HOTEL RESERVATION PREDICTION

PHAM THI THI PHONG
CLASS: DA31

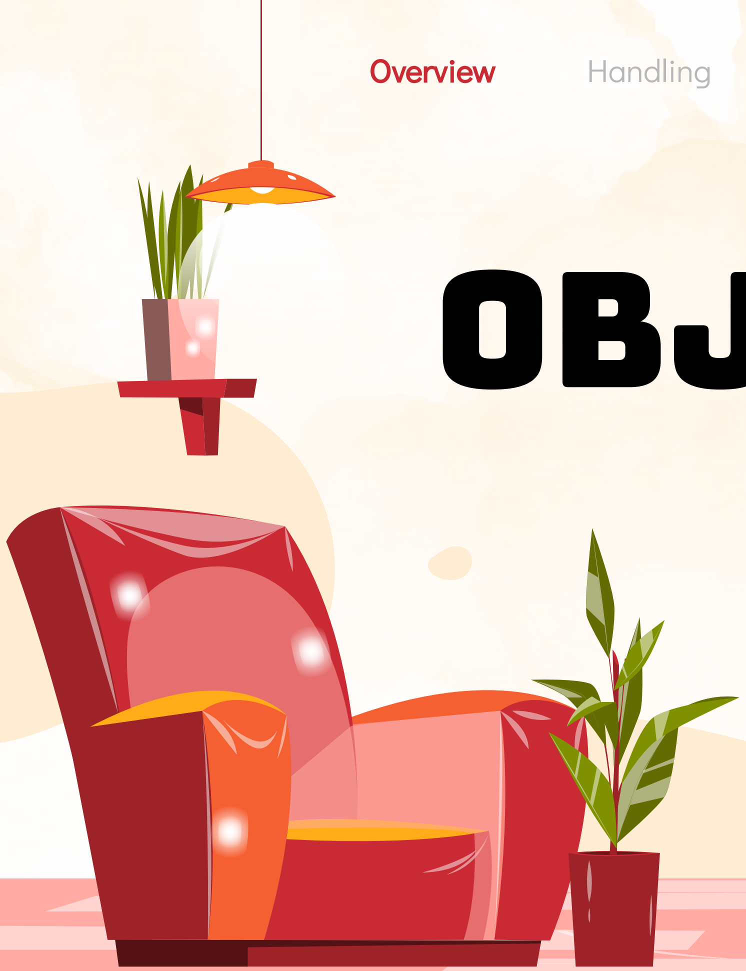
CONTENTS

01

OVERVIEW

DATASET

- Name: **Hotel bookings in Portugal**
- Source: **Kaggle**
- Link: https://www.kaggle.com/datasets/mathsian/hotel-bookings/data
- Original: https://www.researchgate.net/publication/329286343_Hotel_booking_demand_datasets



OBJECTIIVES

- Find out **customer reservation behaviors** via EDA
- Conduct algorithms to **predict/ classify** customers reservation.

ABOUT THE DATA

Data size: 32 columns x 119390 rows	
Customers source	6 columns
Customer character	7 columns
Reservation	13 columns
Recorded time	6 columns

ABOUT THE DATA

Customers Source	Customer Character	Reservation	Recorded time
1. hotel	1. adults	1. is_canceled	1. arrival_date_year
2. market_segment	2. children	2. lead_time	2. arrival_date_month
3. distribution_channel	3. babies	3. days_in_waiting_list	3. arrival_date_week_number
4. agent	4. country	4. stays_in_weekend_nights	4. arrival_date_day_of_month
5. company	5. is_repeated_guest	5. stays_in_week_nights	5. reservation_status
6. customer_type	6. previous_cancellations	6. reserved_room_type	6. reservation_status_date
	7. previous_bookings_not_canceled	7. assigned_room_type	
		8. booking_changes	
		9. deposit_type	
		10. adr	
		11. required_car_parking_spaces	
		12. meal	
		13. total_of_special_requests	

ABOUT THE DATA

Data samples:

```python
hotel_booking.sample(5)
```

✓ 0.0s Python

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_nights	adults	children	babies	meal	cou
74822	City Hotel	1	335	2015	September	38	17	0	1	2	0.000	0	BB	
5418	Resort Hotel	1	14	2016	April	18	29	0	1	2	0.000	0	BB	
49515	City Hotel	1	126	2016	April	16	14	0	3	2	0.000	0	BB	
82102	City Hotel	0	3	2015	December	51	19	2	1	2	0.000	0	BB	
9175	Resort Hotel	1	70	2016	October	45	31	2	5	2	0.000	0	BB	

02

DATA HANDLING

DUPLICATED

As authors claims that:

*"**Each observation** represents **a hotel booking.**"*

*"Since this is hotel **real data**, all data elements pertaining hotel or costumer **identification were deleted.**"*

Action:

Data record is separate customers from separate hotels and coincidentally have the same data. The action here is choose to **keep all the duplicates**.

duplicate

```
# checking total duplicated
hotel_booking.duplicated().sum()
```

✓ 0.1s

31994

NULL AND UNDEFINED

null data

```
# checking null
hotel_booking.isnull().sum().sort_values(ascending=False)
```

✓ 0.0s

```
company                112593
agent                   16340
country                   488
children                    4
reserved_room_type          0
assigned_room_type          0
booking_changes             0
```

```
# checking undefined via listing unique values
for i in hotel_booking.columns:
    print(i)
    print(hotel_booking[i].unique())
    print('_____________')
```

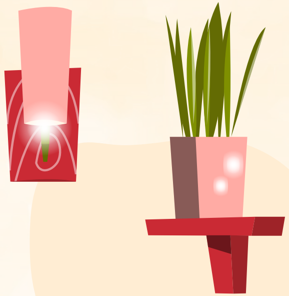
✓ 0.1s

```
_____________
meal
['BB' 'FB' 'HB' 'SC' 'Undefined']
```

in short:

columns	***problem***
agent	has nan
company	has nan
children	has nan
country	has nan
meal	has Undefined
market_segment	has Undefined
distribution_channel	has Undefined

- Nan value: 4 columns
- Undefined value: 3 columns



NULL AND UNDEFINED

As authors claims that:

"In some categorical variables like Agent or Company, ***"NULL" is presented as one of the categories.***
This should not be considered a missing value, but rather as "not applicable".
For example, if a booking "Agent" is defined as ***"NULL" it means that the booking did not came from a travel agent.***

null data

```
# checking null
hotel_booking.isnull().sum().sort_values(ascending=False)
```

✓ 0.0s

```
company                 112593
agent                    16340
country                    488
children                     4
reserved_room_type           0
assigned_room_type           0
booking_changes              0
```

As said:

For the 'agent' and 'company' columns can change 'nan values' into:

- **0** as customer **not** come from agent/company
- **1** as customer **come** from agent/company

```
# create def to change data value
def convert_num(i):
    if i > 0:
        return 1
    return 0

# apply def
hotel_booking['agent_encode'] = hotel_booking.agent.apply(convert_num)
hotel_booking['company_encode'] = hotel_booking.company.apply(convert_num)
```

✓ 0.1s

NULL AND UNDEFINED

```
# 'children' columns has 4 nan values replace by median
hotel_booking['children'] = hotel_booking['children'].fillna(hotel_booking['children'].median())
```

✓ 0.0s

Replace nan in **'children'** column by **median.**

```
# 'country' columns has 488 nulls values
# not using to run model but can use to illustrate customer character
# leave as it is
hotel_booking.country.isnull().sum()
hotel_booking.country.value_counts()
```

✓ 0.0s

```
country
PRT    48590
GBR    12129
FRA    10415
ESP     8568
```

'country' columns has 488 nulls values, this columns is **not using to run model** but can **use to illustrate** customer character therefore **leave as it is.**

NULL AND UNDEFINED

```
# as dictionary states that:
# - Undefined/SC : no meal package
# - BB : Bed & Breakfast
# - HB : Half board (breakfast and one other meal - usually dinner)
# - FB : Full board (breakfast, lunch and dinner)

# as said: encode 'meal' as Undefined/SC: 0 and others: 1
def convert_meal(j):
    if j == 'Undefined' or j == 'SC':
        return 0
    return 1

hotel_booking['meal_encode'] = hotel_booking.meal.apply(convert_meal)
```

✓ 0.0s

'meal' column has **many categories** however there are **2 types** which are reservation with meal package and not.

Action is **encode 'meal'** as:

- **0** is reservation **no** meal package
- **1** is reservation **with** meal package

NULL AND UNDEFINED

```
# 'market_segment' and 'distribution_channel' have less than 6 'undefined' values
# leave as it is
print(hotel_booking.market_segment.value_counts())
print('-----------')
print(hotel_booking.distribution_channel.value_counts())
```

✓ 0.0s

```
market_segment
Online TA        56477
Offline TA/TO    24219
Groups           19811
Direct           12606
Corporate         5295
Complementary      743
Aviation           237
Undefined            2
Name: count, dtype: int64
-----------
distribution_channel
TA/TO        97870
Direct       14645
Corporate     6677
GDS            193
Undefined        5
Name: count, dtype: int64
```

'market_segment' and **'distribution_channel'** have **less than 6 'Undefined'** values.

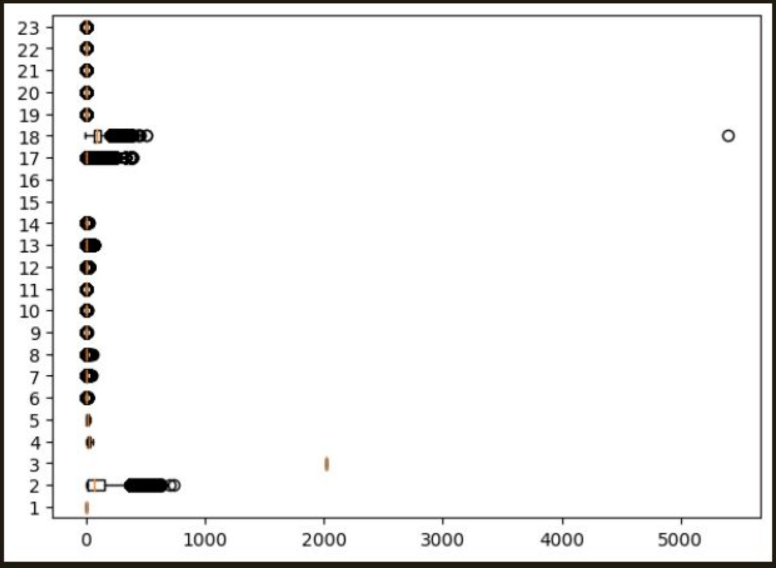
Not many to effect model, **leave as it**.

OUTLIERS

```python
plt.figure(figsize=(7,5))
plt.boxplot(hotel_outliers, vert = False)

plt.show()

# showed that columns 18 which is 'adr' has 1 outliers -> choose to delete that row
```
✓ 0.6s




```python
# drop outliers
hotel_booking = hotel_booking[hotel_booking['adr'] < 1000].reset_index()
```
✓ 0.1s

Column **'adr' has outliers.** Action is to **delete.**

RESIZE DATASET

Merge columns: Merge columns have similar meaning.

merge columns

```
# create columns 'family_size' for illustation
hotel_booking['family_size'] = hotel_booking['adults'] + hotel_booking['children'] + hotel_booking['babies']
```

✓ 0.0s

```
# booking_requests = booking_changes + required_car_parking_spaces + total_of_special_requests : cause of this is all request in booking process.
hotel_booking['booking_requests'] = hotel_booking.booking_changes + hotel_booking.required_car_parking_spaces + hotel_booking.total_of_special_requests
```

✓ 0.0s

```
hotel_booking['stay_in_days'] = hotel_booking.stays_in_weekend_nights + hotel_booking.stays_in_week_nights
```

✓ 0.0s

RESIZE DATASET

Create data for better illustration and calculate:

create data for illustrate and calculate

```python
# create columns 'source' for illustrate customer sources
hotel_booking['source'] = np.where((hotel_booking['agent'] > 0) & (hotel_booking['company'] > 0), 'both',
                          np.where(hotel_booking['agent'] > 0, 'agent',
                          np.where(hotel_booking['company'] > 0, 'comapny',
                          'not applicable')))

# create columns 'meal_request'
meal_dictionary = {'BB': 'Meal', 'FB': 'Meal', 'HB': 'Meal', 'SC': 'No meal', 'Undefined': 'No meal'}
hotel_booking['meal_request'] = hotel_booking['meal'].map(meal_dictionary)

# create columns 'repeated_guest'
repeated_guest_dictionary = {0: 'New guest', 1: 'Old guest'}
hotel_booking['repeated_guest'] = hotel_booking['is_repeated_guest'].map(repeated_guest_dictionary)
```
✓ 0.0s

```python
# create columns 'cancelation_status'
cancel_dictionary = {0: 'Check_in', 1: 'Canceled'}
hotel_booking['cancelation_status'] = hotel_booking['is_canceled'].map(cancel_dictionary)
```
✓ 0.0s

```python
# create columns 'total_booking'
hotel_booking['total_booking'] = hotel_booking.previous_cancellations + hotel_booking.previous_bookings_not_canceled + 1
```
✓ 0.0s

```python
hotel_booking['total_cancelation'] = hotel_booking['previous_cancellations'] + hotel_booking['is_canceled']
```
✓ 0.0s

```python
hotel_booking['cancelation_rate'] = (hotel_booking['total_cancelation'] / hotel_booking['total_booking'] )* 100
```
✓ 0.0s

RESIZE DATASET

Drop columns:

drop unnescessary columns

drop columns below cause of:

- reservation_status: exactly as 'is_canceled' columns
- reservation_status_date: not see the use
- arrival_date_year: this analysis focus on customer behaviours in months and days
- arrival_date_week_number: this analysis focus on customer behaviours in months and days
- company: replace by 'company_encode'
- agent: replace by 'agent_encode'
- market_segment: similar with distribution_channel
- adults, children, babies: replace by 'family size'
- stays_in_weekend_nights, stays_in_week_nights: replace by 'stay_in_days'

```
hotel_booking = hotel_booking.drop(['reservation_status', 'reservation_status_date', 'arrival_date_year', 'arrival_date_week_number',
                                    'company', 'agent',
                                    'market_segment',
                                    'adults', 'children', 'babies',
                                    'booking_changes', 'required_car_parking_spaces', 'total_of_special_requests',
                                    'stays_in_weekend_nights', 'stays_in_week_nights'], axis=1)
```

✓ 0.0s

03

EDA

CUSTOMER SOURCES

82%

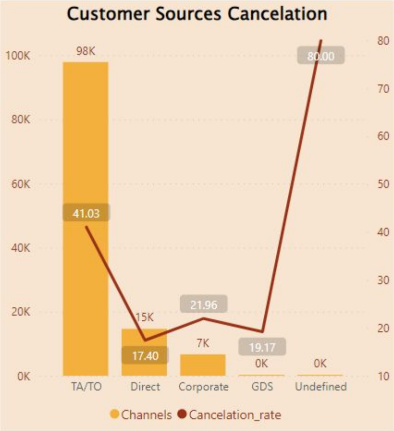
Customers from **Agents**

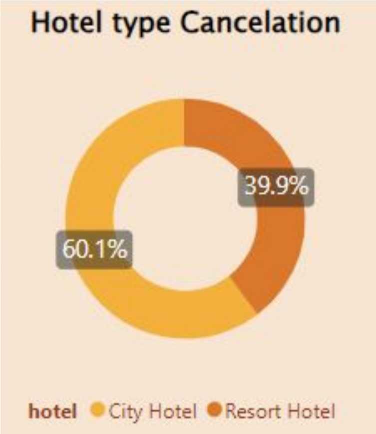
75%

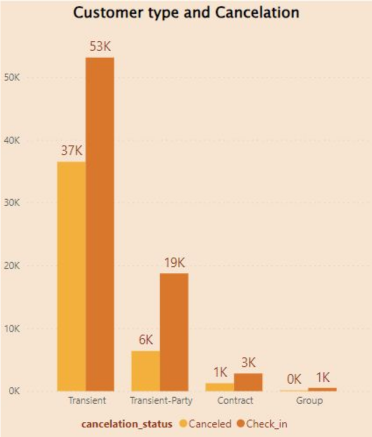
Chose **City Hotel**

75%

Transient

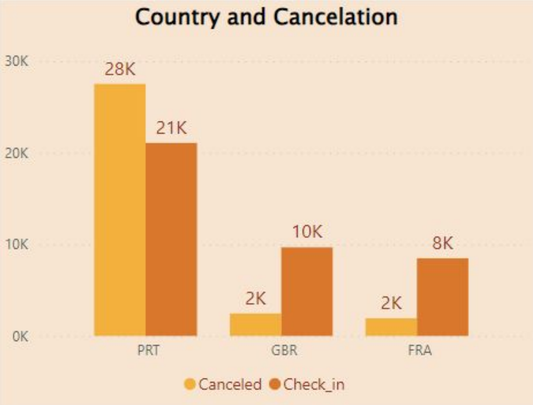






CUSTOMER CHARACTER

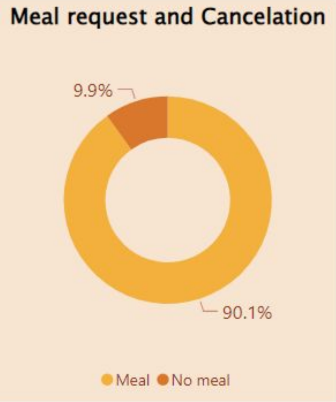
Portuguese is main guests but their **Cancelation** is **higher** than their **Check-in.**



family_size	Cancelation_rate
0	14.18
1	29.01
2	39.69
3	32.18
4	41.87
5	24.82
6	100.00
10	50.00
12	50.00
20	100.00
26	100.00
27	100.00
40	100.00
50	100.00
55	100.00

Smaller the **Family size** **higher** change to **Check-in.**

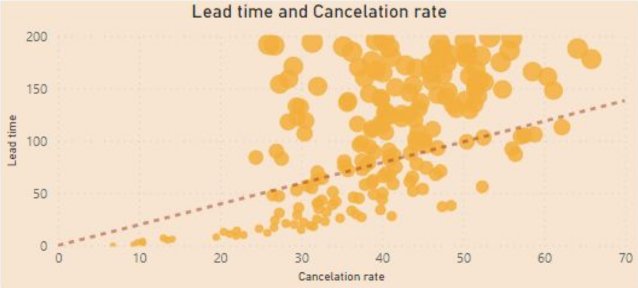
Most guest **demain for meal** during stays.



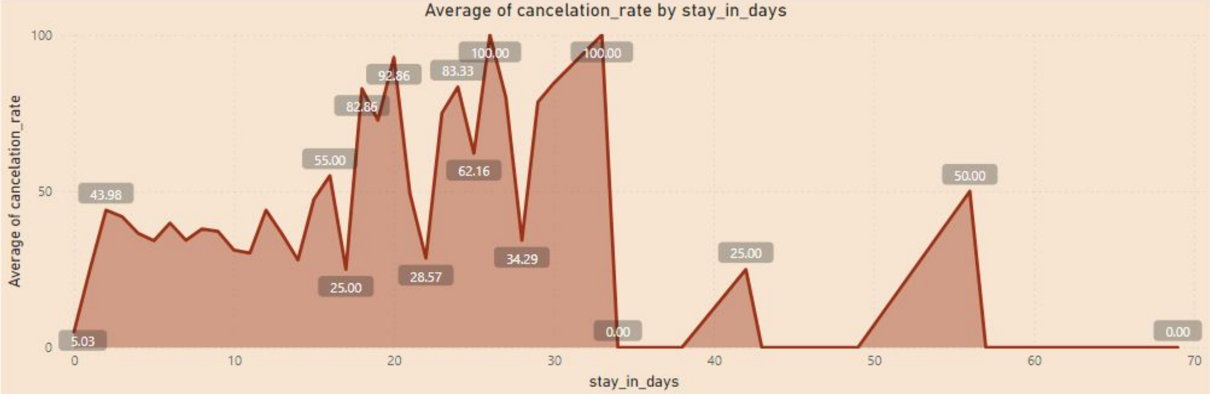
Guest are **new.**

BOOKING EXPERIENCE

Guests in wait list for **30 days** or about **100 days** have **high cancelation rate.**



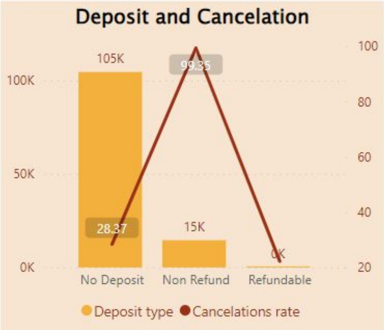
Guest who booked **less than 50 days** ahead have high rate of cancelation.



Guest booked for **more than 35 days** of stay have lower cancelation rate.

BOOKING EXPERIENCE

No Deposit is general in used but have low cancelation rate in **contrast with Non refund** in both indicators.



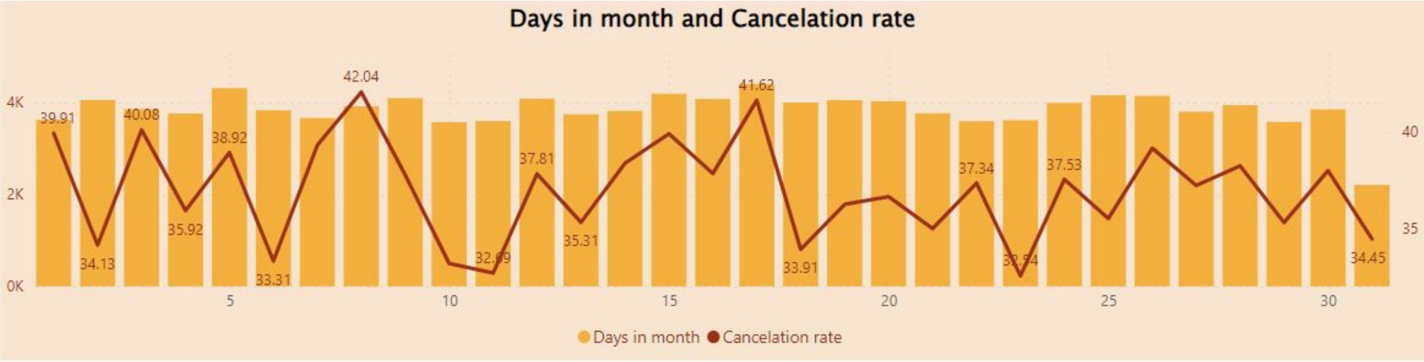
Guests who have **certain amount of requests** are tent to stay more than who do not have any request or have alot of request.

Room A has **highest booking** at the **average price**.

SEASONAL

Customer most booking in **summer months** and **Cancelation** in this months also **higher.**

There are **not much difference** booking values in each day in a months.



04

PREP FOR MODEL

ENCODING

```
# seperate numeric columns
hb_numeric = hotel_booking.select_dtypes(include='number')
```

```
# encode the categories columns using onthot
hotel_dictionary = {'Resort Hotel': 1, 'City Hotel': 0}
hotel_booking['hotel_encode'] = hotel_booking.hotel.map(hotel_dictionary)

onehot_month            = pd.get_dummies(hotel_booking['arrival_date_month'], prefix = 'month').astype(int)
onehot_market           = pd.get_dummies(hotel_booking['distribution_channel'], prefix = 'market').astype(int)
onehot_reserved_room    = pd.get_dummies(hotel_booking['reserved_room_type'], prefix = 'reserved_room').astype(int)
onehot_assigned_room    = pd.get_dummies(hotel_booking['assigned_room_type'], prefix = 'assigned_room').astype(int)
onehot_deposit_type     = pd.get_dummies(hotel_booking['deposit_type'], prefix = 'deposit_type').astype(int)
onehot_customer_type    = pd.get_dummies(hotel_booking['customer_type'], prefix = 'customer_type').astype(int)
```

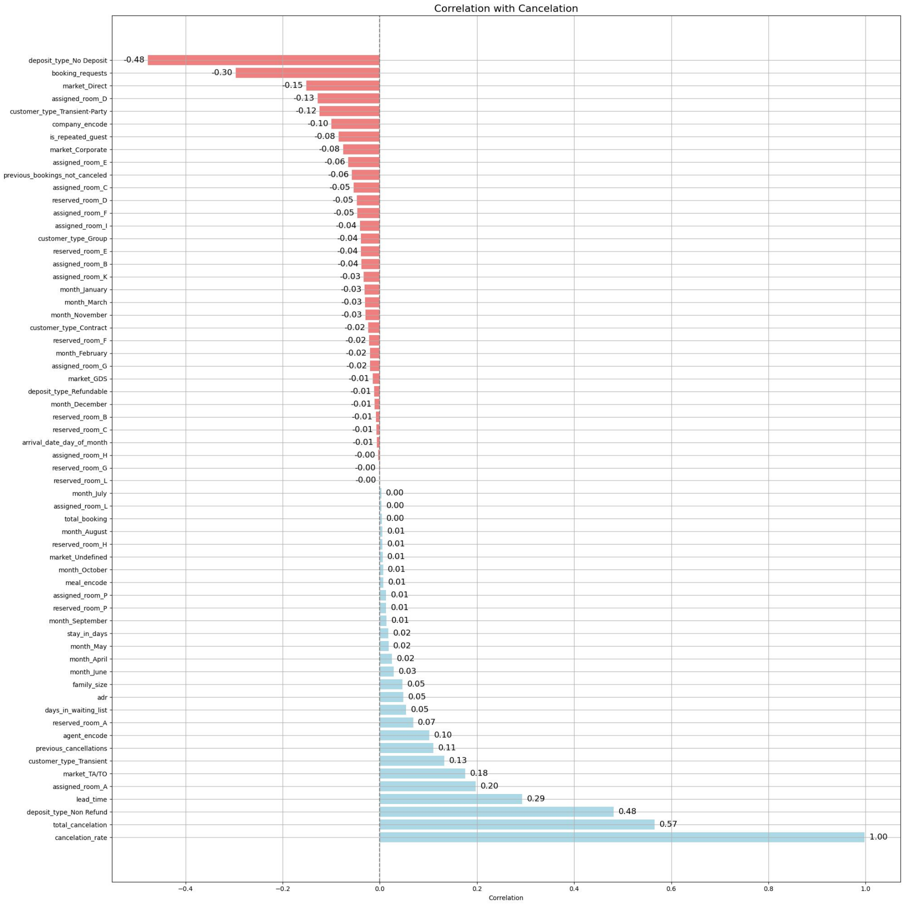
```
# create new dataset to concate 2 datasets

hb_encode = pd.concat([
    hb_numeric, # the dataset
    onehot_month,
    onehot_market,
    onehot_reserved_room,
    onehot_assigned_room,
    onehot_deposit_type,
    onehot_customer_type
], axis = 1)
```

Encode categories columns using **onehot** and **dictionary** method

X, Y DEFINING

Checking correlation:

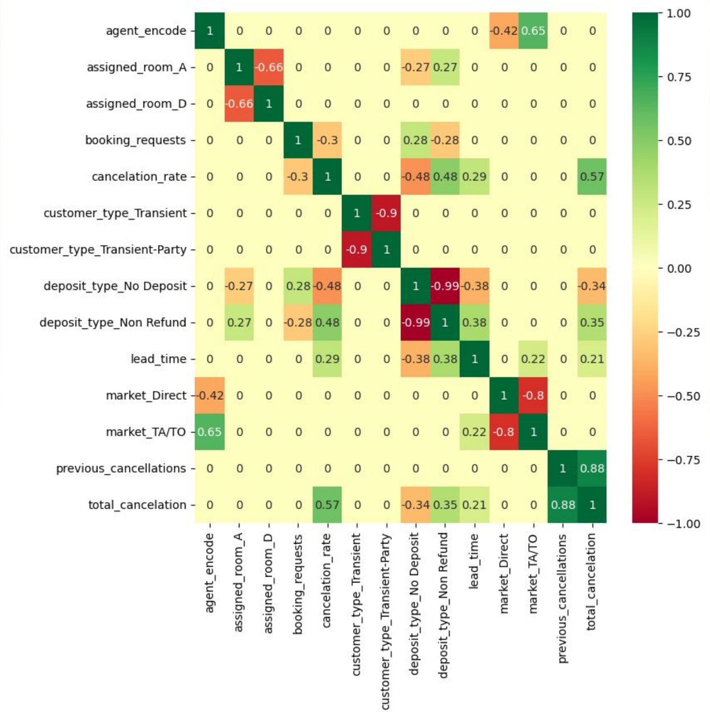


As showed in the graph:

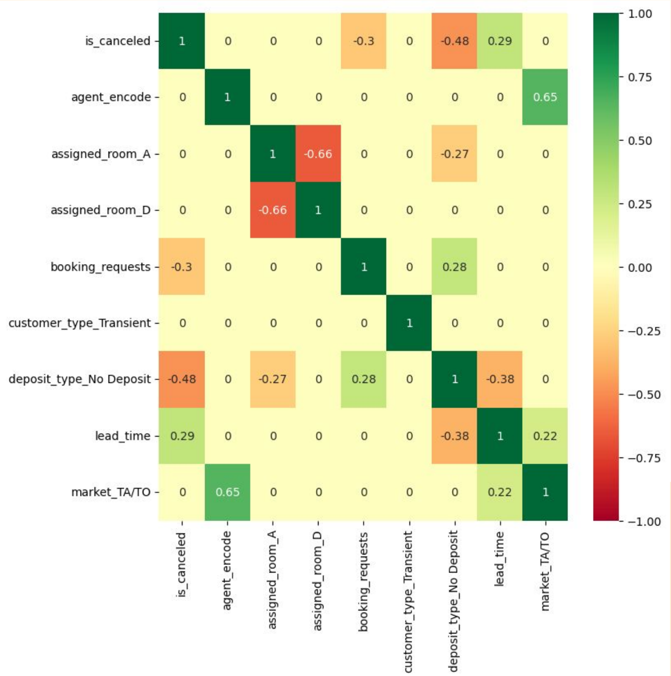
'Cancelation_rate' and **'total_cancelation'** have **high correlate** with **'is_canceled'** this will effect model accuaracy therefore **not use** this columns in model.

X, Y DEFINING

Checking correlation:



There are couples of columns that have **high correlation with each other.** The choice is to **delete 1 of each columns** to prevent **Multicollinearity** (Đa cộng tuyến).



X, Y DEFINING

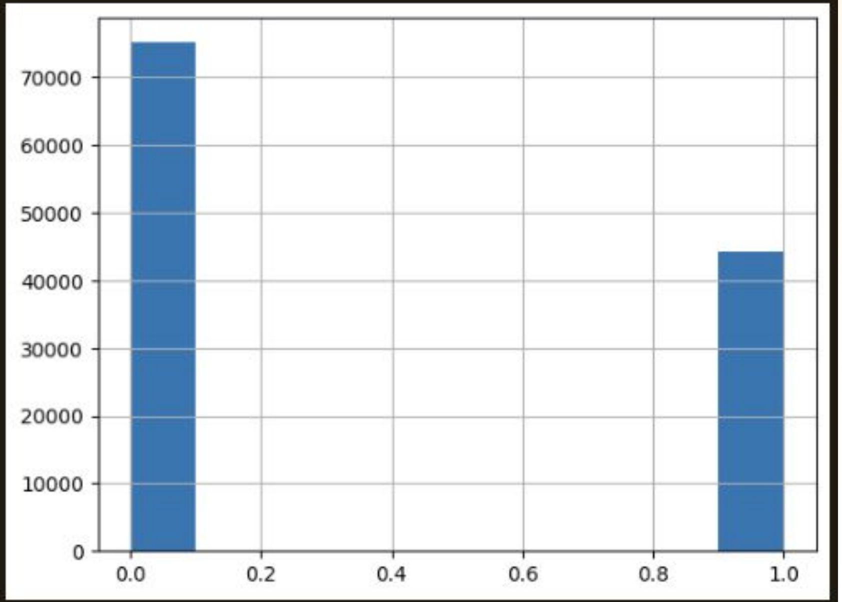
Select X, y for model:

```
# chose X, y

y = hb['is_canceled'].values
X = hb[['agent_encode', 'assigned_room_A', 'assigned_room_D', 'booking_requests', 'customer_type_Transient', 'deposit_type_No Deposit', 'lead_time', 'market_TA/TO']].values
```

IMBALANCE DATA

```
hb_encode_corr['is_canceled'].hist()
✓ 0.3s
<Axes: >
```




```
# dataset not cancel == 0
hbc_0 = hb_encode_corr[hb_encode_corr.is_canceled == 0]

# dataset is canceled == 1
hbc_1 = hb_encode_corr[hb_encode_corr.is_canceled == 1]

# dataset size
hbc_0.shape, hbc_1.shape
# random choose data from hbc_0
hbc_0_resapled = hbc_0.sample(44223, random_state=1)

# new dataset
hbc_0_resapled.shape
# connect hb_0 and hbc_1
hb_balance = pd.concat([hbc_0_resapled,hbc_1])

# y after rebalance
hb_balance['is_canceled'].value_counts()
```

```
is_canceled
0    44223
1    44223
Name: count, dtype: int64
```

NORMALIZATION

normalization: min-max scaler

```
# Scale X2
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(hb_balance.iloc[:, 1:11].values)

# create new dataframe
hb = pd.DataFrame(data = X_scaled, columns = hb_balance.iloc[:, 1:11].columns)

# add y2 column
hb['is_canceled'] = hb_balance['is_canceled']

hb.head()
```

	agent_encode	assigned_room_A	assigned_room_D	booking_requests	customer_type_Transient	deposit_type_Non Refund	lead_time	market_TA/TO	total_cancelation	is_canceled
0	1.000	0.000	1.000	0.143	0.000	0.000	0.267	1.000	0.000	0
1	1.000	0.000	1.000	0.000	1.000	0.000	0.000	1.000	0.000	0
2	1.000	1.000	0.000	0.048	1.000	0.000	0.350	1.000	0.000	0
3	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0
4	1.000	0.000	1.000	0.048	1.000	0.000	0.040	1.000	0.000	0

SPLIT TRAIN-TEST

```python
# run split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

05

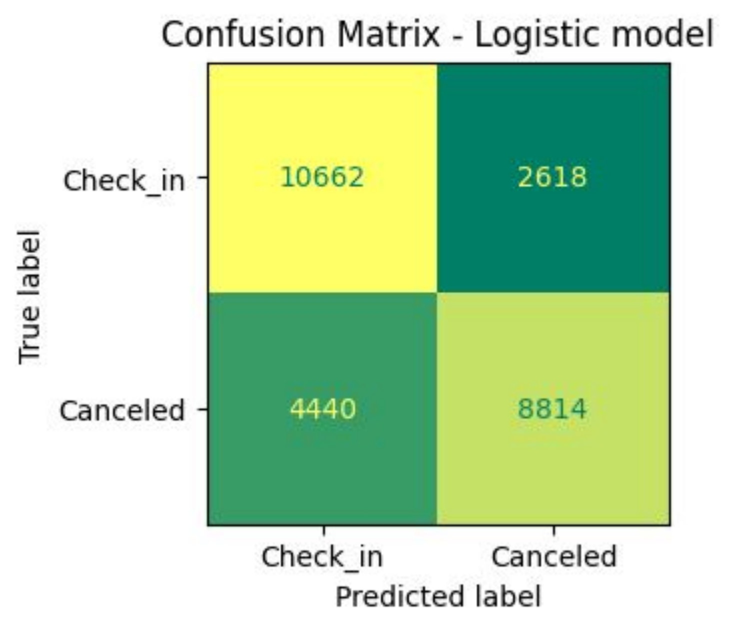
MODELS

MODELS:

Models	Parameter
Logistic Regresion	Default
Gaussian Naive Bayes	Default
Decision Tree	max_depth = 15
Radom Forest	n_estimators =78
K Nearst Neighbor	n_neighbors = 34

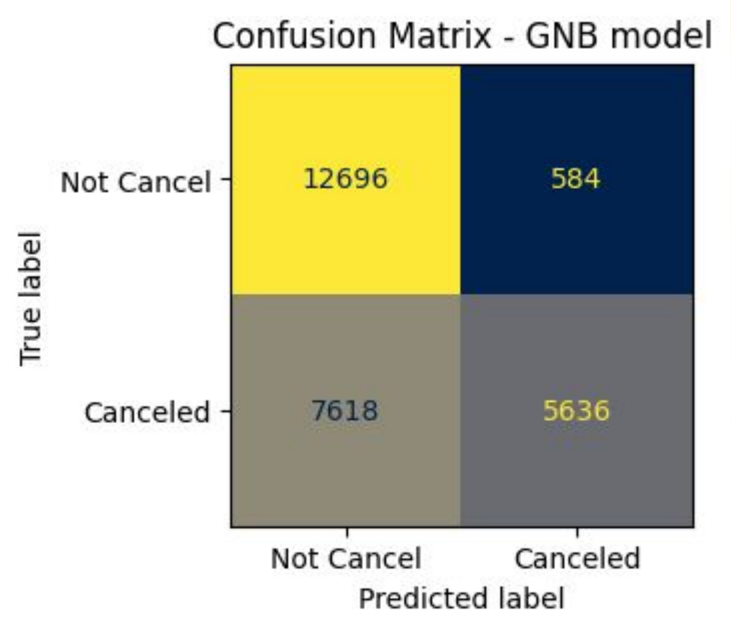
LOGISTIC REGRESION

	precision	recall	f1-score	support
0	0.71	0.80	0.75	13280
1	0.77	0.67	0.71	13254
accuracy			0.73	26534
macro avg	0.74	0.73	0.73	26534
weighted avg	0.74	0.73	0.73	26534

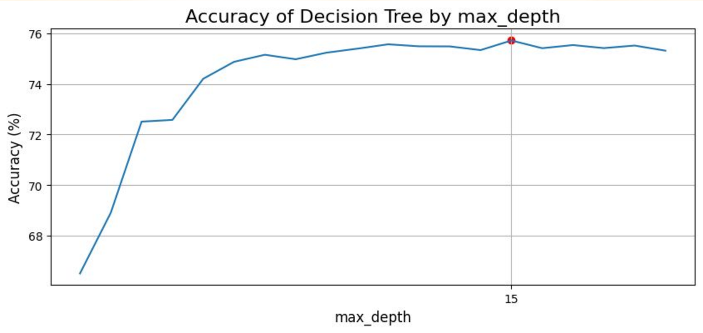


GAUSSIAN NAIVE BAYES

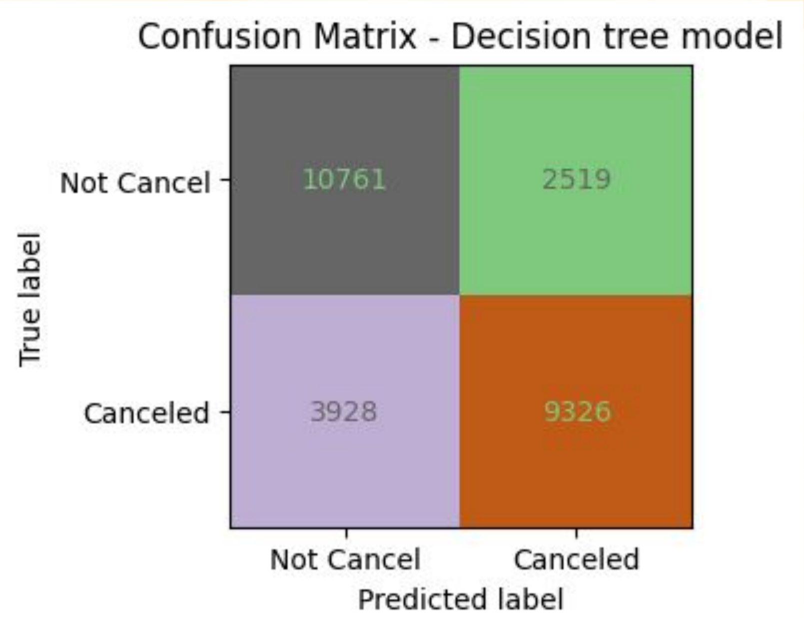
	precision	recall	f1-score	support
0	0.62	0.96	0.76	13280
1	0.91	0.43	0.58	13254
accuracy			0.69	26534
macro avg	0.77	0.69	0.67	26534
weighted avg	0.77	0.69	0.67	26534



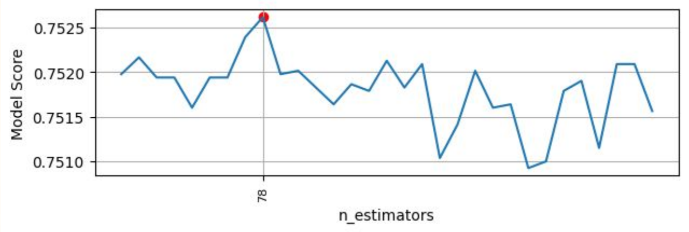
DECISION TREE



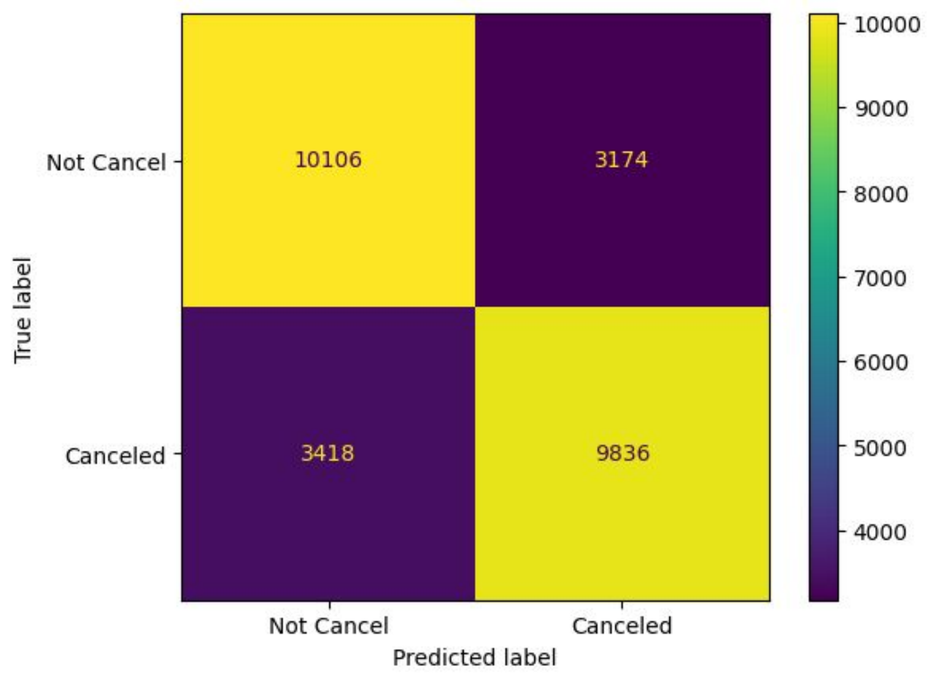
	precision	recall	f1-score	support
0	0.73	0.81	0.77	13280
1	0.79	0.70	0.74	13254
accuracy			0.76	26534
macro avg	0.76	0.76	0.76	26534
weighted avg	0.76	0.76	0.76	26534



RANDOM FOREST

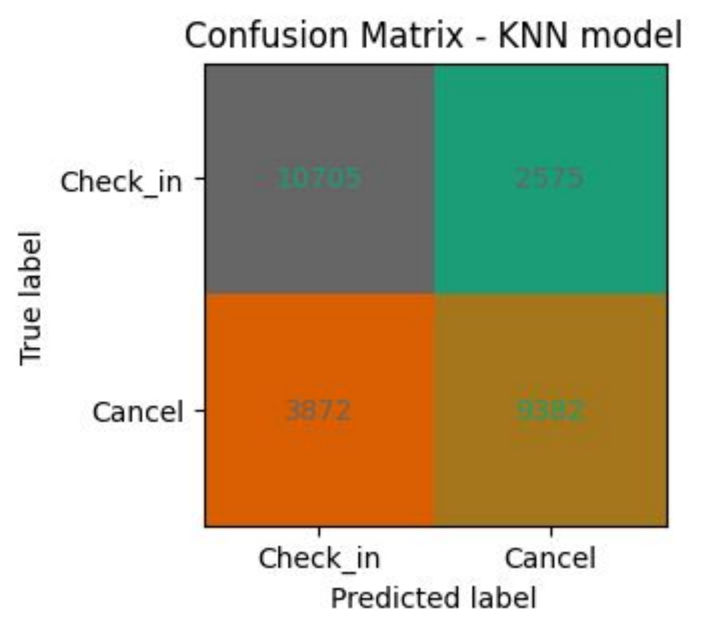


	precision	recall	f1-score	support
0	0.7475	0.7622	0.7548	13280
1	0.7569	0.7420	0.7494	13254
accuracy			0.7521	26534
macro avg	0.7522	0.7521	0.7521	26534
weighted avg	0.7522	0.7521	0.7521	26534



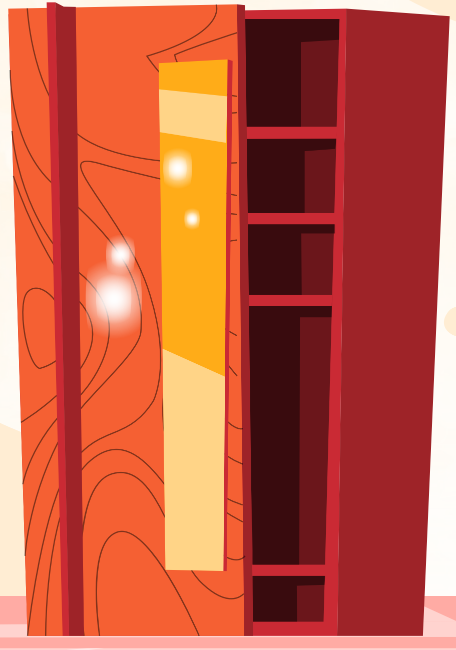
K NEAREST NEIGHBORS

	precision	recall	f1-score	support
0	0.7344	0.8061	0.7686	13280
1	0.7846	0.7079	0.7443	13254
accuracy			0.7570	26534
macro avg	0.7595	0.7570	0.7564	26534
weighted avg	0.7595	0.7570	0.7564	26534

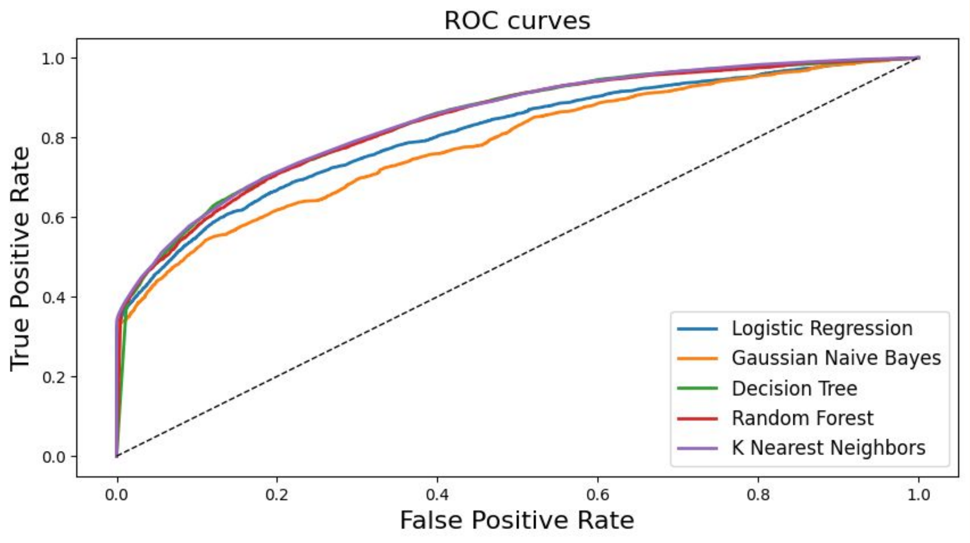


06

RESULT



MODELS COMPARITION



Models	Accuracy Score	F1_Score	Precision	Recall
Decesion Tree	0.76	0.74	0.79	0.70
GNB	0.69	0.58	0.91	0.43
KNN	0.76	0.74	0.78	0.71
Logistic Regresson	0.73	0.71	0.77	0.67

All models have accuracy measure scores are **higher than 0.7**, except for GNB models.

GNB has **highest Precision score** but others scores are low.

Decision Tree has the best performance.

THANK YOU FOR LISTENING!