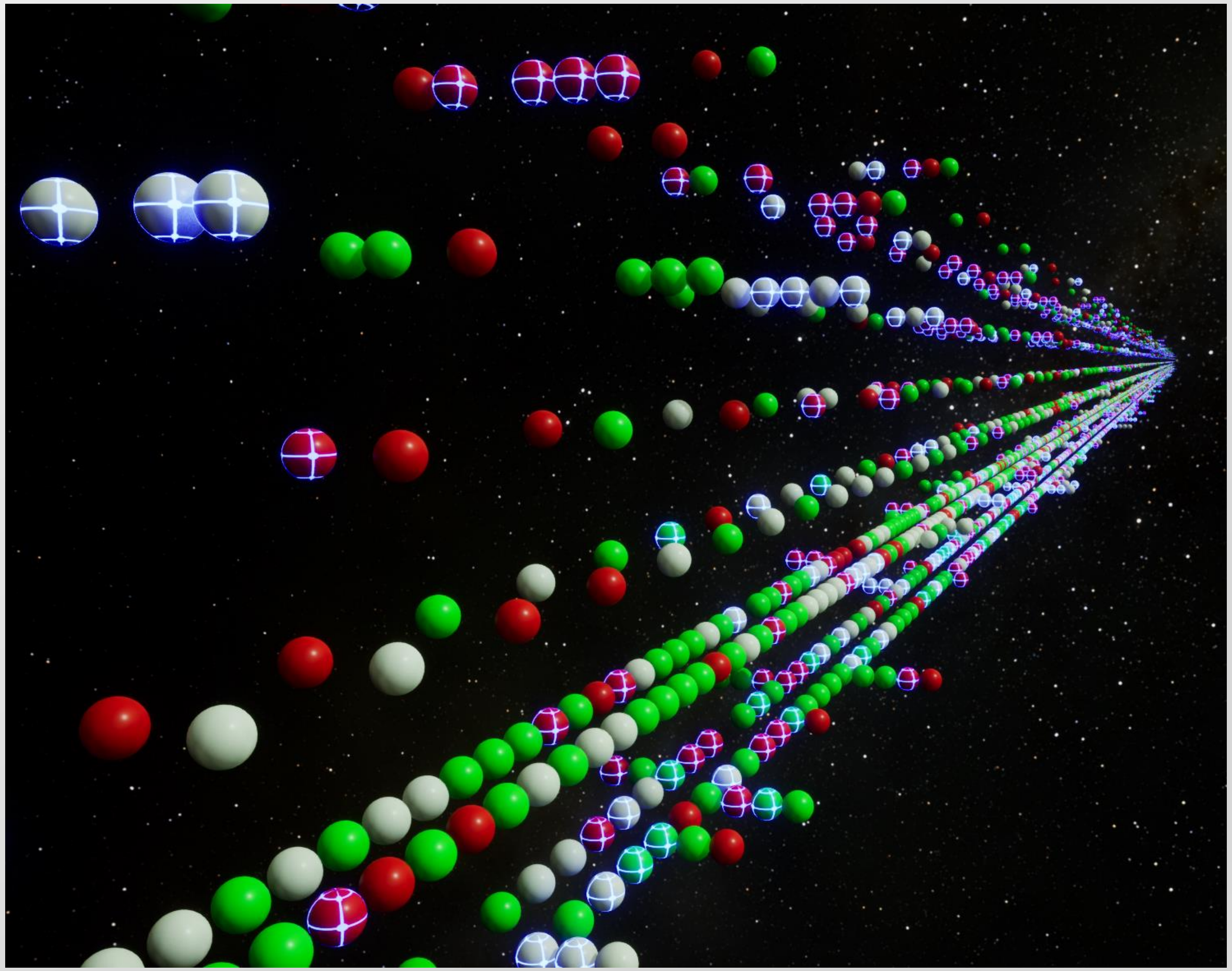
- CSV files with 14 columns and 8693(train)/4277(test) rows
- Roughly 2% missing data
- Predictors:
  - PassengerId – Unique id for each passenger, consisting of a group id portion and an individual (within-group) portion.
  - HomePlanet – The planet the passenger departed from
  - Destination – The passenger’s destination planet
  - CryoSleep – Whether the passenger is in suspended animation or not
  - Cabin – The room, consisting of deck, side, and room number components
  - Age – The age in years of the passenger
  - VIP – Whether the passenger is a VIP or not
  - RoomService, FoodCourt, ShoppingMall, Spa, VRDeck – Amount of money spent at various locations
  - Name – The passenger’s full name
- Response:
  - Transported – Whether the passenger was transported to an alternate dimension or not.

