# Predicting which passengers are transported to an alternate dimension

## 1. Problem Definition

Welcome to the year 2912, where your data science skills are needed to solve a cosmic mystery. We've received a transmission from four lightyears away and things aren't looking good.

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!

***predicting which passengers were transported by the anomaly using records recovered from the spaceship’s damaged computer system***

## 2. Data

The data for the prediction is given by Kaggle it-self. Checkout here:<https://www.kaggle.com/competitions/spaceship-titanic/data>

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

## 3. Evaluation

Submissions are evaluated based on their classification accuracy, the percentage of predicted labels that are correct. <https://www.kaggle.com/competitions/spaceship-titanic/overview>

## 4. Features

* 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.

## Preparing the tools

We're going to use Pandas, Matplotlib and Numpy for data analysis and manipulation.

# Import all the tools we need  
  
# Regular EDA (exploratory data analysis) and plotting libraries  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import LabelEncoder  
  
# we want our plot to appear inside the notebook  
%matplotlib inline   
  
# Import models for Scikit-Learn  
from sklearn.linear\_model import LogisticRegression  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.ensemble import RandomForestClassifier  
  
# Model Evaluation  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score  
from sklearn.model\_selection import RandomizedSearchCV, GridSearchCV  
from sklearn.metrics import confusion\_matrix, classification\_report  
from sklearn.metrics import precision\_score, recall\_score, f1\_score  
from sklearn.metrics import RocCurveDisplay

## Load Data

df\_train = pd.read\_csv("data/train.csv")  
df\_test = pd.read\_csv("data/test.csv")

df\_train.shape

(8693, 14)

## Data exploration (exploratory data analysis or EDA)

The goal here is to find out more about data and become a subject matter expert on the dataset you're working with.

df\_train["Transported"].value\_counts()

Transported  
True 4378  
False 4315  
Name: count, dtype: int64

df\_train.isna().sum()

PassengerId 0  
HomePlanet 201  
CryoSleep 217  
Cabin 199  
Destination 182  
Age 179  
VIP 203  
RoomService 181  
FoodCourt 183  
ShoppingMall 208  
Spa 183  
VRDeck 188  
Name 200  
Transported 0  
dtype: int64

df\_train.info()

<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

df\_train.head(20)

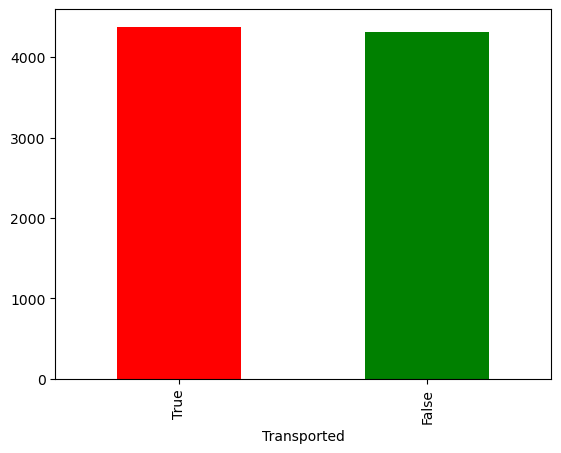
PassengerId HomePlanet CryoSleep Cabin Destination Age VIP \  
0 0001\_01 Europa False B/0/P TRAPPIST-1e 39.0 False   
1 0002\_01 Earth False F/0/S TRAPPIST-1e 24.0 False   
2 0003\_01 Europa False A/0/S TRAPPIST-1e 58.0 True   
3 0003\_02 Europa False A/0/S TRAPPIST-1e 33.0 False   
4 0004\_01 Earth False F/1/S TRAPPIST-1e 16.0 False   
5 0005\_01 Earth False F/0/P PSO J318.5-22 44.0 False   
6 0006\_01 Earth False F/2/S TRAPPIST-1e 26.0 False   
7 0006\_02 Earth True G/0/S TRAPPIST-1e 28.0 False   
8 0007\_01 Earth False F/3/S TRAPPIST-1e 35.0 False   
9 0008\_01 Europa True B/1/P 55 Cancri e 14.0 False   
10 0008\_02 Europa True B/1/P TRAPPIST-1e 34.0 False   
11 0008\_03 Europa False B/1/P 55 Cancri e 45.0 False   
12 0009\_01 Mars False F/1/P TRAPPIST-1e 32.0 False   
13 0010\_01 Earth False G/1/S TRAPPIST-1e 48.0 False   
14 0011\_01 Earth False F/2/P TRAPPIST-1e 28.0 False   
15 0012\_01 Earth False NaN TRAPPIST-1e 31.0 False   
16 0014\_01 Mars False F/3/P 55 Cancri e 27.0 False   
17 0015\_01 Earth False F/4/P 55 Cancri e 24.0 False   
18 0016\_01 Mars True F/5/P TRAPPIST-1e 45.0 False   
19 0017\_01 Earth False G/0/P TRAPPIST-1e 0.0 False   
  
 RoomService FoodCourt ShoppingMall Spa VRDeck Name \  
0 0.0 0.0 0.0 0.0 0.0 Maham Ofracculy   
1 109.0 9.0 25.0 549.0 44.0 Juanna Vines   
2 43.0 3576.0 0.0 6715.0 49.0 Altark Susent   
3 0.0 1283.0 371.0 3329.0 193.0 Solam Susent   
4 303.0 70.0 151.0 565.0 2.0 Willy Santantines   
5 0.0 483.0 0.0 291.0 0.0 Sandie Hinetthews   
6 42.0 1539.0 3.0 0.0 0.0 Billex Jacostaffey   
7 0.0 0.0 0.0 0.0 NaN Candra Jacostaffey   
8 0.0 785.0 17.0 216.0 0.0 Andona Beston   
9 0.0 0.0 0.0 0.0 0.0 Erraiam Flatic   
10 0.0 0.0 NaN 0.0 0.0 Altardr Flatic   
11 39.0 7295.0 589.0 110.0 124.0 Wezena Flatic   
12 73.0 0.0 1123.0 0.0 113.0 Berers Barne   
13 719.0 1.0 65.0 0.0 24.0 Reney Baketton   
14 8.0 974.0 12.0 2.0 7.0 Elle Bertsontry   
15 32.0 0.0 876.0 0.0 0.0 Justie Pooles   
16 1286.0 122.0 NaN 0.0 0.0 Flats Eccle   
17 0.0 1.0 0.0 0.0 637.0 Carry Hughriend   
18 0.0 0.0 0.0 0.0 0.0 Alus Upead   
19 0.0 0.0 0.0 0.0 0.0 Lyde Brighttt   
  
 Transported   
0 False   
1 True   
2 False   
3 False   
4 True   
5 True   
6 True   
7 True   
8 True   
9 True   
10 True   
11 True   
12 True   
13 False   
14 True   
15 False   
16 False   
17 False   
18 True   
19 True

df\_train.describe()

Age RoomService FoodCourt ShoppingMall Spa \  
count 8514.000000 8512.000000 8510.000000 8485.000000 8510.000000   
mean 28.827930 224.687617 458.077203 173.729169 311.138778   
std 14.489021 666.717663 1611.489240 604.696458 1136.705535   
min 0.000000 0.000000 0.000000 0.000000 0.000000   
25% 19.000000 0.000000 0.000000 0.000000 0.000000   
50% 27.000000 0.000000 0.000000 0.000000 0.000000   
75% 38.000000 47.000000 76.000000 27.000000 59.000000   
max 79.000000 14327.000000 29813.000000 23492.000000 22408.000000   
  
 VRDeck   
count 8505.000000   
mean 304.854791   
std 1145.717189   
min 0.000000   
25% 0.000000   
50% 0.000000   
75% 46.000000   
max 24133.000000

df\_train["Transported"].value\_counts().plot(kind = "bar", color = ["red", "green"])

<Axes: xlabel='Transported'>



## Transportation rate according to HomePlanet

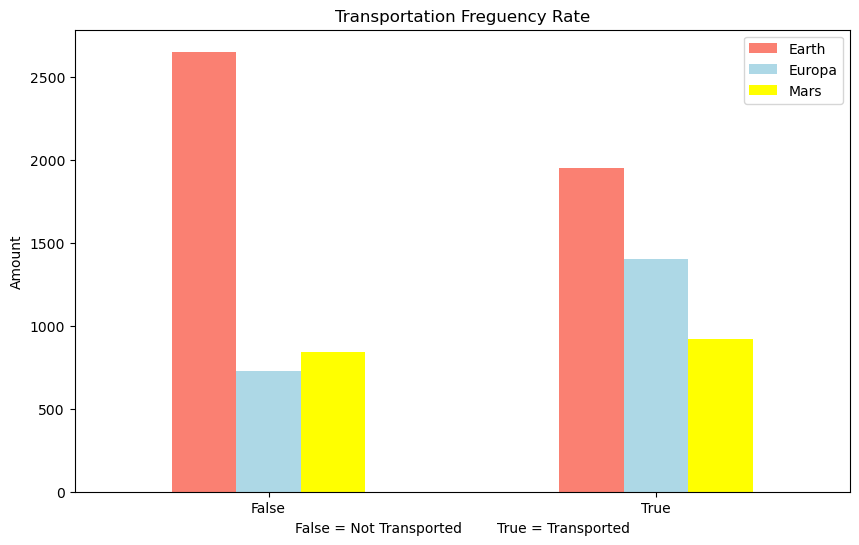
pd.crosstab(df\_train.Transported, df\_train.HomePlanet)

HomePlanet Earth Europa Mars  
Transported   
False 2651 727 839  
True 1951 1404 920

df\_train.columns

Index(['PassengerId', 'HomePlanet', 'CryoSleep', 'Cabin', 'Destination', 'Age',  
 'VIP', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck',  
 'Name', 'Transported'],  
 dtype='object')

pd.crosstab(df\_train.Transported, df\_train.HomePlanet).plot(kind = "bar", figsize = (10, 6), color = ["salmon", "lightblue", "yellow"])  
plt.title("Transportation Freguency Rate")  
plt.xlabel("False = Not Transported True = Transported")  
plt.ylabel("Amount")  
plt.legend(["Earth", "Europa", "Mars"])  
plt.xticks(rotation = 0);



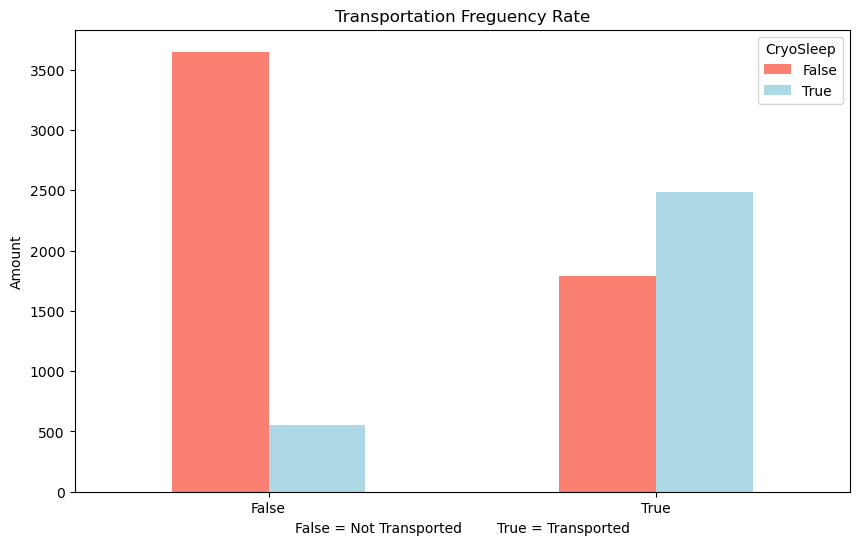
df\_train

PassengerId HomePlanet CryoSleep Cabin Destination Age VIP \  
0 0001\_01 Europa False B/0/P TRAPPIST-1e 39.0 False   
1 0002\_01 Earth False F/0/S TRAPPIST-1e 24.0 False   
2 0003\_01 Europa False A/0/S TRAPPIST-1e 58.0 True   
3 0003\_02 Europa False A/0/S TRAPPIST-1e 33.0 False   
4 0004\_01 Earth False F/1/S TRAPPIST-1e 16.0 False   
... ... ... ... ... ... ... ...   
8688 9276\_01 Europa False A/98/P 55 Cancri e 41.0 True   
8689 9278\_01 Earth True G/1499/S PSO J318.5-22 18.0 False   
8690 9279\_01 Earth False G/1500/S TRAPPIST-1e 26.0 False   
8691 9280\_01 Europa False E/608/S 55 Cancri e 32.0 False   
8692 9280\_02 Europa False E/608/S TRAPPIST-1e 44.0 False   
  
 RoomService FoodCourt ShoppingMall Spa VRDeck Name \  
0 0.0 0.0 0.0 0.0 0.0 Maham Ofracculy   
1 109.0 9.0 25.0 549.0 44.0 Juanna Vines   
2 43.0 3576.0 0.0 6715.0 49.0 Altark Susent   
3 0.0 1283.0 371.0 3329.0 193.0 Solam Susent   
4 303.0 70.0 151.0 565.0 2.0 Willy Santantines   
... ... ... ... ... ... ...   
8688 0.0 6819.0 0.0 1643.0 74.0 Gravior Noxnuther   
8689 0.0 0.0 0.0 0.0 0.0 Kurta Mondalley   
8690 0.0 0.0 1872.0 1.0 0.0 Fayey Connon   
8691 0.0 1049.0 0.0 353.0 3235.0 Celeon Hontichre   
8692 126.0 4688.0 0.0 0.0 12.0 Propsh Hontichre   
  
 Transported   
0 False   
1 True   
2 False   
3 False   
4 True   
... ...   
8688 False   
8689 False   
8690 True   
8691 False   
8692 True   
  
[8693 rows x 14 columns]

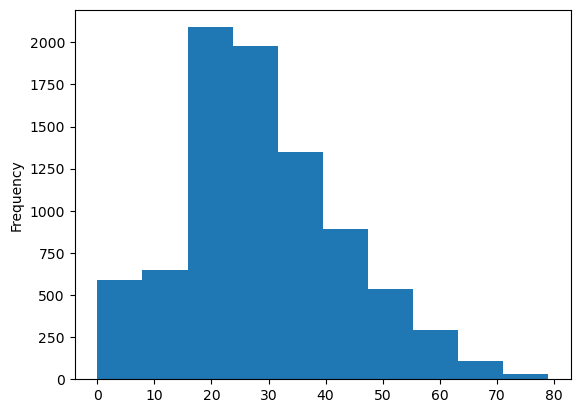
pd.crosstab(df\_train.Transported, df\_train.CryoSleep)

CryoSleep False True   
Transported   
False 3650 554  
True 1789 2483

pd.crosstab(df\_train.Transported, df\_train.CryoSleep).plot(kind = "bar", figsize = (10, 6), color = ["salmon", "lightblue"])  
plt.title("Transportation Freguency Rate")  
plt.xlabel("False = Not Transported True = Transported")  
plt.ylabel("Amount")  
# plt.legend([])  
plt.xticks(rotation = 0);



# Check the distribution of the column with a histogram  
df\_train.Age.plot.hist();



df\_train.Destination.unique()

array(['TRAPPIST-1e', 'PSO J318.5-22', '55 Cancri e', nan], dtype=object)

pd.crosstab(df\_train.Transported, df\_train.Destination)

Destination 55 Cancri e PSO J318.5-22 TRAPPIST-1e  
Transported   
False 702 395 3128  
True 1098 401 2787

df\_train.info()

<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

# Find the columns which contains strings  
for label, content in df\_train.items():  
 if pd.api.types.is\_string\_dtype(content):  
 print(label)

PassengerId

df\_train.describe(include = "all").T

count unique top freq mean std \  
PassengerId 8693 8693 0001\_01 1 NaN NaN   
HomePlanet 8492 3 Earth 4602 NaN NaN   
CryoSleep 8476 2 False 5439 NaN NaN   
Cabin 8494 6560 G/734/S 8 NaN NaN   
Destination 8511 3 TRAPPIST-1e 5915 NaN NaN   
Age 8514.0 NaN NaN NaN 28.82793 14.489021   
VIP 8490 2 False 8291 NaN NaN   
RoomService 8512.0 NaN NaN NaN 224.687617 666.717663   
FoodCourt 8510.0 NaN NaN NaN 458.077203 1611.48924   
ShoppingMall 8485.0 NaN NaN NaN 173.729169 604.696458   
Spa 8510.0 NaN NaN NaN 311.138778 1136.705535   
VRDeck 8505.0 NaN NaN NaN 304.854791 1145.717189   
Name 8493 8473 Gollux Reedall 2 NaN NaN   
Transported 8693 2 True 4378 NaN NaN   
  
 min 25% 50% 75% max   
PassengerId NaN NaN NaN NaN NaN   
HomePlanet NaN NaN NaN NaN NaN   
CryoSleep NaN NaN NaN NaN NaN   
Cabin NaN NaN NaN NaN NaN   
Destination NaN NaN NaN NaN NaN   
Age 0.0 19.0 27.0 38.0 79.0   
VIP NaN NaN NaN NaN NaN   
RoomService 0.0 0.0 0.0 47.0 14327.0   
FoodCourt 0.0 0.0 0.0 76.0 29813.0   
ShoppingMall 0.0 0.0 0.0 27.0 23492.0   
Spa 0.0 0.0 0.0 59.0 22408.0   
VRDeck 0.0 0.0 0.0 46.0 24133.0   
Name NaN NaN NaN NaN NaN   
Transported NaN NaN NaN NaN NaN

# df\_train = df

df\_train

PassengerId HomePlanet CryoSleep Cabin Destination Age VIP \  
0 0001\_01 Europa False B/0/P TRAPPIST-1e 39.0 False   
1 0002\_01 Earth False F/0/S TRAPPIST-1e 24.0 False   
2 0003\_01 Europa False A/0/S TRAPPIST-1e 58.0 True   
3 0003\_02 Europa False A/0/S TRAPPIST-1e 33.0 False   
4 0004\_01 Earth False F/1/S TRAPPIST-1e 16.0 False   
... ... ... ... ... ... ... ...   
8688 9276\_01 Europa False A/98/P 55 Cancri e 41.0 True   
8689 9278\_01 Earth True G/1499/S PSO J318.5-22 18.0 False   
8690 9279\_01 Earth False G/1500/S TRAPPIST-1e 26.0 False   
8691 9280\_01 Europa False E/608/S 55 Cancri e 32.0 False   
8692 9280\_02 Europa False E/608/S TRAPPIST-1e 44.0 False   
  
