1. Locate open source data from the web. Dataset - Spaceship Titanic on https://www.kaggle.com

2. Provide a clear description of the data and its source (i.e., URL of the web site). URL: https://www.kaggle.com/code/muhammadhadi13/titanic-eda-model-application/
   File train.csv - Personal records for about two-thirds (~8700) of the passengers, to be used as training data. Columns: PassengerId,HomePlanet,CryoSleep,Cabin,Destination,Age,VIP,RoomService,FoodCourt,S

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.

In [ ]:
```
# 3. Load the Dataset into the pandas data frame.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the dataset into the pandas data frame
data = pd.read_csv("train.csv")
```

4. Data Preprocessing: check for missing values in the data using pandas insult(), describe() function to get some initial statistics. Provide variable descriptions. Types of variable etc. Check the dimensions of the data frame.

In [ ]:
```
# Check for missing values in the data using pandas insult()
print(data.isnull().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
```

In [ ]:
```
# Describe the data
print(data.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
```

In [ ]:
```
# Types of variable
print(data.dtypes)
```

```
PassengerId      object
HomePlanet       object
CryoSleep        object
Cabin            object
Destination      object
Age             float64
VIP              object
RoomService     float64
FoodCourt       float64
ShoppingMall    float64
Spa             float64
VRDeck          float64
Name             object
Transported        bool
dtype: object
```

In [ ]:
```
# Check the dimensions of the data frame
print(data.shape)
```

```
(8693, 14)
```

5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions.

In [ ]:
```
# Summarize the types of variables by checking the data types
print(data.dtypes)
```

```
PassengerId      object
HomePlanet       object
CryoSleep        object
Cabin            object
Destination      object
Age             float64
VIP              object
RoomService     float64
FoodCourt       float64
ShoppingMall    float64
Spa             float64
VRDeck          float64
Name             object
Transported        bool
dtype: object
```

In [ ]:
```
# fill age with median value
data['Age'].fillna(data['Age'].median(), inplace=True)

# If variables are not in the correct data type, apply proper type conversions
print(data['Age'])
```

```
0       39.0
1       24.0
2       58.0
3       33.0
4       16.0
        ...
8688    41.0
8689    18.0
8690    26.0
8691    32.0
8692    44.0
Name: Age, Length: 8693, dtype: float64
```

In [ ]:
```
# Convert the Age column to integer
data['Age'] = data['Age'].astype(int)

# display age before and after conversion
print(data['Age'])
```

```
0       39
1       24
2       58
3       33
4       16
        ..
8688    41
8689    18
8690    26
8691    32
8692    44
Name: Age, Length: 8693, dtype: int32
```

6. Turn categorical variables into quantitative variables in Python Convert the HomePlanet column to quantitative variables display the HomePlanet column before conversion

In [ ]:
```
print(data['HomePlanet'])
```

```
0       Europa
1        Earth
2       Europa
3       Europa
4        Earth
         ...
8688    Europa
8689     Earth
8690     Earth
8691    Europa
8692    Europa
Name: HomePlanet, Length: 8693, dtype: object
```

In [ ]:
```
# Categorical data
data['HomePlanet'] = pd.Categorical(data['HomePlanet'])
data['HomePlanet'] = data['HomePlanet'].cat.codes

# display
print(data['HomePlanet'])
```

```
0       1
1       0
2       1
3       1
4       0
       ..
8688    1
8689    0
8690    0
8691    1
8692    1
Name: HomePlanet, Length: 8693, dtype: int8
```