Exploratory Data Analysis

Real-Time 3d Visualization in Unreal Engine 5

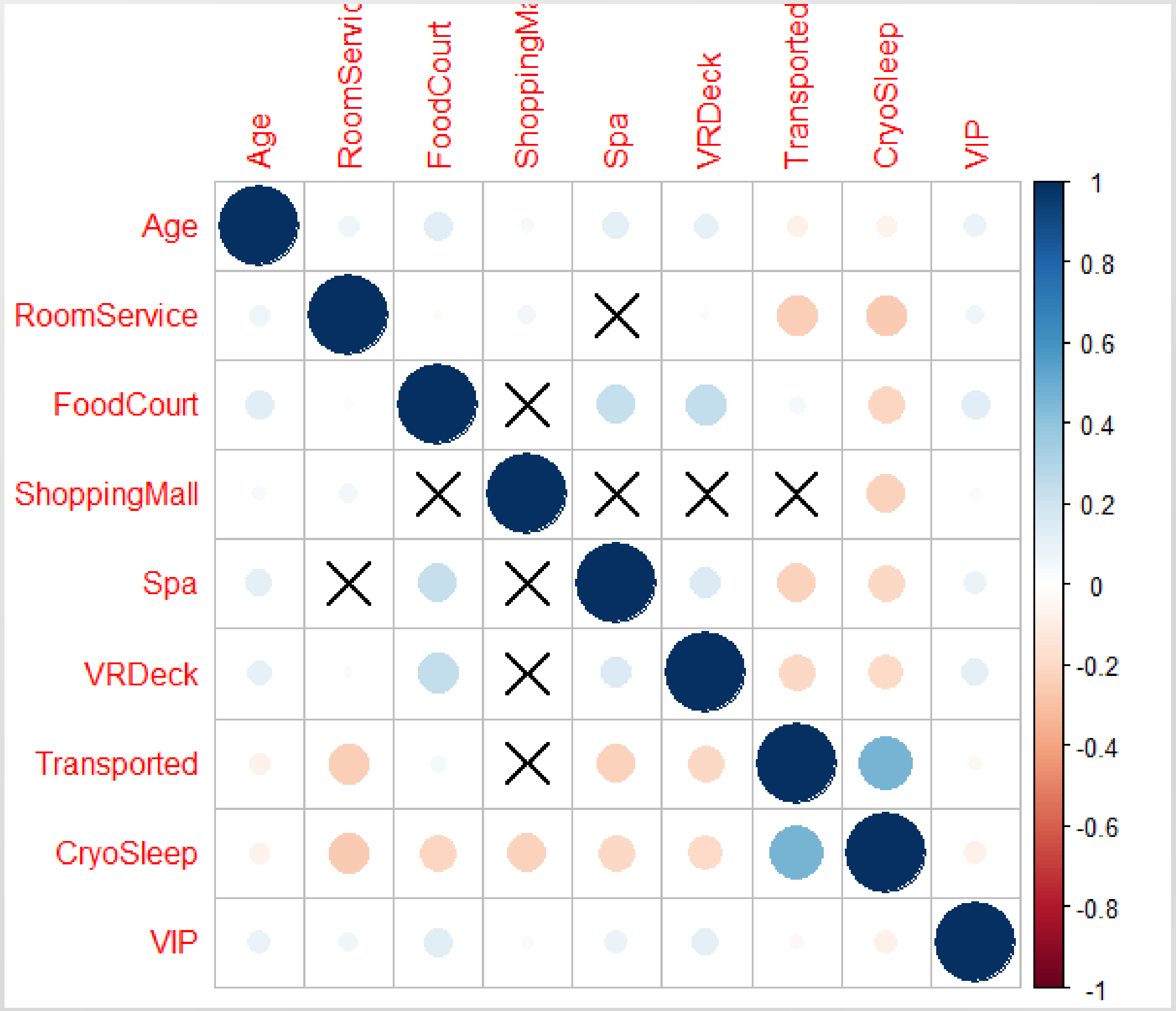
Each sphere represents a passenger. Decks are represented by position on the up/down axis, sides are represented by the separation on the left/right axis, and cabin numbers are represented in order by position on the forward/backward axis. Red represents transported, green not transported, and grey is unknown (in the test set). Passengers with a luminous blue cross indicate cryosleep.

One takeaway from this visualization was the seemingly spatial clustering of transported passengers.



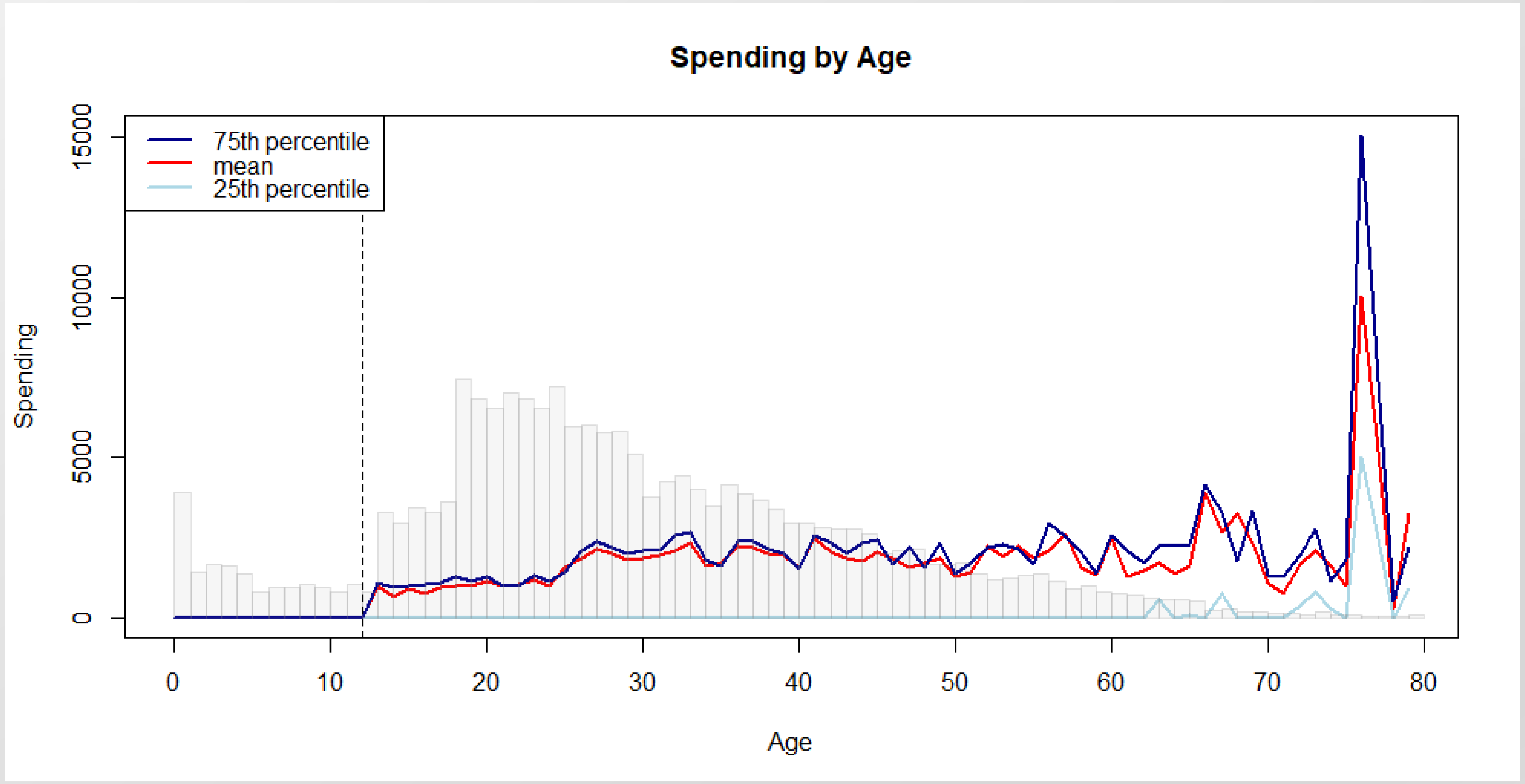
Correlation Analysis

Correlation plot of various predictors and the response. Various correlations exist. Notable are the positive correlations between Transported and the spending predictors, as well as the negative correlation between Transported and CryoSleep.



