Homework 3

Anthony Cu (PSTAT 134/234)

Homework 3

Figure 1: Star Trek ships.

For this homework assignment, we'll be working with a dataset called Spaceship Titanic. It is a simulated dataset used for a popular Kaggle competition, intended to be similar to the very famous Titanic dataset. The premise of the Spaceship Titanic data is that it is currently the year 2912. You have received a transmission from four lightyears away, sent by the Spaceship Titanic, which was launched a month ago.

The Titanic set out with about 13,000 passengers who were emigrating from our solar system to three newly habitable exoplanets. However, it collided with a spacetime anomaly hidden in a dust cloud, and as a result, although the ship remained intact, half of the passengers on board were transported to an alternate dimension. Your challenge is to predict which

passengers were transported, using records recovered from the spaceship's damaged computer system.

The dataset is provided in `/data`, along with a codebook describing each variable. You should read the dataset into your preferred coding language (R or Python) and familiarize yourself with each variable.

We will use this dataset for the purposes of practicing our data visualization and feature engineering skills.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re
from sklearn.preprocessing import StandardScaler

titanic = pd.read_csv('~/pstat-134-234/Homework/Homework 3/data/spaceship_titanic.csv')
```

Exercise 1

Which variables have missing values? What percentage of these variables is missing? What percentage of the overall dataset is missing?

```
# titanic.isnull().sum(), titanic.isna().sum() returns the same

for col in titanic.columns.tolist():
  print('{} column missing values: {}'.format(col, titanic[col].isnull().sum()))
```

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

All variables (except for PassengerID and Transported) have missing values. I searched for all null (and NA) values in the columns and calculated the total of these entries. I print the missing percentage of each of the variables:

```
for w in titanic.columns:
  missing_prop = titanic[w].isnull().sum() / len(titanic)
  print(f"{w}: {missing_prop:.2%}")
```

```
PassengerId: 0.00%
HomePlanet: 2.31%
CryoSleep: 2.50%
Cabin: 2.29%
Destination: 2.09%
Age: 2.06%
VIP: 2.34%
RoomService: 2.08%
FoodCourt: 2.11%
ShoppingMall: 2.39%
Spa: 2.11%
VRDeck: 2.16%
Name: 2.30%
Transported: 0.00%
```

```
((titanic.isnull().sum()).sum() / titanic.size)
```

```
np.float64(0.019095824226389047)
```

In total of the entire dataframe, about 1.91% of the total data is missing.

Exercise 2

Use mode imputation to fill in any missing values of `home_planet`, `cryo_sleep`, `destination`, and `vip`. Drop any observations with a missing value of `cabin` (there are too many possible values).

Use median imputation to fill in any missing values of `age`. Rather than imputing with the overall mean of `age`, impute with the median age of the corresponding `vip` group. (For example, if someone who is a VIP is missing their age, replace their missing age value with the median age of all passengers who are **also** VIPs).

For passengers missing any of the expenditure variables (`room_service`, `food_court`, `shopping_mall`, `spa`, or `vr_deck`), handle them in this way:

- If all their observed expenditure values are 0, **or** if they are in cryo-sleep, replace their missing value(s) with 0.
- For the remaining missing expenditure values, use mean imputation.

```
titanic_cleaned = titanic.copy()

# mode imputation
for i in ['HomePlanet', 'CryoSleep', 'Destination', 'VIP']:
  titanic_cleaned[i] = titanic_cleaned[i].fillna(titanic_cleaned[i].mode()[0])
```

```
<string>:4: FutureWarning: Downcasting object dtype arrays on .fillna, .ffill, .bfill is depr
```

```
# drop observations that are missing values in cabin
titanic_cleaned = titanic_cleaned.dropna(subset=['Cabin'])

# median imputation
age_medians = titanic_cleaned.groupby(['VIP'])['Age'].transform(lambda x: x.fillna(x.median()

titanic_cleaned['Age'] = titanic_cleaned['Age'].fillna(age_medians)

# alternate approach (not vectorized function)
# median_VIP = titanic_cleaned[titanic_cleaned['VIP'] == True]['Age'].median()
# median_no_VIP = titanic_cleaned[titanic_cleaned['VIP'] == False]['Age'].median()
#
# titanic_cleaned.loc[(titanic_cleaned['VIP'] == True) & (titanic_cleaned['Age'].isnull()),
# titanic_cleaned.loc[(titanic_cleaned['VIP'] == False) & (titanic_cleaned['Age'].isnull()),

# missing by expenditure values

# to check if any of these observations are missing or if any of these observations equal 0 (
expenditures = ['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']
rows_to_0 = np.all((titanic_cleaned[expenditures] == 0) | titanic_cleaned[expenditures].isnul

# impute the entires with 0s
titanic_cleaned.loc[rows_to_0, expenditures] = 0

# for remaining mean imputation (not for condition)
for i in ['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']:
  titanic_cleaned[i] = titanic_cleaned[i].fillna(titanic_cleaned[i].mean())

# we check that we have successfully filled in missing values
titanic_cleaned.isnull().sum()
```

```
PassengerId       0
HomePlanet        0
CryoSleep         0
Cabin             0
Destination       0
Age               0
VIP               0
RoomService       0
FoodCourt         0
ShoppingMall      0
Spa               0
VRDeck            0
Name            198
Transported       0
dtype: int64
```

Exercise 3

What are the proportions of both levels of the outcome variable, `transported`, in the dataset?

```
transport_levels = titanic_cleaned['Transported'].value_counts()

transport_levels / transport_levels.sum()
```

```
Transported
True     0.50365
False    0.49635
Name: count, dtype: float64
```

50.37% of the variables are transported, while 49.63% of the variables are not transported.

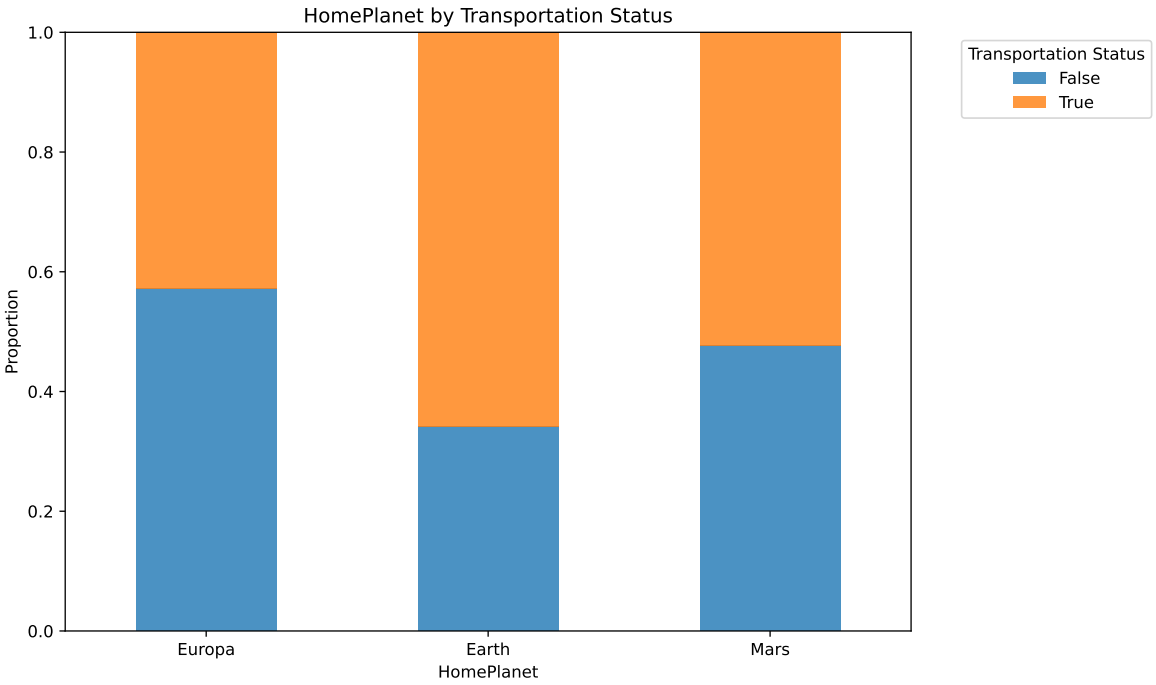
Exercise 4

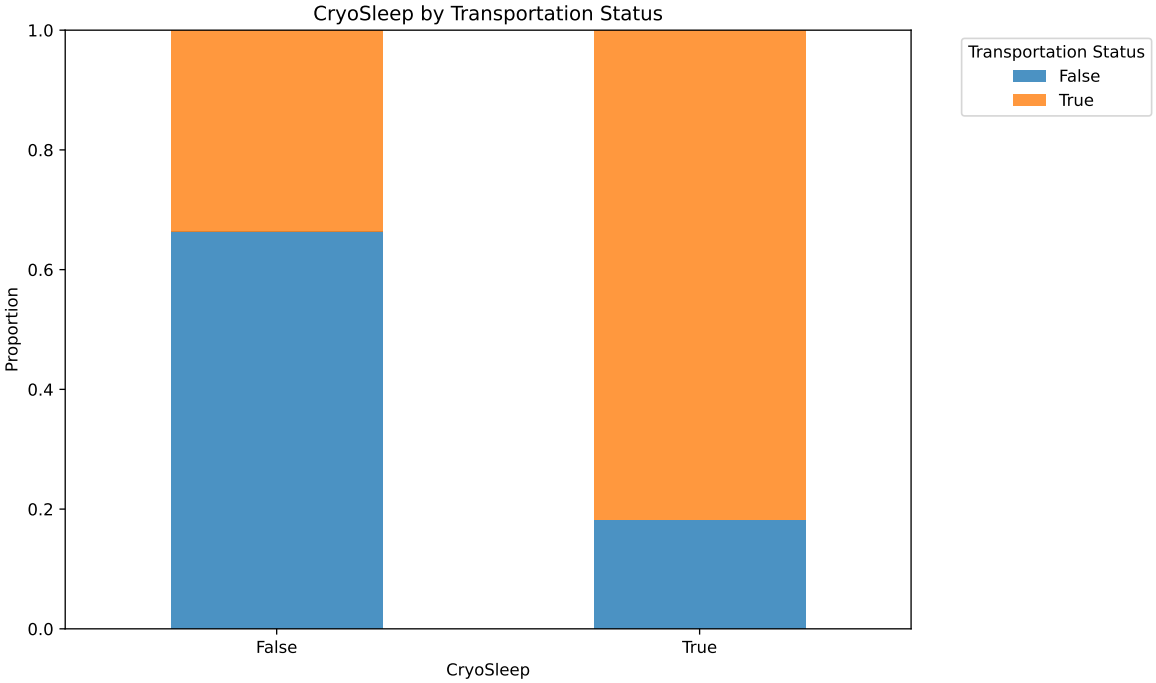
Make proportion stacked bar charts of each of the following. Describe what patterns, if any, you observe.

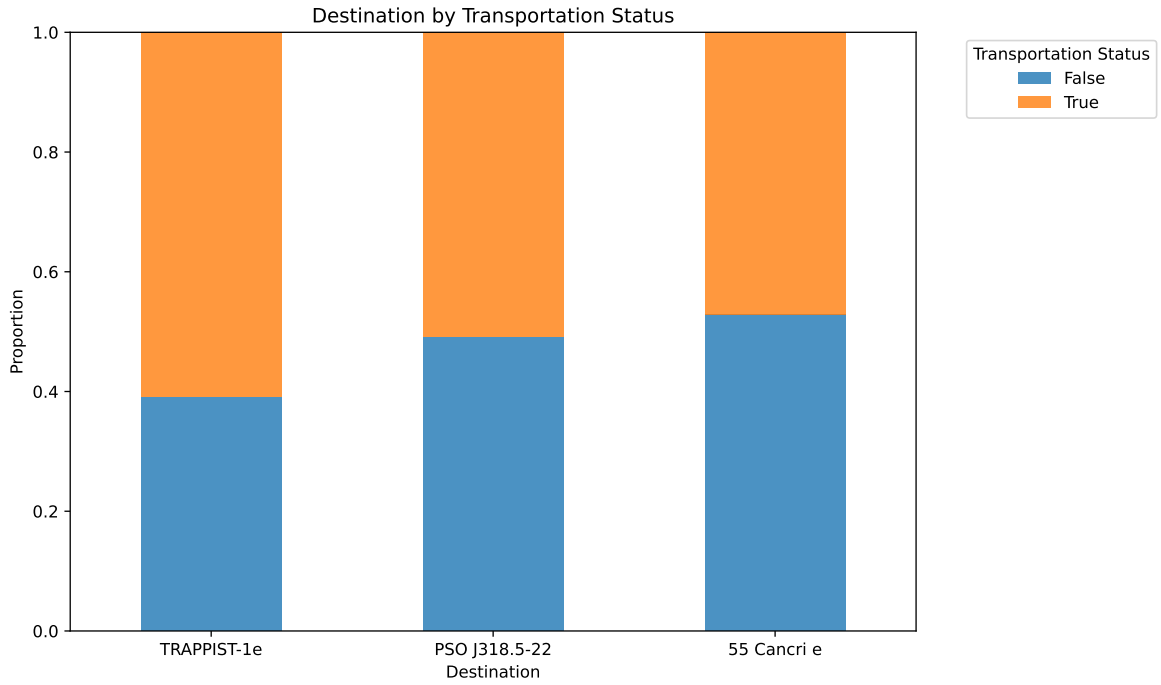
1. `home_planet` and `transported`
2. `cryo_sleep` and `transported`

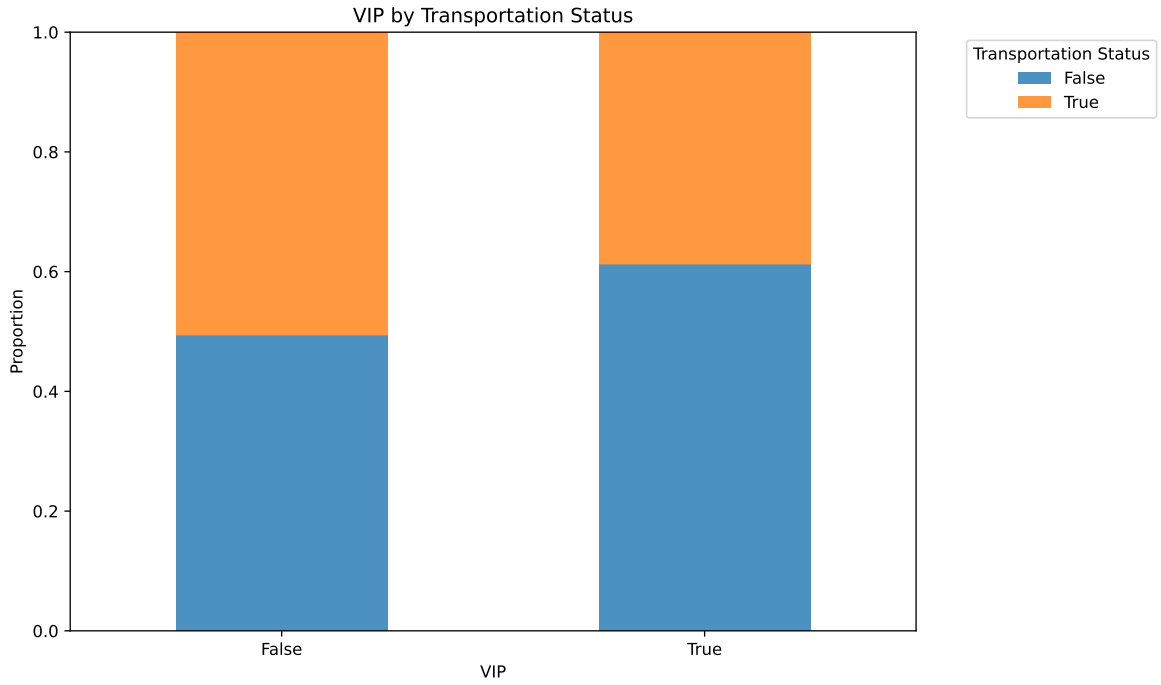
3. `destination` and `transported`
4. `vip` and `transported`

```
# proportion stacked bar charts of transportation status filled by classes
for i in ['HomePlanet', 'CryoSleep', 'Destination', 'VIP']:
  # Calculate counts of each class by home planet
  count_data = titanic_cleaned.groupby([i, 'Transported']).size().reset_index(name='count')

  # Calculate the total counts for each home planet
  total_counts = count_data.groupby(i)['count'].transform('sum')

  # Calculate the proportion
  count_data['proportion'] = count_data['count'] / total_counts

  # Create a pivot table for better plotting
  pivot_data = count_data.pivot(index=i, columns='Transported', values='proportion').fillna(0

  # Plotting
  plt.figure(figsize=(10, 6))

  # Stack the bars
  pivot_data.plot(kind='bar', stacked=True, ax=plt.gca(), alpha=0.8)

  # Set y-axis limits
  plt.ylim(0, 1)
  plt.title(f"{i} by Transportation Status", loc='center')
  plt.xlabel(i)
  plt.xticks(np.arange(len(titanic_cleaned[i].unique())), labels=titanic_cleaned[i].unique()
  plt.ylabel("Proportion")

  # Customize the legend
  plt.legend(title="Transportation Status", bbox_to_anchor=(1.05, 1), loc='upper left')
  plt.tight_layout()

  # Show the plot
  plt.show()
  plt.close()
```


HomePlanet by Transportation Status
Transportation Status
False
True
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
Europa
Earth
Mars
HomePlanet


CryoSleep by Transportation Status
Transportation Status
False
True
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
False
True
CryoSleep

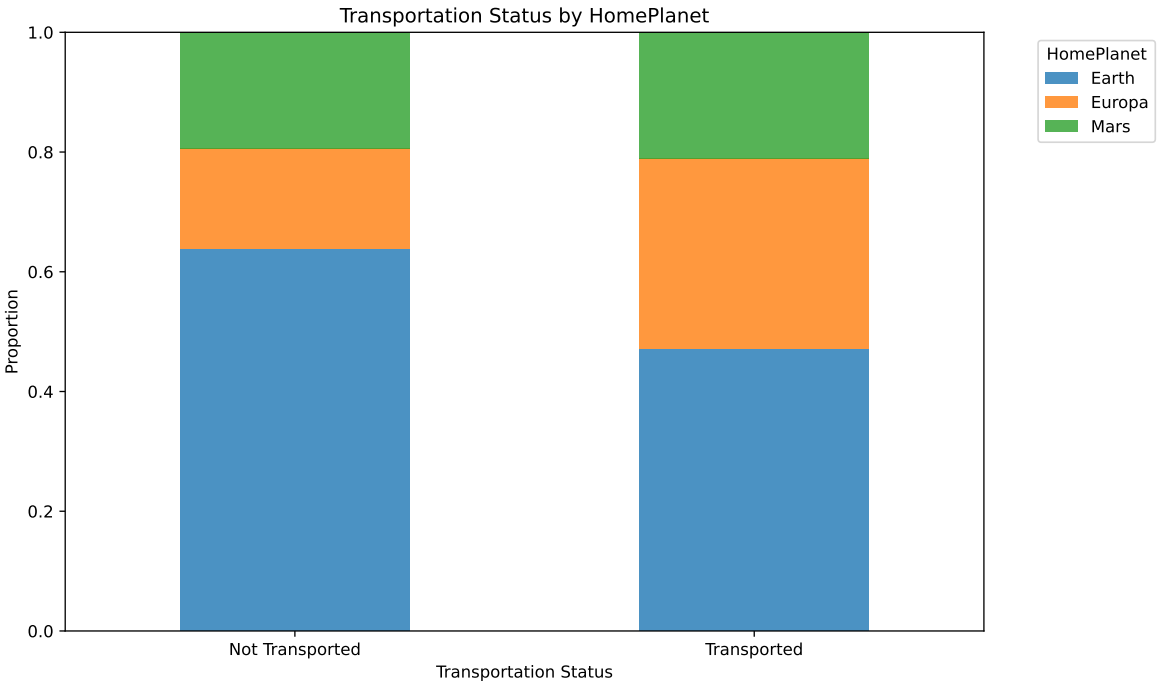


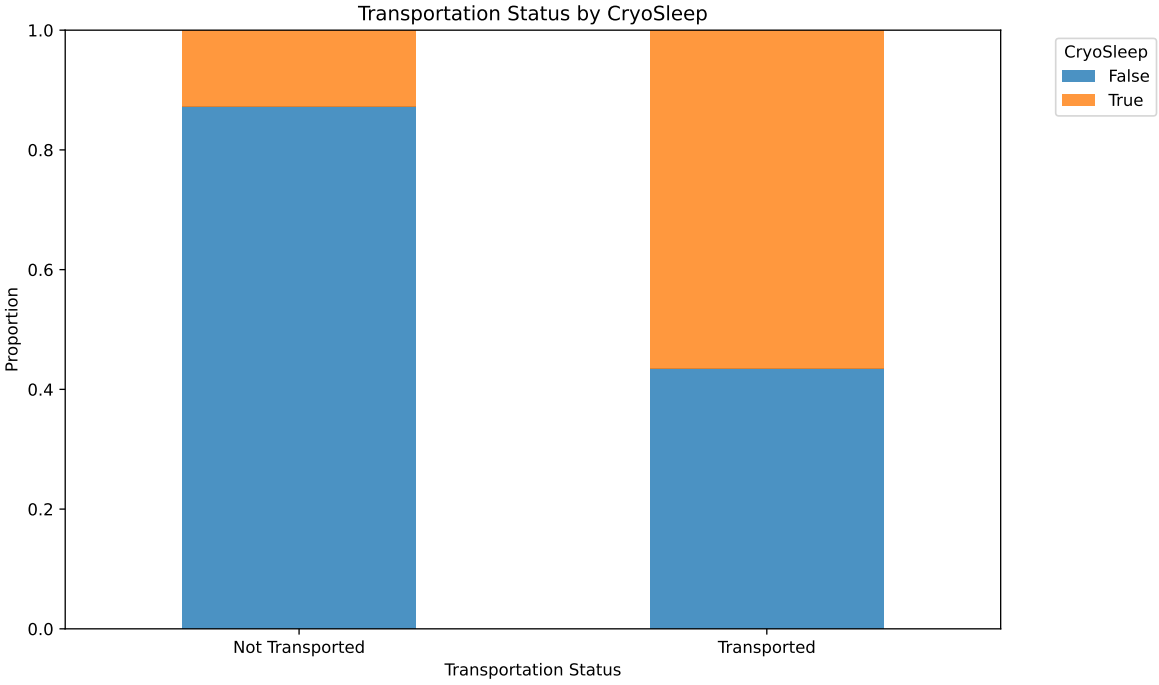


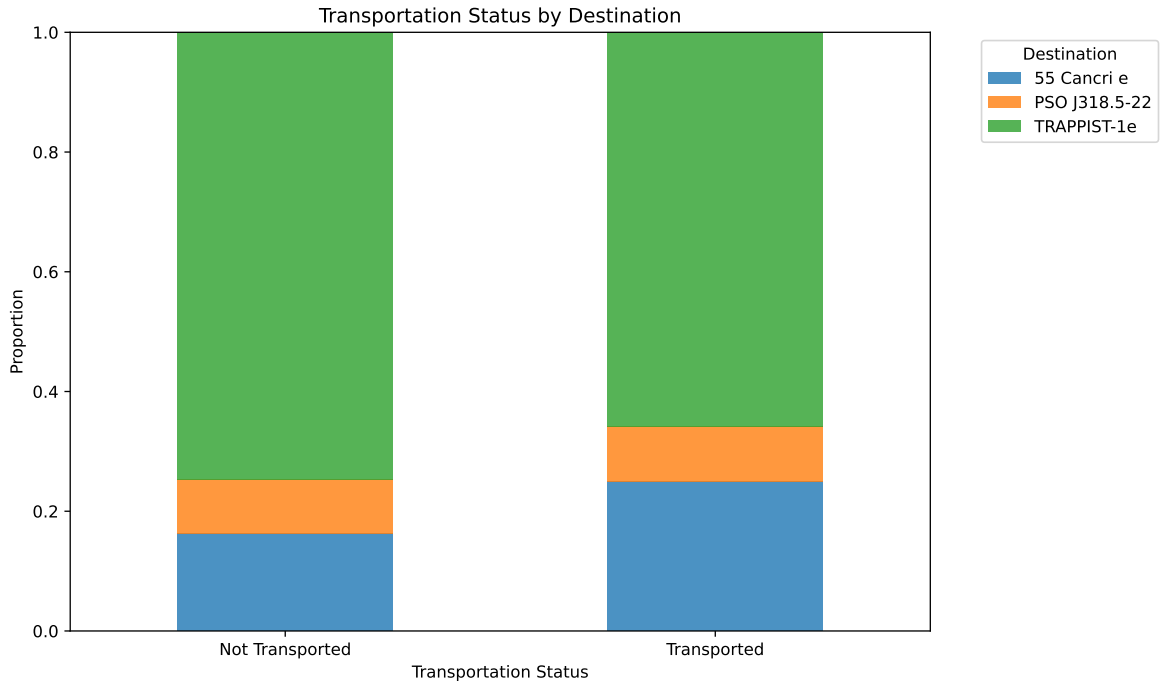
Out of all of the home planets, Earth transports the most observations- transporting over 60% of its passengers. In addition, about 80% of the passengers in CryoSleep are transported. Of

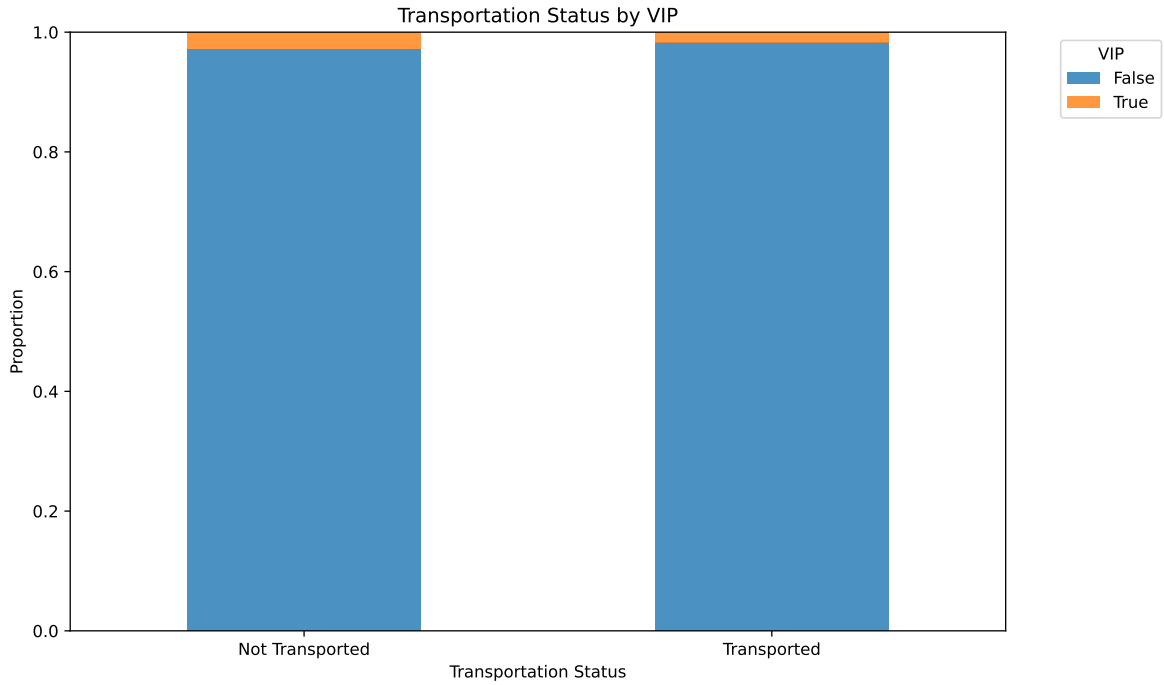
all the destinations, TRAPPIST-1e has the highest rate of successful transportations- roughly 60% of its observations; on the other hand 55 Cancri e has the lowest rate of successful transportation- about 50% of its observations. VIP status tends to cause less successful transportation, as only 40% of VIP passengers are transported. On the other hand, non-VIP members are evenly transported.

```
# proportion stacked bar charts of classes filled by transportation status
for i in ['HomePlanet', 'CryoSleep', 'Destination', 'VIP']:
  # Calculate counts of each class by home planet
  count_data = titanic_cleaned.groupby(['Transported', i]).size().reset_index(name='count')

  # Calculate the total counts for each home planet
  total_counts = count_data.groupby('Transported')['count'].transform('sum')

  # Calculate the proportion
  count_data['proportion'] = count_data['count'] / total_counts

  # Create a pivot table for better plotting
  pivot_data = count_data.pivot(index='Transported', columns=i, values='proportion').fillna(0

  # Plotting
  plt.figure(figsize=(10, 6))

  # Stack the bars
  pivot_data.plot(kind='bar', stacked=True, ax=plt.gca(), alpha=0.8)

  # Set y-axis limits
  plt.ylim(0, 1)
  plt.title(f"Transportation Status by {i}", loc = 'center')
  plt.xlabel('Transportation Status')
  plt.xticks(ticks=[0, 1], labels=['Not Transported', 'Transported'], rotation=0)
  plt.ylabel("Proportion")

  # Customize the legend
  plt.legend(title=i, bbox_to_anchor=(1.05, 1), loc='upper left')
  plt.tight_layout()

  # Show the plot
  plt.show()
  plt.close()
```


Transportation Status by HomePlanet
1.0
0.8
0.6
0.4
0.2
0.0
Proportion
Not Transported
Transported
Transportation Status
HomePlanet
Earth
Europa
Mars


Transportation Status by CryoSleep
1.0
0.8
0.6
0.4
0.2
0.0
Proportion
Not Transported
Transported
Transportation Status
CryoSleep
False
True





These plots emphasize the same interpretations- but with different lenses. Earth passengers consume most of the proportions of transported and non-transported observations. Most of

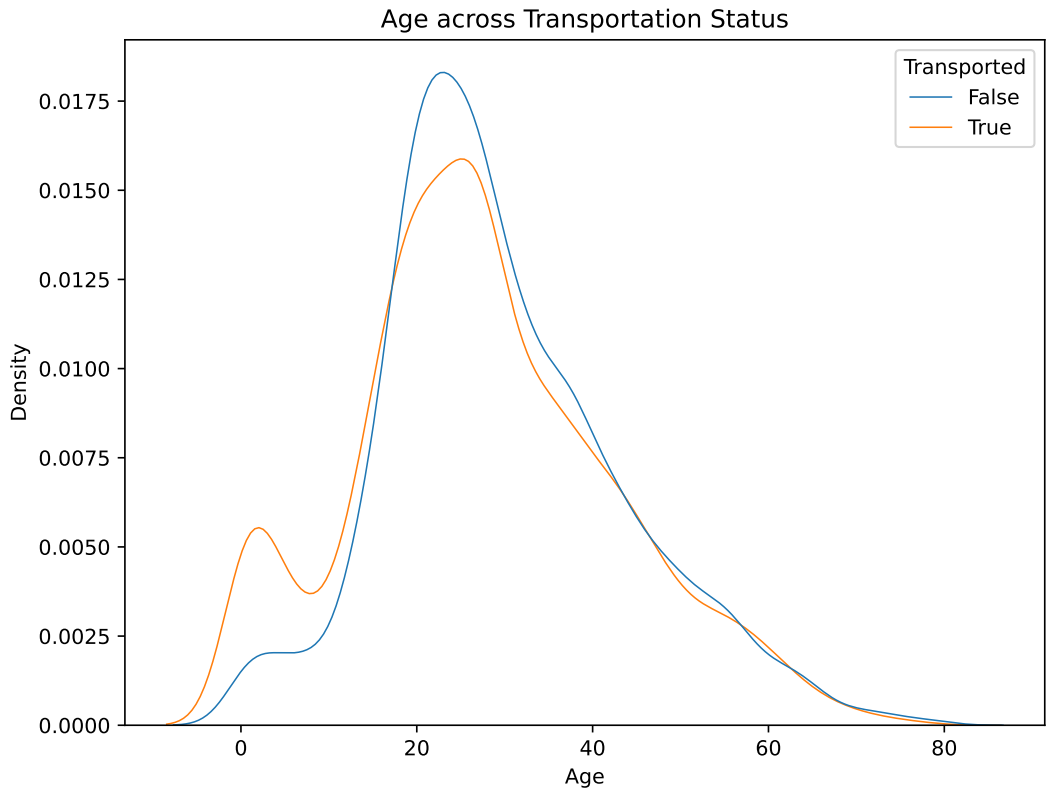
the transported passengers are in CryoSleep while most of the non-transported passengers are *not* in CryoSleep. The destination for both transported and non-transported passengers is TRAPPIST-1e. The VIP distribution for transported and non-transported passengers is very similar- indicating there may not be a correlation between the variables.

Exercise 5

Using box plots, density curves, histograms, dot plots, or violin plots, compare the distributions of the following and describe what patterns you observe, if any.

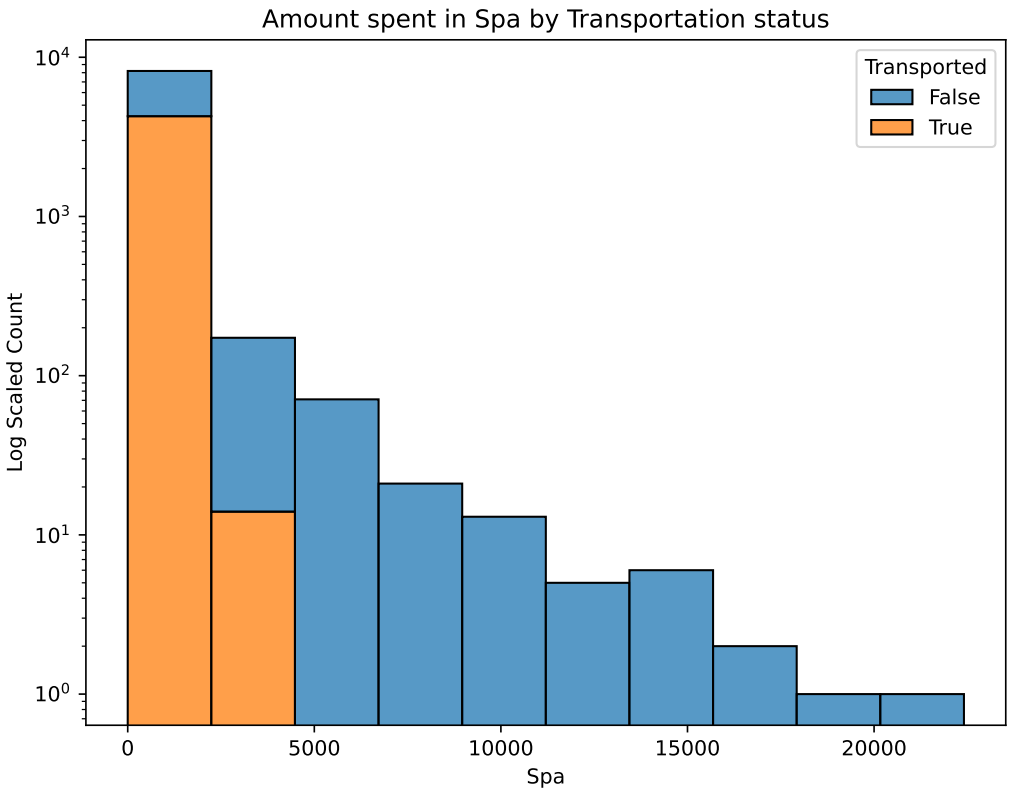
1. `age` across levels of `transported`
2. `room_service` across levels of `transported`
3. `spa` across levels of `transported`
4. `vr_deck` across levels of `transported`

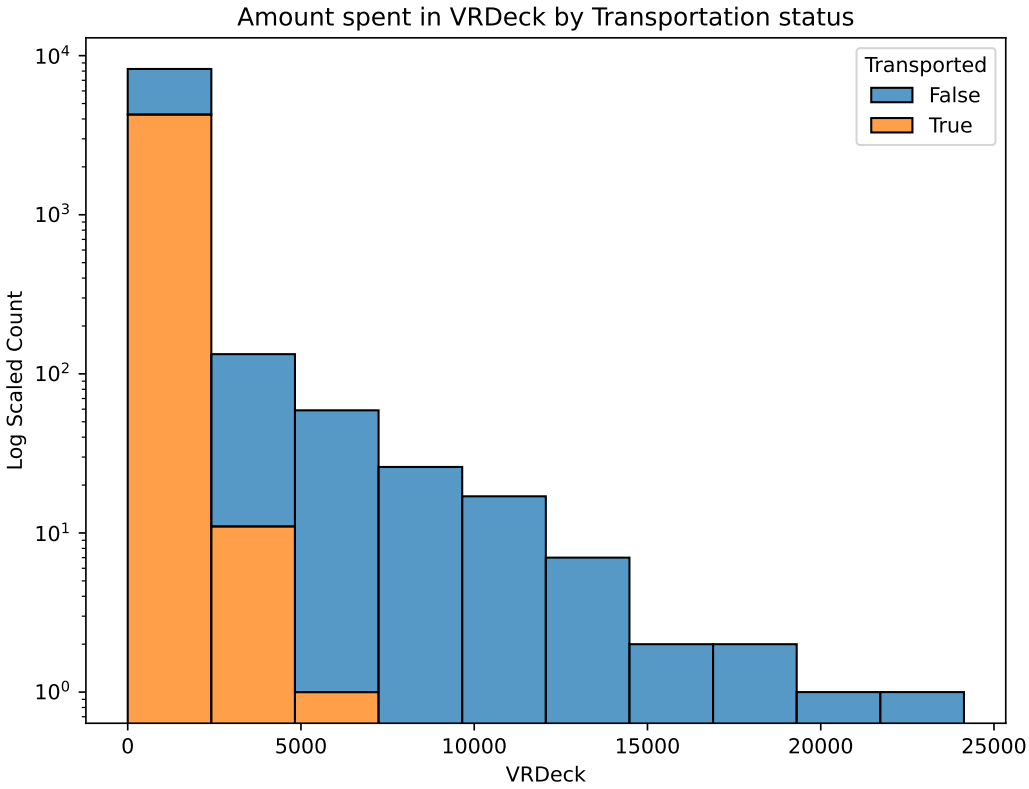
```
plt.figure(figsize=(8, 6))
sns.kdeplot(data=titanic_cleaned, x='Age', hue='Transported', linewidth=0.75)

# Add labels and title
plt.title("Age across Transportation Status")
plt.xlabel("Age")
plt.ylabel("Density")

# Show the plot
plt.show()
```

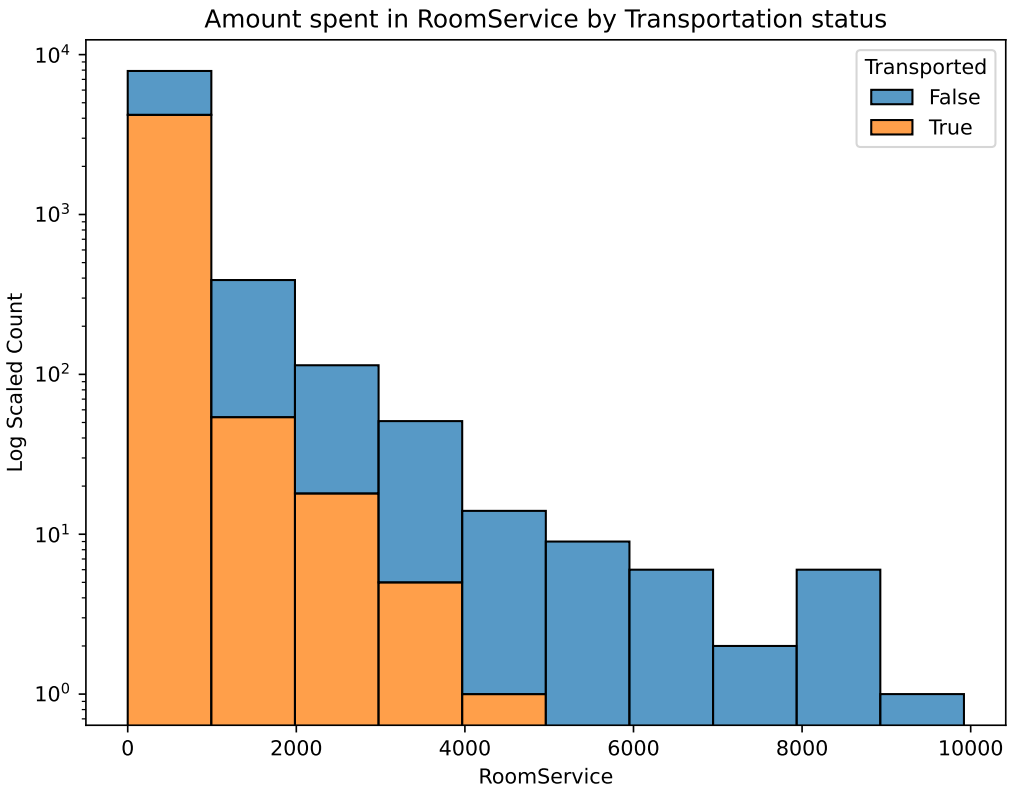



```
plt.close()
```

```
# considerations: log normalize?
for i in ['Spa', 'VRDeck', 'RoomService']:
  plt.figure(figsize=(8, 6))
  sns.histplot(data=titanic_cleaned, x=i, hue='Transported', bins=10, edgecolor='black', sta

  # Add labels and title
  plt.yscale('log')
  plt.title(f"Amount spent in {i} by Transportation status")
  plt.xlabel(i)
  plt.ylabel("Log Scaled Count")

  # Show the plot
  plt.show()
  plt.close()
```


Amount spent in Spa by Transportation status
Transported
False
True
Log Scaled Count
10^4
10^3
10^2
10^1
10^0
0
5000
10000
15000
20000
Spa


Amount spent in VRDeck by Transportation status
Transported
False
True
Log Scaled Count
10^4
10^3
10^2
10^1
10^0
0
5000
10000
15000
20000
25000
VRDeck



Exercise 6

Make a correlogram of the continuous variables in the dataset. What do you observe?

```
library(reticulate)
library(tidyverse)
```

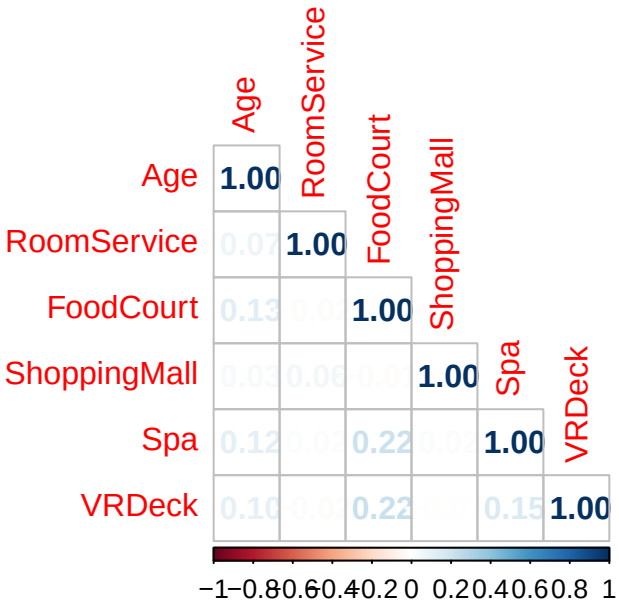
```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

```
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(corrplot)
```

```
corrplot 0.95 loaded
```

```
py$titanic_cleaned %>%
  select_if(is.numeric) %>%
  cor(use = "pairwise.complete.obs") %>%
  corrplot(type = "lower", method = "number")
```



The correlogram indicates weak positive correlations amongst most of the continuous variables. There exists the stronget positive correlation between FoodCourt with Spa and Foodcourt with VRDeck. This indicates that these variables behave very similarly. Age has a weak positive correlation with each of the variables [RoomService, FoodCourt, ShoppingMall, Spa, and VRDeck]

Exercise 7

Use binning to divide the feature `age` into six groups: ages 0-12, 13-17, 18-25, 26-30, 31-50, and 51+.

```
def categorize_age(age):
  if age <= 12:
    return "0-12"
  elif 13 <= age <= 17:
    return "13-17"
  elif 18 <= age <= 25:
    return "18-25"
  elif 26 <= age <= 30:
    return "26-30"
  elif 31 <- age <= 50:
    return "31-50"
  else:
    return "51+"
titanic_cleaned['Age_Group'] = titanic_cleaned['Age'].apply(categorize_age)
```

Exercise 8

For the expenditure variables, do the following:

- Create a new feature that consists of the total expenditure across all five amenities;
- Create a binary feature to flag passengers who did not spend anything (a total expenditure of 0);
- Log-transform the total expenditure to reduce skew.

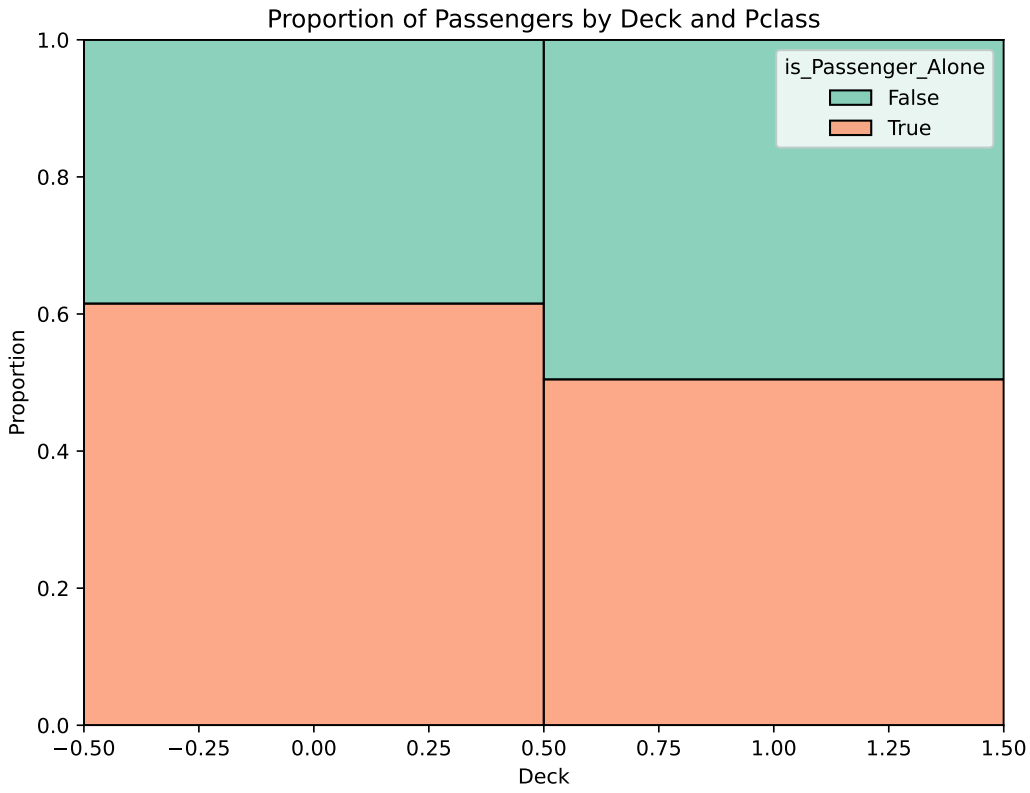
```
titanic_cleaned['Sum_Expenditures'] = titanic_cleaned[expenditures].sum(axis = 1) # axis = 1

titanic_cleaned['is_Spent_Expenditures'] = titanic_cleaned['Sum_Expenditures'] != 0

# log transform with log(1+x) to handle 0s
titanic_cleaned['Sum_Expenditures_log'] = np.log1p(titanic_cleaned['Sum_Expenditures'])
```

Exercise 9

Using the `passenger_id` column, create a new binary-coded feature that represents whether a passenger was traveling alone or not. Make a proportion stacked bar chart of this feature and `transported`. What do you observe?

```
groups = titanic_cleaned['PassengerId'].str[:4].value_counts().to_dict()

def group_passengers(passenger):
  if groups[passenger[:4]] > 1:
    return False
  else:
    return True

titanic_cleaned['is_Passenger_Alone'] = titanic_cleaned['PassengerId'].apply(group_passengers

# alternate (old code)
# group = {}
# for i in titanic_cleaned['PassengerId']:
#   if i[:4] in group:
#     group[i[:4]] += 1
#   else:
#     group[i[:4]] = 1
```

```
titanic_cleaned['Transported'] = titanic_cleaned['Transported'].astype('category')

plt.figure(figsize=(8, 6))
sns.histplot(
    data=titanic_cleaned,
    x='Transported',
    hue='is_Passenger_Alone',
    multiple='fill',
    discrete = True,
    palette="Set2"
)
plt.xlabel("Deck")
plt.ylabel("Proportion")
plt.title("Proportion of Passengers by Deck and Pclass")
plt.show()
```




```
plt.close()
```

Exercise 10

Using the `cabin` variable, extract:

1. Cabin deck (A, B, C, D, E, F, G, or T);
2. Cabin number (0 to 2000);
3. Cabin side (P or S).

Then do the following:

- Drop any observations with a cabin deck of T;
- Bin cabin number into groups of 300 (for example, 0 - 300, 301 - 600, 601- 900, etc.).

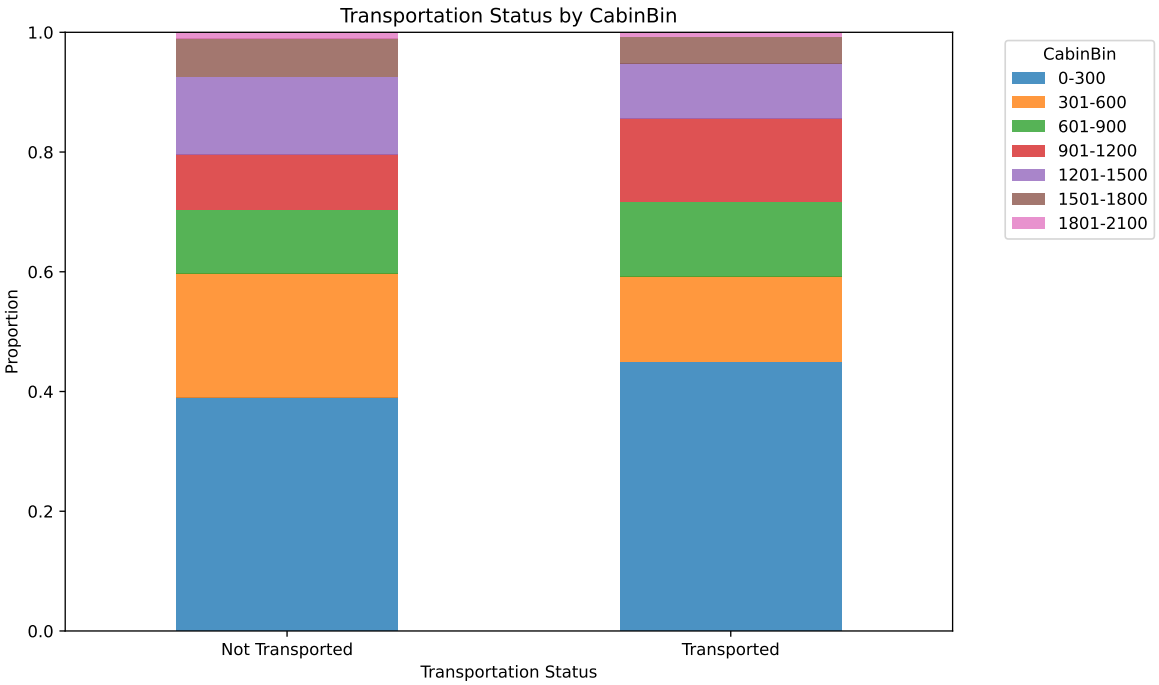
```
# cabin deck is the first letter of each cabin
titanic_cleaned['Deck'] = titanic_cleaned['Cabin'].str[:1]

# cabin number is the numerical part of each cabin
# titanic_cleaned['CabinNumber'] = titanic_cleaned['Cabin'].str.findall(r'\d+\.?\d*')

titanic_cleaned['CabinNumber'] = titanic_cleaned['Cabin'].apply(lambda x: int(''.join(filter

# cabin side is the last letter of each cabin
titanic_cleaned['CabinSide'] = titanic_cleaned['Cabin'].str[-1]
```

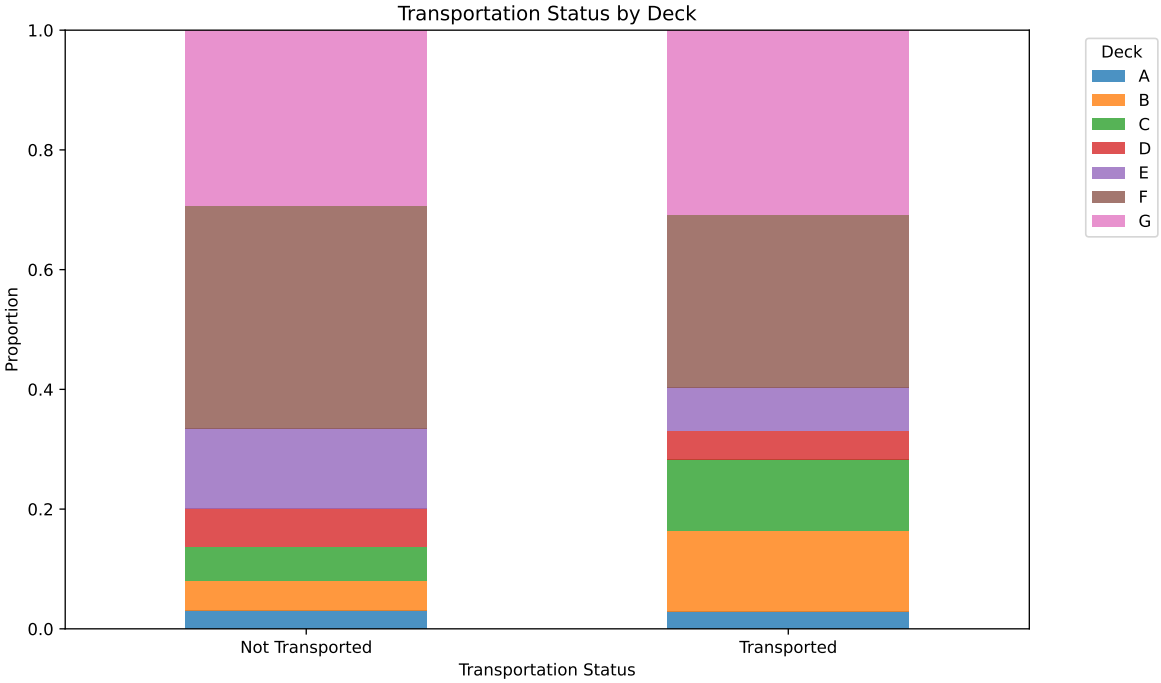
```
titanic_cleaned['Deck']
```

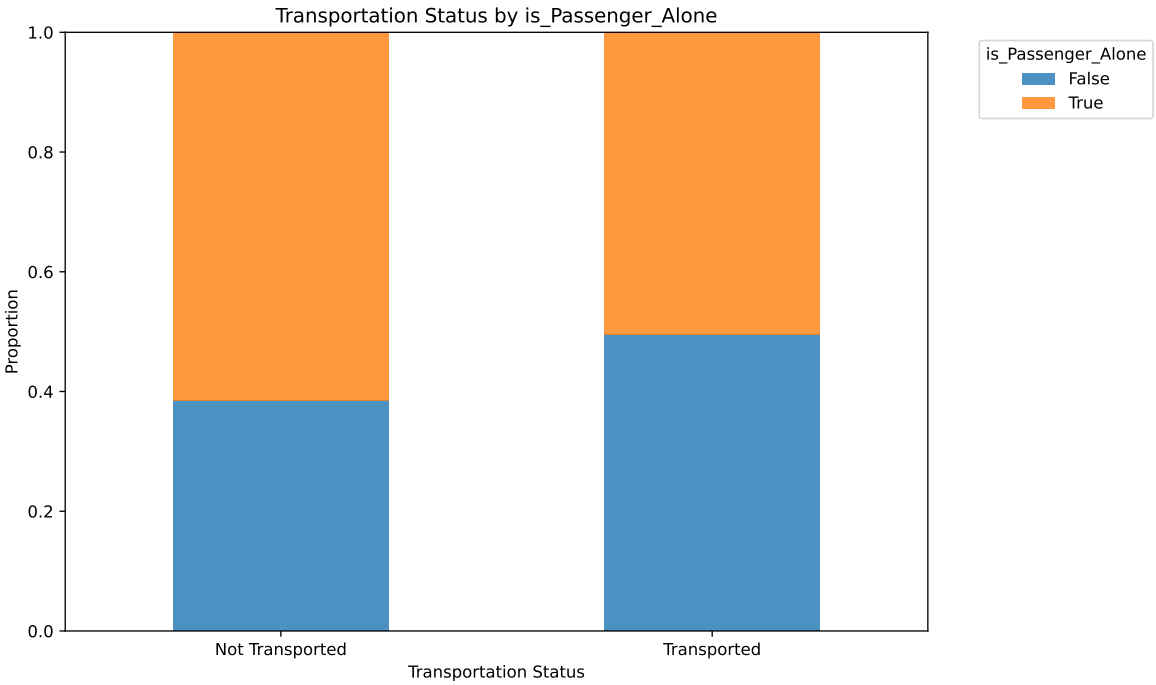
```
0       B
1       F
2       A
3       A
4       F
       ..
8688    A
8689    G
8690    G
8691    E
8692    E
Name: Deck, Length: 8494, dtype: object
```

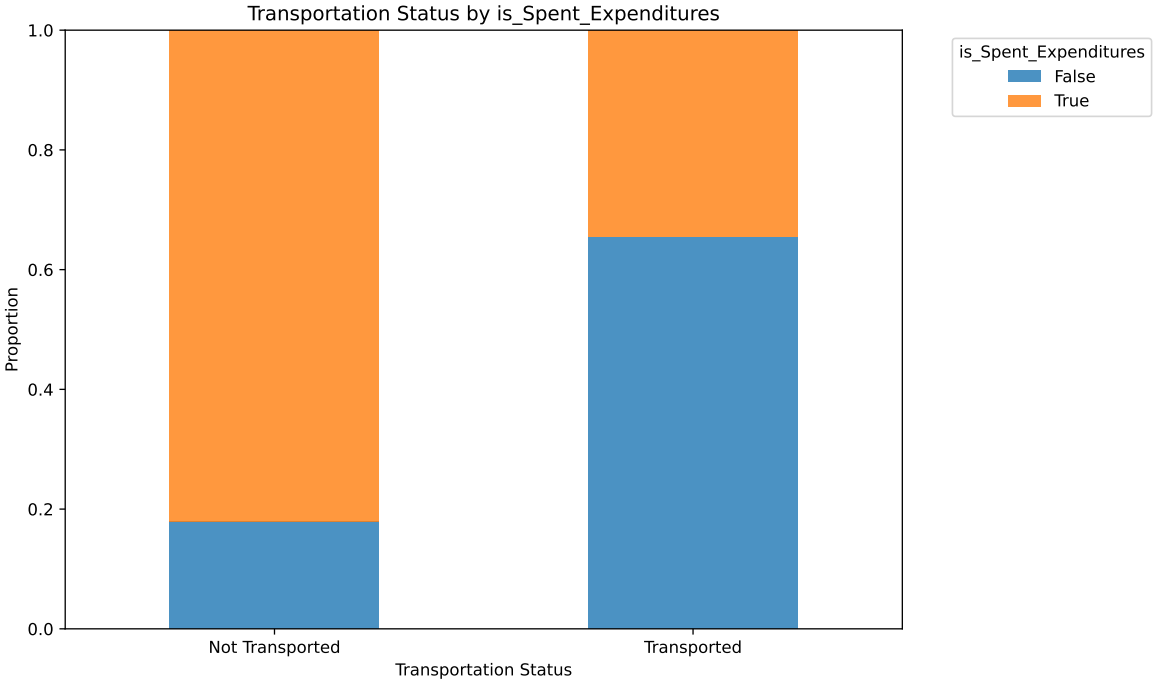
```
# drop observations with a cabin deck of T
titanic_cleaned = titanic_cleaned[titanic_cleaned['Deck'] != 'T']

# bin cabin numbers into groups of 300
bins = range(titanic_cleaned['CabinNumber'].min(), titanic_cleaned['CabinNumber'].max() + 30

labels = []
for i in bins[:-1]:
  if i == 0:
    labels.append(f'{i}-{i+300}')
  else:
    labels.append(f'{i+1}-{i+300}')

# use pd.cut() to categorize into bins
titanic_cleaned['CabinBin'] = pd.cut(titanic_cleaned['CabinNumber'], bins=bins, labels=labels
```

```
# EXTRA
for i in ['CabinBin', 'Deck', 'is_Passenger_Alone', 'is_Spent_Expenditures']:
  # Calculate counts of each class by home planet
  count_data = titanic_cleaned.groupby(['Transported', i]).size().reset_index(name='count')

  # Calculate the total counts for each home planet
  total_counts = count_data.groupby('Transported')['count'].transform('sum')

  # Calculate the proportion
  count_data['proportion'] = count_data['count'] / total_counts

  # Create a pivot table for better plotting
  pivot_data = count_data.pivot(index='Transported', columns=i, values='proportion').fillna(0

  # Plotting
  plt.figure(figsize=(10, 6))

  # Stack the bars
  pivot_data.plot(kind='bar', stacked=True, ax=plt.gca(), alpha=0.8)

  # Set y-axis limits
  plt.ylim(0, 1)
  plt.title(f"Transportation Status by {i}", loc = 'center')
  plt.xlabel('Transportation Status')
  plt.xticks(ticks=[0, 1], labels=['Not Transported', 'Transported'], rotation=0)
  plt.ylabel("Proportion")

  # Customize the legend
  plt.legend(title=i, bbox_to_anchor=(1.05, 1), loc='upper left')
  plt.tight_layout()

  # Show the plot
  plt.show()
  plt.close()
```


Transportation Status by CabinBin
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
Not Transported
Transported
Transportation Status
CabinBin
0-300
301-600
601-900
901-1200
1201-1500
1501-1800
1801-2100


Transportation Status by Deck
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
Not Transported
Transported
Transportation Status
Deck
A
B
C
D
E
F
G


Transportation Status by is_Passenger_Alone
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
Not Transported
Transported
Transportation Status
is_Passenger_Alone
False
True


Transportation Status by is_Spent_Expenditures
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
Not Transported
Transported
Transportation Status
is_Spent_Expenditures
False
True

```
titanic_cleaned['HomePlanet'].value_counts()
```

```
HomePlanet
Earth     4701
Europa    2066
Mars      1722
Name: count, dtype: int64
```

```
titanic_cleaned['CryoSleep'].value_counts()
```

```
CryoSleep
False    5536
True     2953
Name: count, dtype: int64
```

```
titanic_cleaned['VIP'].value_counts()
```

```
VIP
False    8296
True      193
Name: count, dtype: int64
```

```
titanic_cleaned['CabinBin'].value_counts()
```

```
CabinBin
0-300        3564
301-600      1484
901-1200      991
601-900       976
1201-1500     937
1501-1800     459
1801-2100      78
Name: count, dtype: int64
```

```
titanic_cleaned['Deck'].value_counts()
```

```
Deck
F    2794
G    2559
E     876
B     779
C     747
D     478
A     256
Name: count, dtype: int64
```

```
titanic_cleaned['is_Passenger_Alone'].value_counts()
```

```
is_Passenger_Alone
True     4748
False    3741
Name: count, dtype: int64
```

```
titanic_cleaned['is_Spent_Expenditures'].value_counts()
```

```
is_Spent_Expenditures
True     4933
False    3556
Name: count, dtype: int64
```

Exercise 11

Create a new data frame (or tibble) that retains the following features:

1. `home_planet`
2. cabin deck
3. cabin number (binned)
4. cabin side
5. `age` (binned)
6. total expenditures (log-transformed)
7. `cryo_sleep`
8. `destination`
9. whether the passenger was traveling alone (call this `solo`)
10. Transported

To those features, do the following:

- One-hot encode all categorical features
- Center and scale all continuous features

```
titanic_new = pd.DataFrame({'home_planet': titanic_cleaned['HomePlanet'], 'cabin_deck': titan

# alternate
# selected_columns = [
#     'HomePlanet', 'Deck', 'CabinBin', 'Age_Group', 'Sum_Expenditures_log',
#     'CryoSleep', 'Destination', 'is_Passenger_Alone', 'Transported'
# ]
#
# df = titanic_cleaned[selected_columns].copy()
#
# # Rename the columns
# df.columns = [
#     'home_planet', 'cabin_deck', 'cabin_number_bin', 'age_bin',
#     'total_expenditures_log', 'cryo_sleep', 'destination', 'solo', 'transported'
# ]
```

```
titanic_new_hot = pd.get_dummies(titanic_new.drop(columns = ['total_expenditures_log', 'trans

# Initialize the scaler
scaler = StandardScaler()

# Fit and transform the data (center and scale)
titanic_new_hot['total_expenditures_log_scaled'] = scaler.fit_transform(titanic_new[['total_e

titanic_new_hot['transported'] = titanic_new['transported']

titanic_new_hot
```

```
      cryo_sleep   solo  ...  total_expenditures_log_scaled  transported
0          False   True  ...                      -1.159492        False
1          False   True  ...                       0.627373         True
2          False  False  ...                       1.343309        False
3          False  False  ...                       1.154939        False
4          False   True  ...                       0.733779         True
...          ...    ...  ...                            ...          ...
8688       False   True  ...                       1.290304        False
8689        True   True  ...                      -1.159492        False
8690       False   True  ...                       0.879937         True
```

```
8691       False  False  ...                    1.125185        False
8692       False  False  ...                    1.135995         True

[8489 rows x 29 columns]
```

Exercise 12

Write up your analyses thus far in one or two paragraphs. Describe what you have learned about the passengers on the Spaceship Titanic. Describe the relationships you observed between variables. Which features do you think may be the best predictors of transported status? Why or why not?

The information in this dataset has approximately 2% total missing data. There seems to be a positive correlation between passengers in CyroSleep and their transporation status, as most passengers in CyroSleep are also transported. In addition, a negative correlation between VIP passengers and their transportation status, as most passengers who are VIP are not transported. There seems to be no correlation between age and transportation status, as both statuses peak around the ages 20-40. There also seems to be no correlation between the expenditures (room service, spa, VRDeck) with transportation status, as most of these observations were all concentrated at 0 (low values).
There seems to be no correlation between cabin number/bin and deck with the transportation status. Cabin bin 0-300 contains the most passengers, which is why it compromises the most of passengers of both transportation status. However, because the amount of passengers who spent expenditures or do not is fairly similar, we can believe there is a correlation that suggests that passengers who do not spend any expenditures are transported. Thus, I think that CryoSleep, VIP, and whether the passenger spent expenditures is the best predictor of transported status.

Exercises for 234 Students

Exercise 13

Split the dataset into training and testing. Use random sampling. Make sure that 80% of observations are in the training set and the remaining 20% in the testing set.

Exercise 14

Using k-fold cross-validation with k of 5, tune two models:

1. A random forest;
2. An elastic net.

Exercise 15

Select your best **random forest** model and use it to predict your testing set. Present the following:

- Testing accuracy;
- Testing ROC curve (and area under the ROC curve);
- Confusion matrix;
- Variable importance plot.

Exercise 16

Write up your conclusions in one to two paragraphs. Answer the following: How did your models do? Are you happy with their performance? Is there another model (besides these two) that you would be interested in trying? Which features ended up being the most important in terms of predicting whether or not a passenger would be transported?