# Spaceship Titanic Competition (Kaggle)

January 22, 2024

**Nihar Patel**

## 1 1. Importing Libraries

- Importing all the required dependencies for the model.

[1]:
```
# Importing libraries
import os
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt

# Importing scikit-learn preprocessing libraries
from sklearn.preprocessing import MinMaxScaler

# Importing scikit-learn and xgboost modeling libraries
from sklearn.ensemble import RandomForestClassifier
```

## 2 2. Importing Data Files

- Reading the `train.csv` and `test.csv` file from the given data path `"/kaggle/input/spaceship-titanic/train.csv"` and `"/kaggle/input/spaceship-titanic/test.csv"` loading it into variable `X` and `X_test`, using `pd.read_csv` as DataFrame.
- Calculating number of non-null values and data-type of every column in `X`, using `.info()`.

[2]:
```
# Read train Data file from the path
X = pd.read_csv("/kaggle/input/spaceship-titanic/train.csv")
X.head()
```

[2]:

| | 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 | |

```
   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

   Transported
0        False
1         True
2        False
3        False
4         True
```

[3]:
```
# Print shape of the data frame
X.shape
```

[3]:
```
(8693, 14)
```

[4]:
```
# Checking non-null values and dtype of each column
X.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
```

## 3 3. Visualizing the Raw Data

- Listing down all the column names in `X`.
- Getting Descriptive analysis of `X` using `.describe()` and visualizing using `.hist()`.

[5]:
```
X.columns
```

[5]:
```
Index(['PassengerId', 'HomePlanet', 'CryoSleep', 'Cabin', 'Destination', 'Age',
       'VIP', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck',
       'Name', 'Transported'],
      dtype='object')
```

**File and Data Field Descriptions**

- **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.

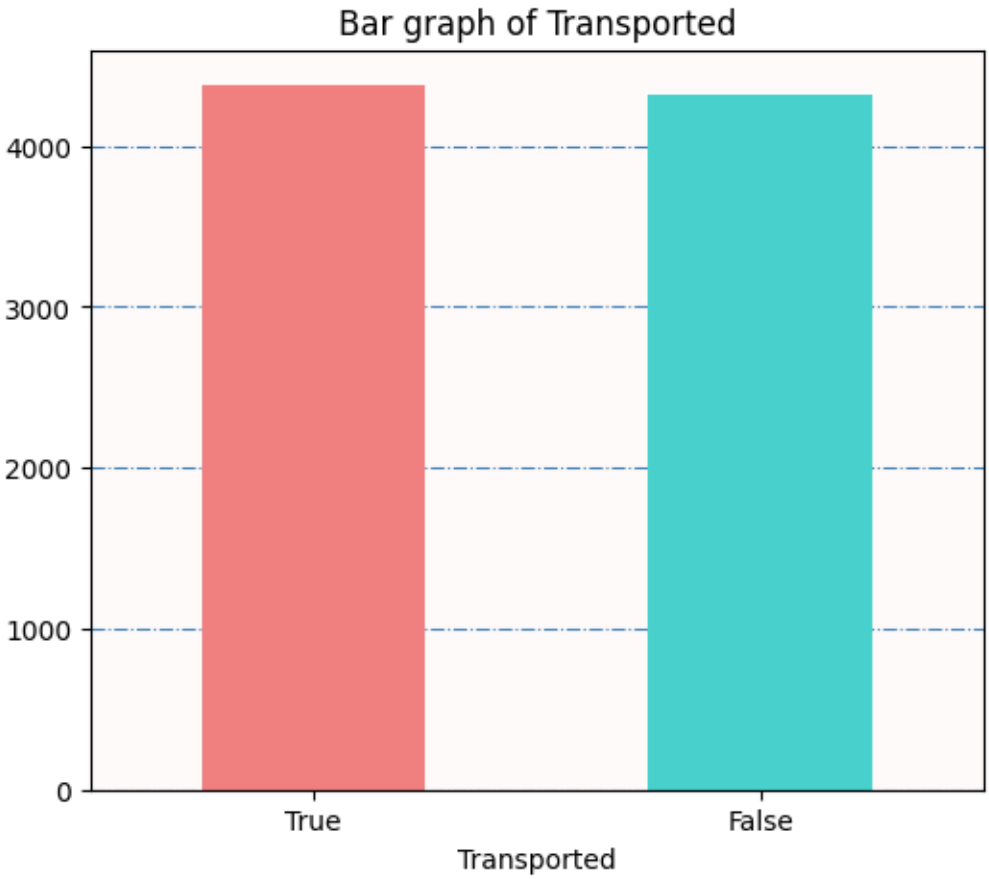
- **sample_submission.csv** - A submission file in the correct format.

  `PassengerId` - Id for each passenger in the test set.

  `Transported` - The target. For each passenger, predict either True or False.

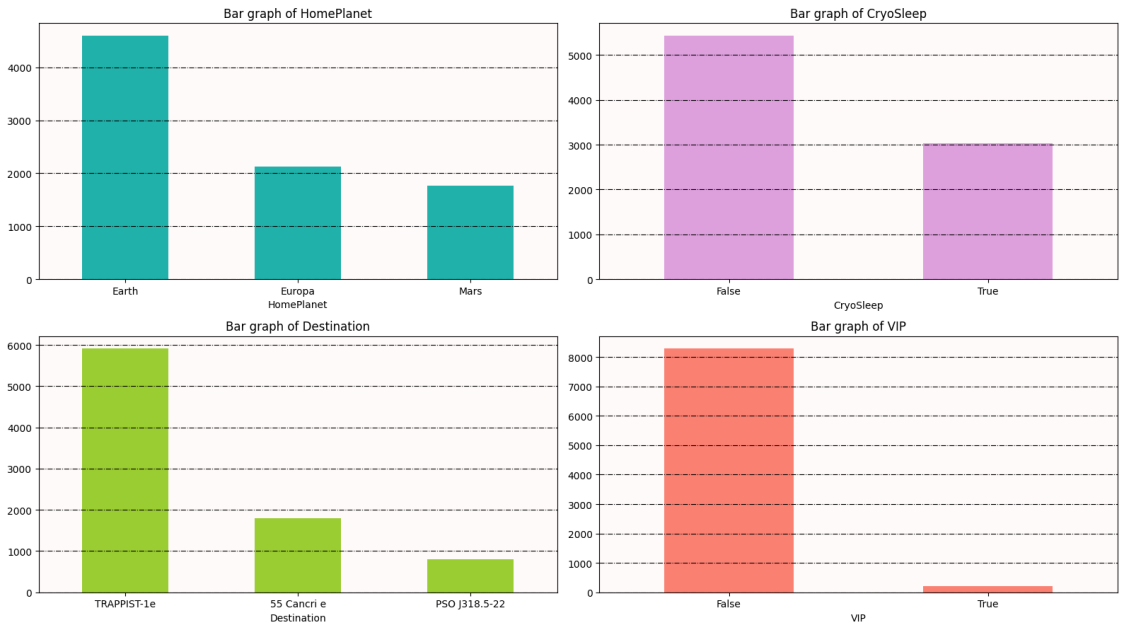
[6]:
```
# Visualization of target (Transported) columns
target = "Transported"

fig, ax = plt.subplots(figsize=(6,5))
X[target].value_counts().plot(kind='bar', ax=ax, color=["lightcoral",␣
 ↪"mediumturquoise"])
ax.set_title(f"Bar graph of {target}")
```

```
ax.tick_params(axis='x', rotation=0)
ax.yaxis.grid(color='steelblue', linestyle="dashdot")
ax.set_facecolor(color="snow")
ax.set_axisbelow(True)
```