Exploration of Spending and Age

This clearly shows the distribution by age, the relationship between age and spending, and how children 12 and younger don't spend money.



Patterns Found that Allow Manual Imputation of Some Missing Values

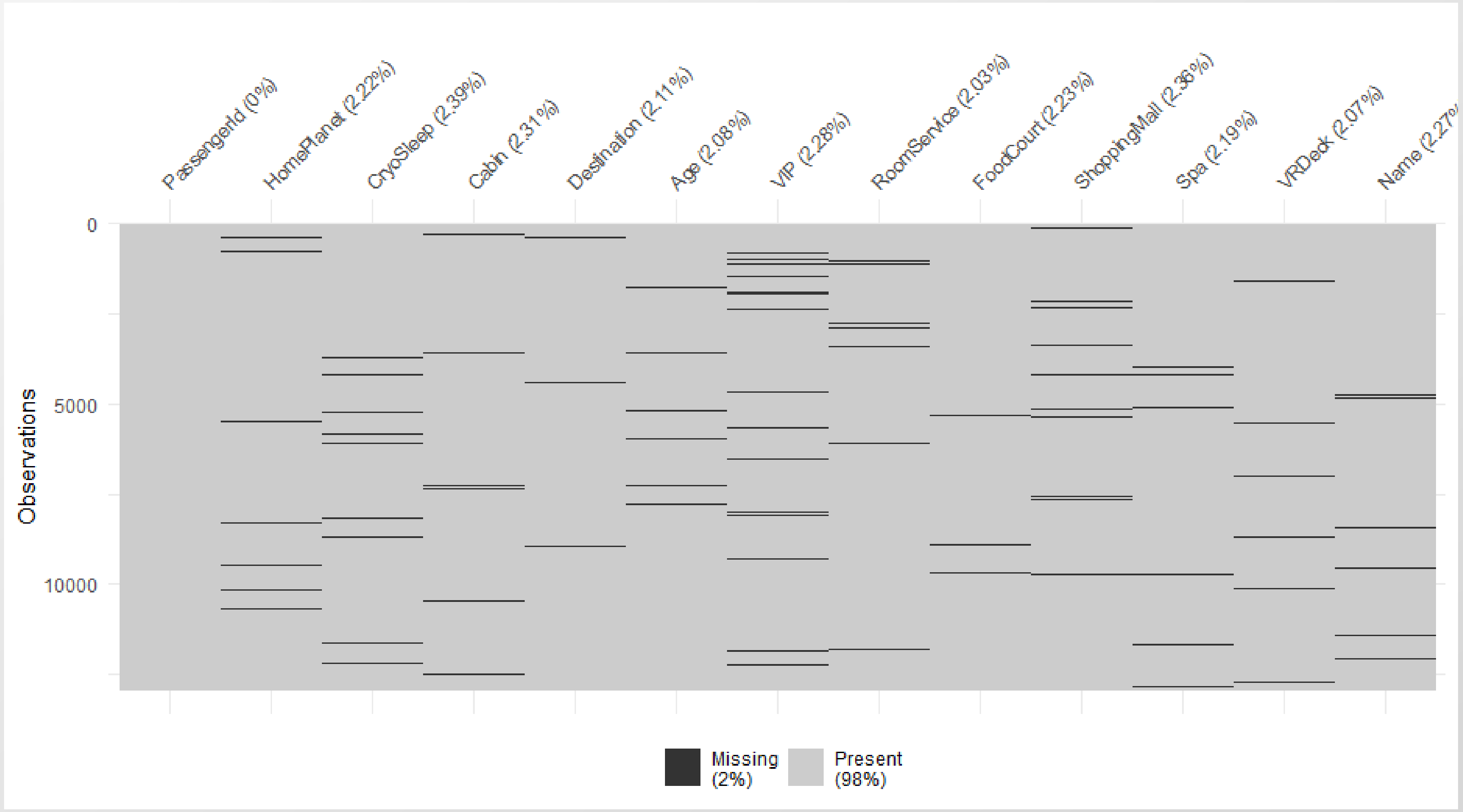
- Passengers in cryosleep don’t spend any money
- Children (Age <= 12) don’t spend any money
- Some decks exclusively have passengers from certain home planets
- People in the same group are on the same side of the ship
- People in a group usually share the same cabin
- Groups always have one home planet, but can have multiple destinations
- No one from Earth has VIP status

Missing Data Imputation

Manual Imputation Based on Strong Patterns

- Some EDA discovered patterns that held in all cases.
- We manually imputed some missing values using those patterns.
- The majority of the missing data still remained after this step.

Analysis of Missing Data



- Little's MCAR test was used to test if the missing data were missing completely at random. The results were in favor of not rejecting the null hypothesis, and thus favored the assumption that the missing data were missing completely at random (p-value of 0.47). The MCAR property allows the imputation of data without introducing bias.

Non-Parametric Missing Value Imputation Using Random Forests

- The missForest R library was used to impute the remaining missing data
- The training and testing data were stacked together, and the response variable was removed
- The combined data set was fed into missForest for imputation
- Multiple iterations of imputation were performed to produce a final, complete dataset
  - For the initial iteration, mean and mode imputation was performed.
  - For each subsequent iteration, for each column, a random forest of 1500 trees was fit with that column as the response and the rest as the predictors, and then that column’s cells that originally had missing data were predicted.
  - Iterations continued until the predictions converged (didn’t change much between iterations)

Feature Engineering

Basic Column Creation

- The cabin was parsed and split into three columns:
  - Deck – The deck letter
  - Side – The side of the ship; either S (starboard) or P (port)
  - Num – The room number
- PassengerId was split into IID (individual) and GID (group) components
- The spending columns were added to create a Spending column
- A Boolean HasSpent column was created
- The name was split into FirstName and LastName columns
- A Boolean IsChild column was created to indicate if passengers were 12 or younger

Group Columns

- Each group was analyzed and new columns created
  - Group size
  - Number of group members transported
  - Number of group members in cryosleep
  - Number of cabins occupied by the group
  - Etc.