 RoomService FoodCourt ShoppingMall Spa VRDeck Name \  
0 0.0 0.0 0.0 0.0 0.0 Maham Ofracculy   
1 109.0 9.0 25.0 549.0 44.0 Juanna Vines   
2 43.0 3576.0 0.0 6715.0 49.0 Altark Susent   
3 0.0 1283.0 371.0 3329.0 193.0 Solam Susent   
4 303.0 70.0 151.0 565.0 2.0 Willy Santantines   
... ... ... ... ... ... ...   
8688 0.0 6819.0 0.0 1643.0 74.0 Gravior Noxnuther   
8689 0.0 0.0 0.0 0.0 0.0 Kurta Mondalley   
8690 0.0 0.0 1872.0 1.0 0.0 Fayey Connon   
8691 0.0 1049.0 0.0 353.0 3235.0 Celeon Hontichre   
8692 126.0 4688.0 0.0 0.0 12.0 Propsh Hontichre   
  
 Transported   
0 False   
1 True   
2 False   
3 False   
4 True   
... ...   
8688 False   
8689 False   
8690 True   
8691 False   
8692 True   
  
[8693 rows x 14 columns]

## Feature Engineering and Missing Values

df\_train.isna().sum()

PassengerId 0  
HomePlanet 201  
CryoSleep 217  
Cabin 199  
Destination 182  
Age 179  
VIP 203  
RoomService 181  
FoodCourt 183  
ShoppingMall 208  
Spa 183  
VRDeck 188  
Name 200  
Transported 0  
dtype: int64

df\_test.isna().sum()

PassengerId 0  
HomePlanet 87  
CryoSleep 93  
Cabin 100  
Destination 92  
Age 91  
VIP 93  
RoomService 82  
FoodCourt 106  
ShoppingMall 98  
Spa 101  
VRDeck 80  
Name 94  
dtype: int64

# From EDA i've concluded that PassengerID, Name, Cabin are not affecting much the Transported label  
del df\_train["PassengerId"]  
del df\_train["Name"]  
del df\_train["Cabin"]  
  
del df\_test["PassengerId"]  
del df\_test["Name"]  
del df\_test["Cabin"]

df\_train.head()

HomePlanet CryoSleep Destination Age VIP RoomService FoodCourt \  
0 Europa False TRAPPIST-1e 39.0 False 0.0 0.0   
1 Earth False TRAPPIST-1e 24.0 False 109.0 9.0   
2 Europa False TRAPPIST-1e 58.0 True 43.0 3576.0   
3 Europa False TRAPPIST-1e 33.0 False 0.0 1283.0   
4 Earth False TRAPPIST-1e 16.0 False 303.0 70.0   
  
 ShoppingMall Spa VRDeck Transported   
0 0.0 0.0 0.0 False   
1 25.0 549.0 44.0 True   
2 0.0 6715.0 49.0 False   
3 371.0 3329.0 193.0 False   
4 151.0 565.0 2.0 True

df\_test.head()

HomePlanet CryoSleep Destination Age VIP RoomService FoodCourt \  
0 Earth True TRAPPIST-1e 27.0 False 0.0 0.0   
1 Earth False TRAPPIST-1e 19.0 False 0.0 9.0   
2 Europa True 55 Cancri e 31.0 False 0.0 0.0   
3 Europa False TRAPPIST-1e 38.0 False 0.0 6652.0   
4 Earth False TRAPPIST-1e 20.0 False 10.0 0.0   
  
 ShoppingMall Spa VRDeck   
0 0.0 0.0 0.0   
1 0.0 2823.0 0.0   
2 0.0 0.0 0.0   
3 0.0 181.0 585.0   
4 635.0 0.0 0.0

df\_train.isna().sum()

HomePlanet 201  
CryoSleep 217  
Destination 182  
Age 179  
VIP 203  
RoomService 181  
FoodCourt 183  
ShoppingMall 208  
Spa 183  
VRDeck 188  
Transported 0  
dtype: int64

df\_test.isna().sum()

HomePlanet 87  
CryoSleep 93  
Destination 92  
Age 91  
VIP 93  
RoomService 82  
FoodCourt 106  
ShoppingMall 98  
Spa 101  
VRDeck 80  
dtype: int64

def ProcessNum(df):  
 num\_data = df.select\_dtypes(['float64'])   
 num\_col = list(num\_data.columns)  
 dict\_num = {i:num\_col[i] for i in range(len(num\_col)) }  
 num\_data.head()  
 imputer = SimpleImputer(strategy='mean')  
 d=imputer.fit\_transform(num\_data)  
 temp=pd.DataFrame(d)  
 temp = temp.rename(columns=dict\_num)   
 return temp  
  
def ProcessObj(df):  
 obj\_data = df.select\_dtypes(['object'])   
 obj\_col = list(obj\_data.columns)  
 for col in list(obj\_data.columns):  
 obj\_data[col] = obj\_data[col].fillna(obj\_data[col].mode()[0])  
 pass  
 z = obj\_data.columns  
 for i in z:  
 un = obj\_data[i].unique()  
 ran = range(1,len(un)+1)  
 obj\_data.replace(dict(zip(un,ran)) ,inplace=True)  
 obj\_data = pd.get\_dummies(obj\_data, columns=['HomePlanet', 'Destination'], prefix = ['HomePlanet', 'Destination'])  
 return obj\_data  
  
def ProcessBool(df):  
   
 bool\_data = df.select\_dtypes(['bool'])  
 col = bool\_data.columns  
 for i in col:  
 bool\_data[i] = LabelEncoder().fit\_transform(bool\_data[i])  
 return bool\_data  
   
  
test\_num = ProcessNum(df\_test)  
train\_num = ProcessNum(df\_train)  
test\_cat = ProcessObj(df\_test)  
train\_cat = ProcessObj(df\_train)  
train\_bool = ProcessBool(df\_train)  
train\_data\_process = pd.concat([train\_num,train\_cat,train\_bool],axis=1)  
test\_data\_process = pd.concat([test\_num,test\_cat],axis=1)

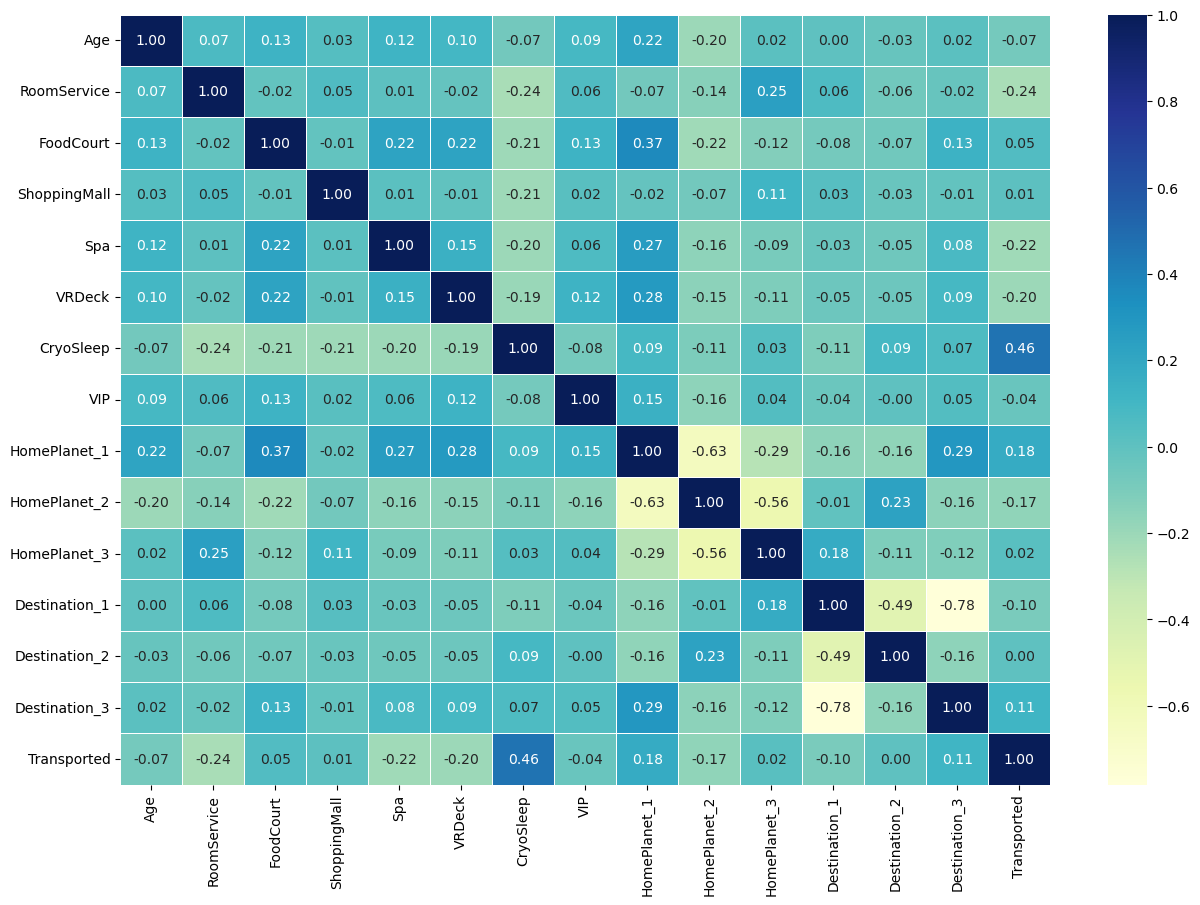
train\_data\_process

Age RoomService FoodCourt ShoppingMall Spa VRDeck CryoSleep \  
0 39.0 0.0 0.0 0.0 0.0 0.0 1   
1 24.0 109.0 9.0 25.0 549.0 44.0 1   
2 58.0 43.0 3576.0 0.0 6715.0 49.0 1   
3 33.0 0.0 1283.0 371.0 3329.0 193.0 1   
4 16.0 303.0 70.0 151.0 565.0 2.0 1   
... ... ... ... ... ... ... ...   
8688 41.0 0.0 6819.0 0.0 1643.0 74.0 1   
8689 18.0 0.0 0.0 0.0 0.0 0.0 2   
8690 26.0 0.0 0.0 1872.0 1.0 0.0 1   
8691 32.0 0.0 1049.0 0.0 353.0 3235.0 1   
8692 44.0 126.0 4688.0 0.0 0.0 12.0 1   
  
 VIP HomePlanet\_1 HomePlanet\_2 HomePlanet\_3 Destination\_1 \  
0 1 True False False True   
1 1 False True False True   
2 2 True False False True   
3 1 True False False True   
4 1 False True False True   
... ... ... ... ... ...   
8688 2 True False False False   
8689 1 False True False False   
8690 1 False True False True   
8691 1 True False False False   
8692 1 True False False True   
  
 Destination\_2 Destination\_3 Transported   
0 False False 0   
1 False False 1   
2 False False 0   
3 False False 0   
4 False False 1   
... ... ... ...   
8688 False True 0   
8689 True False 0   
8690 False False 1   
8691 False True 0   
8692 False False 1   
  
[8693 rows x 15 columns]

# Co-relation matrix  
train\_data\_process.corr()

Age RoomService FoodCourt ShoppingMall Spa \  
Age 1.000000 0.067612 0.127937 0.032655 0.120992   
RoomService 0.067612 1.000000 -0.015521 0.052962 0.009925   
FoodCourt 0.127937 -0.015521 1.000000 -0.013934 0.220587   
ShoppingMall 0.032655 0.052962 -0.013934 1.000000 0.013678   
Spa 0.120992 0.009925 0.220587 0.013678 1.000000   
VRDeck 0.099210 -0.019207 0.224275 -0.007189 0.147957   
CryoSleep -0.070736 -0.243986 -0.205682 -0.206366 -0.198392   
VIP 0.091574 0.056595 0.126006 0.018483 0.060573   
HomePlanet\_1 0.217444 -0.067476 0.365500 -0.021019 0.266323   
HomePlanet\_2 -0.201109 -0.139941 -0.215384 -0.071708 -0.159188   
HomePlanet\_3 0.016081 0.245451 -0.124781 0.111257 -0.088140   
Destination\_1 0.003008 0.061098 -0.078216 0.031603 -0.032948   
Destination\_2 -0.028247 -0.062828 -0.065542 -0.031397 -0.053428   
Destination\_3 0.016709 -0.024288 0.134994 -0.013346 0.075242   
Transported -0.074249 -0.242048 0.046074 0.010019 -0.218791   
  
 VRDeck CryoSleep VIP HomePlanet\_1 HomePlanet\_2 \  
Age 0.099210 -0.070736 0.091574 0.217444 -0.201109   
RoomService -0.019207 -0.243986 0.056595 -0.067476 -0.139941   
FoodCourt 0.224275 -0.205682 0.126006 0.365500 -0.215384   
ShoppingMall -0.007189 -0.206366 0.018483 -0.021019 -0.071708   
Spa 0.147957 -0.198392 0.060573 0.266323 -0.159188   
VRDeck 1.000000 -0.193107 0.123092 0.282118 -0.153676   
CryoSleep -0.193107 1.000000 -0.078281 0.093395 -0.107231   
VIP 0.123092 -0.078281 1.000000 0.147008 -0.162345   
HomePlanet\_1 0.282118 0.093395 0.147008 1.000000 -0.633221   
HomePlanet\_2 -0.153676 -0.107231 -0.162345 -0.633221 1.000000   
HomePlanet\_3 -0.111875 0.032715 0.043523 -0.287022 -0.559658   
Destination\_1 -0.050878 -0.113380 -0.039617 -0.156958 -0.007418   
Destination\_2 -0.045454 0.087764 -0.000592 -0.163308 0.231177   
Destination\_3 0.089818 0.065589 0.045167 0.293517 -0.156169   
Transported -0.204825 0.460132 -0.037261 0.176916 -0.168845   
  
 HomePlanet\_3 Destination\_1 Destination\_2 Destination\_3 \  
Age 0.016081 0.003008 -0.028247 0.016709   
RoomService 0.245451 0.061098 -0.062828 -0.024288   
FoodCourt -0.124781 -0.078216 -0.065542 0.134994   
ShoppingMall 0.111257 0.031603 -0.031397 -0.013346   
Spa -0.088140 -0.032948 -0.053428 0.075242   
VRDeck -0.111875 -0.050878 -0.045454 0.089818   
CryoSleep 0.032715 -0.113380 0.087764 0.065589   
VIP 0.043523 -0.039617 -0.000592 0.045167   
HomePlanet\_1 -0.287022 -0.156958 -0.163308 0.293517   
HomePlanet\_2 -0.559658 -0.007418 0.231177 -0.156169   
HomePlanet\_3 1.000000 0.177243 -0.111260 -0.120996   
Destination\_1 0.177243 1.000000 -0.486554 -0.783137   
Destination\_2 -0.111260 -0.486554 1.000000 -0.162240   
Destination\_3 -0.120996 -0.783137 -0.162240 1.000000   
Transported 0.019544 -0.096319 0.000092 0.108722   
  
 Transported   
Age -0.074249   
RoomService -0.242048   
FoodCourt 0.046074   
ShoppingMall 0.010019   
Spa -0.218791   
VRDeck -0.204825   
CryoSleep 0.460132   
VIP -0.037261   
HomePlanet\_1 0.176916   
HomePlanet\_2 -0.168845   
HomePlanet\_3 0.019544   
Destination\_1 -0.096319   
Destination\_2 0.000092   
Destination\_3 0.108722   
Transported 1.000000

# Lets make our correlation matrix a little pettier  
import seaborn as sns  
corr\_matrix = train\_data\_process.corr()  
fig, ax = plt.subplots(figsize = (15, 10))  
ax = sns.heatmap(corr\_matrix,  
 annot=True,  
 linewidths=0.5,  
 fmt=".2f",  
 cmap="YlGnBu");



## Modeling

# Splitting the data  
X = train\_data\_process.drop('Transported', axis=1)  
y = train\_data\_process['Transported']

X.shape

(8693, 14)

X

Age RoomService FoodCourt ShoppingMall Spa VRDeck CryoSleep \  
0 39.0 0.0 0.0 0.0 0.0 0.0 1   
1 24.0 109.0 9.0 25.0 549.0 44.0 1   
2 58.0 43.0 3576.0 0.0 6715.0 49.0 1   
3 33.0 0.0 1283.0 371.0 3329.0 193.0 1   
4 16.0 303.0 70.0 151.0 565.0 2.0 1   
... ... ... ... ... ... ... ...   
8688 41.0 0.0 6819.0 0.0 1643.0 74.0 1   
8689 18.0 0.0 0.0 0.0 0.0 0.0 2   
8690 26.0 0.0 0.0 1872.0 1.0 0.0 1   
8691 32.0 0.0 1049.0 0.0 353.0 3235.0 1   
8692 44.0 126.0 4688.0 0.0 0.0 12.0 1   
  
 VIP HomePlanet\_1 HomePlanet\_2 HomePlanet\_3 Destination\_1 \  
0 1 True False False True   
1 1 False True False True   
2 2 True False False True   
3 1 True False False True   
4 1 False True False True   
... ... ... ... ... ...   
8688 2 True False False False   
8689 1 False True False False   
8690 1 False True False True   
8691 1 True False False False   
8692 1 True False False True   
  
 Destination\_2 Destination\_3   
0 False False   
1 False False   
2 False False   
3 False False   
4 False False   
... ... ...   
8688 False True   
8689 True False   
8690 False False   
8691 False True   
8692 False False   
  
[8693 rows x 14 columns]

y.shape

(8693,)

# Splitting the data into train and test sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

X\_train

Age RoomService FoodCourt ShoppingMall Spa VRDeck CryoSleep \  
2333 28.0 0.000000 55.0 0.0 656.0 0.0 1   
2589 17.0 0.000000 1195.0 31.0 0.0 0.0 1   
8302 28.0 0.000000 0.0 0.0 0.0 0.0 2   
8177 20.0 224.687617 2.0 289.0 976.0 0.0 1   
500 36.0 0.000000 0.0 0.0 0.0 0.0 2   
... ... ... ... ... ... ... ...   
5734 18.0 14.000000 2.0 144.0 610.0 0.0 1   
5191 50.0 690.000000 0.0 30.0 762.0 428.0 1   
5390 22.0 158.000000 0.0 476.0 0.0 26.0 1   
860 34.0 379.000000 0.0 1626.0 0.0 0.0 1   
7270 28.0 7.000000 489.0 0.0 4.0 6027.0 1   
  
 VIP HomePlanet\_1 HomePlanet\_2 HomePlanet\_3 Destination\_1 \  
2333 1 False True False True   
2589 1 False True False True   
8302 1 True False False False   
8177 1 False False True True   
500 1 True False False False   
... ... ... ... ... ...   
5734 1 False True False True   
5191 1 False False True True   
5390 1 False True False False   
860 1 False False True True   
7270 1 True False False False   
  
 Destination\_2 Destination\_3   
2333 False False   
2589 False False   
8302 False True   
8177 False False   
500 False True   
... ... ...   
5734 False False   
5191 False False   
5390 True False   
860 False False   
7270 False True   
  
[6954 rows x 14 columns]

Now we've got our data split into training and test sets, it's time to a machine learning model.

We'll train it(find patterns) on the training set.

And we'll test it(use the patterns) on the test set.

We're going to try 3 machine learning models:

1. Logistic Regression
2. K-Nearest Neighbours Classifier
3. Random Forest Classifier

# Put models to a dictionary  
models = {"Logistic Regression": LogisticRegression(),  
 "KNN": KNeighborsClassifier(),  
 "Random Forest Classifier": RandomForestClassifier()}  
  
# Creating a funtion to fit and score models  
def fit\_and\_score\_models(models, X\_train, X\_test, y\_train, y\_test):  
 """  
 Fits and evaluates given machine learning models.  
 models: a dict of diff Scikit-Learn machine learning models  
 X\_train: Training data(no labels)  
 X\_test: Testing data(no labels)  
 y\_train: Training labels  
 y\_test: Testing labels  
 """  
 # Random Seed  
 np.random.seed(42)  
   
 # A dictionary to store the scores  
 model\_scores = {}  
   
 # Loop through models  
 for name, model in models.items():  
 #Fit the model with train data  
 model.fit(X\_train, y\_train)  
 # Score the model with test data  
 model\_scores[name] = model.score(X\_test, y\_test)  
   
 return model\_scores

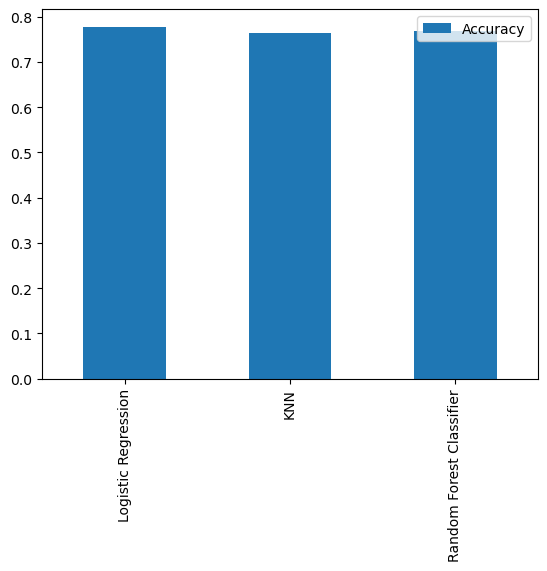
model\_scores = fit\_and\_score\_models(models = models, X\_train=X\_train, X\_test=X\_test, y\_train=y\_train, y\_test=y\_test)  
model\_scores

C:\Users\Harshith-Raj\space-titanic\env\lib\site-packages\sklearn\linear\_model\\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.  
  
Increase the number of iterations (max\_iter) or scale the data as shown in:  
 https://scikit-learn.org/stable/modules/preprocessing.html  
Please also refer to the documentation for alternative solver options:  
 https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression  
 n\_iter\_i = \_check\_optimize\_result(

{'Logistic Regression': 0.7780333525014376,  
 'KNN': 0.7630822311673375,  
 'Random Forest Classifier': 0.7694077055779184}

## Model Comparision

model\_compare = pd.DataFrame(model\_scores, index = ["Accuracy"])  
model\_compare.T.plot.bar();



Now we've got a baseline model with some Accuracy score let's try to increase the score by further Tuning the model.

## Hyperparameter Tuning

# Lets tune the KNN  
train\_scores = []  
test\_scores = []  
  
# List of diff n\_neighbors  
neighbors = range(1,21)  
  
# model instance  
knn = KNeighborsClassifier()  
  
# Looping through  
for i in neighbors:  
 knn.set\_params(n\_neighbors = i)  
   
 # Fit the model  
 knn.fit(X\_train, y\_train)  
   
 # Score the model on train set  
 train\_scores.append(knn.score(X\_train, y\_train))  
   
 # Score the model on test set  
 test\_scores.append(knn.score(X\_test, y\_test))

train\_scores

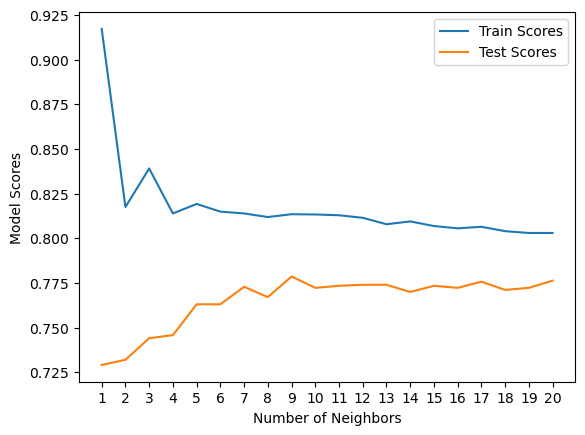
[0.917169974115617,  
 0.8175150992234685,  
 0.8390854184641933,  
 0.813920046016681,  
 0.8192407247627265,  
 0.8149266609145815,  
 0.813920046016681,  
 0.8119068162208801,  
 0.8134886396318666,  
 0.8133448375035951,  
 0.8129134311187806,  
 0.8114754098360656,  
 0.8078803566292782,  
 0.8094621800402646,  
 0.8068737417313776,  
 0.8055795225769341,  
 0.8064423353465632,  
 0.8039976991659477,  
 0.8029910842680472,  
 0.8029910842680472]

test\_scores

[0.7291546866014951,  
 0.7320299022426682,  
 0.7441058079355952,  
 0.745830937320299,  
 0.7630822311673375,  
 0.7630822311673375,  
 0.772857964347326,  
 0.7671075330649799,  
 0.7786083956296722,  
 0.7722829212190915,  
 0.7734330074755607,  
 0.7740080506037953,  
 0.7740080506037953,  
 0.7699827487061529,  
 0.7734330074755607,  
 0.7722829212190915,  
 0.7757331799884991,  
 0.7711328349626222,  
 0.7722829212190915,  
 0.7763082231167338]

plt.plot(neighbors, train\_scores, label = "Train Scores")  
plt.plot(neighbors, test\_scores, label = "Test Scores")  
plt.xticks(np.arange(1,21,1))  
plt.xlabel("Number of Neighbors")  
plt.ylabel("Model Scores")  
plt.legend()  
  
print(f"Maximum score on training data set: {max(test\_scores)\*100:.2f}%")

Maximum score on training data set: 77.86%



## Hyperparameter Tuning

We are gonna tune these two models,

* Logistic Regression
* Random Forest Classifier ..usng RandomizedSearchCV

# Create a hyperparameter grid for LogisticRegression()  
log\_reg\_grid = {"C": np.logspace(-4, 4, 20),  
 "solver": ["liblinear"]}  
  
# Create a hyperparameter grid for RandomForestClassifier()  
rf\_grid = {"n\_estimators": np.arange(10, 1000, 50),  
 "max\_depth": [None, 3, 5, 10],  
 "min\_samples\_split": np.arange(2, 20, 2),  
 "min\_samples\_leaf": np.arange(1, 20, 2)}

Now we've got parameter grids lets tune the model using RandomizedSearchCV

# Tune Logistic Model  
np.random.seed(42)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)  
  
# Setup random hyperparameter search for LogisticRegression()  
rs\_log\_reg = RandomizedSearchCV(LogisticRegression(),  
 param\_distributions=log\_reg\_grid,  
 cv = 5,  
 n\_iter = 20,  
 verbose = True)  
  
# Fit random hyperparameter search model for LogisticRegression()  
rs\_log\_reg.fit(X\_train, y\_train);

Fitting 5 folds for each of 20 candidates, totalling 100 fits

rs\_log\_reg.best\_params\_

{'solver': 'liblinear', 'C': 0.012742749857031334}

rs\_log\_reg.score(X\_test, y\_test)

0.777458309373203

# Tune Random Forest Model  
np.random.seed(42)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)  
  
# Setup random hyperparameter search for LogisticRegression()  
rs\_rf = RandomizedSearchCV(RandomForestClassifier(),  
 param\_distributions=rf\_grid,  
 cv = 5,  
 n\_iter = 20,  
 verbose = True)  
  
# Fit random hyperparameter search model for LogisticRegression()  
rs\_rf.fit(X\_train, y\_train);

Fitting 5 folds for each of 20 candidates, totalling 100 fits

rs\_rf.best\_params\_

{'n\_estimators': 810,  
 'min\_samples\_split': 2,  
 'min\_samples\_leaf': 13,  
 'max\_depth': None}

rs\_rf.score(X\_test, y\_test)

0.7883841288096607

model\_scores

{'Logistic Regression': 0.7780333525014376,  
 'KNN': 0.7630822311673375,  
 'Random Forest Classifier': 0.7694077055779184}

## Hyperparaameter tuning with GridSearchCV

Since our RandomForestClassifier model provides the best scores so far, we''ll try and improve them by again using GridSearchCV()

# Create a grid with best params  
rs\_rf\_grid = {'n\_estimators': [810],  
 'min\_samples\_split': [2],  
 'min\_samples\_leaf': [13],  
 'max\_depth': [None]}  
  
  
# Setup grid hyperparameter search for LogisticRegression  
gs\_rf = GridSearchCV(RandomForestClassifier(),  
 param\_grid=rs\_rf\_grid,  
 cv=5,  
 verbose=True)  
  
# Fit the model  
gs\_rf.fit(X\_train, y\_train)

Fitting 5 folds for each of 1 candidates, totalling 5 fits

GridSearchCV(cv=5, estimator=RandomForestClassifier(),  
 param\_grid={'max\_depth': [None], 'min\_samples\_leaf': [13],  
 'min\_samples\_split': [2], 'n\_estimators': [810]},  
 verbose=True)

gs\_rf.score(X\_test, y\_test)

0.7889591719378953

y\_preds = gs\_rf.predict(X\_test)

y\_preds[:10]

array([0, 1, 1, 0, 1, 1, 0, 0, 1, 0], dtype=int64)

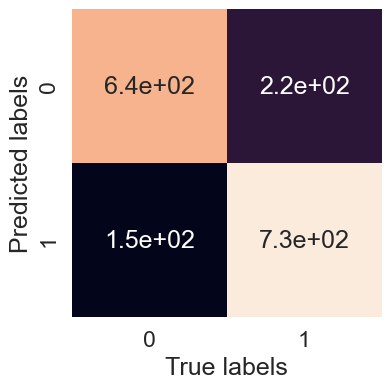
np.array(y\_test[:10])

array([1, 0, 0, 1, 1, 1, 0, 0, 1, 0], dtype=int64)

# Confusion Matrix  
print(confusion\_matrix(y\_test, y\_preds))

[[644 217]  
 [150 728]]

# Plot the confusion matrix with seaborn  
sns.set(font\_scale = 1.5)  
  
def plot\_conf\_mat(y\_test, y\_preds):  
 """  
 Plots a nice looking confusion matrix using Seaborns heatmap()  
 """  
   
 fgi, ax = plt.subplots(figsize = (4, 4))  
 ax = sns.heatmap(confusion\_matrix(y\_test, y\_preds),  
 annot=True,  
 cbar=False)  
 plt.xlabel("True labels")  
 plt.ylabel("Predicted labels")  
  
plot\_conf\_mat(y\_test, y\_preds)



### Calculate evaluatin metrics using cross-validation

we're going to calculae accuracy, precision, recall and f1-score of our model using cross-validation and to do so we'll be using cross\_val\_score

gs\_rf.best\_params\_

{'max\_depth': None,  
 'min\_samples\_leaf': 13,  
 'min\_samples\_split': 2,  
 'n\_estimators': 810}

# Create a new model with best params  
clf = RandomForestClassifier(max\_depth=None,  
 min\_samples\_leaf=13,  
 min\_samples\_split=2,  
 n\_estimators=810)

# Cross validation accuracy  
cv\_acc = cross\_val\_score(clf, X, y, cv=5, scoring="accuracy")  
cv\_acc

array([0.78895917, 0.78320874, 0.79125934, 0.80955121, 0.80724971])

cv\_acc = np.mean(cv\_acc)

cv\_acc

0.7960456355285334

# Submitting to Kaggle (formatting)  
submission = pd.read\_csv('data/sample\_submission.csv')  
features\_test = np.array(test\_data\_process)  
y\_test\_pred = gs\_rf.predict(features\_test)

C:\Users\Harshith-Raj\space-titanic\env\lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names  
 warnings.warn(

y\_test\_pred

array([1, 0, 1, ..., 1, 1, 1], dtype=int64)

submission['Transported'] = y\_test\_pred.astype(bool)  
submission.to\_csv('data/submission.csv', index=False)