In [1]:
```
import numpy as np
import pandas as pd


import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/hotel-reservations-classification-dataset/Hotel Reservations.csv
```

In [2]:
```
import pandas as pd
```

In [3]:
```
df = pd.read_csv("/kaggle/input/hotel-reservations-classification-dataset/Hotel Res
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36275 entries, 0 to 36274
Data columns (total 19 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   Booking_ID                            36275 non-null  object 
 1   no_of_adults                          36275 non-null  int64  
 2   no_of_children                        36275 non-null  int64  
 3   no_of_weekend_nights                  36275 non-null  int64  
 4   no_of_week_nights                     36275 non-null  int64  
 5   type_of_meal_plan                     36275 non-null  object 
 6   required_car_parking_space            36275 non-null  int64  
 7   room_type_reserved                    36275 non-null  object 
 8   lead_time                             36275 non-null  int64  
 9   arrival_year                          36275 non-null  int64  
 10  arrival_month                         36275 non-null  int64  
 11  arrival_date                          36275 non-null  int64  
 12  market_segment_type                   36275 non-null  object 
 13  repeated_guest                        36275 non-null  int64  
 14  no_of_previous_cancellations          36275 non-null  int64  
 15  no_of_previous_bookings_not_canceled  36275 non-null  int64  
 16  avg_price_per_room                    36275 non-null  float64
 17  no_of_special_requests                36275 non-null  int64  
 18  booking_status                        36275 non-null  object 
dtypes: float64(1), int64(13), object(5)
memory usage: 5.3+ MB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Booking_ID                            0
no_of_adults                          0
no_of_children                        0
no_of_weekend_nights                  0
no_of_week_nights                     0
type_of_meal_plan                     0
required_car_parking_space            0
room_type_reserved                    0
lead_time                             0
arrival_year                          0
arrival_month                         0
arrival_date                          0
market_segment_type                   0
repeated_guest                        0
no_of_previous_cancellations          0
no_of_previous_bookings_not_canceled  0
avg_price_per_room                    0
no_of_special_requests                0
booking_status                        0
dtype: int64
```

In [6]:
```
df.head()
```

Out[6]:

	Booking_ID	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_me
0	INN00001	2	0	1	2	Me
1	INN00002	2	0	2	3	Not
2	INN00003	1	0	2	1	Me
3	INN00004	2	0	0	2	Me
4	INN00005	2	0	1	1	Not

In [7]:
```
from sklearn.preprocessing import LabelEncoder
l1=LabelEncoder()
```

In [8]:
```
df["Booking_ID"]=l1.fit_transform(df["Booking_ID"])
df["type_of_meal_plan"]=l1.fit_transform(df["type_of_meal_plan"])
df["room_type_reserved"]=l1.fit_transform(df["room_type_reserved"])
df["market_segment_type"]=l1.fit_transform(df["market_segment_type"])
df["booking_status"]=l1.fit_transform(df["booking_status"])
```

In [9]:
```
df
```

Out[9]:

	Booking_ID	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_
0	0	2	0	1	2	
1	1	2	0	2	3	
2	2	1	0	2	1	
3	3	2	0	0	2	
4	4	2	0	1	1	
...	...	...	...	...	...	
36270	36270	3	0	2	6	
36271	36271	2	0	1	3	
36272	36272	2	0	2	6	
36273	36273	2	0	0	3	
36274	36274	2	0	1	2	

36275 rows × 19 columns

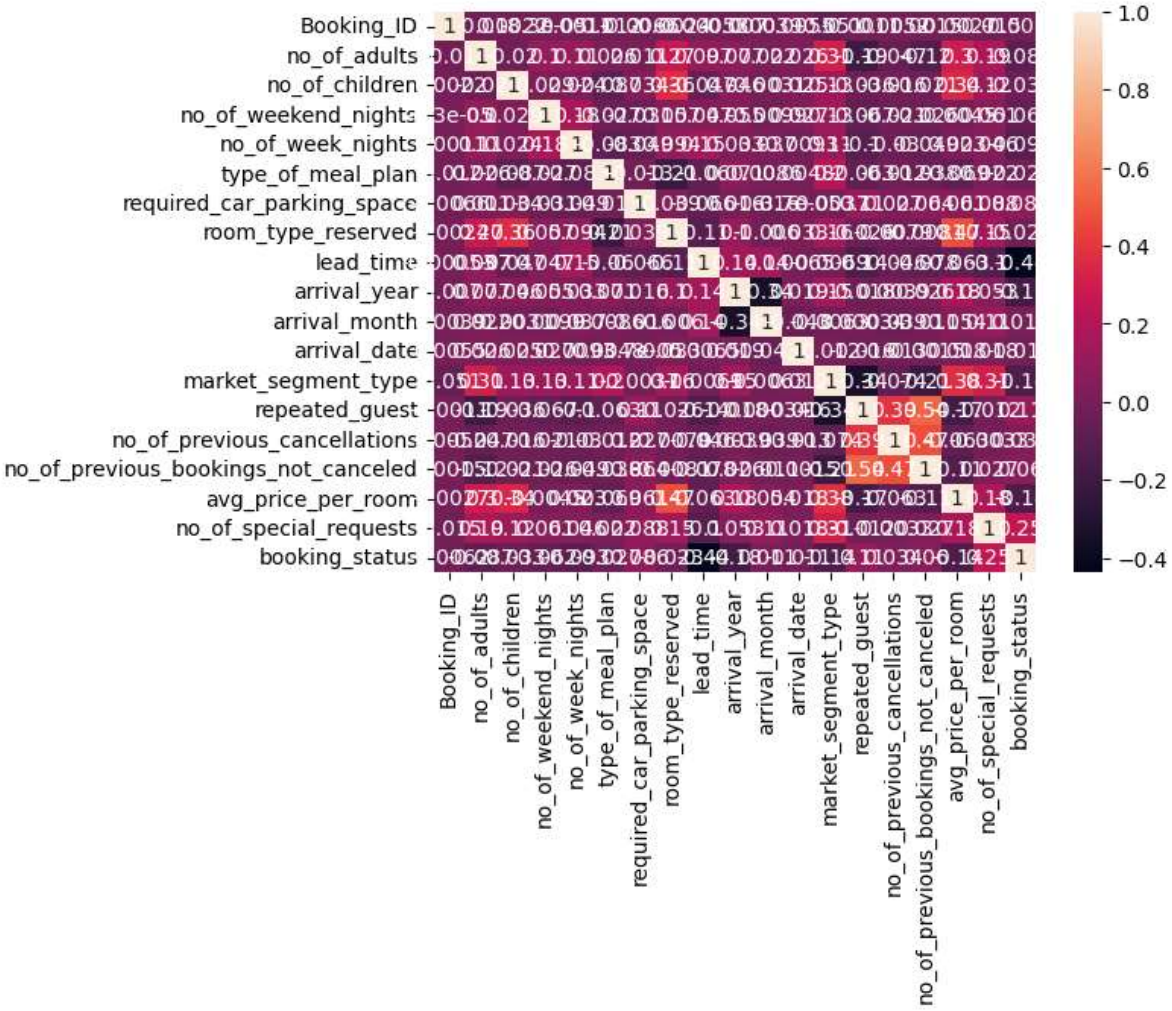
In [10]:
```
df.corr()
```

Out[10]:

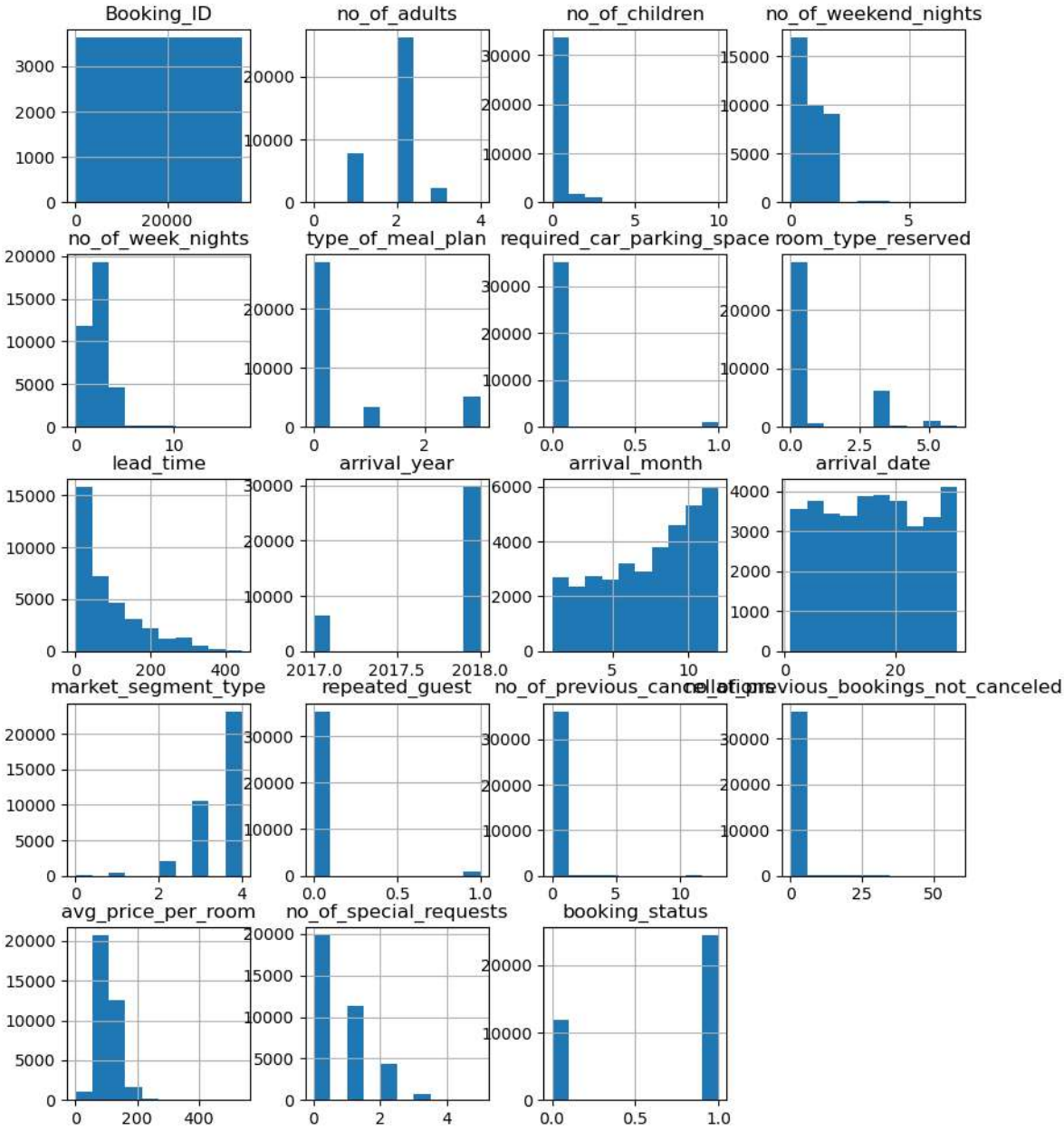
	Booking_ID	no_of_adults	no_of_children	no_of_weekend_ni
Booking_ID	1.000000	-0.009994	0.002153	-0.00
no_of_adults	-0.009994	1.000000	-0.019787	0.10
no_of_children	0.002153	-0.019787	1.000000	0.02
no_of_weekend_nights	-0.000083	0.103316	0.029478	1.00
no_of_week_nights	0.001107	0.105622	0.024398	0.17
type_of_meal_plan	-0.012476	0.025555	-0.086764	-0.02
required_car_parking_space	-0.006579	0.011429	0.034244	-0.03
room_type_reserved	-0.002375	0.270348	0.364073	0.05
lead_time	-0.000535	0.097287	-0.047091	0.04
arrival_year	-0.006980	0.076719	0.045983	0.05
arrival_month	0.003942	0.021841	-0.003076	-0.00
arrival_date	0.005462	0.026338	0.025482	0.02
market_segment_type	-0.051493	0.314103	0.130618	0.12
repeated_guest	-0.001076	-0.192277	-0.036348	-0.06
no_of_previous_cancellations	0.005227	-0.047426	-0.016390	-0.02
no_of_previous_bookings_not_canceled	0.001465	-0.119166	-0.021189	-0.02
avg_price_per_room	-0.002687	0.296886	0.337728	-0.00
no_of_special_requests	-0.014795	0.189401	0.124486	0.06
booking_status	-0.006237	-0.086920	-0.033078	-0.06

In [11]:
```
import seaborn as sns
sns.heatmap(data=df.corr(),annot=True)
```

Out[11]:
```
<AxesSubplot:>
```



In [12]:
```
import matplotlib.pyplot as plt
df.hist(figsize=(10,12))
plt.show()
```



In [13]:
```
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

In [ ]:
```
```

In [14]:
```
from imblearn.over_sampling import SMOTE
sm= SMOTE()
x_data,y_data = sm.fit_resample(x,y)
```

In [15]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_data,y_data,test_size=0.18,random_
```

In [16]:
```
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
x=std.fit_transform(x_train)
```

In [17]:
```
#from sklearn.tree import DecisionTreeClassifier
#dt = DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=12)
#dt.fit(x_train,y_train)


#from sklearn.linear_model import LogisticRegression
```

```
#lg = LogisticRegression()
#lg.fit(x_train,y_train)


#from sklearn.svm import SVC
#svc=SVC(kernel='linear',random_state=11)
#svc.fit(x_train,y_train)

#from sklearn.neighbors import KNeighborsClassifier
#knn = KNeighborsClassifier()
#knn.fit(x_train,y_train)
```

In [ ]:

In [18]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=40)


from sklearn.linear_model import LogisticRegression
lg = LogisticRegression()


from sklearn.naive_bayes import GaussianNB
nvb = GaussianNB()


from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()


from sklearn.ensemble import VotingClassifier
vot = VotingClassifier(estimators=[("DT",dt),("LG",lg),("NVB",nvb),("KNN",knn)])
vot.fit(x_train,y_train)


from sklearn.metrics import accuracy_score
y_pred_vot = vot.predict(x_test)


from sklearn.metrics import accuracy_score
ac_vot=accuracy_score(y_test,y_pred_vot)
ac_vot
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Conv
ergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

Out[18]:
```
0.7593668147135861
```

In [19]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=12)
dt.fit(x_train,y_train)
```

Out[19]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=12)
```

In [20]:
```
from sklearn.linear_model import LogisticRegression
lg = LogisticRegression()
```

```
lg.fit(x_train,y_train)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Conv
ergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

Out[20]:
```
LogisticRegression()
```

In [21]:
```
#from sklearn.svm import SVC
#svc=SVC(kernel='linear',random_state=11)
#svc.fit(x_train,y_train)
```

In [22]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
```

Out[22]:
```
KNeighborsClassifier()
```

In [23]:
```
y_pred_dt=dt.predict(x_test)
from sklearn.metrics import accuracy_score
ac_dt=accuracy_score(y_test,y_pred_dt)*100
print("ac_dt = ",ac_dt)
print(f'------------------')


y_pred_lg=lg.predict(x_test)
from sklearn.metrics import accuracy_score
ac_lg=accuracy_score(y_test,y_pred_lg)*100
print("ac_lg = ",ac_lg)
print(f'------------------')



#y_pred_svm=svc.predict(x_test)
#from sklearn.metrics import accuracy_score
##ac_svm=accuracy_score(y_test,y_pred_svm)*100
#print("ac_svm = ",ac_svm)
#print(f'------------------')

y_pred_knn=knn.predict(x_test)
from sklearn.metrics import accuracy_score
ac_knn=accuracy_score(y_test,y_pred_knn)*100
print("ac_knn = ",ac_knn)
print(f'------------------')
```

```
ac_dt =  85.05864935656531
------------------
ac_lg =  71.27889761986106
------------------
ac_knn =  73.19211934859355
------------------
```

In [24]:
```
from sklearn.ensemble import RandomForestClassifier
ran = RandomForestClassifier(n_estimators=40, max_features=4)
ran.fit(x_train,y_train)


from sklearn.metrics import accuracy_score
```

```
y_pred_ran = ran.predict(x_test)


from sklearn.metrics import accuracy_score
ac_ran=accuracy_score(y_test,y_pred_ran)
ac_ran
```

Out[24]:
```
0.9221045439016058
```

In [25]:
```
from sklearn.ensemble import BaggingClassifier
bag = BaggingClassifier(base_estimator=dt,n_estimators=5,random_state=40)
bag.fit(x_train,y_train)


from sklearn.metrics import accuracy_score
y_pred_bag = bag.predict(x_test)


from sklearn.metrics import accuracy_score
ac_bag=accuracy_score(y_test,y_pred_bag)
ac_bag
```

Out[25]:
```
0.8509281403029267
```

In [26]:
```
print("ac_vot = ",ac_vot*100)
print(f'------------------')

print("ac_ran = ",ac_ran*100)
print(f'------------------')

print("ac_bag = ",ac_bag*100)
print(f'------------------')

print("ac_dt = ",ac_dt)
print(f'------------------')

print("ac_lg = ",ac_lg)
print(f'------------------')

print("ac_knn = ",ac_knn)
print(f'------------------')
```

```
ac_vot =  75.93668147135861
------------------
ac_ran =  92.21045439016058
------------------
ac_bag =  85.09281403029267
------------------
ac_dt =  85.05864935656531
------------------
ac_lg =  71.27889761986106
------------------
ac_knn =  73.19211934859355
------------------
```

**Report for random forest classification.

In [29]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_ran))
```

	precision	recall	f1-score	support
0	0.93	0.91	0.92	4383
1	0.91	0.94	0.92	4398
accuracy			0.92	8781
macro avg	0.92	0.92	0.92	8781
weighted avg	0.92	0.92	0.92	8781

In [30]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test,y_pred_ran)
print(confusion_matrix)
```

```
[[3982  401]
 [ 283 4115]]
```

Inference

Using DecisionTreeClassifier, the model's accuracy was around 92%; precision, recall, and f1-score were92%,92%, and 92% after the model had been trained.

In [ ]:
```

```