Cabin Columns

- Similarly, cabins were analyzed and new cabin-based columns created
  - Cabin size
  - Number of people in the cabin transported
  - Etc.
- Further columns were created based on the passenger’s surrounding cabins
  - E.g. Side neighbor cabin’s size
  - The goal was to create proximity-based columns to aid the machine learning models in finding spatial clusters of transported passengers.

Column Data Type Preparation

- Some models require data to have certain data types
- In order to comply with all models, the data were transformed into numeric variables
- Categorical variables were transformed into binary 1/0 dummy variables
- Integer variables were transformed into continuous variables (R’s numeric type)

Modeling Methodology

Modeling Methodology

- Three base models were tuned using 10-fold cross validation on the training data
  - LASSO logistic regression
  - Random forest
  - XGBoost
- Stacking models were created by taking the out-of-fold predictions of those same three models, and using them as input to a logistic regression meta model
  - Meta model with raw probability predictions as its predictor variables
  - Meta model with Boolean predictions as its predictor variables

Base Models

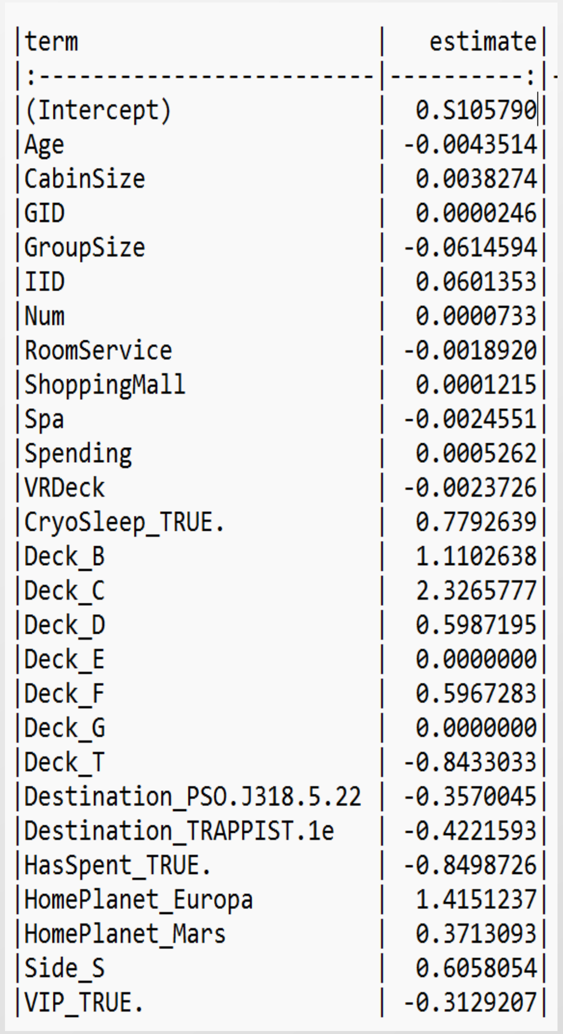
- R’s glmnet, randomForest, and XGBoost libraries were used for LASSO logistic regression, random forest, and gradient boosting models, respectively
- Each model had 200 sets of hyperparameters generated using a latin hypercube space-filling method
- The 600 total models underwent 10-fold cross-validation to find the best hyperparameters
    - LASSO hyperparameters – L1 penalty
    - Random forest hyperparameters – Number of trees, number of sampled predictors, and minimum node size
    - XGBoost hyperparameters – Number of trees, number of sampled predictors, minimum node size, tree depth, learning rate, minimum loss reduction, and the sample size proportion
- The most accurate model from each class was chosen, refit on the whole training data, and used to predict the test data.

Meta Models

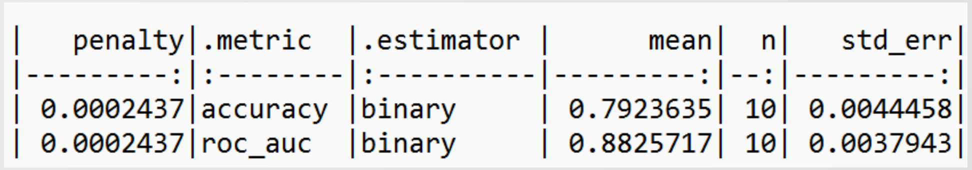
- Out-of-sample predictions were gathered from the results of the best base models’ 10-fold cross-validation processes
- Those results were used to create six new predictors
  - Two for each of the selected base models – probability and Boolean predictions
- A logistic regression model was fit on the three probability predictions, with Transported as the response. Another was fit on the Boolean predictions
- The base models’ predictions of on the test set were used as input to produce the meta models’ final predictions for the test set.

