EIN6934- Statistical foundations of data analysis

SPACESHIP TITANIC

Predict which passengers are transported to an alternate dimension

Final Project Report – Fall 2022

TEAM 2

Chandni Indiraraj

Aravind Menon

Lekha Anand

Course Instructor

Dr. Walter Silva

TABLE OF CONTENTS

1. ABSTRACT
2. INTRODUCTION
3. MOTIVATION
4. DATA DESCRIPTION
5. METHODOLOGY
    - DATA CLEANING
    - FEATURE ENGINEERING
    - LOGISTIC REGRESSION
    - SUPPORT VECTOR CLUSTERING
    - XGBOOST
    - GRADIENT BOOSTING
    - K-NEAREST NEIGHBORS
    - RANDOM FOREST
    - DECISION TREE
6. CONCLUSION
7. REFERENCES

ABSTRACT

A set of data is collected to predict the passengers that are transported to the alternate dimension when a collision with a space-time anomaly. The data is recovered from the computer in the spaceship which can help us use them for retrieving lost passengers. This article helps in predicting the passengers transported through this anomaly and can be used to save them.

Using the data provided in the “Spaceship- Titanic” competition of Kaggle data competition, this article provides information about the overall data required for the process, and the relationship between various variables, and visualizes them for better understanding. Various techniques such as Logistic Regression, Decision trees, Random forest, and KNN model, were built to predict the transported variable. By comparing the predictions, we find which of the models gives us the best output and better accuracy. The training data set contains 8700 observations and the testing data set has 4300 of passengers where our task is to predict the transported value of passengers of this set.

INTRODUCTION

The Kaggle competition gives the opportunity to take part in an online competition named “Spaceship – Titanic “ which predicts the passengers that are transported to an alternate dimension. A whole data set of 13,000 passengers are on board the spaceship that travels in search of habitable exoplanets. In this voyage, they were hit by a space-time anomaly by which a set of passengers are transported to an alternate dimension. This article uses various techniques in both R studio and Python to build models to predict the transported passengers in the testing set of the data.

The process starts with data exploration which involves its cleaning, preprocessing, and visualization, and later gets to a stage of model building. The models are trained and fitted and finally, the total process ends with predictions.

MOTIVATION FOR PROJECT

Data analytics and its application in today's world have major importance. It is applied in many different fields. By the end of this project, we can predict the number of passengers that are transported, and it can be vital data for the rescue operation. This method can be applied on different occasions. Prediction and analysis are two important aspects of data analytics that are applied here.

DATA SET DESCRIPTION

The data set obtained for this project is taken from the Kaggle competition. The "Spaceship Titanic" host competition provides us with train, test, and submission data. The train data has two-thirds of the whole data while the test data has the rest. The train and test have the same variables except that the test data set does not have the transported variable which is to be predicted by us.

The data description from Kaggle competition as per the project :

Train.csv - Personal records for about two-thirds (~8700) of the passengers, to be used as training data.

- PassengerId - A unique Id for each passenger. Each Id takes the form gggg_pp where gggg indicates a group the passenger is travelling with and pp is their number within the group. People in a group are often family members, but not always.
- HomePlanet - The planet the passenger departed from, typically their planet of permanent residence.
- CryoSleep - Indicates whether the passenger elected to be put into suspended animation for the duration of the voyage. Passengers in CryoSleep are confined to their cabins.
- Cabin - The cabin number where the passenger is staying. Takes the form deck/num/side, where side can be either P for Port or S for Starboard.
- Destination - The planet the passenger will be debarking to.
- Age - The age of the passenger.
- VIP - Whether the passenger has paid for special VIP service during the voyage.
- RoomService, FoodCourt, ShoppingMall, Spa, VRDeck - Amount the passenger has billed at each of the Spaceship Titanic's many luxury amenities.
- Name - The first and last names of the passenger.
- Transported - Whether the passenger was transported to another dimension. This is the target, the column you are trying to predict.

- test.csv - Personal records for the remaining one-third (~4300) of the passengers, to be used as test data. Your task is to predict the value of Transported for the passengers in this set.
- sample_submission.csv - A submission file in the correct format.
    - PassengerId - Id for each passenger in the test set.
    - Transported - The target. For each passenger, predict either True or False.