[7]:
```
# Visualization of Catagorical columns
cols = ["HomePlanet", "CryoSleep", "Destination", "VIP"]

fig, ax = plt.subplots(figsize=(16,9), nrows=2, ncols=2)

X[cols[0]].value_counts().plot(kind='bar', ax=ax[0,0], color="lightseagreen")
ax[0, 0].set_title(f"Bar graph of {cols[0]}")
ax[0, 0].tick_params(axis='x', rotation=0)
ax[0, 0].yaxis.grid(color='k', linestyle="dashdot")
ax[0, 0].set_facecolor(color="snow")

X[cols[1]].value_counts().plot(kind='bar', ax=ax[0,1], color="plum")
ax[0, 1].set_title(f"Bar graph of {cols[1]}")
ax[0, 1].tick_params(axis='x', rotation=0)
```

```
ax[0, 1].yaxis.grid(color='k', linestyle="dashdot")
ax[0, 1].set_facecolor(color="snow")

X[cols[2]].value_counts().plot(kind='bar', ax=ax[1,0], color="yellowgreen")
ax[1, 0].set_title(f"Bar graph of {cols[2]}")
ax[1, 0].tick_params(axis='x', rotation=0)
ax[1, 0].yaxis.grid(color='k', linestyle="dashdot")
ax[1, 0].set_facecolor(color="snow")

X[cols[3]].value_counts().plot(kind='bar', ax=ax[1,1], color="salmon")
ax[1, 1].set_title(f"Bar graph of {cols[3]}")
ax[1, 1].tick_params(axis='x', rotation=0)
ax[1, 1].yaxis.grid(color='k', linestyle="dashdot")
ax[1, 1].set_facecolor(color="snow")

plt.tight_layout()
```




[8]:
```
# Descriptive statistics of numeric columns
X.describe()
```

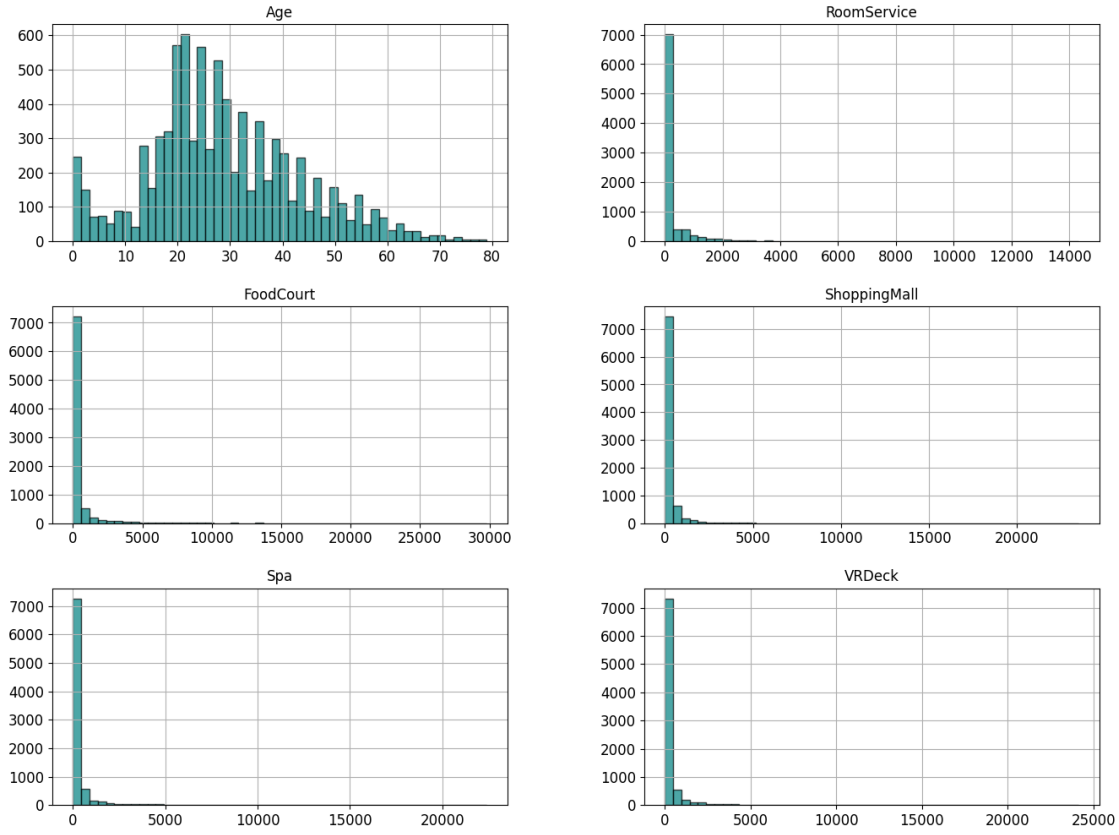
[8]:

| | 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
```

[9]:
```
# Histogram of Continious columns
ax = X.hist(figsize=(16,12), bins=50, xlabelsize=12, ylabelsize=12, grid=True,␣
↪color="teal", edgecolor='black', alpha=0.7);
```



# 4 4. Data Pre-processing

- Calculating total number of null values in `X` column wise.
- Dropping unwanted columns from `X` using `.drop()` function.
- Dropping all the records having at least one `NaN` value.
- Checking once again total number of null values in `X`.

[10]:
```
# Calculating null values in each column
X.isnull().sum()
```

[10]:
```
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
```

[11]:
```
# Removing unwanted columns
X.drop(columns=["PassengerId","Name"], inplace=True)

# Removing records having NaN values
X.dropna(inplace=True)

# Re-calculating null values in each column
X.isnull().sum()
```

[11]:
```
HomePlanet      0
CryoSleep       0
Cabin           0
Destination     0
Age             0
VIP             0
RoomService     0
FoodCourt       0
ShoppingMall    0
Spa             0
VRDeck          0
Transported     0
dtype: int64
```

### 4.1 (i). Tranforming Data

- Listing down number of unique values and their count, using `.unique()`.
- Transforming every column into one-hot encoded column.

[12]:
```
# Printing Unique values and their counts for each column
for i in X.columns:
    print(f"\n{i}")
    print(f"{X[i].unique()}")
    print(f"Count of Unique Values: {X[i].unique().shape[0]}")
```

```
HomePlanet
['Europa' 'Earth' 'Mars']
Count of Unique Values: 3

CryoSleep
[False True]
Count of Unique Values: 2

Cabin
['B/0/P' 'F/0/S' 'A/0/S' ... 'G/1499/S' 'G/1500/S' 'E/608/S']
Count of Unique Values: 5413

Destination
['TRAPPIST-1e' 'PSO J318.5-22' '55 Cancri e']
Count of Unique Values: 3

Age
[39. 24. 58. 33. 16. 44. 26. 35. 14. 45. 32. 48. 28.  0.  1. 49. 10.  7.
 21. 15. 34. 43. 47.  2. 23. 20. 17. 55.  4. 56. 25. 38. 27. 36. 22. 62.
 18. 42. 19. 37. 13.  8. 40.  3. 54.  9.  6. 31. 29. 64. 67. 61. 50. 11.
 51. 41. 30. 46. 60. 63. 57.  5. 79. 68. 59. 74. 12. 65. 53. 71. 52. 75.
 76. 78. 70. 73. 66. 69. 72. 77.]
Count of Unique Values: 80