Results

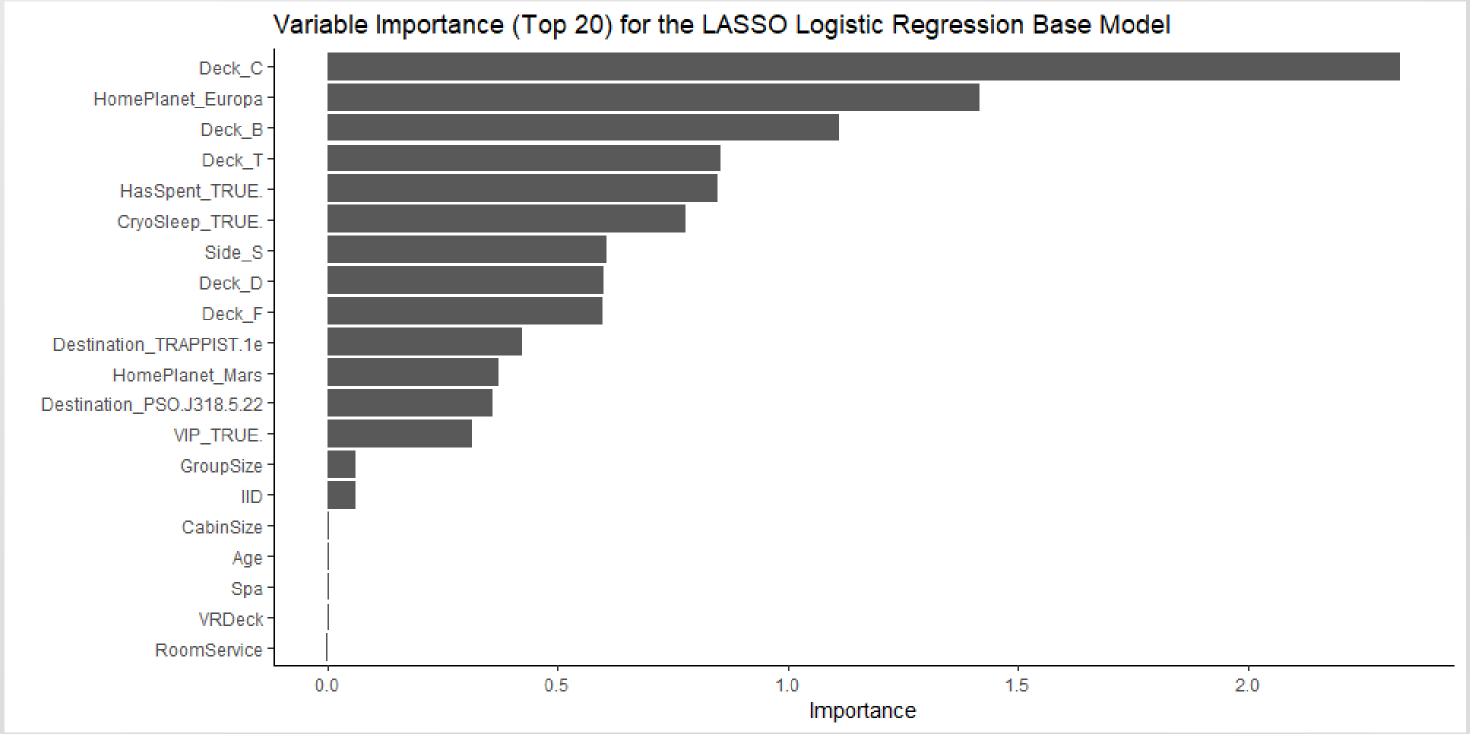
LASSO Logistic Regression



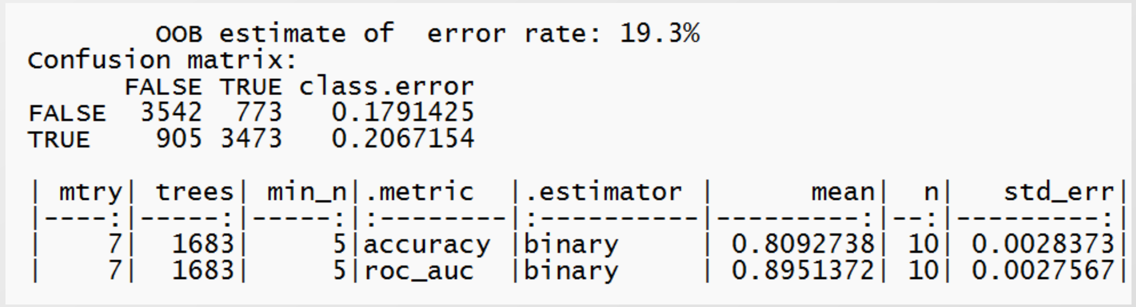
```
|term                       |  estimate|
|:--------------------------|---------:|
|(Intercept)                | 0.S105790|
|Age                        |-0.0043514|
|CabinSize                  | 0.0038274|
|GID                        | 0.0000246|
|GroupSize                  |-0.0614594|
|IID                        | 0.0601353|
|Num                        | 0.0000733|
|RoomService                |-0.0018920|
|ShoppingMall               | 0.0001215|
|Spa                        |-0.0024551|
|Spending                   | 0.0005262|
|VRDeck                     |-0.0023726|
|CryoSleep_TRUE.            | 0.7792639|
|Deck_B                     | 1.1102638|
|Deck_C                     | 2.3265777|
|Deck_D                     | 0.5987195|
|Deck_E                     | 0.0000000|
|Deck_F                     | 0.5967283|
|Deck_G                     | 0.0000000|
|Deck_T                     |-0.8433033|
|Destination_PSO.J318.5.22  |-0.3570045|
|Destination_TRAPPIST.1e    |-0.4221593|
|HasSpent_TRUE.             |-0.8498726|
|HomePlanet_Europa          | 1.4151237|
|HomePlanet_Mars            | 0.3713093|
|Side_S                     | 0.6058054|
|VIP_TRUE.                  |-0.3129207|
```



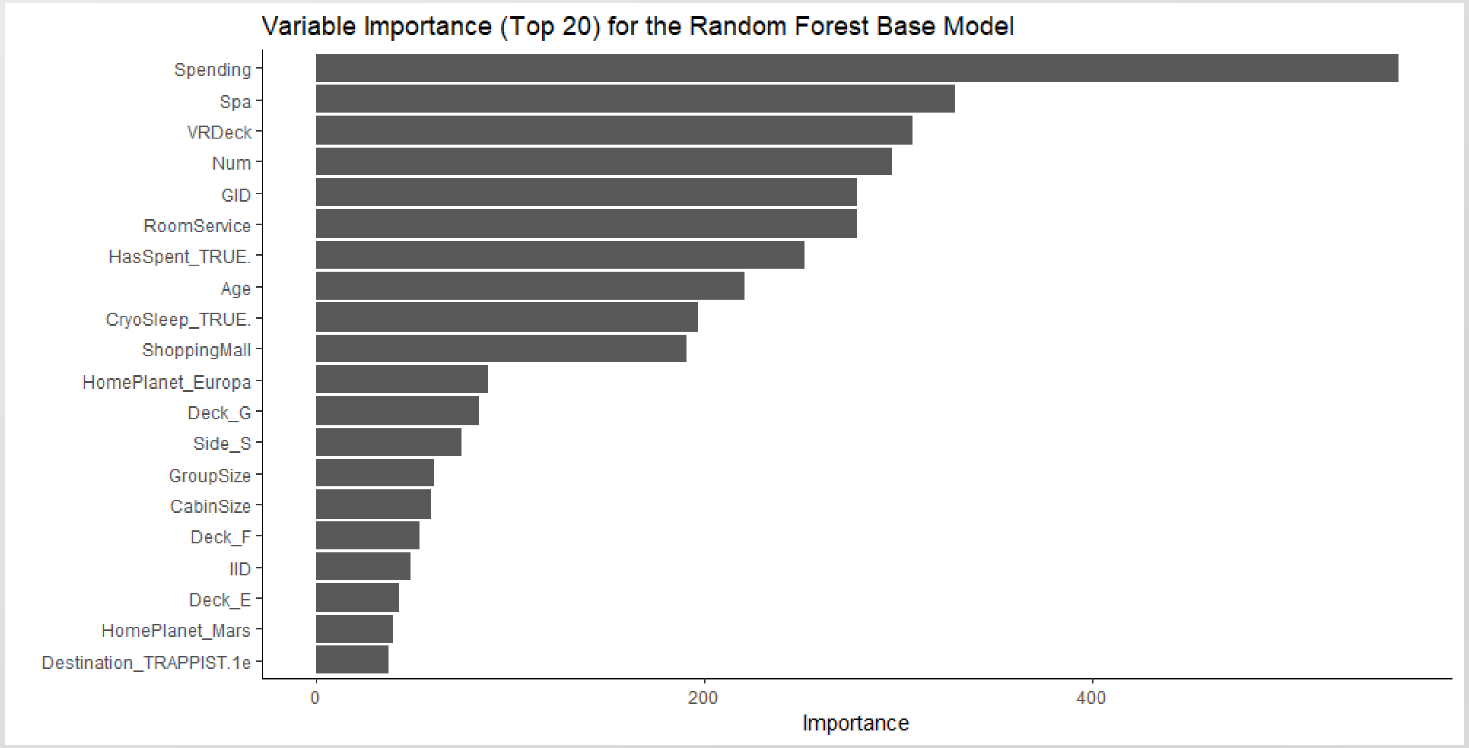
```
|   penalty|.metric  |.estimator |      mean|  n|   std_err|
|---------:|:--------|:----------|---------:|--:|---------:|
| 0.0002437|accuracy |binary     | 0.7923635| 10| 0.0044458|
| 0.0002437|roc_auc  |binary     | 0.8825717| 10| 0.0037943|
```



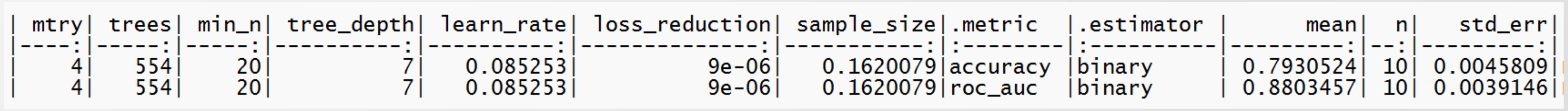
Random Forest



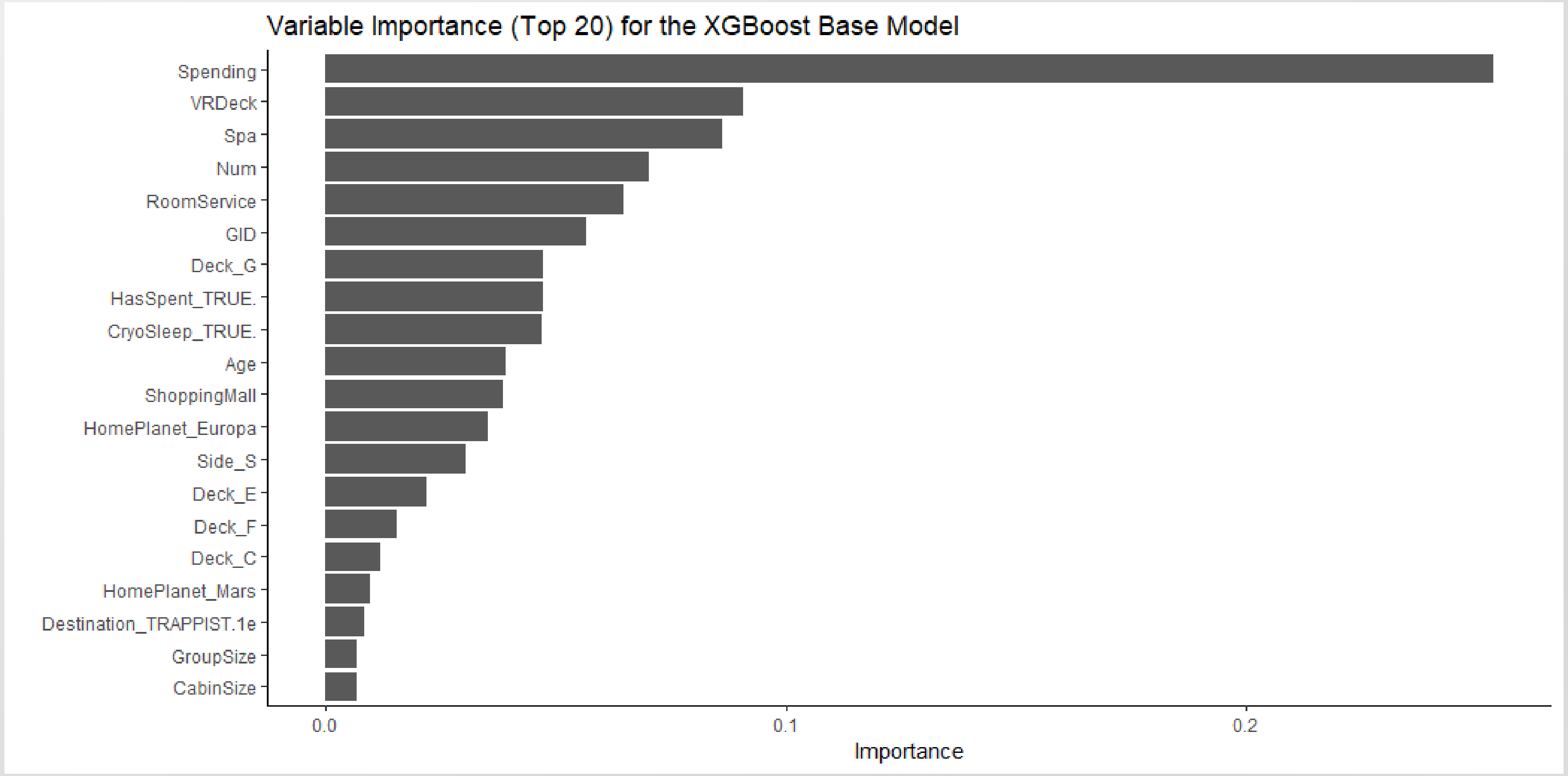
```
        OOB estimate of  error rate: 19.3%
Confusion matrix:
      FALSE TRUE class.error
FALSE  3542  773   0.1791425
TRUE    905 3473   0.2067154

| mtry| trees| min_n|.metric  |.estimator |      mean|  n|   std_err|
|----:|-----:|-----:|:--------|:----------|---------:|--:|---------:|
|    7|  1683|     5|accuracy |binary     | 0.8092738| 10| 0.0028373|
|    7|  1683|     5|roc_auc  |binary     | 0.8951372| 10| 0.0027567|
```



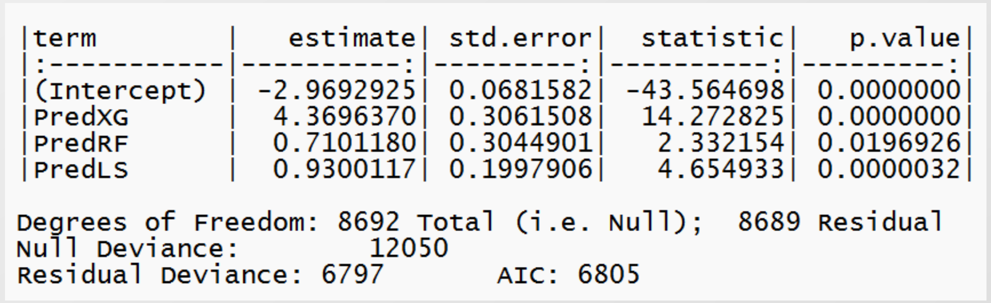
XGBoost



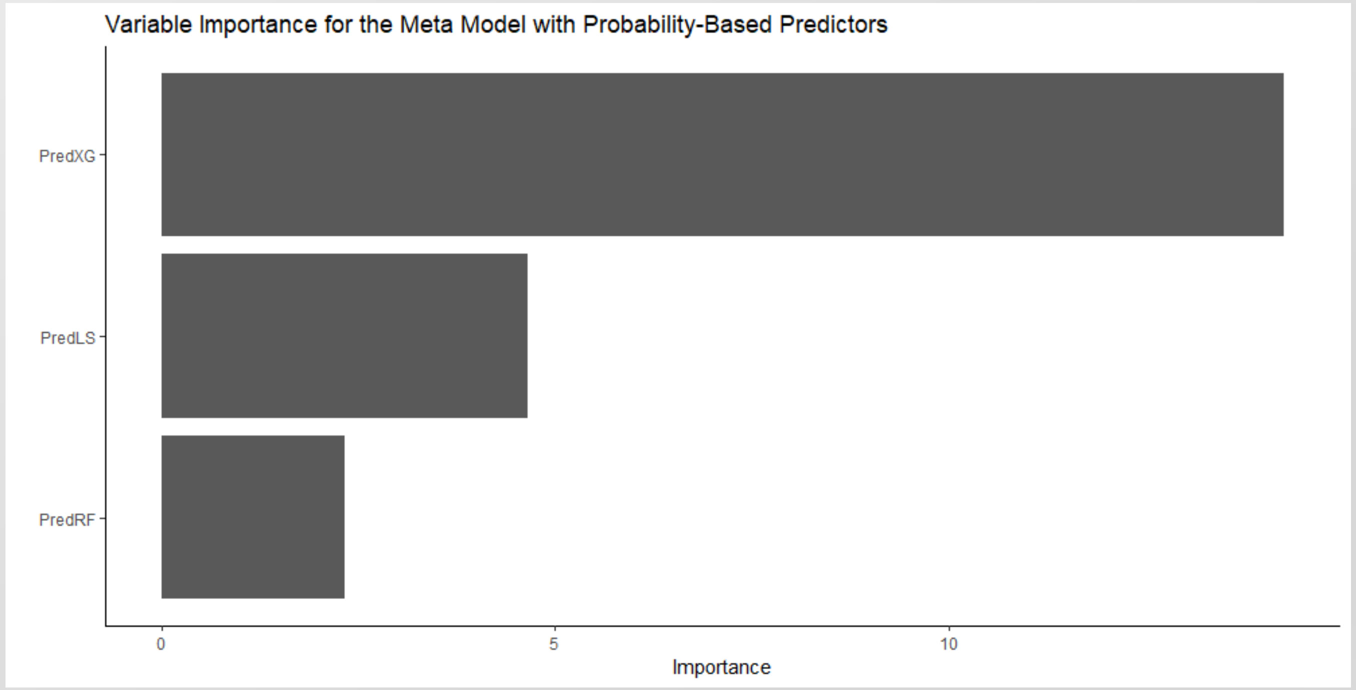
```
| mtry| trees| min_n| tree_depth| learn_rate| loss_reduction| sample_size|.metric  |.estimator |      mean|  n|   std_err|
|----:|-----:|-----:|----------:|----------:|--------------:|-----------:|:--------|:----------|---------:|--:|---------:|
|    4|   554|    20|          7|   0.085253|          9e-06|   0.1620079|accuracy |binary     | 0.7930524| 10| 0.0045809|
|    4|   554|    20|          7|   0.085253|          9e-06|   0.1620079|roc_auc  |binary     | 0.8803457| 10| 0.0039146|
```



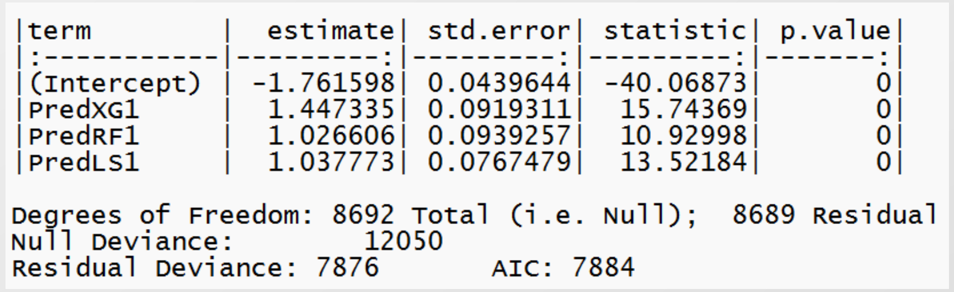
Meta Model with Probability Predictors



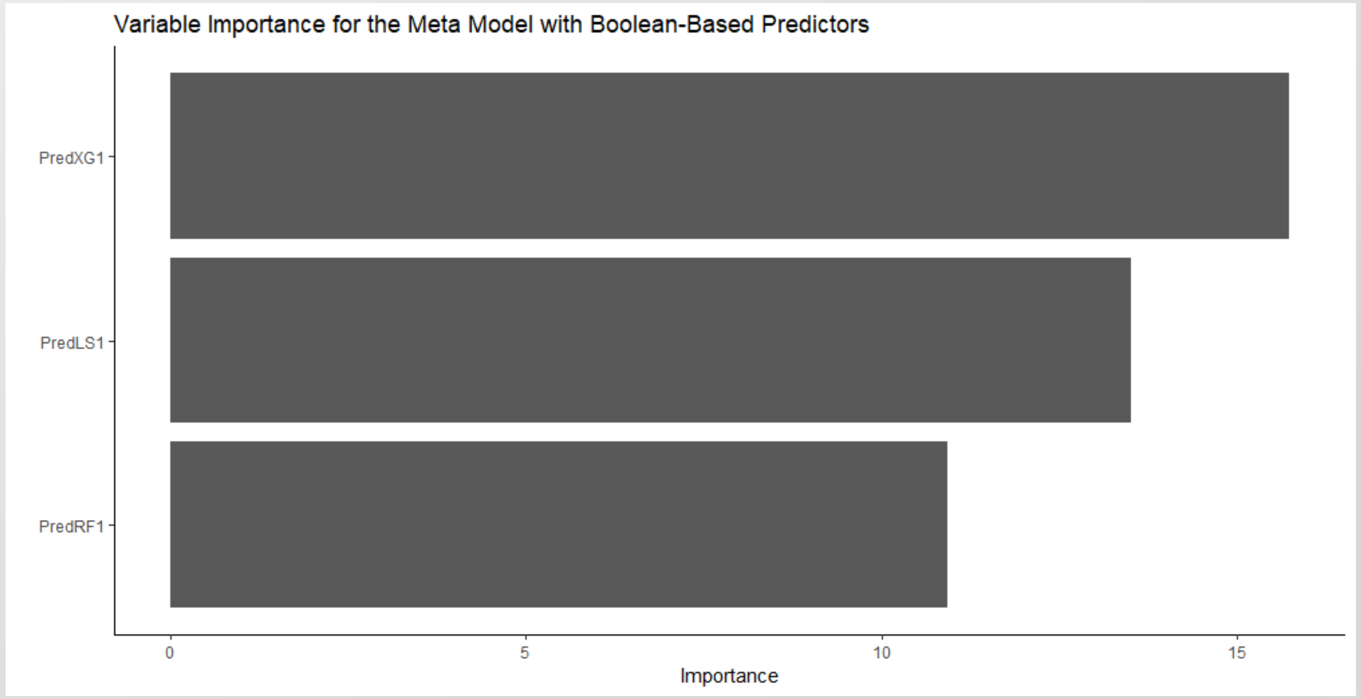
```
|term        |   estimate| std.error|  statistic|   p.value|
|:-----------|----------:|---------:|----------:|---------:|
|(Intercept) | -2.9692925| 0.0681582| -43.564698| 0.0000000|
|PredXG      |  4.3696370| 0.3061508|  14.272825| 0.0000000|
|PredRF      |  0.7101180| 0.3044901|   2.332154| 0.0196926|
|PredLS      |  0.9300117| 0.1997906|   4.654933| 0.0000032|

Degrees of Freedom: 8692 Total (i.e. Null);  8689 Residual
Null Deviance:       12050
Residual Deviance: 6797      AIC: 6805
```



Meta Model with Boolean Predictors



```
|term        |  estimate| std.error| statistic| p.value|
|:-----------|---------:|---------:|---------:|-------:|
|(Intercept) | -1.761598| 0.0439644| -40.06873|       0|
|PredXG1     |  1.447335| 0.0919311|  15.74369|       0|
|PredRF1     |  1.026606| 0.0939257|  10.92998|       0|
|PredLS1     |  1.037773| 0.0767479|  13.52184|       0|

Degrees of Freedom: 8692 Total (i.e. Null);  8689 Residual
Null Deviance:       12050
Residual Deviance: 7876       AIC: 7884
```



Test Set Classification Accuracy

LASSO	Random Forest	XGBoost	Meta (Prob)	Meta (Bool)
79.73%	79.50%	80.20%	80.43%	80.22%

- The Meta (Prob) performance achieved position 398 out of 2268 on the competition’s leaderboard

Discussion

Model Performance

- XGBoost was the best performing base model, which is in line with empirical results shown in other competitions and studies.
- LASSO performed well and valued predictors differently than the other two base models. The different “perspective” that LASSO offered may have contributed to the effectiveness of the meta models.
- The probability-based meta model performed better than the Boolean-based one. This makes sense since converted a numeric probability to a Boolean value results in significant data loss.

Future Extensions

- Add some or all of the original predictors to the set of predictors for the meta models
- Build different stacking models by varying the training and cross-validation approaches
- Add additional base models
  - BART
  - NNs
  - XGBoost models with different hyperparameters
- Use sequential hyperparameter search instead of a fixed starting set of hyperparameters to test

End