METHODOLOGY

The problem statement of the dataset is to predict whether the passenger was transported or not based on the other features available in the dataset. Training dataset consists of 14columns and 8693 passengers in total where all the columns are considered as features. Test data set replicates same features as in training data set except transported. Here's transported is the target features which has to be predicted through our models. In our training dataset it is observed that we have both categorical and continuous data or features.

Categorical features are those columns which has set off date of values which is repeating over the complete dataset. In Titanic dataset categorical columns are HomePlanet, CryoSleep, Cabin, Destination, VIP and the target column Transported. All the remaining features like passengerId come on RoomService, Age, FoodCourt, shoppingMall, Spa, VRDeck are all continuous features which are non-repetitive as well as numeric. Continuous features may or may not have identical values throughout the dataset. The dataset is subjected to cleaning to replace all the missing values followed by pre-processing techniques like scaling and transformations which is finally run through the model to predict for send passenger being transported or not.

DATA CLEANING

We have witnessed missing values and NA values in both training and testing dataset. It is essential to clean the dataset before running it through neural network models to predict for the target variable upon studying the behaviors of existing features in the dataset. The column Name and passenger ID are identical in every pattern and to have a fair prediction irrespective of person in place, we have dropped Name and Passenger ID as features in our analytical model.

The training dataset comprises of missing values in the following ratio:

FEATURES	NULL PERCENTAGE
AGE	2.05
ROOMSERVICE	2.08
DESTINATION	2.09
FOODCOURT	2.1
SPA	2.1
VRDECK	2.16
CABIN	2.28
NAME	2.30
HOMEPLANET	2.33
VIP	2.35
SHOPPINGMALL	2.39
CRYOSLEEP	2.42

The testing dataset comprises of missing values in the following ratio:

FEATURES	NULL PERCENTAGE
AGE	2.17

ROOMSERVICE	1.91
DESTINATION	2.51
FOODCOURT	2.47
SPA	2.36
VRDECK	1.87
CABIN	2.33
NAME	2.19
HOMEPLANET	2.02
VIP	2.17
SHOPPINGMALL	2.29
CRYOSLEEP	2.17

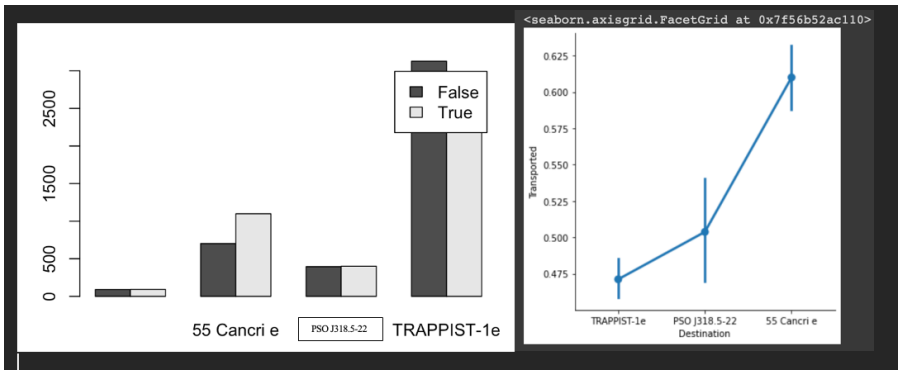
The training dataset and the testing dataset were subjected to cleaning process separately. We have used interpolate function available in python functions to replace all the missing values with its mean. The parameter method was set to linear in the interpolate function to substitute the missing values with most accurate value possible as in the feature.

```
tr=tr.interpolate(method='linear')
te=te.interpolate(method='linear')
```

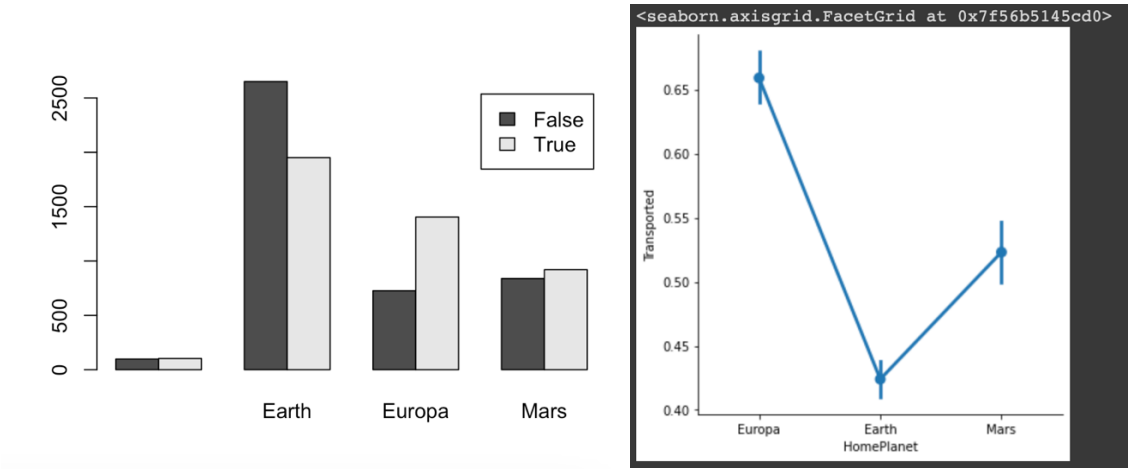
categorical features

There are 5 categorical features in titanic dataset to predict of transported. Categorical columns consist of unique set of data values which is repeated in the complete dataset. Following are the categorical columns:

Destination - Destination has 3 unique values and missing values used diversly in the dataset. The unique values are 'TRAPPIST-1e', 'PSO J318.5-22', '55 Cancri e'. The correlation of this feature with the target feature is visualized using bar plot.

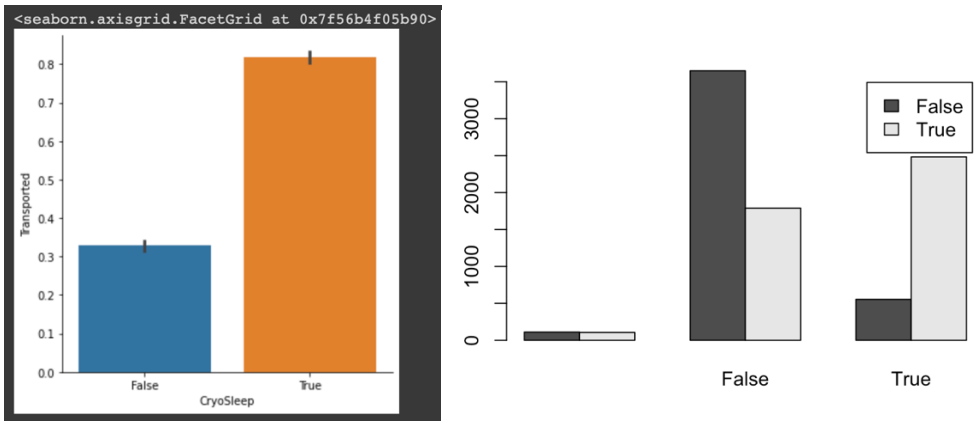


HomePlanet - Home planet has 3 unique values and missing values used diversly in the dataset. The unique values are Europa, Earth, and Mars. The correlation of this feature with the target feature is visualized using bar plot.



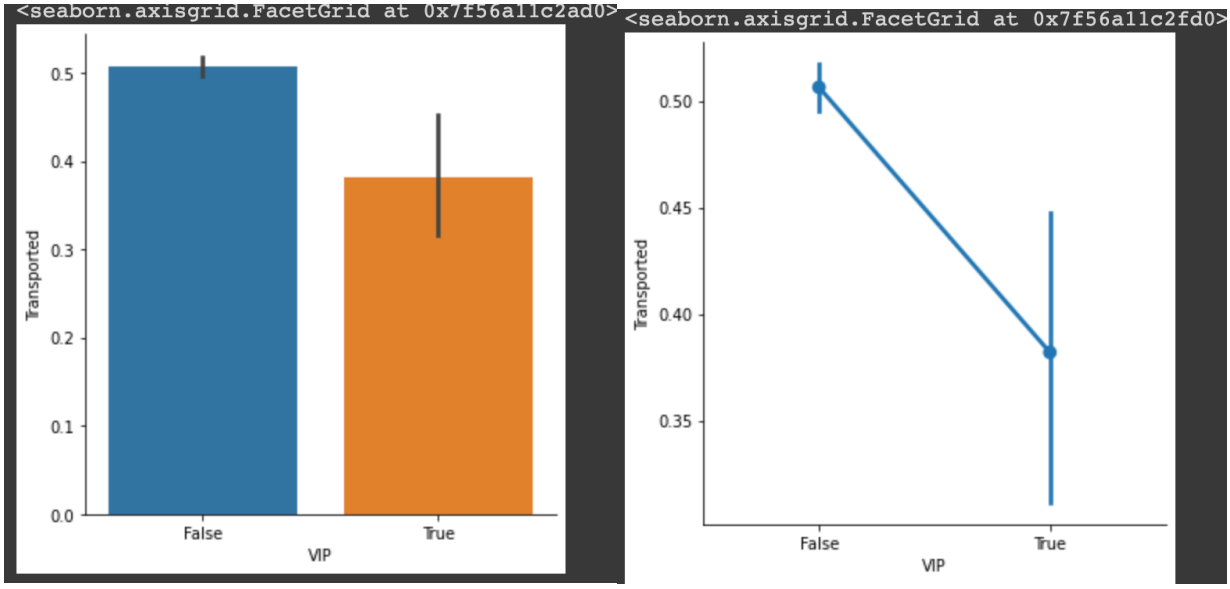
Passengers with HomePlanet as Europa are more likely to be transported than Mars followed by Earth.

CryoSleep - CryoSleep is a Boolean feature with true or false values in it. The correlation of CryoSleep with Transported is as follows. CryoSleep consists of null values which are linearly interpolated based on other features.



Passengers in CryoSleep are more likely to be transported.

VIP - VIP is a feature to express whether a passenger was in VIP suite or not and it is recorded as a Boolean feature with true or false values in it. The correlation of VIP with Transported is visualized using bar and point plot. VIP consists of null values which are linearly interpolated based on other features.

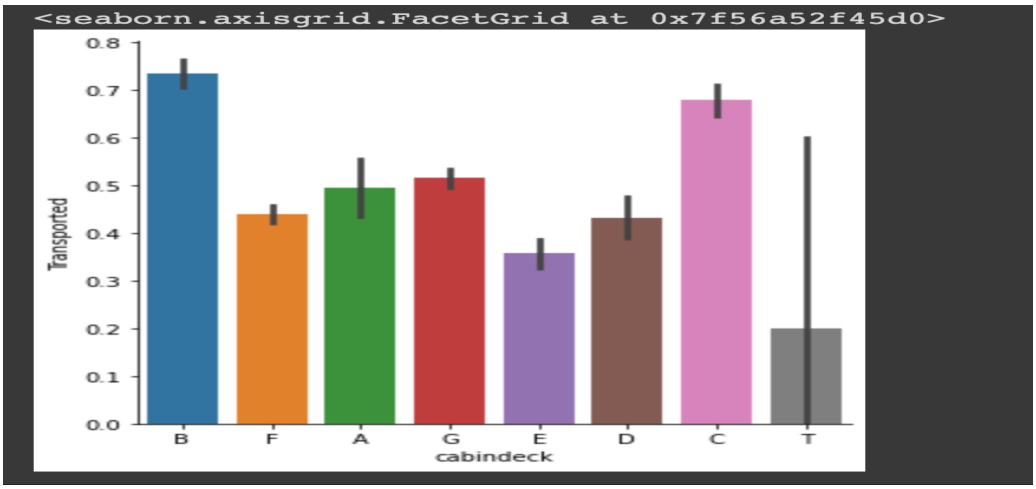


Passengers not in VIP suite are more likely to be transported.

Cabin- Cabin is a combination of Cabin deck, Cabin number and Cabin Side. Cabin numbers are continuous data points, while cabin side and deck are categorical with identical values os P, S for cabin side and groups A, B, C, D, E, F, G, T for cabin decks.

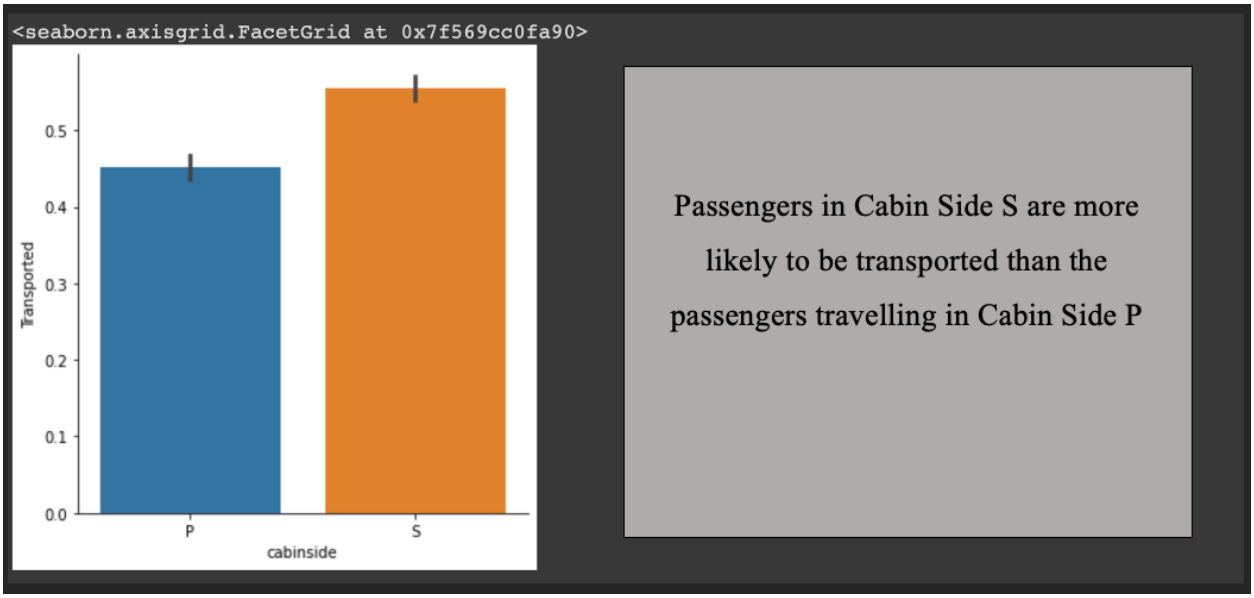
The feature Cabin is split into three different features Cabinnum, Cabindeck, CabinSide using str_split function and used a three different features for analytical predictions.

CabinDeck holds values as A, B, C, D, E, F, G, T and to find its relation with transported, we have visulaized using barplot.



It is clearly seen that passengers in deck B as highly subjected to be transported.

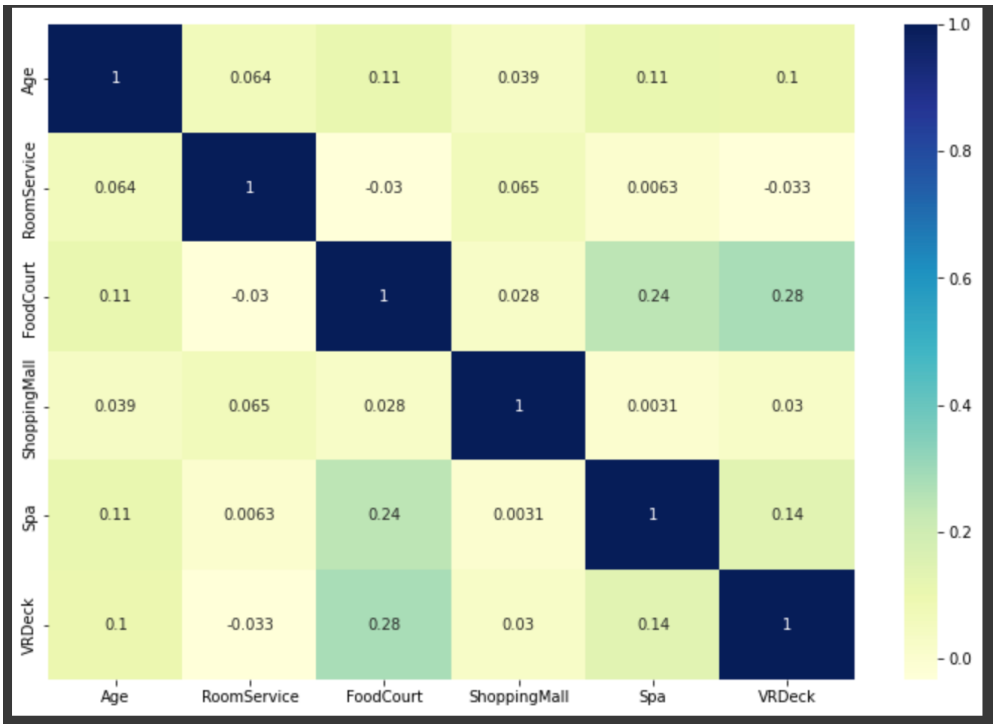
CabinSide holds 2 unique values P and S with some missing values. The correlation between CabinSide and Transported are visualized using bar and point plot.



