```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression

import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('/content/train.csv')
```

```
df.head()
```

	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	
0	0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.0	False	0.0	0.0	0.0	0.0	0.0	M Of
1	0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.0	False	109.0	9.0	25.0	549.0	44.0	`
2	0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.0	True	43.0	3576.0	0.0	6715.0	49.0	
3	0003_02	Europa	False	A/0/S	TRAPPIST-1e	33.0	False	0.0	1283.0	371.0	3329.0	193.0	
4	0004_01	Earth	False	F/1/S	TRAPPIST-1e	16.0	False	303.0	70.0	151.0	565.0	2.0	Santa

```
df.shape
```

```
(8693, 14)
```

```
df.info()
```

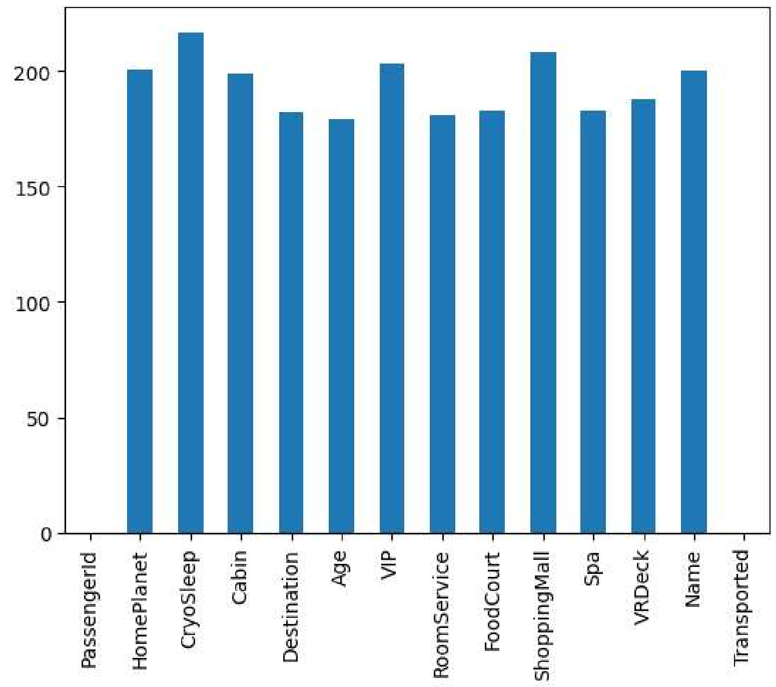
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   PassengerId   8693 non-null   object 
 1   HomePlanet    8492 non-null   object 
 2   CryoSleep     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.describe()
```

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

```
df.isnull().sum().plot.bar()
plt.show()
```



```
col = df.loc[:,'RoomService':'VRDeck'].columns
df.groupby('VIP')[col].mean()
```

	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
VIP					
False	217.218527	426.336536	173.876298	301.711045	282.718056
True	473.615385	1811.393782	247.726804	760.710660	1234.856410

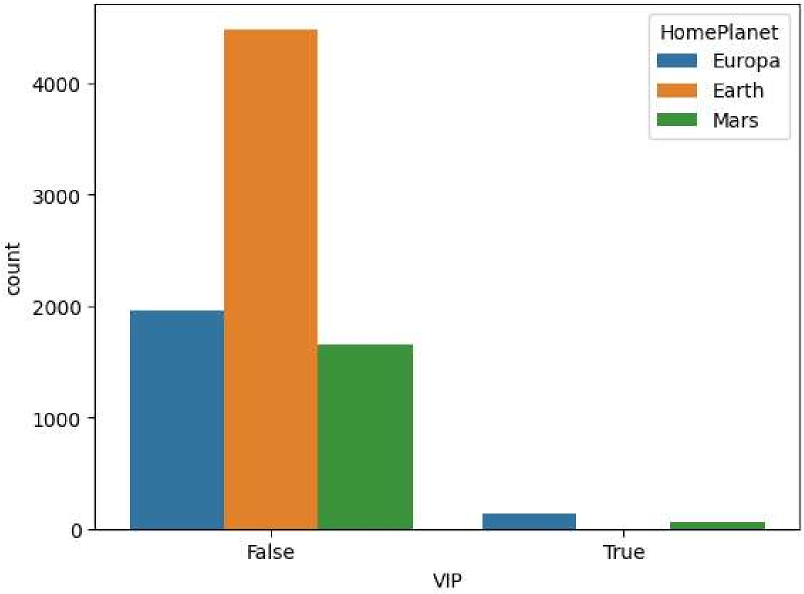
```
df.groupby('CryoSleep')[col].mean()
```

	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
CryoSleep					
False	350.146772	713.004316	270.586504	486.09294	475.716165

```
temp = df['CryoSleep'] == True
df.loc[temp, col] = 0.0
```

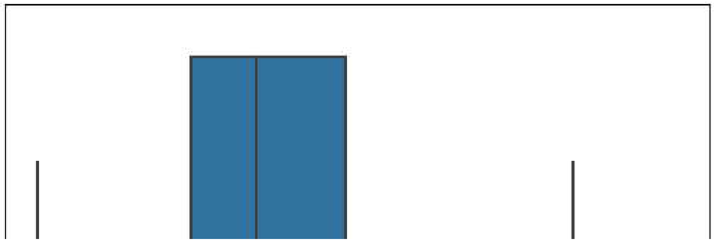
```
for c in col:
    for val in [True, False]:
        temp = df['VIP'] == val
        k = df[temp].mean()
        df.loc[temp, c] = df.loc[temp, c].fillna(k)
```

```
sb.countplot(data=df, x='VIP',
            hue='HomePlanet')
plt.show()
```



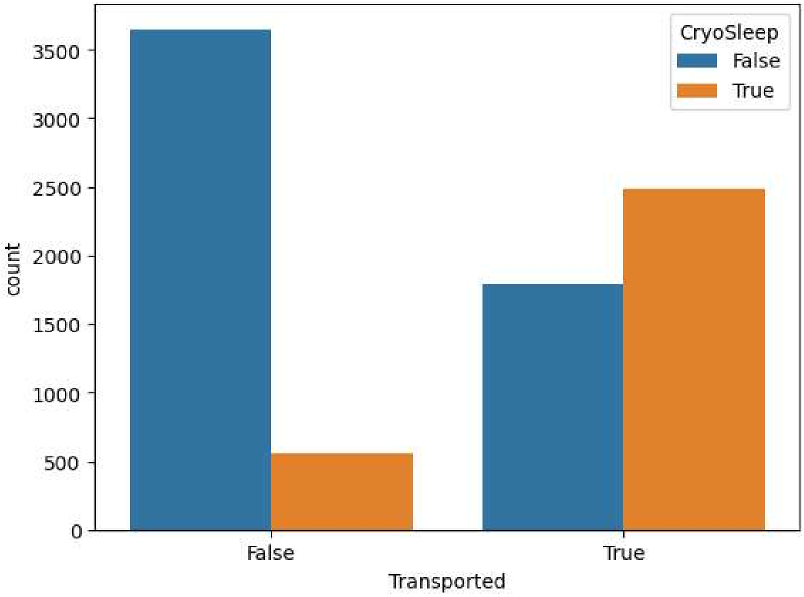
```
col = 'HomePlanet'
temp = df['VIP'] == False
df.loc[temp, col] = df.loc[temp, col].fillna('Earth')

temp = df['VIP'] == True
df.loc[temp, col] = df.loc[temp, col].fillna('Europa')
```

```
sb.boxplot(df['Age'],orient='h')
plt.show()
```



```
temp = df[df['Age'] < 61]['Age'].mean()
df['Age'] = df['Age'].fillna(temp)
```

```
sb.countplot(data=df,
            x='Transported',
            hue='CryoSleep')
plt.show()
```



```
df.isnull().sum().plot.bar()
plt.show()
```

```
for col in df.columns:
    # Check for null values presence
    if df[col].isnull().sum() == 0:
        continue

    # If the data type is categorical filling by mode.
    if df[col].dtype == object or df[col].dtype == bool:
        df[col] = df[col].fillna(df[col].mode()[0])

    # Else by mean
    else:
        df[col] = df[col].fillna(df[col].mean())

df.isnull().sum().sum()
```

```
0
```

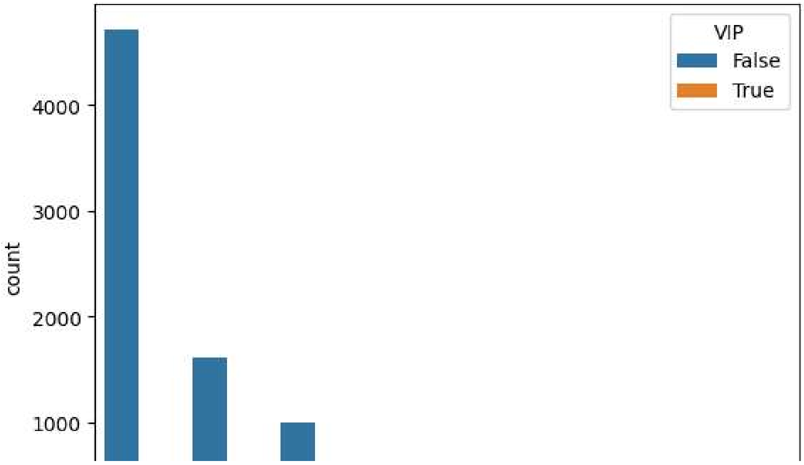
```
df.head()
```

	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	Food
0	0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.0	False	0.0	
1	0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.0	False	109.0	
2	0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.0	True	43.0	3
3	0003_02	Europa	False	A/0/S	TRAPPIST-1e	33.0	False	0.0	1
4	0004_01	Earth	False	F/1/S	TRAPPIST-1e	16.0	False	303.0	

```
new = df["PassengerId"].str.split("_", n=1, expand=True)
df["RoomNo"] = new[0].astype(int)
df["PassengerNo"] = new[1].astype(int)

df.drop(['PassengerId', 'Name'],
        axis=1, inplace=True)
```

```
data = df['RoomNo']
for i in range(df.shape[0]):
    temp = data == data[i]
    df['PassengerNo'][i] = (temp).sum() #this was throwing an indentation error
```

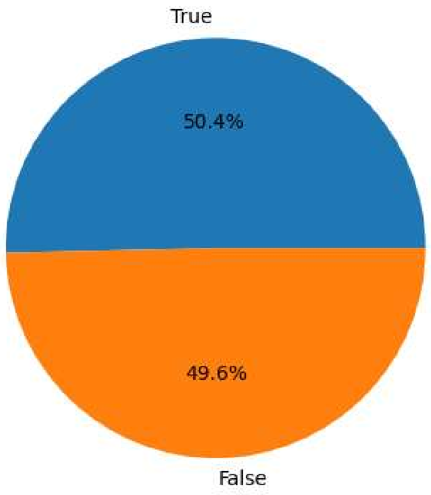
```
df.drop(['RoomNo'], axis=1,
        inplace=True)

sb.countplot(data=df,
            x = 'PassengerNo',
            hue='VIP')
plt.show()
```



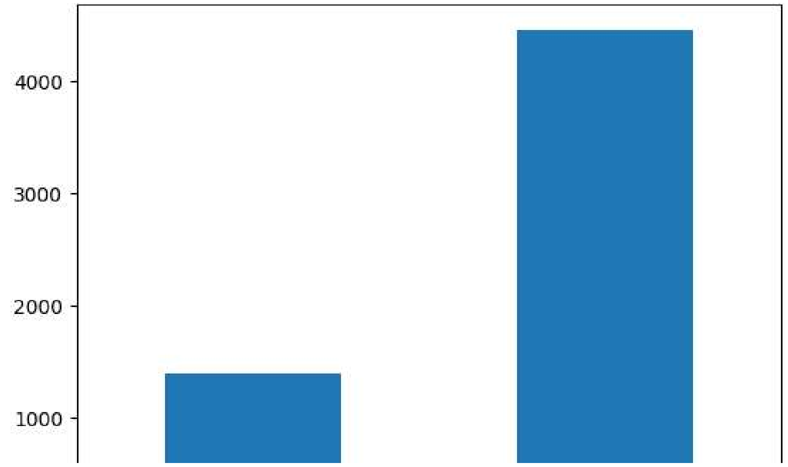
```
new = df["Cabin"].str.split("/", n=2, expand=True)
data["F1"] = new[0]
df["F2"] = new[1].astype(int)
df["F3"] = new[2]

df.drop(['Cabin'], axis=1,
        inplace=True)
```

```
df['LeasureBill'] = df['RoomService'] + df['FoodCourt']\
+ df['ShoppingMall'] + df['Spa'] + df['VRDeck']
```

```
x = df['Transported'].value_counts()
plt.pie(x.values,
        labels=x.index,
        autopct='%1.1f%%')
plt.show()
```



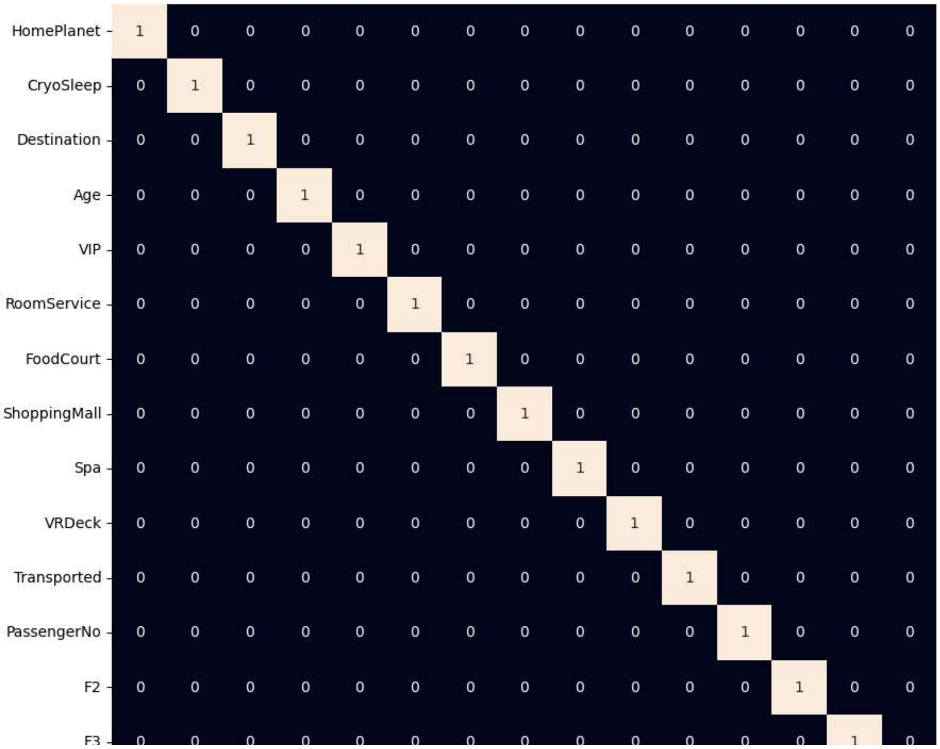
```
df.groupby('VIP').mean()['LeasureBill'].plot.bar()
plt.show()
```



```
for col in df.columns:
    # In case of categorical column
    # encode them
    if df[col].dtype == object:
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])

    # In case of boolean data type
    # convert them to binary
    if df[col].dtype == 'bool':
        df[col] = df[col].astype(int)

df.head()
```

	HomePlanet	CryoSleep	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	
0	1	0	2	39.0	0	0.0	0.0	0.0	
1	0	0	2	24.0	0	109.0	9.0	25.0	
2	1	0	2	58.0	1	43.0	3576.0	0.0	6
3	1	0	2	33.0	0	0.0	1283.0	371.0	3
4	0	0	2	16.0	0	303.0	70.0	151.0	

```
plt.figure(figsize=(10,10))
sb.heatmap(df.corr()>0.8,
    annot=True,
    cbar=False)
plt.show()
```



```
features = df.drop(['Transported'], axis=1)
target = df.Transported

X_train, X_val,\
    Y_train, Y_val = train_test_split(features, target,
                                      test_size=0.1,
                                      random_state=22)

X_train.shape, X_val.shape
```

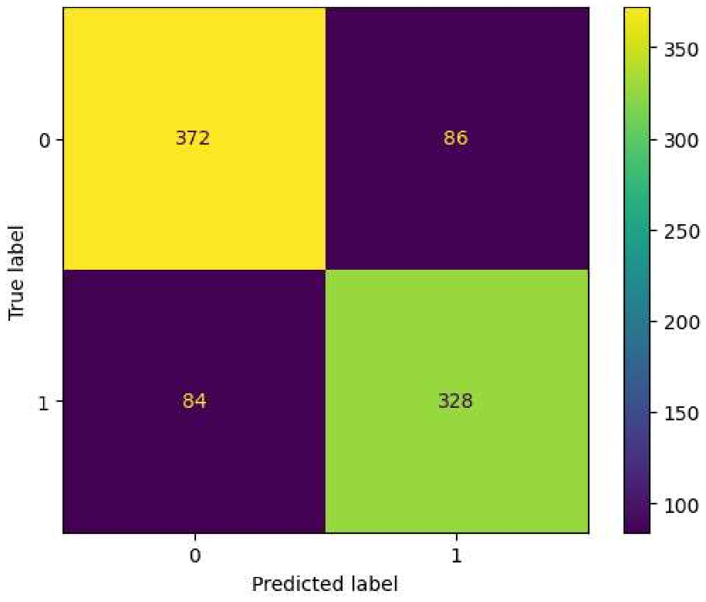
```
     ((7823, 14), (870, 14))
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
```

```
from sklearn.metrics import roc_auc_score as ras
models = [LogisticRegression(), XGBClassifier(),
        SVC(kernel='rbf', probability=True)]

for i in range(len(models)):
    models[i].fit(X_train, Y_train)

    print(f'{models[i]} : ')

    train_preds = models[i].predict_proba(X_train)[:, 1]
    print('Training Accuracy : ', ras(Y_train, train_preds))

    val_preds = models[i].predict_proba(X_val)[:, 1]
    print('Validation Accuracy : ', ras(Y_val, val_preds))
    print()
```

```
     LogisticRegression() :
     Training Accuracy :  0.8690305894110832
     Validation Accuracy :  0.8571352863865691

     XGBClassifier(base_score=None, booster=None, callbacks=None,
                   colsample_bylevel=None, colsample_bynode=None,
                   colsample_bytree=None, device=None, early_stopping_rounds=None,
                   enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...) :
Training Accuracy :  0.9854435831348938
Validation Accuracy :  0.8802491838724722

SVC(probability=True) :
Training Accuracy :  0.8885367142620493
Validation Accuracy :  0.8619631576715987
```

```
y_pred = models[1].predict(X_val)
cm = metrics.confusion_matrix(Y_val, y_pred)
disp = metrics.ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```



```
print(metrics.classification_report
    (Y_val, models[1].predict(X_val)))
```

```
              precision    recall  f1-score   support

           0       0.82      0.81      0.81       458
           1       0.79      0.80      0.79       412

    accuracy                           0.80       870
   macro avg       0.80      0.80      0.80       870
weighted avg       0.80      0.80      0.80       870
```