VIP
[False True]
Count of Unique Values: 2

RoomService
[   0.  109.   43. ... 1003. 8586.  745.]
Count of Unique Values: 1112

FoodCourt
[   0.    9. 3576. ... 1146. 6819. 4688.]
Count of Unique Values: 1318

ShoppingMall
```

```
[0.0000e+00 2.5000e+01 3.7100e+02 1.5100e+02 3.0000e+00 1.7000e+01
 5.8900e+02 1.1230e+03 6.5000e+01 1.2000e+01 1.0000e+00 6.9000e+01
 1.3600e+02 8.1000e+01 1.4110e+03 4.5000e+01 1.8000e+02 1.9380e+03
 3.3740e+03 1.4800e+02 1.9700e+02 4.4600e+02 9.0000e+00 5.0000e+00
 7.3800e+02 1.0180e+03 1.2950e+03 3.1000e+01 6.7000e+02 9.5700e+02
 5.2600e+02 3.2600e+02 8.7000e+01 8.0400e+02 4.2900e+02 4.9000e+01
 2.9600e+02 7.3000e+02 1.2600e+02 2.1000e+02 1.0000e+01 4.8000e+01
 1.6000e+01 1.6100e+02 6.7300e+02 4.2600e+02 4.0100e+02 8.6700e+02
 7.1900e+02 6.6000e+01 1.3700e+02 3.0700e+02 1.1900e+02 1.8800e+02
 3.0100e+02 2.2100e+02 6.6500e+02 2.5600e+02 7.9600e+02 2.8700e+02
 5.4100e+02 4.0800e+02 3.1300e+02 5.9200e+02 1.2800e+02 2.0780e+03
 2.9000e+01 3.2000e+01 8.8000e+01 1.3240e+03 1.5000e+01 2.5900e+02
 1.3590e+03 1.5700e+02 7.4000e+01 2.5100e+02 2.1100e+02 6.3400e+02
 1.4340e+03 2.0200e+02 2.9750e+03 4.5300e+02 3.3000e+01 4.4470e+03
 5.5800e+02 7.8600e+02 4.0000e+00 7.0000e+00 6.0000e+00 5.9100e+02
 8.8100e+02 3.5600e+02 5.5200e+02 3.6000e+01 5.2000e+01 1.1160e+03
 1.0100e+02 5.9520e+03 2.6000e+01 2.2000e+01 8.4000e+02 1.0200e+02
 7.9500e+02 7.4700e+02 1.8200e+02 8.6300e+02 2.0000e+00 5.4400e+02
 3.8000e+01 1.7490e+03 4.7000e+01 1.1000e+01 4.1000e+01 7.5000e+02
 1.7330e+03 3.8400e+02 3.1600e+02 4.7900e+02 8.3800e+02 8.3100e+02
 7.0400e+02 2.6600e+02 5.8000e+02 3.3200e+02 4.0200e+02 6.1200e+02
 3.1000e+02 2.1000e+01 5.2100e+02 3.7000e+01 6.0100e+02 1.0920e+03
 6.1600e+02 5.5100e+02 4.6000e+01 8.2600e+02 9.0800e+02 1.8000e+01
 5.0000e+01 8.2000e+01 1.4100e+02 1.0870e+03 7.0800e+02 7.3600e+02
 2.5200e+02 9.7000e+01 7.8300e+02 1.9520e+03 2.2600e+02 6.8900e+02
 7.2200e+02 3.2200e+02 4.8400e+02 5.5000e+01 6.7900e+02 3.2500e+02
 1.3990e+03 8.7000e+02 1.0000e+02 7.0100e+02 5.1000e+01 7.1400e+02
 3.4300e+02 2.6950e+03 7.1000e+01 1.9000e+01 3.2000e+02 4.4400e+02
 5.4000e+02 7.1600e+02 2.5520e+03 4.2100e+02 9.3000e+01 9.8400e+02
 1.6260e+03 2.3400e+02 8.7100e+02 5.5500e+02 1.3830e+03 8.0000e+00
 8.3400e+02 3.2300e+02 4.4800e+02 1.3000e+01 7.1800e+02 4.9000e+02
 3.5000e+01 9.8000e+01 8.8000e+02 7.1500e+02 1.2200e+02 1.0600e+02
 1.9730e+03 8.4000e+01 6.2000e+01 1.5260e+03 8.2800e+02 2.7000e+01
 6.0000e+01 6.4200e+02 2.9800e+02 9.9000e+01 2.6300e+02 7.5600e+02
 6.1240e+03 3.6300e+02 1.6320e+03 8.8400e+02 7.6000e+01 1.0950e+03
 7.2000e+01 9.2600e+02 2.7900e+02 6.0000e+02 6.5400e+02 4.6400e+02
 3.7000e+03 7.5200e+02 5.1300e+02 8.4300e+02 1.9500e+02 9.0700e+02
 1.2500e+03 1.0610e+03 3.3010e+03 4.3100e+02 8.9600e+02 2.8900e+02
 6.1900e+02 4.8200e+02 1.0840e+03 2.6800e+02 6.7500e+02 4.8170e+03
 1.0900e+02 2.9200e+02 2.3000e+01 3.4000e+02 2.6000e+02 1.9300e+02
 1.1280e+03 5.6100e+02 9.4600e+02 5.3700e+02 6.9000e+02 3.7700e+02
 2.8000e+02 2.1200e+02 2.2400e+02 3.3400e+02 5.2900e+02 2.8000e+01
 1.0130e+03 1.2430e+03 2.4300e+02 1.7210e+03 2.1160e+03 2.2800e+02
 3.9000e+01 2.1190e+03 4.2200e+02 7.3000e+01 7.4400e+02 9.1300e+02
 4.9700e+02 6.8000e+01 1.2170e+03 6.8300e+02 1.3600e+03 5.3300e+02
 6.6100e+02 4.3000e+01 1.4600e+02 3.7200e+02 1.4870e+03 7.9800e+02
 9.6500e+02 8.8500e+02 2.1800e+02 1.4700e+02 2.1300e+02 2.0800e+02
 1.1550e+03 2.0100e+02 6.8700e+02 5.1600e+02 1.2470e+03 8.9000e+01
```

6.4600e+02 1.2460e+03 4.7300e+02 4.9900e+02 3.8700e+02 4.9600e+02
1.2400e+02 8.0900e+02 2.6610e+03 4.5400e+02 2.5700e+02 8.6400e+02
6.5700e+02 9.7400e+02 1.0810e+03 1.7890e+03 1.7600e+02 1.5800e+02
3.9700e+02 3.4500e+02 3.4000e+01 2.8800e+02 1.0700e+02 1.5080e+03
2.9500e+02 1.4760e+03 7.8700e+02 5.3000e+01 3.9800e+02 1.1340e+03
4.8990e+03 6.7700e+02 9.4000e+01 5.6000e+01 6.9300e+02 1.3900e+02
3.5200e+02 8.0200e+02 2.4900e+02 5.4800e+02 1.0510e+03 5.0300e+02
6.9400e+02 2.6400e+03 8.7200e+02 8.0100e+02 2.3150e+03 6.1100e+02
7.3400e+02 1.4040e+03 1.0580e+03 4.8000e+02 6.9800e+02 1.6100e+03
6.1000e+01 2.0270e+03 9.9100e+02 9.0200e+02 5.6000e+02 1.8700e+02
8.5700e+02 6.7100e+02 7.4100e+02 4.6900e+02 9.2100e+02 8.0000e+01
1.4000e+01 1.7100e+02 1.8500e+02 6.4700e+02 1.2570e+03 4.7000e+02
1.1270e+03 2.5590e+03 4.3500e+02 7.0000e+01 3.8100e+02 9.5000e+01
7.9100e+02 6.2600e+02 3.8800e+02 2.0000e+01 5.7000e+01 6.4800e+02
7.8000e+01 9.9400e+02 2.6700e+02 1.4320e+03 6.5900e+02 6.7800e+02
3.0000e+01 4.4000e+01 3.1900e+02 1.3000e+02 1.0790e+03 1.5590e+03
4.9300e+02 1.1680e+03 5.7800e+02 4.5000e+02 5.4000e+01 5.2500e+02
1.1450e+03 1.3420e+03 3.4400e+02 1.1000e+02 1.8900e+02 1.6820e+03
4.5100e+02 2.4000e+01 9.2200e+02 9.6000e+01 1.3800e+02 5.5700e+02
7.4300e+02 2.8500e+02 2.6280e+03 1.6460e+03 1.1310e+03 2.7100e+02
1.5580e+03 1.7950e+03 1.9940e+03 3.8130e+03 1.8650e+03 1.1600e+02
3.4800e+02 3.6200e+02 1.3300e+02 1.7000e+02 1.6610e+03 4.3330e+03
6.6000e+02 3.6600e+02 6.4000e+01 1.5400e+02 9.8100e+02 1.7400e+02
1.1100e+02 7.7000e+01 4.4200e+02 1.6170e+03 9.2000e+01 3.6480e+03
1.2820e+03 1.6980e+03 7.7900e+02 4.4300e+02 2.0050e+03 1.5090e+03
3.1500e+02 5.6900e+02 3.7400e+02 9.3300e+02 1.3200e+02 7.0900e+02
1.4410e+03 2.5160e+03 2.7000e+02 1.7520e+03 1.1810e+03 9.4700e+02
1.6500e+03 1.7930e+03 1.3490e+03 6.3300e+02 1.5500e+02 6.5600e+02
2.9900e+02 2.6460e+03 6.1800e+02 3.9400e+02 5.9000e+01 9.7100e+02
1.4200e+02 6.9900e+02 1.5530e+03 6.6900e+02 7.2600e+02 7.5000e+01
4.0700e+02 1.1000e+03 1.2710e+03 2.3800e+02 5.9900e+02 2.3600e+02
3.4100e+02 1.1700e+02 1.5000e+02 5.0400e+02 7.6200e+02 5.3200e+02
9.0000e+01 5.8400e+02 7.0500e+02 1.2640e+03 8.3000e+02 2.6900e+02
8.9800e+02 4.1500e+02 1.8100e+02 5.8100e+02 4.0000e+02 1.4420e+03
9.1000e+01 3.9500e+02 2.9080e+03 1.2310e+03 6.7400e+02 1.7630e+03
6.8800e+02 2.7400e+02 8.0000e+02 1.7770e+03 1.1580e+03 5.0800e+02
2.6870e+03 1.2150e+03 4.5900e+02 6.4000e+02 2.7600e+02 8.2700e+02
3.3500e+02 7.8800e+02 3.5700e+02 2.4600e+02 9.8000e+02 2.3200e+02
1.7690e+03 6.3200e+02 8.6900e+02 1.3100e+02 1.1200e+02 7.1700e+02
2.5800e+02 1.5030e+03 4.5090e+03 2.8400e+02 1.3760e+03 2.4200e+02
1.9000e+02 1.5640e+03 6.2700e+02 2.2700e+02 7.3500e+02 5.9700e+02
1.8830e+03 2.1500e+02 6.2300e+02 1.1890e+03 1.4620e+03 5.5300e+02
7.1100e+02 2.3160e+03 8.0500e+02 4.7700e+02 2.5500e+02 7.6400e+02
1.1360e+03 4.2400e+02 9.7000e+02 9.2700e+02 2.2780e+03 1.3400e+02
6.6200e+02 2.2300e+02 5.3100e+02 3.7800e+02 3.6000e+02 2.4330e+03
1.8280e+03 1.5460e+03 1.2300e+02 1.6300e+02 4.8900e+02 1.1440e+03
7.7700e+02 7.0300e+02 3.6270e+03 1.8600e+02 9.5900e+02 2.0740e+03
3.2700e+02 7.4200e+02 3.5800e+02 4.1800e+02 1.8300e+02 3.5300e+02

5.2400e+02 4.5500e+02 1.0800e+03 1.6700e+02 1.6700e+03 1.4350e+03
7.7400e+02 1.1320e+03 1.1500e+02 6.6400e+02 6.0800e+02 8.8200e+02
4.7900e+03 1.0200e+03 2.2200e+02 2.7160e+03 2.3500e+02 7.8500e+02
3.6400e+02 4.8600e+02 2.1400e+03 8.3000e+01 5.6700e+02 7.1200e+02
7.5400e+02 4.8740e+03 1.1290e+03 9.4500e+02 9.4900e+02 4.2850e+03
4.4100e+02 2.0600e+02 6.1000e+02 4.2000e+01 1.5130e+03 1.4140e+03
4.7100e+02 7.7100e+02 2.4140e+03 3.1100e+02 1.4550e+03 6.3800e+02
5.0700e+02 1.7420e+03 1.3390e+03 8.9500e+02 1.2900e+02 8.8900e+02
8.7700e+02 3.2460e+03 3.7500e+02 2.1840e+03 1.9900e+03 1.6900e+02
2.1700e+02 6.0300e+02 2.5540e+03 8.5000e+01 2.0030e+03 4.6500e+02
3.9200e+02 1.4450e+03 1.6770e+03 6.1500e+02 1.0300e+02 2.4540e+03
2.4100e+02 4.5810e+03 4.9400e+03 2.0670e+03 3.8000e+02 1.1800e+02
1.3560e+03 1.9800e+02 4.7600e+02 8.8800e+02 1.2100e+02 9.2900e+02
3.7830e+03 7.4600e+02 2.7200e+02 1.5200e+02 4.3700e+02 1.8750e+03
6.5500e+02 9.4100e+02 7.1300e+02 8.6800e+02 7.8000e+02 5.6350e+03
5.0100e+02 8.4700e+02 1.2970e+03 1.9100e+02 2.7500e+03 2.1020e+03
3.0900e+02 1.0424e+04 9.1200e+02 1.9190e+03 7.0600e+02 3.2100e+02
2.8320e+03 8.4600e+02 5.0500e+02 6.0600e+02 1.2500e+02 4.3000e+02
7.8900e+02 1.5480e+03 2.2510e+03 6.0500e+02 5.8600e+02 1.9050e+03
4.1300e+02 5.7100e+02 4.4900e+02 5.6500e+02 8.5300e+02 1.4400e+02
1.0400e+03 1.3200e+03 5.2800e+02 2.1530e+03 2.5740e+03 3.5000e+02
1.0470e+03 1.5300e+02 1.6600e+02 1.8780e+03 2.1600e+02 6.6300e+02
2.1960e+03 1.0890e+03 5.5900e+02 6.8050e+03 7.7800e+02 5.4500e+02
6.1700e+02 6.6700e+02 1.0060e+03 2.9290e+03 2.4730e+03 5.4200e+02
3.0200e+02 1.5940e+03 2.1340e+03 2.6400e+02 1.6090e+03 1.1100e+03
2.3320e+03 7.2500e+02 9.7300e+02 3.1700e+02 5.9800e+02 7.2400e+02
8.5900e+02 1.0370e+03 2.0900e+02 6.8100e+02 1.4900e+02 5.9500e+02
2.5400e+02 2.0770e+03 1.1420e+03 8.9000e+02 1.9400e+02 8.4800e+02
8.9200e+02 7.9000e+02 3.0000e+02 3.2400e+02 1.1990e+03 4.1400e+02
1.7610e+03 1.4300e+02 8.9100e+02 8.0800e+02 7.6100e+02 3.6700e+02
2.6500e+02 9.0580e+03 2.3870e+03 1.2240e+03 6.5300e+02 1.0390e+03
3.9490e+03 2.3660e+03 2.9100e+02 1.0480e+03 1.3260e+03 1.4370e+03
7.0000e+02 5.3400e+02 1.6570e+03 4.0300e+02 1.3130e+03 1.1670e+03
1.4500e+02 8.4100e+02 6.3600e+02 7.5100e+02 1.5730e+03 1.5410e+03
1.3170e+03 5.4680e+03 2.3100e+02 4.7800e+02 6.2200e+02 2.9150e+03
1.7200e+03 1.6500e+02 8.5800e+02 5.2700e+02 7.6300e+02 1.5900e+03
7.1000e+02 4.3300e+02 1.6800e+02 1.6860e+03 3.3600e+02 8.4900e+02
1.8640e+03 2.0760e+03 1.1300e+02 1.1850e+03 2.9740e+03 9.7600e+02
8.3500e+02 3.3900e+02 8.4500e+02 1.2700e+02 1.2253e+04 2.3410e+03
5.2000e+02 1.7910e+03 1.5010e+03 6.4900e+02 3.0400e+02 1.1470e+03
1.1190e+03 2.6140e+03 1.4000e+02 8.1300e+02 6.3500e+02 1.3730e+03
6.9600e+02 2.3980e+03 1.8980e+03 2.3830e+03 1.0000e+03 2.0100e+03
7.8100e+03 1.2360e+03 4.0580e+03 1.1790e+03 9.0900e+02 8.6000e+02
9.0100e+02 1.7200e+02 1.6600e+03 1.1640e+03 1.0150e+03 1.5830e+03
1.0410e+03 1.7700e+02 2.0900e+03 1.2320e+03 8.5100e+02 1.9670e+03
6.3100e+02 6.8200e+02 1.6900e+03 1.7500e+02 1.2110e+03 2.3300e+03
1.6420e+03 8.4400e+02 1.9110e+03 2.0800e+03 4.7610e+03 4.0000e+01
6.5100e+02 7.7300e+02 7.6600e+02 8.7900e+02 1.7900e+02 8.7400e+02

```
 1.0280e+03 2.7300e+02 2.4010e+03 2.8850e+03 1.9070e+03 1.0500e+03
 1.9600e+02 2.2900e+02 2.9700e+02 4.5800e+02 1.5400e+03 3.9440e+03
 5.3600e+02 4.3900e+02 5.7500e+02 3.8600e+02 2.9560e+03 4.2500e+02
 9.1000e+02 1.1210e+03 1.1040e+03 6.4300e+02 7.0200e+02 1.4680e+03
 4.1600e+02 5.7200e+02 1.8260e+03 1.3090e+03 1.3500e+02 1.1400e+02
 1.2880e+03 4.8100e+02 1.0250e+03 5.8000e+01 7.2800e+02 4.1270e+03
 3.0500e+02 2.6210e+03 3.7900e+02 1.9120e+03 2.3900e+02 3.4600e+02
 4.1900e+02 3.4900e+02 6.4400e+02 3.9580e+03 8.1700e+02 4.4700e+02
 1.0705e+04 6.0400e+02 6.2400e+02 6.8600e+02 3.3100e+02 9.6000e+02
 4.9200e+02 3.4200e+02 2.6200e+02 2.7780e+03 5.9400e+02 1.6290e+03
 2.5660e+03 5.7600e+02 4.6000e+02 8.7600e+02 1.0080e+03 2.2670e+03
 1.9290e+03 2.7700e+02 6.0200e+02 1.7920e+03 4.0400e+02 7.3900e+02
 5.4600e+02 1.6410e+03 1.5360e+03 6.3000e+01 1.0830e+03 6.3310e+03
 9.7700e+02 2.0360e+03 2.9000e+02 3.9600e+02 1.7850e+03 7.9700e+02
 6.2210e+03 1.9080e+03 1.3640e+03 4.5600e+02 1.7290e+03 1.1690e+03
 9.2800e+02 3.0800e+02 2.2000e+02 2.3700e+03 1.0630e+03 1.6400e+02
 1.4010e+03 7.8200e+02 2.8600e+02 6.3700e+02 8.5500e+02 1.3630e+03
 3.5400e+02 2.0000e+02 4.1200e+02 7.1480e+03 6.2000e+02 1.2000e+02
 1.5980e+03 1.2910e+03 4.4070e+03 5.6300e+02 1.1050e+03 2.3492e+04
 8.1600e+02 4.0900e+02 1.9700e+03 2.0400e+02 1.3570e+03 3.3800e+02
 1.0720e+03 1.0690e+03 7.5800e+02 1.3530e+03 7.8400e+02 1.8400e+02
 1.2960e+03 7.9200e+02 6.8400e+02 8.0700e+02 4.8300e+02 1.3380e+03
 9.1800e+02 2.0500e+02 5.1000e+02 1.8720e+03]
Count of Unique Values: 1000

Spa
[   0.  549. 6715. ... 2868. 1107. 1643.]
Count of Unique Values: 1162

VRDeck
[   0.   44.   49. ... 1164.  971. 3235.]
Count of Unique Values: 1125

Transported
[False  True]
Count of Unique Values: 2
```

[13]:
```
X.head(5)
```

[13]:

| | HomePlanet | CryoSleep | Cabin | Destination | Age | VIP | RoomService | FoodCourt | ShoppingMall | Spa | VRDeck | Transported |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Europa | False | B/0/P | TRAPPIST-1e | 39.0 | False | 0.0 | | | | | |
| 1 | Earth | False | F/0/S | TRAPPIST-1e | 24.0 | False | 109.0 | | | | | |
| 2 | Europa | False | A/0/S | TRAPPIST-1e | 58.0 | True | 43.0 | | | | | |
| 3 | Europa | False | A/0/S | TRAPPIST-1e | 33.0 | False | 0.0 | | | | | |
| 4 | Earth | False | F/1/S | TRAPPIST-1e | 16.0 | False | 303.0 | | | | | |

```
0        0.0           0.0     0.0     0.0        False
1        9.0          25.0   549.0    44.0         True
2     3576.0           0.0  6715.0    49.0        False
3     1283.0         371.0  3329.0   193.0        False
4       70.0         151.0   565.0     2.0         True
```

#### 4.1.1 HomePlanet, CryoSleep, Destination, and VIP Columns

- Filtering out all the catagorical columns having `few` number of catagories.
- Converting catagorical columns into one-hot encoded columns, using `.get_dummies`.

[14]:
```
# One-hot encoding
X = pd.get_dummies(X, columns=["HomePlanet", "CryoSleep", "Destination", "VIP"],␣
↪dtype=float)
```

#### 4.1.2 Cabin Column

- Filtering out all the catagorical columns having `moderate` amount of catagories.
- Spliting the `Cabin` column values to `deck`,`num` and `side` column, to gain better information of each element, using `.split()`.
- Converting only `deck` and `side` columns into one-hot encoded form, using `.get_dummies()`, because `num` column is a `numerical` column.

[15]:
```
cabin_col = X["Cabin"]
cabin_col
```

[15]:
```
0          B/0/P
1          F/0/S
2          A/0/S
3          A/0/S
4          F/1/S
           ...
8688      A/98/P
8689    G/1499/S
8690    G/1500/S
8691     E/608/S
8692     E/608/S
Name: Cabin, Length: 6764, dtype: object
```

[16]:
```
# Spliting Cabin column into 'deck', 'num' and 'side'
deck, num, side = np.array([]), np.array([]), np.array([])

for every_entry in cabin_col:
    temp = str(every_entry).split("/")
    deck = np.append(deck, temp[0])
    num = np.append(num, np.int32(temp[1]))
    side = np.append(side, temp[2])
```

```
num = num.astype(np.integer)

X["deck"], X["num"], X["side"] = deck, num, side
X.drop(columns=["Cabin"], inplace=True)
```

```
/tmp/ipykernel_19/4169490143.py:10: DeprecationWarning: Converting `np.integer`
or `np.signedinteger` to a dtype is deprecated. The current result is
`np.dtype(np.int_)` which is not strictly correct. Note that the result depends
on the system. To ensure stable results use may want to use `np.int64` or
`np.int32`.
  num = num.astype(np.integer)
```

[17]:
```
# Most common value in Cabin columns based on customized method
comman_deck = str(X["deck"].value_counts().first_valid_index())
comman_num = int(np.mean(X["num"]))
comman_side = str(X["side"].value_counts().first_valid_index())
common_val = comman_deck+"/"+str(comman_num)+"/"+comman_side

print(f"The most common occurance based on our algorithm is: {common_val}")
```

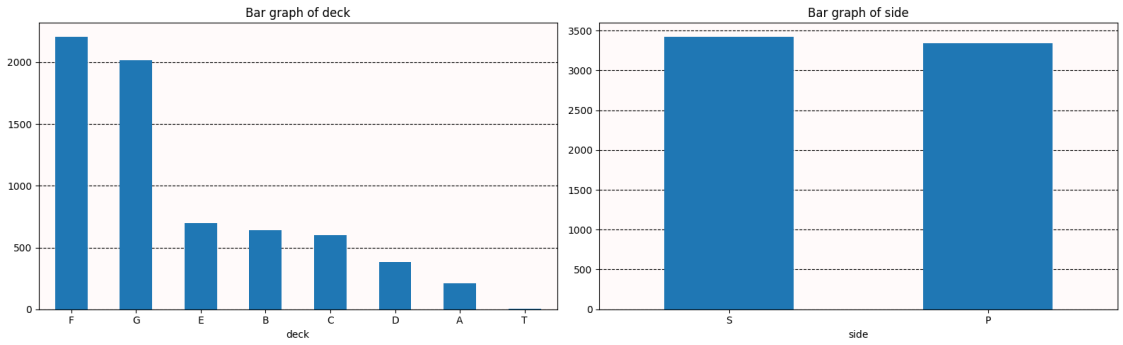
```
The most common occurance based on our algorithm is: F/598/S
```

[18]:
```
X.columns
```

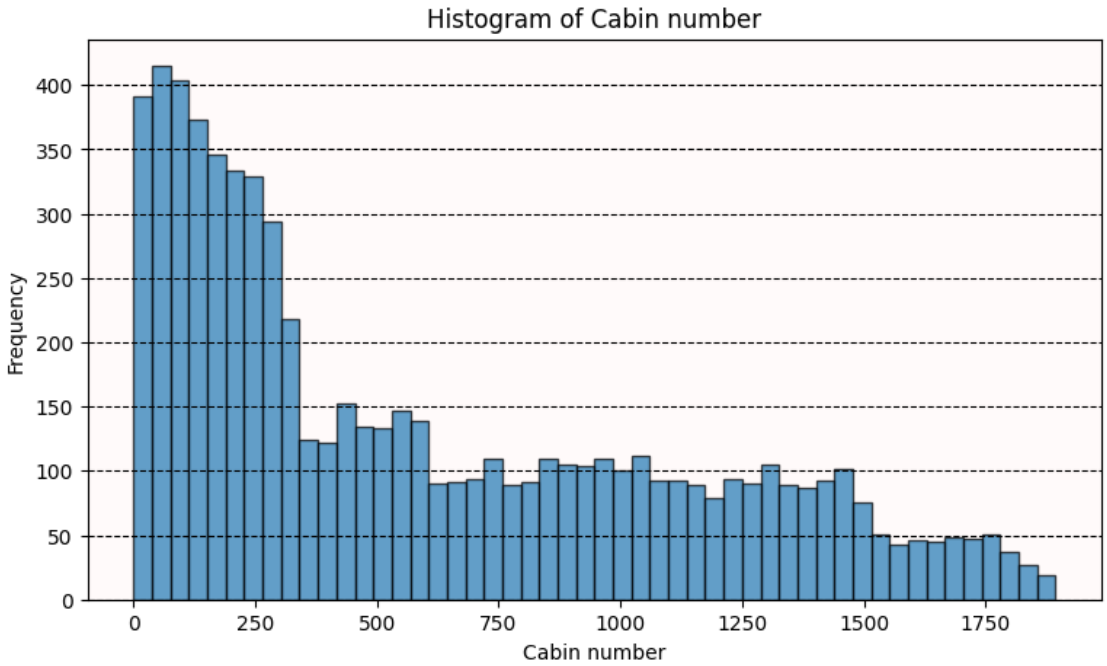
[18]:
```
Index(['Age', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck',
       'Transported', 'HomePlanet_Earth', 'HomePlanet_Europa',
       'HomePlanet_Mars', 'CryoSleep_False', 'CryoSleep_True',
       'Destination_55 Cancri e', 'Destination_PSO J318.5-22',
       'Destination_TRAPPIST-1e', 'VIP_False', 'VIP_True', 'deck', 'num',
       'side'],
      dtype='object')
```

[19]:
```
# Visualization of Catagorical columns generated by the Cabin column
cols = ["deck", "side"]

fig, ax = plt.subplots(figsize=(16,5), ncols=2)

X[cols[0]].value_counts().plot(kind='bar', ax=ax[0])
ax[0].set_title(f"Bar graph of {cols[0]}")
ax[0].tick_params(axis='x', rotation=0)
ax[0].yaxis.grid(linestyle='--', color='k')
ax[0].set_facecolor(color="snow")
ax[0].set_axisbelow(True)

X[cols[1]].value_counts().plot(kind='bar', ax=ax[1])
ax[1].set_title(f"Bar graph of {cols[1]}")
ax[1].tick_params(axis='x', rotation=0)
ax[1].yaxis.grid(linestyle='--', color='k')
```

```
ax[1].set_facecolor(color="snow")
ax[1].set_axisbelow(True)

plt.tight_layout()
```




[20]:
```
# Visualization of Numerical columns generated by the Cabin column
ax = X["num"].hist(figsize=(9, 5), bins=50, xlabelsize=10, ylabelsize=10,␣
 ↪grid=False, edgecolor='black', alpha=0.7)
ax.yaxis.grid(linestyle='--', color='k')
ax.set_xlabel("Cabin number")
ax.set_ylabel("Frequency")
ax.set_title("Histogram of Cabin number")
ax.set_facecolor(color="snow");
```



[21]:
```
# Converting Catagorical columns to one-hot encoding form generated from Cabin
X = pd.get_dummies(X, columns=["deck", "side"], dtype=float)
X.head()
```

[21]:

| | Age | RoomService | FoodCourt | ShoppingMall | Spa | VRDeck | Transported | HomePlanet_Earth | HomePlanet_Europa | HomePlanet_Mars | ... | deck_A | deck_B | deck_C | deck_D | deck_E | deck_F | deck_G | deck_T | side_P | side_S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 39.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | False | 0.0 | 1.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| 1 | 24.0 | 109.0 | 9.0 | 25.0 | 549.0 | 44.0 | True | 1.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 2 | 58.0 | 43.0 | 3576.0 | 0.0 | 6715.0 | 49.0 | False | 0.0 | 1.0 | 0.0 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 3 | 33.0 | 0.0 | 1283.0 | 371.0 | 3329.0 | 193.0 | False | 0.0 | 1.0 | 0.0 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 4 | 16.0 | 303.0 | 70.0 | 151.0 | 565.0 | 2.0 | True | 1.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 |

[5 rows x 28 columns]

### 4.1.3 `Transported` Column

- Converting the target column `Transported` from `boolean` to `int`.

[22]:
```
# Converting target column, from boolean to int
X["Transported"] = X["Transported"].astype(np.int32)

X.head(5)
```

[22]:

| | Age | RoomService | FoodCourt | ShoppingMall | Spa | VRDeck | Transported | HomePlanet_Earth | HomePlanet_Europa | HomePlanet_Mars | ... | deck_A | deck_B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 39.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0.0 | 1.0 | 0.0 | ... | 0.0 | 1.0 |
| 1 | 24.0 | 109.0 | 9.0 | 25.0 | 549.0 | 44.0 | 1 | 1.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |
| 2 | 58.0 | 43.0 | 3576.0 | 0.0 | 6715.0 | 49.0 | 0 | 0.0 | 1.0 | 0.0 | ... | 1.0 | 0.0 |
| 3 | 33.0 | 0.0 | 1283.0 | 371.0 | 3329.0 | 193.0 | 0 | 0.0 | 1.0 | 0.0 | ... | 1.0 | 0.0 |
| 4 | 16.0 | 303.0 | 70.0 | 151.0 | 565.0 | 2.0 | 1 | 1.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |

```
2              0.0               1.0              0.0  ...      1.0     0.0
3              0.0               1.0              0.0  ...      1.0     0.0
4              1.0               0.0              0.0  ...      0.0     0.0

   deck_C  deck_D  deck_E  deck_F  deck_G  deck_T  side_P  side_S
0     0.0     0.0     0.0     0.0     0.0     0.0     1.0     0.0
1     0.0     0.0     0.0     1.0     0.0     0.0     0.0     1.0
2     0.0     0.0     0.0     0.0     0.0     0.0     0.0     1.0
3     0.0     0.0     0.0     0.0     0.0     0.0     0.0     1.0
4     0.0     0.0     0.0     1.0     0.0     0.0     0.0     1.0

[5 rows x 28 columns]
```

### 4.2 (ii). Data Spliting

- Seperating the target column `Transported` from `X_train`, to `y_train`, and dropping from the `X_train`.

[23]:
```
# Seperating the target column from X_train
y_train = X["Transported"]
X_train = X.drop(columns=["Transported"])

X_train.head(5)
```

[23]:
```
    Age  RoomService  FoodCourt  ShoppingMall     Spa  VRDeck  \
0  39.0          0.0        0.0           0.0     0.0     0.0
1  24.0        109.0        9.0          25.0   549.0    44.0
2  58.0         43.0     3576.0           0.0  6715.0    49.0
3  33.0          0.0     1283.0         371.0  3329.0   193.0
4  16.0        303.0       70.0         151.0   565.0     2.0

   HomePlanet_Earth  HomePlanet_Europa  HomePlanet_Mars  CryoSleep_False  ...  \
0               0.0                1.0              0.0              1.0  ...
1               1.0                0.0              0.0              1.0  ...
2               0.0                1.0              0.0              1.0  ...
3               0.0                1.0              0.0              1.0  ...
4               1.0                0.0              0.0              1.0  ...

   deck_A  deck_B  deck_C  deck_D  deck_E  deck_F  deck_G  deck_T  side_P  \
0     0.0     1.0     0.0     0.0     0.0     0.0     0.0     0.0     1.0
1     0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0     0.0
2     1.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
3     1.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
4     0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0     0.0

   side_S
0     0.0
```

```
1     1.0
2     1.0
3     1.0
4     1.0

[5 rows x 27 columns]
```

### 4.3 (iii). Data Scaling

- Applying `MinMaxScaler` to scale down all the vlaues in the `X_train`, which will help in modeling and fitting the model efficiently.

[24]:
```
cont_col = ["Age", "RoomService", "FoodCourt", "ShoppingMall", "Spa", "VRDeck",␣
 ↪"num"]
scaler = MinMaxScaler()
X_train[cont_col] = scaler.fit_transform(X_train[cont_col])

X_train.head(5)
```

[24]:
```
        Age  RoomService  FoodCourt  ShoppingMall       Spa    VRDeck  \
0  0.493671     0.000000   0.000000      0.000000  0.000000  0.000000
1  0.303797     0.010988   0.000302      0.001064  0.024500  0.002164
2  0.734177     0.004335   0.119948      0.000000  0.299670  0.002410
3  0.417722     0.000000   0.043035      0.015793  0.148563  0.009491
4  0.202532     0.030544   0.002348      0.006428  0.025214  0.000098

   HomePlanet_Earth  HomePlanet_Europa  HomePlanet_Mars  CryoSleep_False  ...  \
0               0.0                1.0              0.0              1.0  ...
1               1.0                0.0              0.0              1.0  ...
2               0.0                1.0              0.0              1.0  ...
3               0.0                1.0              0.0              1.0  ...
4               1.0                0.0              0.0              1.0  ...

   deck_A  deck_B  deck_C  deck_D  deck_E  deck_F  deck_G  deck_T  side_P  \
0     0.0     1.0     0.0     0.0     0.0     0.0     0.0     0.0     1.0
1     0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0     0.0
2     1.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
3     1.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
4     0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0     0.0

   side_S
0     0.0
1     1.0
2     1.0
3     1.0
4     1.0
```

[5 rows x 27 columns]

### 4.4 (iv.) Test Transform

- Reading `test.csv` and storing the data as a dataframe`X_test`.
- Calculating total null values in each column, from `X_test`.
- Transforming `X_test`, such that there are no records (`PassengerId`) to be droped.
- Data imputation has been done by `.ffill()`.

[25]:
```
# Read train Data file from the path
X_test = pd.read_csv("/kaggle/input/spaceship-titanic/test.csv")

passenger_Id = X_test["PassengerId"]
X_test.drop(columns=["PassengerId"], inplace=True)

X_test.head()
```

[25]:

| | HomePlanet | CryoSleep | Cabin | Destination | Age | VIP | RoomService | FoodCourt | ShoppingMall | Spa | VRDeck | Name |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Earth | True | G/3/S | TRAPPIST-1e | 27.0 | False | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | Nelly Carsoning |
| 1 | Earth | False | F/4/S | TRAPPIST-1e | 19.0 | False | 0.0 | 9.0 | 0.0 | 2823.0 | 0.0 | Lerome Peckers |
| 2 | Europa | True | C/0/S | 55 Cancri e | 31.0 | False | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | Sabih Unhearfus |
| 3 | Europa | False | C/1/S | TRAPPIST-1e | 38.0 | False | 0.0 | 6652.0 | 0.0 | 181.0 | 585.0 | Meratz Caltilter |
| 4 | Earth | False | F/5/S | TRAPPIST-1e | 20.0 | False | 10.0 | 0.0 | 635.0 | 0.0 | 0.0 | Brence Harperez |

[26]:
```
X_test.isnull().sum()
```

[26]:
```
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
```

[27]:
```
def test_transformer(df_X):

    # Removing unwanted columns and NaN valued records
    df_X.drop(columns=["Name"], inplace=True)

    # Replacing None cells to NaN values
    df_X.fillna(value=np.nan, inplace=True)

    # Converting catagorical columns to One-Hot Encoded form
    cols = ["HomePlanet", "CryoSleep", "Destination", "VIP"]
    df_X = pd.get_dummies(df_X, columns=cols, dtype=float)

    df_X.ffill(inplace=True)

    # Transforming "Cabin" column
    cabin_col = df_X["Cabin"]
    deck, num, side = np.array([]), np.array([]), np.array([])

    for every_entry in cabin_col:
        temp = str(every_entry).split("/")
        deck = np.append(deck, temp[0])
        num = np.append(num, np.int32(temp[1]))
        side = np.append(side, temp[2])

    num = num.astype(np.integer)

    df_X["deck"], df_X["num"], df_X["side"] = deck, num, side
    df_X.drop(columns=["Cabin"], inplace=True)

    # Converting catagroical columns to One-Hot Encoded form
    df_X = pd.get_dummies(df_X, columns=["deck", "side"], dtype=float)

    return df_X
```

[28]:
```
X_test = test_transformer(X_test)

# Applying scaler transform
X_test[cont_col] = scaler.transform(X_test[cont_col])
```

```
/tmp/ipykernel_19/3254332854.py:25: DeprecationWarning: Converting `np.integer`
or `np.signedinteger` to a dtype is deprecated. The current result is
`np.dtype(np.int_)` which is not strictly correct. Note that the result depends
on the system. To ensure stable results use may want to use `np.int64` or
`np.int32`.
  num = num.astype(np.integer)
```

[29]:
```
X_test
```

[29]:

| | Age | RoomService | FoodCourt | ShoppingMall | Spa | VRDeck | \ |
|---|---|---|---|---|---|---|---|
| 0 | 0.341772 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | |
| 1 | 0.240506 | 0.000000 | 0.000302 | 0.000000 | 0.125982 | 0.000000 | |
| 2 | 0.392405 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | |
| 3 | 0.481013 | 0.000000 | 0.223124 | 0.000000 | 0.008077 | 0.028767 | |
| 4 | 0.253165 | 0.001008 | 0.000000 | 0.027030 | 0.000000 | 0.000000 | |
| ... | ... | ... | ... | ... | ... | ... | |
| 4272 | 0.430380 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | |
| 4273 | 0.531646 | 0.000000 | 0.028410 | 0.000724 | 0.000446 | 0.007081 | |
| 4274 | 0.531646 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | |
| 4275 | 0.531646 | 0.000000 | 0.089894 | 0.000000 | 0.000000 | 0.025718 | |
| 4276 | 0.544304 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | |

| | HomePlanet_Earth | HomePlanet_Europa | HomePlanet_Mars | CryoSleep_False | \ |
|---|---|---|---|---|---|
| 0 | 1.0 | 0.0 | 0.0 | 0.0 | |
| 1 | 1.0 | 0.0 | 0.0 | 1.0 | |
| 2 | 0.0 | 1.0 | 0.0 | 0.0 | |
| 3 | 0.0 | 1.0 | 0.0 | 1.0 | |
| 4 | 1.0 | 0.0 | 0.0 | 1.0 | |
| ... | ... | ... | ... | ... | |
| 4272 | 1.0 | 0.0 | 0.0 | 0.0 | |
| 4273 | 1.0 | 0.0 | 0.0 | 1.0 | |
| 4274 | 0.0 | 0.0 | 1.0 | 0.0 | |
| 4275 | 0.0 | 1.0 | 0.0 | 1.0 | |
| 4276 | 1.0 | 0.0 | 0.0 | 0.0 | |

| | ... | deck_A | deck_B | deck_C | deck_D | deck_E | deck_F | deck_G | deck_T | \ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| 1 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | |
| 2 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 3 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 4 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 4272 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| 4273 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |
| 4274 | ... | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 4275 | ... | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 4276 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | |

| | side_P | side_S |
|---|---|---|
| 0 | 0.0 | 1.0 |
| 1 | 0.0 | 1.0 |
| 2 | 0.0 | 1.0 |
| 3 | 0.0 | 1.0 |
| 4 | 0.0 | 1.0 |
| ... | ... | ... |
| 4272 | 0.0 | 1.0 |

```
4273     0.0     1.0
4274     1.0     0.0
4275     1.0     0.0
4276     0.0     1.0

[4277 rows x 27 columns]
```

# 5 Modeling

- For model selection, the best model to trian with will be `RandomForestClassifier`. ## Random Forest Classifier

[30]:
```
rfc = RandomForestClassifier(max_depth=14, random_state=42)
rfc.fit(X_train, y_train)
```

[30]:
```
RandomForestClassifier(max_depth=14, random_state=42)
```

[31]:
```
y_pred = rfc.predict(X_test)
y_pred
```

[31]:
```
array([0, 0, 1, ..., 1, 1, 0], dtype=int32)
```

# 6 Creating Submition File

- Converting `y_pred`'s datatype form `int32` to `boolean`.
- Storing `PassengerId` and `Transported` into a dictionary.

[32]:
```
y_pred = y_pred.astype(bool)
y_pred
```

[32]:
```
array([False, False,  True, ...,  True,  True, False])
```

[33]:
```
passenger_Id
```

[33]:
```
0       0013_01
1       0018_01
2       0019_01
3       0021_01
4       0023_01
         ...
4272    9266_02
4273    9269_01
4274    9271_01
4275    9273_01
4276    9277_01
Name: PassengerId, Length: 4277, dtype: object
```

[34]:
```
y_pred.shape
```

[34]:
```
(4277,)
```

[35]:
```
data = {'PassengerId': passenger_Id,
        'Transported': y_pred}

submission = pd.DataFrame(data)
submission.to_csv('/kaggle/working/submission.csv', index=False)
submission
```

[35]:

| | PassengerId | Transported |
|---|---|---|
| 0 | 0013_01 | False |
| 1 | 0018_01 | False |
| 2 | 0019_01 | True |
| 3 | 0021_01 | True |
| 4 | 0023_01 | True |
| ... | ... | ... |
| 4272 | 9266_02 | True |
| 4273 | 9269_01 | True |
| 4274 | 9271_01 | True |
| 4275 | 9273_01 | True |
| 4276 | 9277_01 | False |

[4277 rows x 2 columns]

[ ]: