In [1]:

```
# import important libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as war
war.filterwarnings('ignore')
pd.set_option('max_columns',None)
import re
```

In [2]:

```
# load dataset
df = pd.read_csv('Data_Science_Internship - Dump.csv')
df.shape
```

Out[2]:

```
(46608, 16)
```

In [3]:

```
df.head()
```

Out[3]:

	Unnamed: 0		Agent_id	status	lost_reason	budg
0	0	1deba9e96f404694373de9749ddd1ca8aa7bb823145a6f...		LOST	Not responding	Na
1	1	299ae77a4ef350ae0dd37d6bba1c002d03444fb1edb236...		LOST	Low budget	Na
2	2	c213697430c006013012dd2aca82dd9732aa0a1a6bca13...		LOST	Not responding	£12 £1 P We
3	3	eac9815a500f908736d303e23aa227f0957177b0e6756b...		LOST	Low budget	0
4	4	1deba9e96f404694373de9749ddd1ca8aa7bb823145a6f...		LOST	Junk lead	Na

Create New Dataframe With Important Column

In [4]:

```
# create new dataframe
dfx = df[['status', 'lost_reason', 'budget', 'lease',
       'movein','room_type']]
dfx.head()
```

Out[4]:

	status	lost_reason	budget	lease	movein	room_type
0	LOST	Not responding	NaN	NaN	NaN	NaN
1	LOST	Low budget	NaN	NaN	NaN	NaN
2	LOST	Not responding	£121 - £180 Per Week	Full Year Course Stay 40 - 44 weeks	31/08/22	Ensuite
3	LOST	Low budget	0-0	0	NaN	NaN
4	LOST	Junk lead	NaN	NaN	NaN	NaN

Handle Null And Duplicate Values

In [5]:

```
# find null values
print(f'Null Values:-\n\n{dfx.isnull().sum()}\n\nNull Values In Percentage:-\n\n{(dfx.is
```

```
Null Values:-

status             0
lost_reason     3364
budget          3700
lease           2341
movein         13638
room_type      23547
dtype: int64

Null Values In Percentage:-

status          0.000000
lost_reason     7.217645
budget          7.938551
lease           5.022743
movein         29.261071
room_type      50.521370
dtype: float64
```

In [6]:

```
# fill null values with forward fill method
dfx.lost_reason.fillna(method='ffill',inplace=True)
dfx.budget.fillna(method='ffill',inplace=True)
dfx.lease.fillna(method='ffill',inplace=True)
dfx.movein.fillna(method='ffill',inplace=True)
```

In [7]:

```
# check null values after cleaning
dfx.isnull().sum()
```

Out[7]:

```
status             0
lost_reason        0
budget             2
lease              2
movein             2
room_type      23547
dtype: int64
```

We Can See That Some Null Values Are Still Present Because We Are Using Forward Fill Method, So We Have To Fill It With Backward Fill Method

In [8]:

```
# fill null values with backward fill method
dfx.budget.fillna(method='bfill',inplace=True)
dfx.lease.fillna(method='bfill',inplace=True)
dfx.movein.fillna(method='bfill',inplace=True)
```

In [9]:

```
# drop room type column because it contains more than 50% of null values
dfx.drop('room_type',inplace = True, axis = 1)
```

In [10]:

```
# check duplicate values
dfx.duplicated().sum()
print(f'Duplicate Values:-\n\n{dfx.duplicated().sum()}\n\nDuplicate Values In Percentage
```

```
Duplicate Values:-

20612

Duplicate Values In Percentage:-

44.22416752488843
```

We Can Clearly See More Then 44% Of Duplicate Data Is Present In Our Dataset, For Better Model Building First We Have To Clean It

In [11]:

```
# drop duplicate values
dfx.drop_duplicates(inplace=True)
```

In [12]:

```
dfx.head()
```

Out[12]:

	status	lost_reason	budget	lease	movein
0	LOST	Not responding	£121 - £180 Per Week	Full Year Course Stay 40 - 44 weeks	31/08/22
1	LOST	Low budget	£121 - £180 Per Week	Full Year Course Stay 40 - 44 weeks	31/08/22
3	LOST	Low budget	0-0	0	31/08/22
4	LOST	Junk lead	0-0	0	31/08/22
5	LOST	Wants private accommodation	120	semester-stay	31/08/22

Feature Engineering

In [13]:

```
# create copy of dataframe
df2 = dfx.copy()
```

In [14]:

```
# take values whose count is more than 100 in lost reason column
lost = df2.lost_reason.value_counts()
lost_100 = lost[lost>100]
```

In [15]:

```
# map values to others whose count is less than 100
df2['lost_reason'] = df2['lost_reason'].map(lambda x: 'Others' if x not in lost_100 else
```

In [16]:

```
df2.head(2)
```

Out[16]:

	status	lost_reason	budget	lease	movein
0	LOST	Not responding	£121 - £180 Per Week	Full Year Course Stay 40 - 44 weeks	31/08/22
1	LOST	Low budget	£121 - £180 Per Week	Full Year Course Stay 40 - 44 weeks	31/08/22

In [17]:

```
# extract numbers from budget column
list_1 = []
budget_1 = []
budget_2 = []
for i in df2.budget:
    for x in i:
        if x == '£':
            i = i.replace(x,'')
    list_1.append(i)
for data in list_1:
    splt = data.split('-')
    if len(splt) == 2:
        if re.findall(r'\d+',splt[0]) and re.findall(r'\d+',splt[1]):
            budget_1.append(re.findall(r'\d+',splt[0]))
            budget_2.append(re.findall(r'\d+',splt[1]))
        else:
            budget_1.append(0)
            budget_2.append(0)

    elif len(splt) == 1:
        budget_1.append(re.findall(r'\d+',splt[0]))
        budget_2.append(0)

    else:
        budget_1.append(0)
        budget_2.append(0)

# create new columns for budget
df2['budget_1'] = budget_1
df2['budget_2'] = budget_2
```

In [18]:

```
# change datatype of budget_1 and budget_2 column
df2['budget_1'] = df2['budget_1'].str[0].fillna(0).astype(float).astype(int)
df2['budget_2'] = df2['budget_2'].str[0].fillna(0).astype(float).astype(int)
```

In [19]:

```
# create new column for average budget
new_budget = []
for i,j in zip(df2.budget_1,df2.budget_2):
    if i>0 and j>0:
        val = (i+j)/2
        new_budget.append(int(val))
    else:
        new_budget.append(int(i)+int(j))

df2['new_budget'] = new_budget
```

In [20]:

```
df2.head()
```

Out[20]:

	status	lost_reason	budget	lease	movein	budget_1	budget_2	new_budget
0	LOST	Not responding	£121 - £180 Per Week	Full Year Course Stay 40 - 44 weeks	31/08/22	121	180	150
1	LOST	Low budget	£121 - £180 Per Week	Full Year Course Stay 40 - 44 weeks	31/08/22	121	180	150
3	LOST	Low budget	0-0	0	31/08/22	0	0	0
4	LOST	Junk lead	0-0	0	31/08/22	0	0	0
5	LOST	Wants private accommodation	120	semester-stay	31/08/22	120	0	120

In [21]:

```
# extract numbers from lease column
lease_1 = []
lease_2 = []
for data in df2.lease:
    splt = data.split('-')
    if len(splt) == 2:
        if re.findall(r'\d+',splt[0]) and re.findall(r'\d+',splt[1]):
            lease_1.append(re.findall(r'\d+',splt[0]))
            lease_2.append(re.findall(r'\d+',splt[1]))
        else:
            lease_1.append(0)
            lease_2.append(0)

    elif len(splt) == 1:
        lease_1.append(re.findall(r'\d+',splt[0]))
        lease_2.append(0)

    else:
        lease_1.append(0)
        lease_2.append(0)
df2['lease_1'] = lease_1
df2['lease_2'] = lease_2
```

In [22]:

```
# change datatype of lease_1 and lease_2 column
df2['lease_1'] = df2['lease_1'].str[0].fillna(0).astype(float).astype(int)
df2['lease_2'] = df2['lease_2'].str[0].fillna(0).astype(float).astype(int)
```

In [23]:

```
# create new column for average lease
new_lease = []
for i,j in zip(df2.lease_1,df2.lease_2):
    if i>0 and j>0:
        val = (i+j)/2
        new_lease.append(int(val))
    else:
        new_lease.append(int(i)+int(j))

df2['new_lease'] = new_lease
```

In [24]:

```
df2.head()
```

Out[24]:

	status	lost_reason	budget	lease	movein	budget_1	budget_2	new_budget	leas
0	LOST	Not responding	£121 - £180 Per Week	Full Year Course Stay 40 - 44 weeks	31/08/22	121	180	150	
1	LOST	Low budget	£121 - £180 Per Week	Full Year Course Stay 40 - 44 weeks	31/08/22	121	180	150	
3	LOST	Low budget	0-0	0	31/08/22	0	0	0	
4	LOST	Junk lead	0-0	0	31/08/22	0	0	0	
5	LOST	Wants private accommodation	120	semester-stay	31/08/22	120	0	120	

In [25]:

```
# split day month and year from the movein column
day = []
month = []
year = []
for dd in df2['movein']:
    spl = dd.split('/')
    day.append(int(spl[0]))
    month.append(int(spl[1]))
    year.append(int(spl[2]))

# create new columns for day month and year column
df2['day'] = day
df2['month'] = month
df2['year'] = year
```

In [26]:

```
# create new dataframe for selected columns
df3 = df2[['status', 'lost_reason','new_budget','new_lease','day', 'month', 'year']]
df3.head()
```

Out[26]:

	status	lost_reason	new_budget	new_lease	day	month	year
0	LOST	Not responding	150	42	31	8	22
1	LOST	Low budget	150	42	31	8	22
3	LOST	Low budget	0	0	31	8	22
4	LOST	Junk lead	0	0	31	8	22
5	LOST	Wants private accommodation	120	0	31	8	22

In [27]:

```
# data encoding for status column
df3['status'] = df3['status'].map({'LOST':0, 'WON':1, 'OPPORTUNITY':2, 'CONTACTED':3, 'P
```

In [28]:

```
# one hot encoding of lost reason column
data = pd.get_dummies(df3['lost_reason'],drop_first=True)
```

In [29]:

```
# concat data with dataframe
df3 = pd.concat([df3,data],axis = 1)
```

In [30]:

```
# drop lost reason column
df3.drop('lost_reason',axis = 1,inplace = True)
```

In [31]:

```
df3.head(2)
```

Out[31]:

	status	new_budget	new_lease	day	month	year	Booked with competitor	Booked with manager	Distance issue	Inadeq d
0	0	150	42	31	8	22	0	0	0	
1	0	150	42	31	8	22	0	0	0	

In [32]:

```
# take only those data where status is 0 or 1
df4 = df3[(df3['status'] == 0) | (df3['status'] == 1)]
```

In [33]:

```
# check target column distribution
df4.status.value_counts()
print(f'Status Distribution:-\n\n{df4.status.value_counts()}\n\nStatus Distribution In P
```

```
Status Distribution:-

0    23155
1     2563
Name: status, dtype: int64

Status Distribution In Percentage:-

0    90.034217
1     9.965783
Name: status, dtype: float64
```

We Can Clearly See Our Data Set Is Imbalanced, In Target Column More Than 90% Data Comes With 0 And Only 9% Data Comes With 1, So Before Model Building We Have To Balance It

Data Spliting

In [34]:

```
X = df4.drop('status',axis = 1)
y = df4['status']
```

Data Balancing

In [35]:

```
# data balance with smotetomek
from imblearn.over_sampling import SMOTE
smt = SMOTE(sampling_strategy=0.80)
X, y = smt.fit_resample(X, y)
```

In [36]:

```
# train test split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

Data Scaling

In [37]:

```
# data scaling with standard scaler
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Model Building

In [38]:

```
# train models and check accuracy score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from xgboost import XGBClassifier
from sklearn.metrics import roc_auc_score

ml_models = [("Logistic Regression",LogisticRegression()),
       ("KNN Classifier", KNeighborsClassifier()),
       ("RandomForest",RandomForestClassifier()),
      ("AdaBoost",AdaBoostClassifier()),
      ("XGBoost",XGBClassifier())]
for name,model in ml_models:
    model.fit(X_train,y_train)
    y_pred1 = model.predict(X_train)
    train_acc = roc_auc_score(y_train,y_pred1)
    y_pred2 = model.predict(X_test)
    test_acc = roc_auc_score(y_test,y_pred2)
    print(f"For {name}:-\nThe Training Accuracy is: {train_acc}\nThe Testing Accuracy is
    print("--"*40)
```

```
For Logistic Regression:-
The Training Accuracy is: 0.8396069026743475
The Testing Accuracy is: 0.8312025927059925
--------------------------------------------------------------------------
------
For KNN Classifier:-
The Training Accuracy is: 0.904608760620295
The Testing Accuracy is: 0.8635268345408669
--------------------------------------------------------------------------
------
For RandomForest:-
The Training Accuracy is: 0.959514062407828
The Testing Accuracy is: 0.8830469383816851
--------------------------------------------------------------------------
------
For AdaBoost:-
The Training Accuracy is: 0.8375808121997035
The Testing Accuracy is: 0.8299356555952416
--------------------------------------------------------------------------
------
For XGBoost:-
The Training Accuracy is: 0.8826401039357186
The Testing Accuracy is: 0.8647274823322333
--------------------------------------------------------------------------
------
```