For continuous features:
The Continuous features in the dataset are all numeric values and the missing values of these features are substituted with most appropriate values from its respective set of records.

The correlation of the training dataset is as follows:

The correlation of the testing dataset using sns heatmap is as follows:



Normalizing dataset:

The non-dependent columns like PassengerId and Name were dropped from both training and test dataset before subjected to analytical model.

#Python codes

```
tr=tr.drop(['Name'],axis=1)
te=te.drop(['Name'],axis=1)
tr =tr.drop(['PassengerId'], axis=1)
te=te.drop(['PassengerId'], axis=1)
```

The feature Cabin was split into three different features comprising deck/group , number and side.

#Python codes

```
te[['cabindeck','cabinnum','cabinside']] = te.Cabin.str.split("/", expand=True)
te=te.drop(['Cabin'], axis=1)
tr[['cabindeck','cabinnum','cabinside']] = tr.Cabin.str.split("/", expand=True)
tr=tr.drop(['Cabin'], axis=1)
```

FEATURE ENGINEERING:

All the categorical columns are factorized into groups or clusters of numeric values.

HomePlanet is categorised as 3 groups/ levels for Earth, Mars, Europa.
Destination is categorized as 3 groups/ levels for 'TRAPPIST-1e', 'PSO J318.5-22', '55 Cancri e'.
Cabin Side is categorised into 2 group levels for side P and S.
Cabin Deck is grouped into 8 levels for 8 unique values A, B, C, D, E, F, G, T.

#Python code:

```
tr['HomePlanet']=pd.factorize(tr.HomePlanet)[0]
tr['cabindeck']=pd.factorize(tr.cabindeck)[0]
tr['cabinside']=pd.factorize(tr.cabinside)[0]
tr['Destination']=pd.factorize(tr.Destination)[0] #similarly for test data as well.
```

CryoSleep and VIP are converted from Boolean to integers with values as 1 and 0 respectively.

```
tr[['CryoSleep', 'VIP','Transported']] = (tr[['CryoSleep', 'VIP','Transported']] == True).astype(int)
te[['CryoSleep', 'VIP']] = (te[['CryoSleep', 'VIP']] == True).astype(int)
```

Scaling the features for the model.

trainX and trainY are the computational and result column for training the train dataset te and similarly testX is the computational dataset for test data.

```
trainX = tr.drop("Transported”, axis=1)
trainY = tr["Transported"]
testX = te
```

Training the model

We have used multiple analytical models to train the dataset and predict for the result column. Starting with logistic regression, we continued to run the model with standard vector clustering SVC, Boosting techniques like Gradient Boosting

and XGBoost, K-Nearest Neighbour KNN, Random Forest and Decision Tree. we built the model and fit the model with our feature engineered dataset. Which is observed for its working and efficiency before using the model for test dataset.

LOGISTIC REGRESSION:

Logistic Regression understands how the target column or feature is related to underlying features in the dataset. This model estimates the output variable as a probabilistic function after studying pattern of the dataset. The first target was to determine the non-essential features and remove from the dataset followed by pre-processing techniques to clean the dataset. This model is then subjected to logistic transformations and predicted for transported feature. The model is run using the training dataset trainX and the predicted value is compared to trainY to determine the training accuracy which is 79%. The same model was run for the test data set to predict for the target feature transported in test dataset. The result is extracted along with the passenger Id and reported to Kaggle competition. Kaggle score for Logistic regression is 76.1%.

In Python:

```
from sklearn.linear_model import LogisticRegression

modelglm = LogisticRegression()
modelglm.fit(trainX, trainY)
predg = modelglm.predict(trainX)
predgt = modelknn.predict(testX)
accg = accuracy_score(trainY, predg) #79
```

In R:

```
model2 = glm(Transported ~ HomePlanet + CryoSleep + Destination + Age + VIP
             + RoomService + FoodCourt + ShoppingMall + Spa + VRDeck, data = train,
             family = binomial)
summary(model2)
```

SUPPORT VECTOR CLUSTERING (SVC):

The next approach we chose was to use a supervised learning method. SVC is used to explore and analyse the data for classification or regression problems. All the datapoints are categorized as levels of data as part of pre-processing stages and are used to build the SVC model to determine the target or result column using non-probabilistic classifier model which is linear and clusters of similar data. The model was run using training dataset and the accuracy was found to be 78.5% and the same model was run with test data to predict for transported. The results were extracted along with passenger ID and reported to Kaggle competition where the score was found to be 67.9%.

```
#SVC
from sklearn.svm import SVC
modelsvc = SVC()
modelsvc.fit(trainX, trainY)
predsvc = modelsvc.predict(testX)
modelsvc.score(trainX, trainY) #accuracy 78.5%
```

```
0.7858046704244794
```

XGBOOST:

The analytical process were providing low accuracy in SVC model. So we thought of using Lasso or Ridge regression as our next approach. Since XGBoost includes Lasso regression as L1 component and Ridge regression as L2 component, it was prominent to use XGBoost. Using these functions, it is easy to prevent overfitting of the model. The model was run using training dataset and the accuracy was found to be 92.5% with parameter tuning. Choosing max_depth as 2, and number of estimators as 500 with learning rate as 0.8, we got the highest training accuracy. The same model was run with test data to predict for transported. The results were extracted along with passenger ID and reported to Kaggle competition where the score was found to be 71.5%.

```
import xgboost
mod = xgboost.XGBClassifier(max_depth=2, n_estimators=500, learning_rate=0.8)
mod.fit(trainX, trainY)
predtr1= mod.predict(trainX)
acc1 = accuracy_score(trainY, predtr)
acc1
```

```
0.92626251006557
```

GRADIENT BOOSTING:

Gradient Boosting is an ensemble model that works for regressions and classification problems. Gradient boosting is an advanced version of decision tree, which plays as an back up when decision tree ends up in overfitting of data. It performs data analysis as layers of tress as in other boosting methods but the loss function is less. The best accuracy was achieved by setting the parameters, max_depth to 2, estimators to 750 and learnings to 0.7 to achieve 92.6% training accuracy. The same model was run with test data to predict for transported. The results were extracted along with passenger ID and reported to Kaggle competition where the score was found to be 76.8%.

```
from sklearn.ensemble import GradientBoostingClassifier
modelxg1 = GradientBoostingClassifier(
    max_depth=2,
    n_estimators=750,
    learning_rate=0.7)
modelxg1.fit(trainX, trainY)
from sklearn.metrics import accuracy_score
predtr= modelxg1.predict(trainX)
acc = accuracy_score(trainY, predtr)
acc #92.6% training accuracy
```

K-Nearest NEIGHBORS KNN:

The dataset is cleaned and pre-processed in forms of clusters of similar data. The factorized dataset will find for the nearest Neighbors and invoke the characters from the near class in contact. We have set the number of Neighbors to 3 and preferred Minkowski Formula over Euclidian by setting p value to 1 and weights

to uniform to product best training accuracy of 86.2%. The P value denotes that the KNN model used Manhattan distance to find nearest objects. The same model was run with test data to predict for transported. The results were extracted along with passenger ID and reported to Kaggle competition where the score was found to be 78.53%.

```
#KNN
from sklearn.neighbors import KNeighborsClassifier
modelknn = KNeighborsClassifier(n_neighbors=3,weights='uniform', p=1)
modelknn.fit(trainX, trainY)
predknn = modelknn.predict(trainX)
accknn = accuracy_score(trainY, predknn)
accknn
#modelknn.score(trainX, trainY)
```

```
0.8619578971586334
```

RANDOM FOREST:

Random forest is an ensemble method used to solve analytical problems which focuses on regressions and classifications. The model uses multiple dimensions of decision trees to pick the most accurate pattern and predict for the result column upon the behaviour followed in the respective computational features. The model was overfitting initially, but with number of estimators as 120, we found this to be the best model with highest training accuracy of 96.5%. The model the output a as the mean of predictions of individual decision trees or nodes given by the estimators. The same model was run with test data to predict for transported. The results were extracted along with passenger ID and reported to Kaggle competition where the score was found to be 78.79%.

```
#rf
from sklearn.ensemble import RandomForestClassifier
modelrf = RandomForestClassifier(n_estimators=120)
modelrf.fit(trainX, trainY)
predrf = modelrf.predict(trainX)
predrft = modelrf.predict(testX)
accrf = accuracy_score(trainY, predrf)
accrf #96.2%
```

DECISION TREE:

Random forest was giving the best training accuracy but the Kaggle score was far away from what we expected. Decision tree is an predictive model that runs on the principle of tree and leaves structure. The tree branches are referred to as the computational features involved for predicting and the predicted features is the leaf node of the tree or the final target column and the whole tree is referred to as model used for classification. Since the categorical columns re factorised into continuous data , Decision tree works as a regression tree to compute whether the passengers were transported or not. The model was subjected to overfitting even after turning the parameters with all possibilities. We have got a training accuracy of 99.2% which is more evident to be overfitting. The model was run with test data and the Kaggle score dropped down to 70.02%.

```
#dt
from sklearn.tree import DecisionTreeClassifier
modeldt = DecisionTreeClassifier(random_state=12)
modeldt.fit(trainX, trainY)
preddt = modeldt.predict(testX)
modeldt.score(trainX, trainY) #accuracy 99.9%
```

CONCLUSION

Various methods were used to during this project. Every time a model was run, we always expected it to be more accurate than the prior approach. This project helped us to learn Machine learning models and also made us more confident with the pre-processing steps including Visualising the data, cleaning the data, facing the outliers, working with the missing values and merging the tables. Once we were done with the pre-processing steps and feature engineering, we implemented different types of Machine Learning models to increase our accuracy in time. We used different methods and all possible approaches to make this model better. This project really helped us to gain more knowledge In exploring and analysing large datasets in terms of analytical model and determine every single column as a potential feature essential for the predictive model. It was a good opportunity to work on multiple models where we could understand how the data set was responding for each data model and how it affected the prediction accuracy respectively. In our analysis random forest get the most accurate model with training accuracy of around 96 percentage but the testing accuracy dropped to 78.8 percentage in our Kaggle submission. It was a visual treat and fun activity to play with the dataset for predicting whether the passengers were transported or not. the top 3 models in our analysis are random forest, KNN and gradient boosting ensemble method.

REFERENCES

1. B, M., A, C., Brenda, V., & A, R. (2005). *Advanced and Multivariate Statistical methods: practical application and interpretation (3rd Edition).* https://usf-flvc.primo.exlibrisgroup.com/permalink/01FALSC_USF/8i1ivu/alma99379625187706599
2. Pathak, Manish. (Nov,2019). *Using XGBoost in Python Tutorial*: https://www.datacamp.com/tutorial/xgboost-in-python
3. Koehrsen, Will. (Dec,2017). *Random Forest in Python*: https://towardsdatascience.com/random-forest-in-python
4. https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
5. https://www.geeksforgeeks.org
6. https://www.kaggle.com/learn/feature-engineering
7. https://labelbox.com/learn/library/complete-guide-data-engines-for-ai
8. https://www.kaggle.com/learn/intermediate-machine-learning
9. https://www.freecodecamp.org
10. https://stackoverflow.com