REPORT

CS\_559\_GROUP\_6\_FINAL\_PROJECT

By: Mandava Rakeshchandra (20012192), Divya Sagar Reddy, Shasidhar Naidu

INTRODUCTION:

The final class project was to train 3 parametric and 3 non parametric algorithm and report the results. The project is distributed among the three teammates and each team performed 1 parametric and 1 non-parametric algorithm. We also trained a stacking classifier in the end and choose a meta classifier. We used different preprocessing techniques for better performance of the models.

Dataset Description:

There are three different dataset files for the project namely train\_.csv, test\_.csv and validation\_.csv to train, test and validate the models respectively. train\_.csv contains about two-thirds (~8700) of the passengers, to be used as training data while test\_.csv contains records for the remaining one-third (~4300) of the passengers, to be used as test data and validation\_.csv contains (~682) records for the validation.  The three files contain the following attributes:  
PassengerId, HomePlanet, CryoSleep, Cabin, Destination, Age, VIP, RoomService, FoodCourt, ShoppingMall, Spa, VRDeck, Name and Transported.

Project Work flow:

1. We first decided on which algorithms we will be using for the project and started working together.
2. EDA was done by
3. Pre-Processing the data was done by Rakeshchandra using different techniques.

* The Pre-Processing included the following steps
* Split 'PassengerId & ‘Cabin' column in the train & test datasets using lambda function
* Compute 'FamilySize' in the train, test, and validation datasets based on GroupNumber Column
* Create a FamilyType based on the no.of people. Single- if 1-person, Small Family- if >=4, and Large if greater than 4 people
* #Dropping unnecessary columns from all the datasets. The columns ‘Name’, ‘PassengerID’, ‘Cabin’
* #Filling the missing values using mean of the column. SimpleImuter is used to replace the missing values with the mean of the respective column.
* Replacing the missing categorical values for the datasets. The missing values are replaced with the most common value in the columns.
* Converting the Dataframes into a numerical value for the datasets using pd.to\_numeric function.
* label encoding for categorical columns and converting Boolean to integer type for all the datasets
* Splitting the dataset into testing and training datasets. 20% data for Testing and 80% for Training.

1. Modelling: Each person in the team performed 1 parametric and 1 non-parametric models. The following are the models we implemented:

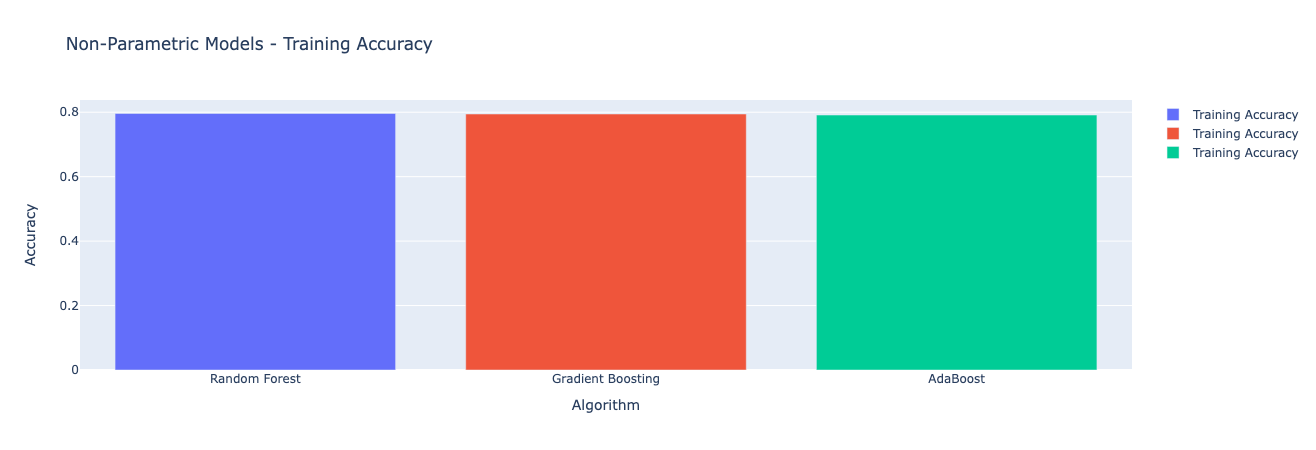
|  |  |
| --- | --- |
| MODEL | PERSON |
| Logistic Regression | Divya Sagar Reddy |
| Simple Neural Network | Mandava Rakeshchandra |
| Linear Discriminant Analysis | Shashidhar Naidu |
| Random Forest | Shashidhar Naidu |
| Gradient Boosting | Mandava Rakeshchandra |
| AdaBoost | Divya Sagar Reddy |

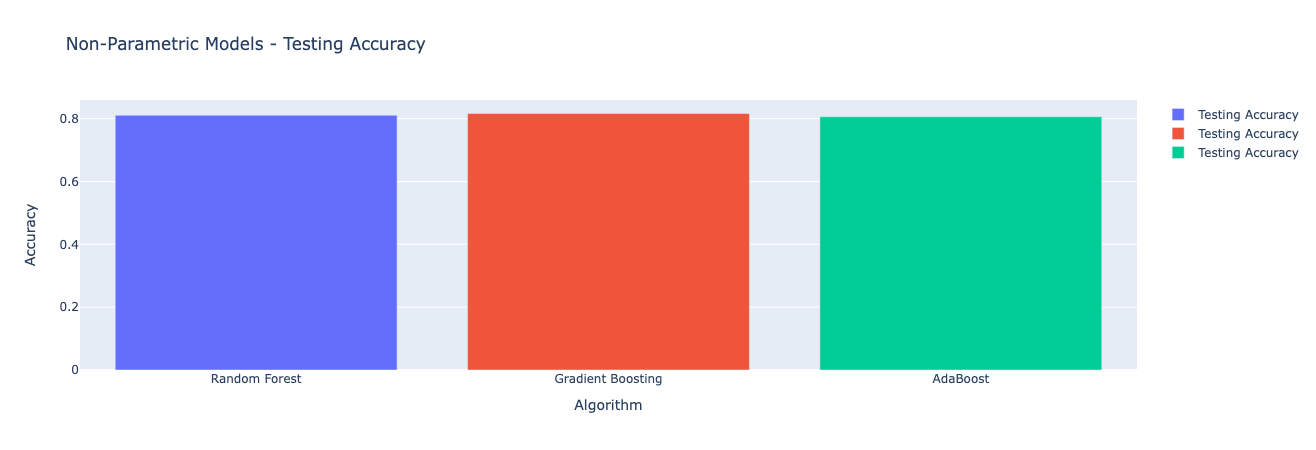
For all the models, we performed Hyperparameters tuning using GridSearchCV.

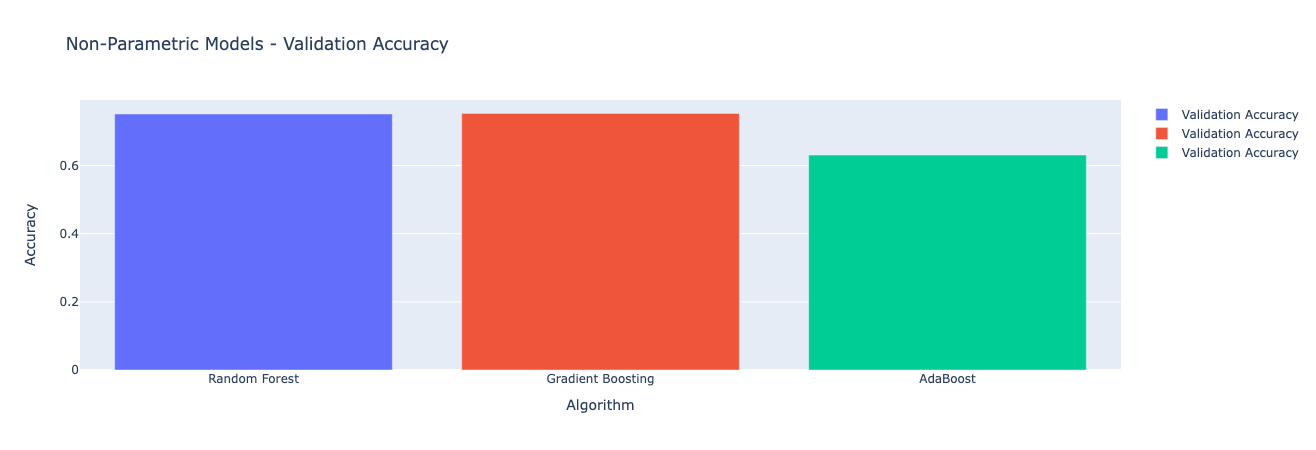
We implemented all the models in a similar way to ensure the readability of the code better. We defined a Parametric Grid for each model and performed grid search using the defined parameters. The grid search model is fitted on the training data and predicted the labels for the testing data. We also found the best hyper\_parameters of all the models and reported the accuracy and generated a bar chart which compares Accuracy of Testing & Training data of all parametric and non-parametric models. The LDA will not have any hyperparameters so the best\_params will be left empty.

Though all the models gave back similar accuracies, Gradient Boosting performed the best with scoring 81.5%.

The following are the Plots generated for Non-Parametric Models:

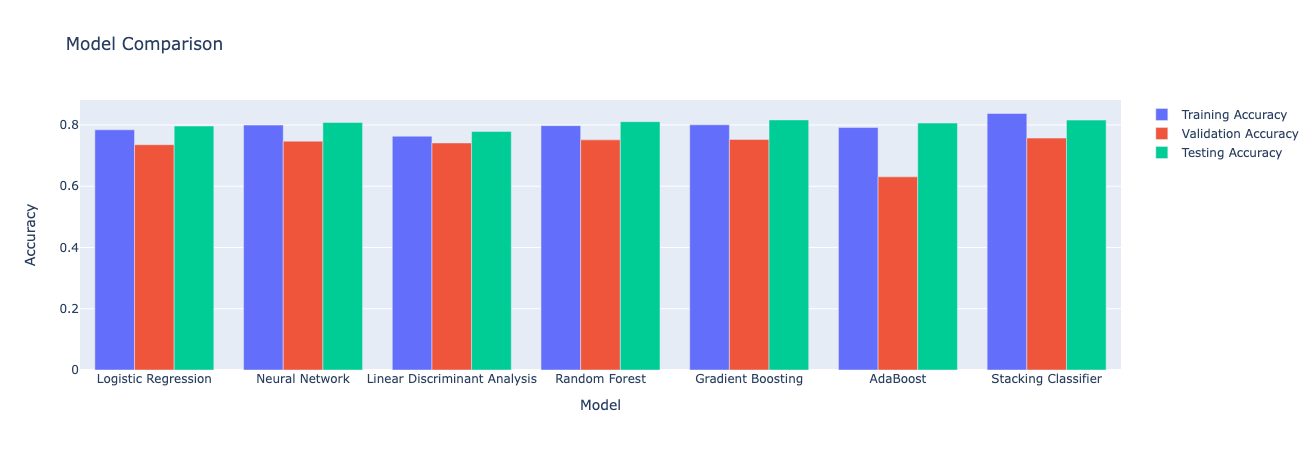






We then created a tuple list “Estimators” which will contain the Algorithms implemented and the best estimators of all the models. A stacking model is then created which will take the list of estimators as input and a final estimator. It is then trained on training dataset using fit method and the predictions on test data is done with the help predict. The accuracy of stacking model ( is calculated and represented along with all other Algorithms in a Bar Chart for comparison.

The following image is the comparison of all 7 models:



For the Kaggle Competition we selected the best performing model to generate the target variable on the testing dataset. We got an accuracy of 80.5% and rank of 385 when we submitted the file.

Conclusion:

In conclusion, we evaluated six models to predict whether a passenger was transported to another dimension or not. Among the individual models, logistic regression performed the best, achieving an accuracy of 80% on the test sets. To further improve the predictions, we employed a stacking ensemble model, which combined the predictions of multiple models. The stacking model also achieved an accuracy of 80% on the test set, which was nearly equal to that of logistic regression, Gradient Boosting.