We Can Clearly See XG Boost Classifier Is Giving The Best Result, So Will Take XG Boost Classifier As Our Final Model

In [39]:

```
# find model best parameters
from sklearn.model_selection import RandomizedSearchCV
xgb = XGBClassifier()
parameters = {"n_estimators": [50,100,150,200,250,300,350,400],
              "max_depth": np.arange(2,10),
              "learning_rate": np.arange(0.01,0.1,0.02),
              'subsample': np.arange(0.5, 1.0, 0.1),
              'colsample_bytree': np.arange(0.4, 1.0, 0.1),
              'colsample_bylevel': np.arange(0.4, 1.0, 0.1)}
ran_xgb = RandomizedSearchCV(xgb, parameters, cv = 5, random_state= 42)
ran_xgb.fit(X_train,y_train)
ran_xgb.best_params_
```

Out[39]:

```
{'subsample': 0.7999999999999999,
 'n_estimators': 250,
 'max_depth': 8,
 'learning_rate': 0.08999999999999998,
 'colsample_bytree': 0.7,
 'colsample_bylevel': 0.8999999999999999}
```

Tune Model With Best Parameters

In [40]:

```
# tune model
xgb = XGBClassifier(subsample = 0.79, n_estimators = 250,
                    max_depth = 8, learning_rate = 0.09,
                    colsample_bytree = 0.7, colsample_bylevel = 0.89)
xgb.fit(X_train,y_train)
y_pred = xgb.predict(X_test)
score = roc_auc_score(y_test,y_pred)*100
score
```

Out[40]:

```
87.3268796946303
```

We Are Getting More Then 87% Of Accuracy Score With XG Boost Model

In [41]:

```
# feature impotance
feature_importance = pd.DataFrame({'Feature Importance':xgb.feature_importances_*100},in
feature_importance = feature_importance['Feature Importance'].sort_values(ascending=Fals
feature_importance
```

Out[41]:

```
Not a student                  6.292204
Low budget                     6.089373
Wants private accommodation    5.894108
Just Enquiring                 5.318133
Booked with manager            5.317873
Short stay                     5.053607
Not interested                 5.008458
Not responding                 4.853704
Semester stay                  4.250910
Inadequate details             4.212471
Junk lead                      4.181040
Low availability               4.156651
No supply                      4.023013
Booked with competitor         3.970712
Repeat lead                    3.924649
year                           3.857918
Distance issue                 3.611315
Not going to university        3.585965
```

Check Performance Of Model

In [42]:

```
# create dummy data for check performance of model
dummy = X_train[0].reshape(1,-1)
```

In [43]:

```
# scale dummy data
dummy = sc.transform(dummy)
```

In [44]:

```
# check prediction of dummy data
xgb.predict(dummy)
```

Out[44]:

```
array([0])
```

We Can See, We Are Getting Good Prediction With Our Final Model

Important Points:-

1. In the given dataset most of the columns contain totally null values.

2. Dataset came with a large amount of null and duplicate values.
3. In the budget and lease column there were text values with numbers, I had to first extract numbers from the text.
4. After extracting values I generated average values for them.
5. Movien column was contain date and for model building I had split day, month and year from movien column.
6. I have done label encoding for the status column and one hot encoding for the lost reason column.
7. Before model building I removed values from the status column except 0 and 1.
8. The dataset was imbalanced, so before model building, I balanced it with the SMOT library.
9. After data balancing I did data scaling.
10. I used different - different ML models for prediction and XG Boost gave the best result.
11. I used auroc score for evaluation because the dataset was imbalanced.
12. In the documentation it was mentioned that we have to take all columns as categorical but the budget and lease columns had continuous values after extracting numbers, that's why I toot its average and consider it to be int.

I Hope You Will Like My Work

Thank You :)

In [45]:

```
df2.head()
```

Out[45]:

s	lost_reason	budget	lease	movein	budget_1	budget_2	new_budget	lease_1	leas
T	Not responding	£121 - £180 Per Week	Full Year Course Stay 40 - 44 weeks	31/08/22	121	180	150	40	
T	Low budget	£121 - £180 Per Week	Full Year Course Stay 40 - 44 weeks	31/08/22	121	180	150	40	
T	Low budget	0-0	0	31/08/22	0	0	0	0	
T	Junk lead	0-0	0	31/08/22	0	0	0	0	
T	Wants private accommodation	120	semester-stay	31/08/22	120	0	120	0	

In [ ]: