projet-kaggle

November 17, 2023

1 Projet Kaggle : Spaceship Titanic

Predict which passengers are transported to an alternate dimension https://www.kaggle.com/competitions/spaceship-titanic/overview

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!

To help rescue crews and retrieve the lost passengers, you are challenged to predict which passengers were transported by the anomaly using records recovered from the spaceship's damaged computer system.

Help save them and change history! We want to predict if a passenger has been transported or not. For this purpose we are going to use 2 models : logistic regression and a random forest. Then compare between them.

1.1 At first we are going to import those data and explore them.

[2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

[3]:
```
chemin = "train.csv"
data_train = pd.read_csv(chemin)
print('The size of my training data is ',data_train.shape)

chemin1 = "test.csv"
data_test = pd.read_csv(chemin1)
print('The size of my test data is ',data_test.shape)
```

```
The size of my training data is  (8693, 14)
The size of my test data is  (4277, 13)
```

1.1.1 Let's have a look to the variables and first informations

[4]:
```python
print("Variables of the dataset are :",data_train.columns,"\n")
print("Let's have a look of the dataset \n",data_train.head(),"\n")
print("Let's have a look to the type of the variables \n",data_train.info())
```

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

Let's have a look of the dataset
   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

<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
Let's have a look to the type of the variables
 None
```

The target variable is "Transported" wich is a boolean. Some variables are "object" but we can change it because of the context. We are going to see it later. We have 13 variables.

[5]:
```
for col in data_train.columns:
    nb_modalite = data_train[col].nunique()
    print(f" The variable '{col}' got {nb_modalite} values possible")
```

```
 The variable 'PassengerId' got 8693 values possible
 The variable 'HomePlanet' got 3 values possible
 The variable 'CryoSleep' got 2 values possible
 The variable 'Cabin' got 6560 values possible
 The variable 'Destination' got 3 values possible
 The variable 'Age' got 80 values possible
 The variable 'VIP' got 2 values possible
 The variable 'RoomService' got 1273 values possible
 The variable 'FoodCourt' got 1507 values possible
 The variable 'ShoppingMall' got 1115 values possible
 The variable 'Spa' got 1327 values possible
 The variable 'VRDeck' got 1306 values possible
 The variable 'Name' got 8473 values possible
 The variable 'Transported' got 2 values possible
```

[6]:
```
nombre_doublons = data_train.duplicated().sum()

print(f'The total number of duplicatness : {nombre_doublons}')
```

```
The total number of duplicatness : 0
```

[7]:
```
data_train.describe(include="all")
```

[7]:
```
       PassengerId HomePlanet CryoSleep    Cabin  Destination          Age
count         8693       8492      8476     8494         8511  8514.000000  \
unique        8693          3         2     6560            3          NaN
top        0001_01      Earth     False  G/734/S  TRAPPIST-1e          NaN
freq             1       4602      5439        8         5915          NaN
```

```
mean          NaN        NaN       NaN      NaN            NaN    28.827930
std           NaN        NaN       NaN      NaN            NaN    14.489021
min           NaN        NaN       NaN      NaN            NaN     0.000000
25%           NaN        NaN       NaN      NaN            NaN    19.000000
50%           NaN        NaN       NaN      NaN            NaN    27.000000
75%           NaN        NaN       NaN      NaN            NaN    38.000000
max           NaN        NaN       NaN      NaN            NaN    79.000000

          VIP   RoomService     FoodCourt  ShoppingMall           Spa
count    8490   8512.000000   8510.000000   8485.000000   8510.000000  \
unique      2           NaN           NaN           NaN           NaN
top     False           NaN           NaN           NaN           NaN
freq     8291           NaN           NaN           NaN           NaN
mean      NaN    224.687617    458.077203    173.729169    311.138778
std       NaN    666.717663   1611.489240    604.696458   1136.705535
min       NaN      0.000000      0.000000      0.000000      0.000000
25%       NaN      0.000000      0.000000      0.000000      0.000000
50%       NaN      0.000000      0.000000      0.000000      0.000000
75%       NaN     47.000000     76.000000     27.000000     59.000000
max       NaN  14327.000000  29813.000000  23492.000000  22408.000000

              VRDeck            Name Transported
count    8505.000000            8493        8693
unique           NaN            8473           2
top              NaN  Gollux Reedall        True
freq             NaN               2        4378
mean      304.854791             NaN         NaN
std      1145.717189             NaN         NaN
min         0.000000             NaN         NaN
25%         0.000000             NaN         NaN
50%         0.000000             NaN         NaN
75%        46.000000             NaN         NaN
max     24133.000000             NaN         NaN
```

1.1.2 Missing Values

Nevertheless the dataset may got missing values to find a solution we need to see how many and the disparity.

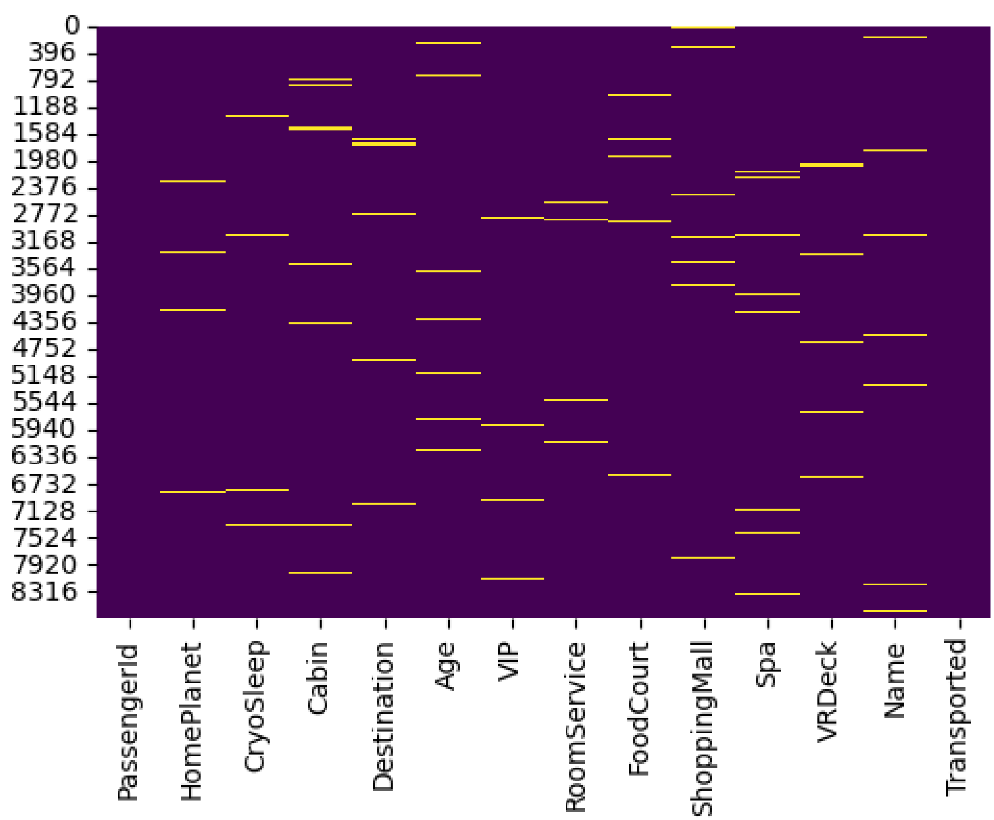
[8]:
```python
print(f"The missing values :\n \n{data_train.isna().sum()}")
```

```
The missing values :

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
```

Here a represensation by using the a graphic to see the distribution of the missing values

[9]:
```python
missing_values = data_train.isna()
plt.figure(figsize=(6, 4))
sns.heatmap(data_train.isna(), cbar = False, cmap='viridis')
plt.show()
```



[10]:
```python
data_train = data_train.dropna()
```

We do not see any distribution or pattern of missing values we can just delete it and the number of passengers who got missing values are pretty small.

1.2 Few word about the logistic regression :

We want to find a way to predict which passenger as been transported. Our target variables only got 2 values possibles False and True (0 and 1). Our features are quantitatives and qualitatives. We want to apply this method to see if we can find anything interesting (a pattern, a modality that is more important than the other etc).

We are going to create our target variable and the our features

1.2.1 For this part only we are going to use only the data_train and divide it again, because data_test don't provide the target variable. We are going to devide the data_train set into 2 dataset one data_train and one data_test but this time we are going to have the y_test that we are going to use.

[11]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
```

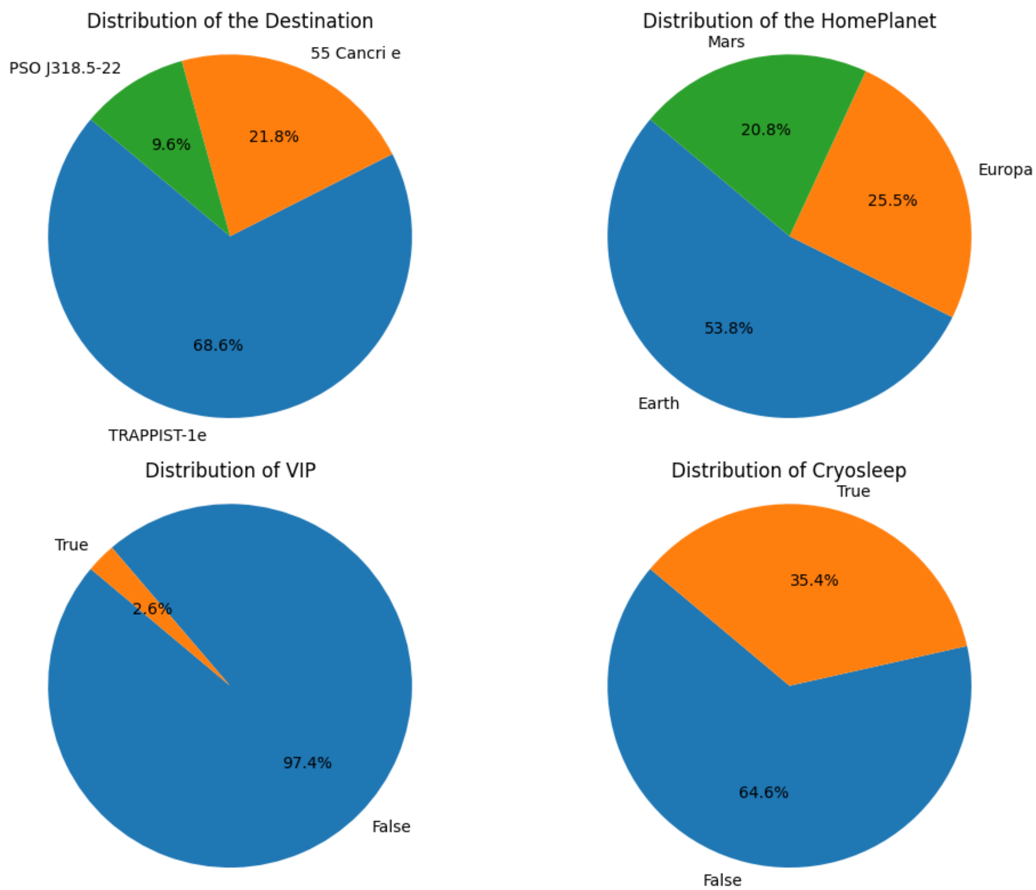
[12]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_train.
 ↪drop("Transported", axis=1), data_train["Transported"], test_size=1/3,␣
 ↪random_state=0)
```

1.2.2 Graphic, plot, histogram

Let's have a representation of our dataset

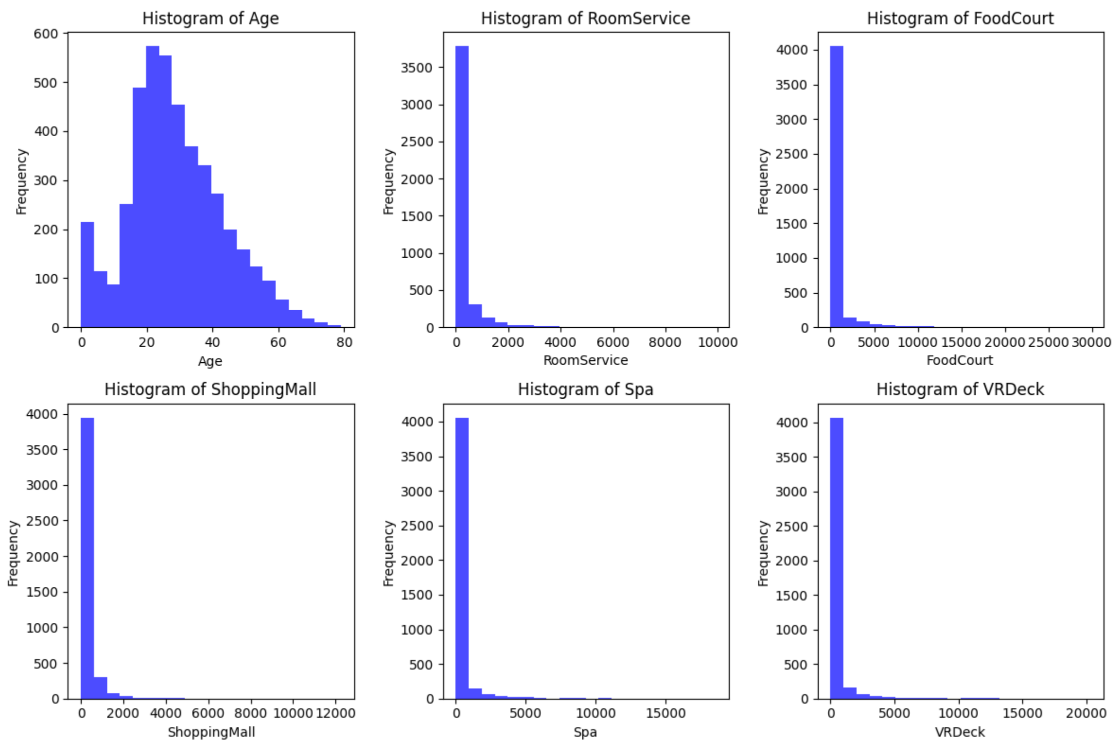
[13]:
```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))

destination = X_train['Destination'].value_counts()
axes[0, 0].pie(destination, labels=destination.index, autopct='%1.1f%%',␣
 ↪startangle=140)
axes[0, 0].set_title('Distribution of the Destination')
axes[0, 0].axis('equal')

homeplanet = X_train['HomePlanet'].value_counts()
axes[0, 1].pie(homeplanet, labels=homeplanet.index, autopct='%1.1f%%',␣
 ↪startangle=140)
axes[0, 1].set_title('Distribution of the HomePlanet')
axes[0, 1].axis('equal')

VIP = X_train["VIP"].value_counts()
axes[1, 0].pie(VIP, labels=VIP.index, autopct='%1.1f%%', startangle=140)
axes[1, 0].set_title('Distribution of VIP')
axes[1, 0].axis('equal')
```

```
cryosleep = X_train["CryoSleep"].value_counts()
axes[1, 1].pie(cryosleep, labels=cryosleep.index, autopct='%1.1f%%',␣
↪startangle=140)
axes[1, 1].set_title('Distribution of Cryosleep')
axes[1, 1].axis('equal')

plt.tight_layout()
plt.show()
```



[14]:
```
fig, axes = plt.subplots(2, 3, figsize=(12, 8))

variables = ["Age", "RoomService", "FoodCourt", "ShoppingMall", "Spa", "VRDeck"]

for i, var in enumerate(variables):
    row, col = divmod(i, 3)
    data = X_train[var]
    ax = axes[row, col]
    ax.hist(data, bins=20, color='blue', alpha=0.7)
    ax.set_title(f'Histogram of {var}')
```

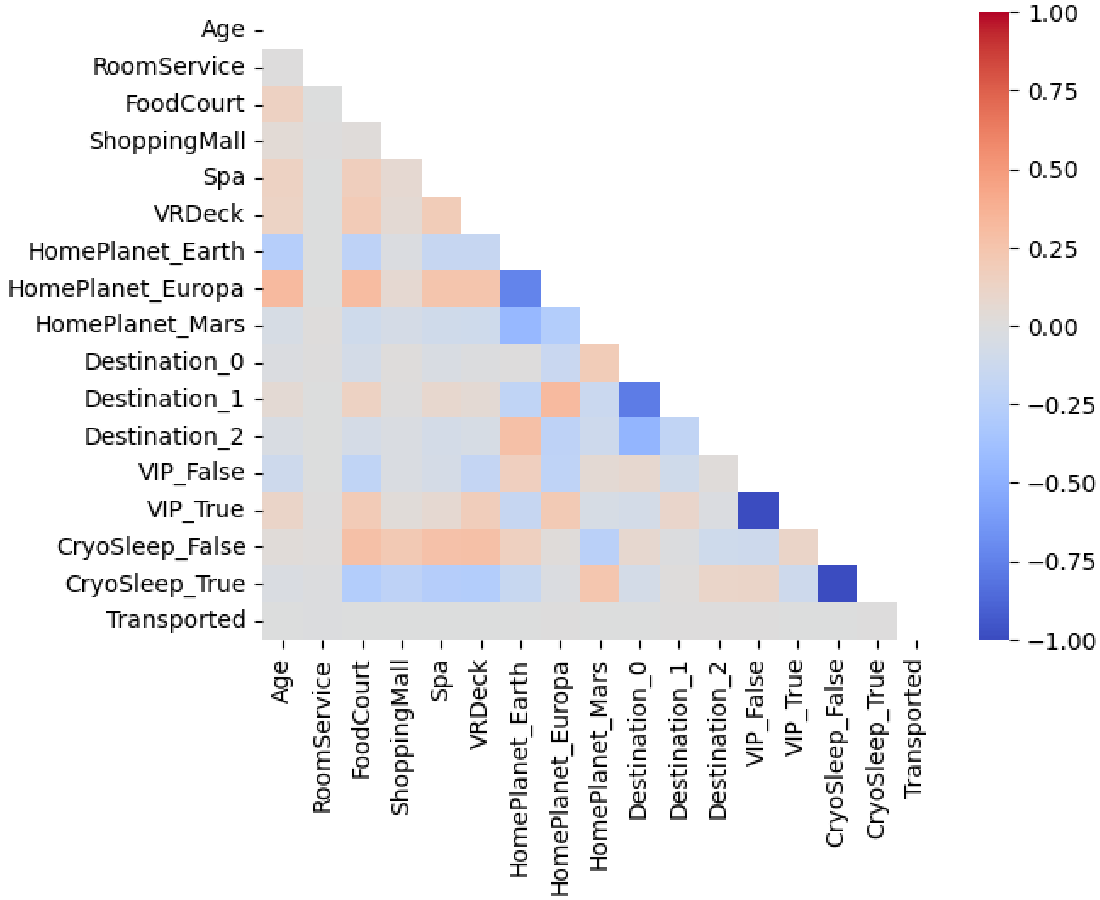
```python
    ax.set_xlabel(var)
    ax.set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```



[15]:
```python
X_train['HomePlanet'] = X_train['HomePlanet'].astype(str)
X_train['Destination'] = pd.factorize(X_train['Destination'])[0]
X_train['VIP'] = X_train['VIP'].astype(str)
X_train['CryoSleep'] = X_train['CryoSleep'].astype(str)
convert = ['HomePlanet','Destination','VIP','CryoSleep']
X_train = pd.get_dummies(X_train, columns=convert, prefix=convert)
X_train.drop(["Cabin","PassengerId","Name"],axis=1,inplace=True)
```

[16]:
```python
df_merged = pd.merge(X_train, data_train[["RoomService","Transported"]], on='RoomService', how='left')
upp_mat = np.triu(df_merged.corr())
sns.heatmap(df_merged.corr(), vmin=-1, vmax=+1, annot=False, cmap="coolwarm", mask=upp_mat)
```

[16]: <AxesSubplot: >



We can see that there is more individuals going to "TRAPPIST-1e", more individuals coming from earth, not being VIP and doing cryo. Unfortunatly there is no significant features that is in correlation with another. To be sure if there is a feature more important than other we are going to use a logistic regression.

We want to use the logistic regression from the scikit-learn library. We can not let the objet variable and we need to change to a categorical variable. (We are also drop variable that we do not need)

[17]:
```
print(X_train.dtypes)
print(X_train.head())
```

```
Age                  float64
RoomService          float64
FoodCourt            float64
ShoppingMall         float64
Spa                  float64
VRDeck               float64
HomePlanet_Earth        bool
HomePlanet_Europa       bool
HomePlanet_Mars         bool
```

```
Destination_0           bool
Destination_1           bool
Destination_2           bool
VIP_False               bool
VIP_True                bool
CryoSleep_False         bool
CryoSleep_True          bool
dtype: object
       Age  RoomService  FoodCourt  ShoppingMall     Spa  VRDeck  \
1528  35.0          0.0        0.0           0.0     0.0     0.0
357   48.0          0.0      111.0           0.0  1508.0     0.0
1101  17.0          0.0        0.0           0.0     0.0     0.0
4688  30.0       1120.0        0.0           0.0     0.0   794.0
6415  24.0          1.0        0.0           0.0   691.0     0.0

      HomePlanet_Earth  HomePlanet_Europa  HomePlanet_Mars  Destination_0  \
1528             False              False             True           True
357               True              False            False           True
1101             False               True            False           True
4688              True              False            False           True
6415              True              False            False          False

      Destination_1  Destination_2  VIP_False  VIP_True  CryoSleep_False  \
1528          False          False       True     False            False
357           False          False       True     False             True
1101          False          False       True     False            False
4688          False          False       True     False             True
6415           True          False       True     False             True

      CryoSleep_True
1528            True
357            False
1101            True
4688           False
6415           False
```

1.3 Logistic Regression

[18]:
```
modele_logit = LogisticRegression(penalty='none',solver='newton-cg')
modele_logit.fit(X_train,y_train)
pd.DataFrame(np.concatenate([modele_logit.intercept_.reshape(-1,1),
                             modele_logit.coef_],axis=1),index = ["coef"],
             columns = ["constante"]+list(X_train.columns)).T
```

```
C:\Users\vvicn\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\linear_model\_logistic.py:1173: FutureWarning:
`penalty='none'`has been deprecated in 1.2 and will be removed in 1.4. To keep
the past behaviour, set `penalty=None`.
```

```
  warnings.warn(
```

[18]:

```
                        coef
constante           0.471800
Age                -0.008423
RoomService        -0.001413
FoodCourt           0.000487
ShoppingMall        0.000628
Spa                -0.002002
VRDeck             -0.001923
HomePlanet_Earth   -0.835272
HomePlanet_Europa   1.482388
HomePlanet_Mars    -0.175317
Destination_0      -0.062344
Destination_1       0.522045
Destination_2       0.012098
VIP_False           0.629136
VIP_True           -0.157336
CryoSleep_False    -0.391802
CryoSleep_True      0.863601
```

1.4 Interpretation

The coefficients mean that if a value is low it interfer not that much, in the other hand if it's high it does interfer. Futhermore if it's positive it's increasing the probability that the passenger is missing. We see that Destination_2 which is 55-Cranri is influential for the probability of missing the same with Cryosleep_True and HomePlanet_Europa which seem logical according to the problem and the context. We can also had that if you come from Earth that avoid the passenger from missing a little more.

[19]:

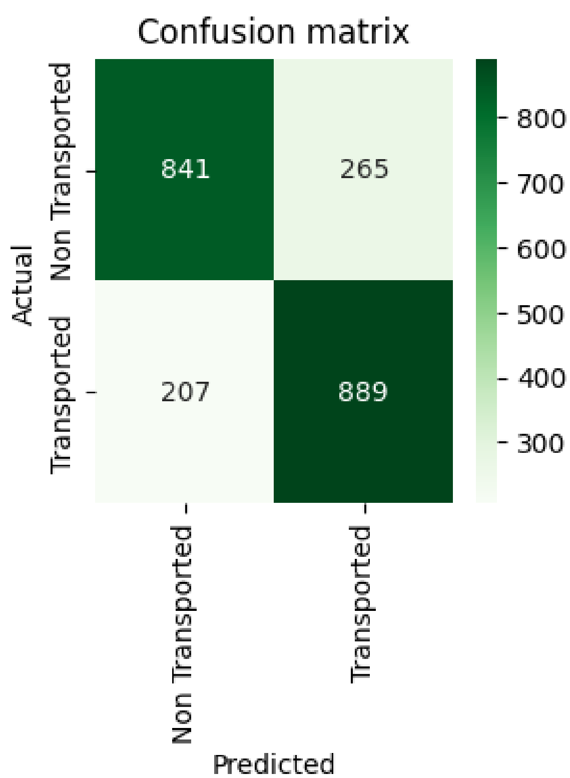
```
X_test['HomePlanet'] = X_test['HomePlanet'].astype(str)
X_test['Destination'] = pd.factorize(X_test['Destination'])[0]
X_test['VIP'] = X_test['VIP'].astype(str)
X_test['CryoSleep'] = X_test['CryoSleep'].astype(str)
convert = ['HomePlanet','Destination','VIP','CryoSleep']
X_test = pd.get_dummies(X_test, columns=convert, prefix=convert)
X_test.drop(["Cabin","PassengerId","Name"],axis=1,inplace=True)
print(X_test.dtypes)
```

```
Age                     float64
RoomService             float64
FoodCourt               float64
ShoppingMall            float64
Spa                     float64
VRDeck                  float64
HomePlanet_Earth           bool
HomePlanet_Europa          bool
HomePlanet_Mars            bool
```

```
Destination_0            bool
Destination_1            bool
Destination_2            bool
VIP_False                bool
VIP_True                 bool
CryoSleep_False          bool
CryoSleep_True           bool
dtype: object
```

1.5 Limit of the model

[20]:
```python
from sklearn.metrics import confusion_matrix

y_pred = modele_logit.predict(X_test)
confusion = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(3, 3))
sns.heatmap(confusion, annot=True, fmt='d', cmap='Greens', xticklabels=['Non␣
↪Transported', 'Transported'], yticklabels=['Non Transported', 'Transported'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion matrix')
plt.show()
```

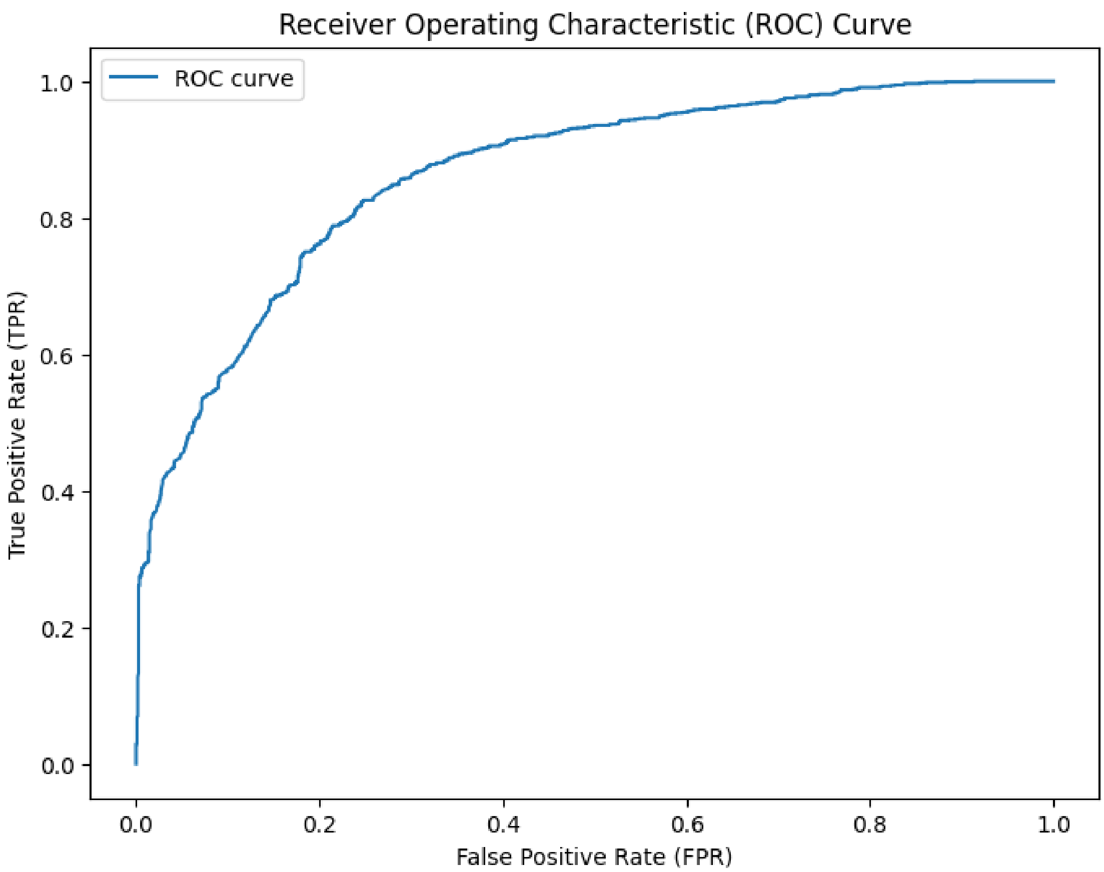


[21]:
```
TP = confusion[0][0]
FP = confusion[0][1]
FN = confusion[1][0]
TN = confusion[1][1]
precision = TP/(TP + FP)
recall = TP/(TP+FN)
print("The accuracy is ",precision)
print("The recall is ",recall)
print("The F1_score is ",2*(precision*recall)/(precision+recall))
```

```
The accuracy is  0.7603978300180831
The recall is  0.8024809160305344
The F1_score is  0.7808727948003714
```

The interpreation of those values show that the precision of good prediction is about 76.8%, the recall 79.7% which is pretty good in our case because that mean we don't reject the one who has been truly transported. The $F_1 - score$ is the harmonic mean that show a good balance between the fact that we don't want to consider the one who has been transported non-transporter and the one who has not been transported as transported.

[22]:
```
from sklearn.metrics import roc_curve, roc_auc_score

y_pred_proba = modele_logit.predict_proba(X_test)[:, 1]

fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label='ROC curve')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()

auc_roc = roc_auc_score(y_test, y_pred_proba)
print(f'AUC-ROC: {auc_roc}')
```




```
AUC-ROC: 0.863104450178852
```

AUC-ROC is close to 1 we can say that the model is pretty good

[23]:
```
import statsmodels.api as sm

col = ['HomePlanet_Earth',
'HomePlanet_Europa',
'HomePlanet_Mars',
'Destination_0',
'Destination_1',
'Destination_2',
'VIP_False',
'VIP_True',
'CryoSleep_False',
'CryoSleep_True' ]

for i in col:
    X_train[i] = X_train[i].astype(int)
```

[24]:
```python
import statsmodels.api as sm

model = sm.Logit(y_train, X_train)
result = model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.433557
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:            Transported   No. Observations:                 4404
Model:                          Logit   Df Residuals:                     4391
Method:                           MLE   Df Model:                           12
Date:                Fri, 17 Nov 2023   Pseudo R-squ.:                  0.3744
Time:                        01:18:05   Log-Likelihood:                -1909.4
converged:                       True   LL-Null:                       -3052.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
=====
                        coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Age                  -0.0084      0.003     -3.008      0.003      -0.014
-0.003
RoomService          -0.0014      0.000    -11.174      0.000      -0.002
-0.001
FoodCourt             0.0005   5.18e-05      9.414      0.000       0.000
0.001
ShoppingMall          0.0006   9.73e-05      6.451      0.000       0.000
0.001
Spa                  -0.0020      0.000    -13.455      0.000      -0.002
-0.002
VRDeck               -0.0019      0.000    -13.493      0.000      -0.002
-0.002
HomePlanet_Earth     -0.7409        nan        nan        nan         nan
nan
HomePlanet_Europa     1.5768        nan        nan        nan         nan
nan
HomePlanet_Mars      -0.0810        nan        nan        nan         nan
nan
Destination_0         0.0320        nan        nan        nan         nan
nan
Destination_1         0.6164        nan        nan        nan         nan
nan
Destination_2         0.1065        nan        nan        nan         nan
```

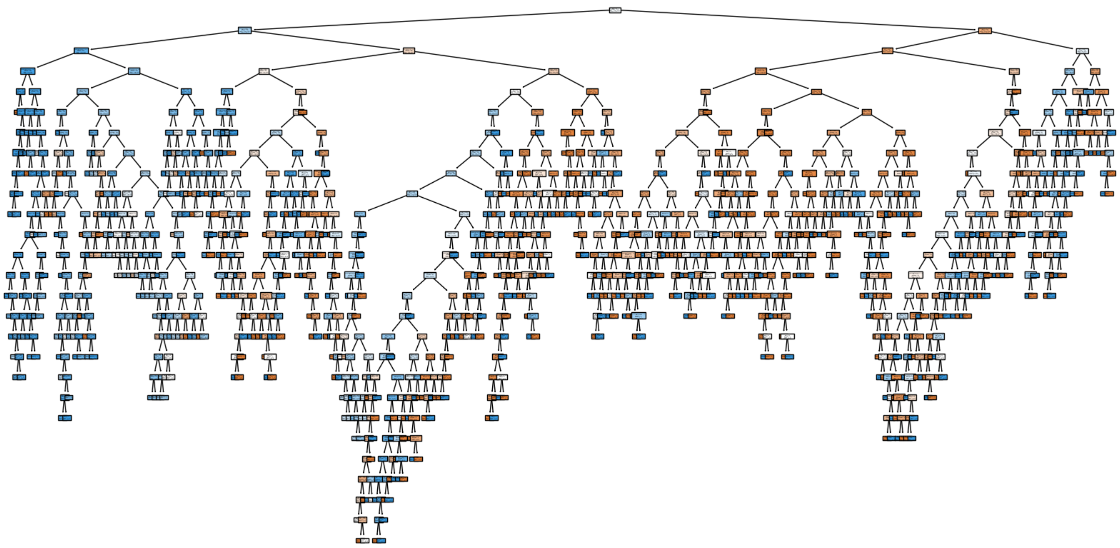
```
nan
VIP_False              0.7707        nan        nan        nan         nan
nan
VIP_True              -0.0158        nan        nan        nan         nan
nan
CryoSleep_False       -0.2503        nan        nan        nan         nan
nan
CryoSleep_True         1.0051        nan        nan        nan         nan
nan
================================================================================
=====
```

Nevertheless we see that the Log-Likelihood and the Pseudo R_square is low, the model is most likely not fitted to be used in this circumstance, which is predicting who is transported or not. Thoses variables are maybe not good enough to explain our target variables.

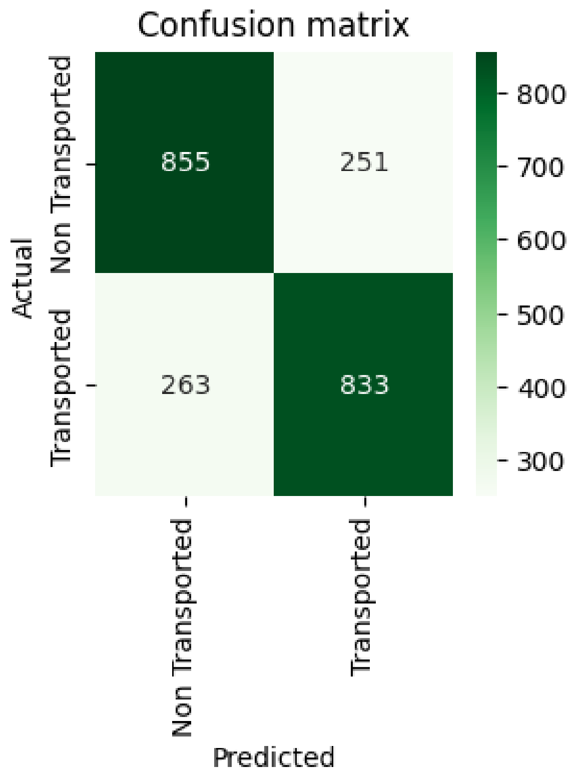
1.6 Random Forest

[25]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import plot_tree
```

[26]:
```
import matplotlib.pyplot as plt

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

tree_to_plot = model.estimators_[0]

plt.figure(figsize=(20, 10))
plot_tree(tree_to_plot, filled=True, feature_names=X_train.columns,␣
 ↪class_names=['Class 0', 'Class 1'])
plt.show()
```



[27]:
```
y_pred = model.predict(X_test)
confusion = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(3, 3))
sns.heatmap(confusion, annot=True, fmt='d', cmap='Greens', xticklabels=['Non␣
↪Transported', 'Transported'], yticklabels=['Non Transported', 'Transported'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion matrix')
plt.show()
```

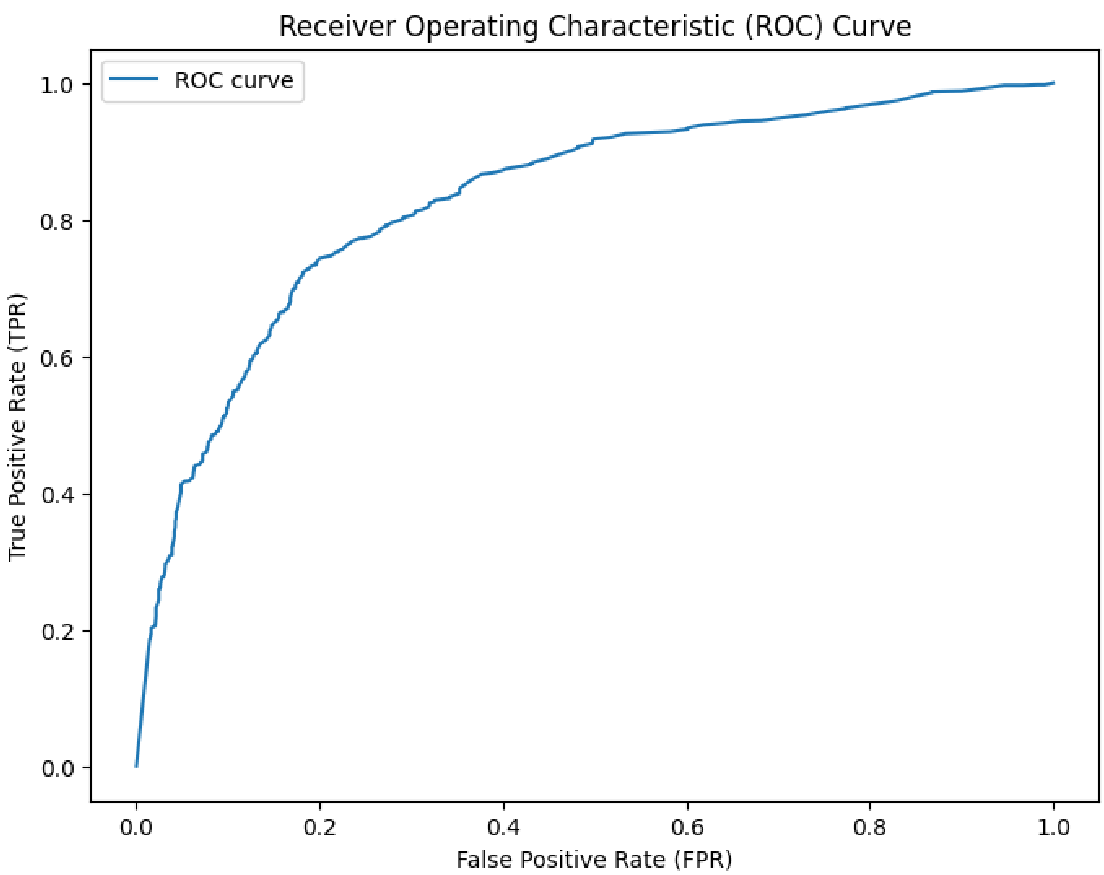


[28]:
```
TP = confusion[0][0]
FP = confusion[0][1]
FN = confusion[1][0]
TN = confusion[1][1]
precision = TP/(TP + FP)
recall = TP/(TP+FN)
print("The accuracy is ",precision)
print("The recall is ",recall)
print("The F1_score is ",2*(precision*recall)/(precision+recall))
```

```
The accuracy is  0.7730560578661845
The recall is  0.7647584973166368
The F1_score is  0.768884892086331
```

[29]:
```
from sklearn.metrics import roc_curve, roc_auc_score

y_pred_proba = model.predict_proba(X_test)[:, 1]

fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label='ROC curve')
```

```python
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()

auc_roc = roc_auc_score(y_test, y_pred_proba)
print(f'AUC-ROC: {auc_roc}')
```




```
AUC-ROC: 0.8302577348503848
```

1.7 Comparative

[30]:
```python
data = {
    'Metric': ['accuracy', 'recall', 'F1-Score', 'AUC-ROC'],
    'Logistic regression': ['77%', '76,5%', '77%', '86%'],
    'Random Forest': ['77%', '76,5%', '76,9%', '83%']
}
```

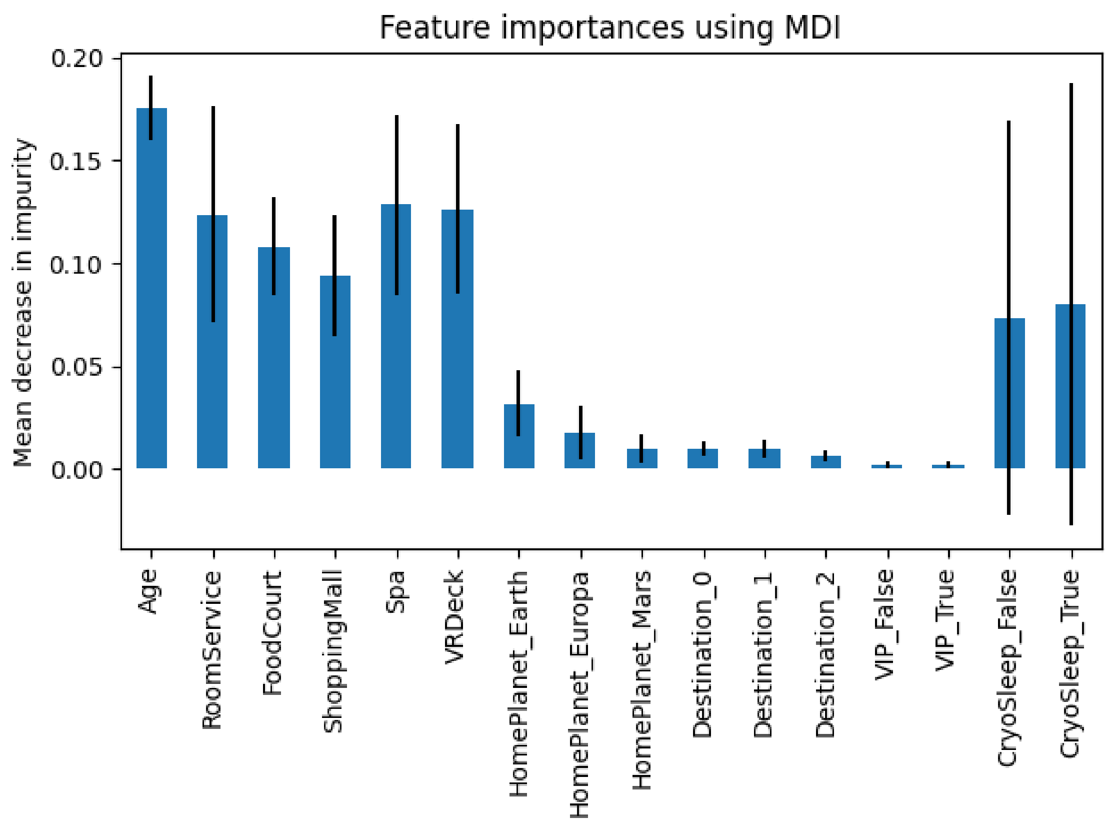
```
tableau_recap = pd.DataFrame(data)

print(tableau_recap.to_string(index=False))
```

```
  Metric Logistic regression Random Forest
accuracy                 77%           77%
  recall               76,5%         76,5%
F1-Score                 77%         76,9%
 AUC-ROC                 86%           83%
```

[31]:
```
import time

import numpy as np

start_time = time.time()
importances = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_], axis=0)
elapsed_time = time.time() - start_time

print(f"Elapsed time to compute the importances: {elapsed_time:.3f} seconds")
```

```
Elapsed time to compute the importances: 0.013 seconds
```

[32]:
```
forest_importances = pd.Series(importances, index=X_train.columns)

fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title("Feature importances using MDI")
ax.set_ylabel("Mean decrease in impurity")
fig.tight_layout()
```



[33]:
```
X_train
```

[33]:

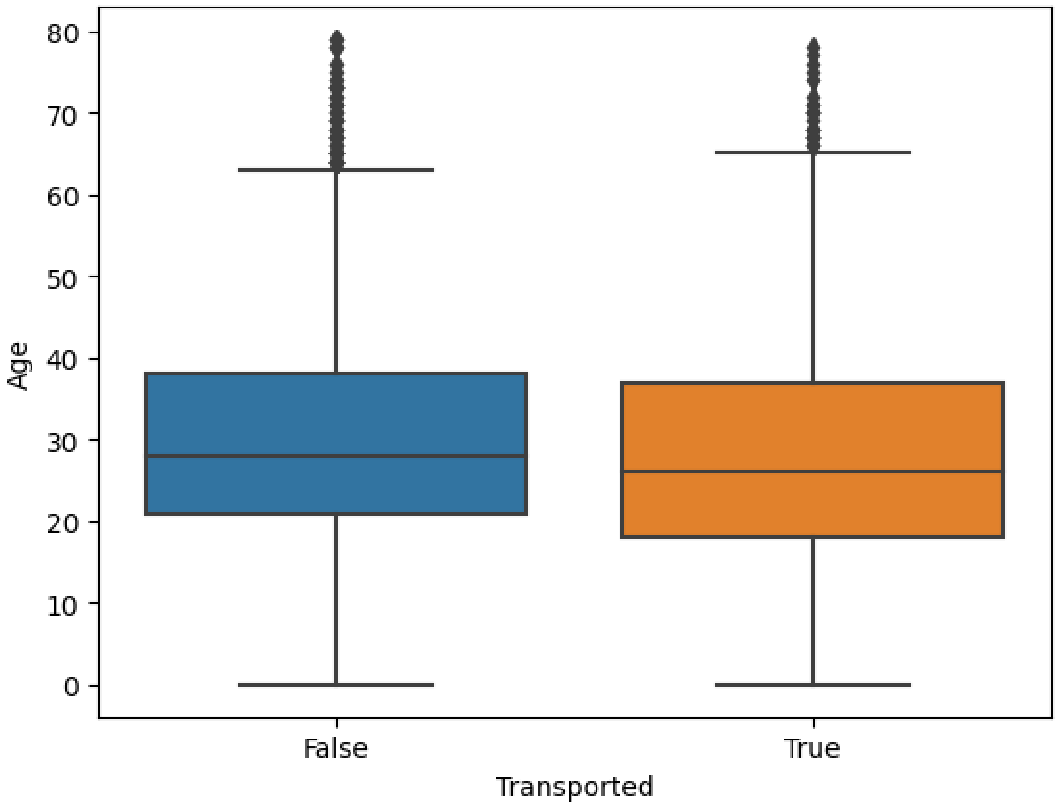
	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	
1528	35.0	0.0	0.0	0.0	0.0	0.0	\
357	48.0	0.0	111.0	0.0	1508.0	0.0	
1101	17.0	0.0	0.0	0.0	0.0	0.0	
4688	30.0	1120.0	0.0	0.0	0.0	794.0	
6415	24.0	1.0	0.0	0.0	691.0	0.0	
…	…	…	…	…	…	…	
6518	53.0	0.0	0.0	0.0	0.0	0.0	
4317	36.0	0.0	0.0	0.0	725.0	2.0	
2214	36.0	0.0	4756.0	0.0	7818.0	96.0	
3468	34.0	0.0	4.0	0.0	685.0	1779.0	
3642	14.0	2.0	881.0	0.0	578.0	214.0	

	HomePlanet_Earth	HomePlanet_Europa	HomePlanet_Mars	Destination_0	
1528	0	0	1	1	\
357	1	0	0	1	
1101	0	1	0	1	
4688	1	0	0	1	
6415	1	0	0	0	

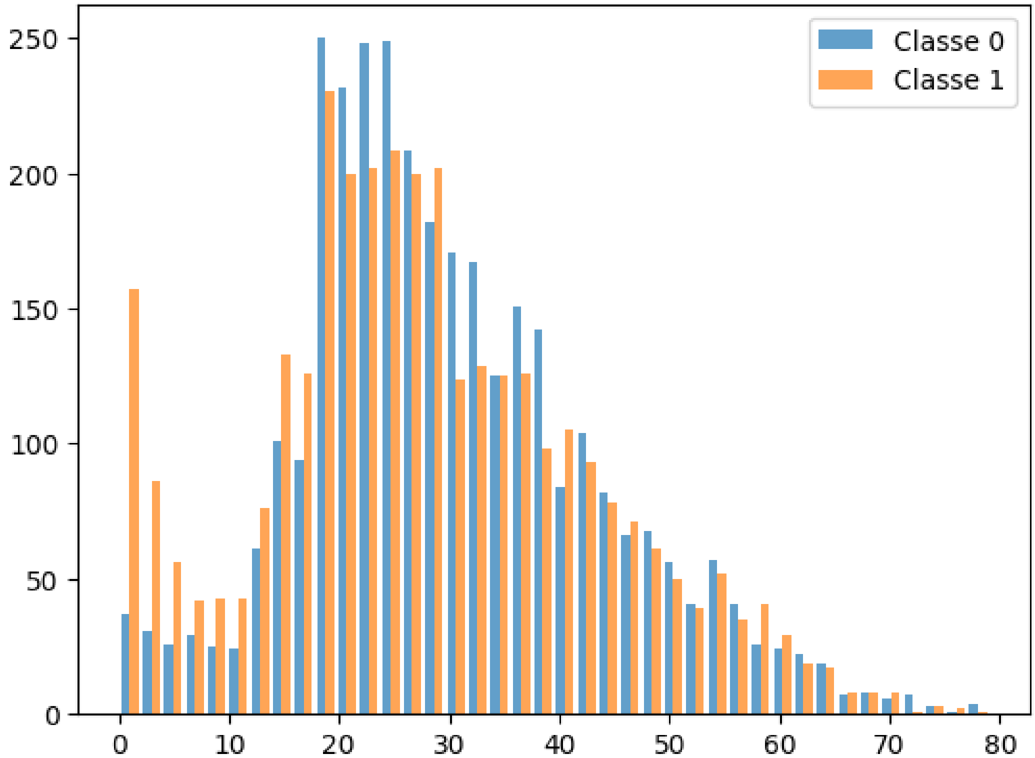
```
…                   …                   …                …              …
6518                0                   0                1              1
4317                1                   0                0              0
2214                0                   1                0              1
3468                0                   1                0              1
3642                0                   1                0              1

      Destination_1  Destination_2  VIP_False  VIP_True  CryoSleep_False  \
1528              0              0          1         0                0
357               0              0          1         0                1
1101              0              0          1         0                0
4688              0              0          1         0                1
6415              1              0          1         0                1
…               …            …          …        …                 …
6518              0              0          1         0                1
4317              1              0          1         0                1
2214              0              0          1         0                1
3468              0              0          0         1                1
3642              0              0          1         0                1

      CryoSleep_True
1528               1
357                0
1101               1
4688               0
6415               0
…                …
6518               0
4317               0
2214               0
3468               0
3642               0

[4404 rows x 16 columns]
```

[42]:
```python
sns.boxplot(x='Transported', y='Age', data=data_train)
plt.show()

from scipy.stats import ttest_ind, mannwhitneyu

group_0 = data_train[data_train['Transported'] == 0]['Age']
group_1 = data_train[data_train['Transported'] == 1]['Age']

t_stat, p_value_t = ttest_ind(group_0, group_1)
mwu_stat, p_value_mw = mannwhitneyu(group_0, group_1)

print(f"Test t : t-statistic = {t_stat}, p-value = {p_value_t}")
```

```
print(f"Test de Mann-Whitney : U-statistic = {mwu_stat}, p-value =␣
  ↪{p_value_mw}")

plt.hist([group_0, group_1], bins='auto', alpha=0.7, label=['Classe 0', 'Classe␣
  ↪1'])
plt.legend()
plt.show()
```




```
Test t : t-statistic = 6.73163661796557, p-value = 1.8187751348139O2e-11
Test de Mann-Whitney : U-statistic = 5965926.0, p-value = 4.1285598462864007e-11
```



1.8 Interpretation

Globaly the precision is very similar, we can even say that a logistic regression is maybe better nevertheless we saw before that the regression model is not that pertinent for this problem. But we can find all the transported passenger pretty much.