PROJECT REPORT

MAJOR PROJECT (1 AND 2)

RINEX – 7 NOVEMBER

MACHINE LEARNING

NAME = ANUVRAT VERMA

GIT HUB = https://github.com/Anuvrat-Verma

COLLEGE NAME = VELLORE INSTITUTE OF TECHNOLOGY BHOPAL

COLLEGE ROLL NUMBER = 21BCE10273

BRANCH = COMPUTER SCIENCE

DATE = 1/02/2023

EMAIL = anuvratverma@gmail.com

GRADUATION YEAR = 2025

CONTACT NUMBER = 8849329261

1) Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR .

DATASET = STAR DATASET (KAGGLE)
DATASET LINK = https://www.kaggle.com/datasets/deepu1109/star-dataset

I HAVE APPLIED LINEAR REGRESSION TO FIND THE STAR TYPE AND SPECTRAL CLASS USING ALL OTHER COLUMNS AS INPUTS

```
# STAR TYPE DATASET
# 1. CREATING DATA FRAME
import pandas as pd
import numpy as np
star = pd.read_csv('star type.csv')
star
```

	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
0	3068	0.002400	0.1700	16.12	0	Red	M
1	3042	0.000500	0.1542	16.60	0	Red	M
2	2600	0.000300	0.1020	18.70	0	Red	M
3	2800	0.000200	0.1600	16.65	0	Red	M
4	1939	0.000138	0.1030	20.06	0	Red	M
...	...	...	...	...	...	...	...
235	38940	374830.000000	1356.0000	-9.93	5	Blue	O
236	30839	834042.000000	1194.0000	-10.63	5	Blue	O
237	8829	537493.000000	1423.0000	-10.73	5	White	A
238	9235	404940.000000	1112.0000	-11.23	5	White	A
239	37882	294903.000000	1783.0000	-7.80	5	Blue	O

240 rows × 7 columns

```
[154] star.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Temperature (K)         240 non-null    int64
 1   Luminosity(L/Lo)        240 non-null    float64
 2   Radius(R/Ro)            240 non-null    float64
 3   Absolute magnitude(Mv)  240 non-null    float64
 4   Star type               240 non-null    int64
 5   Star color              240 non-null    object
 6   Spectral Class          240 non-null    object
dtypes: float64(3), int64(2), object(2)
memory usage: 13.2+ KB
```

```
[155] #2 CLEANING DATA
      # WILL CONVERT ALL STRING/OBJECT TO INT TYPE
      star['Star color'].unique()
```

```
array(['Red', 'Blue White', 'White', 'Yellowish White', 'Blue white',
       'Pale yellow orange', 'Blue', 'Blue-white', 'Whitish',
       'yellow-white', 'Orange', 'White-Yellow', 'white', 'Blue ',
       'yellowish', 'Yellowish', 'Orange-Red', 'Blue white ',
       'Blue-White'], dtype=object)
```

```
# CLEANING COLUMN STAR COLOR
# ENCODING COLORS WITH INTEGERS USING REPLACE FUNCTION , AND MANAGING RUBBISH VALUES
star['Star color'] = star['Star color'].str.replace('Red','1')
star['Star color'] = star['Star color'].str.replace('Blue White','2')
star['Star color'] = star['Star color'].str.replace('Blue-white','2')
star['Star color'] = star['Star color'].str.replace('Blue white','2')
star['Star color'] = star['Star color'].str.replace('Blue-White','2')
star['Star color'] = star['Star color'].str.replace('white','3')
star['Star color'] = star['Star color'].str.replace('White','3')
star['Star color'] = star['Star color'].str.replace('Whitish','4')
star['Star color'] = star['Star color'].str.replace('Yellowish White','5')
star['Star color'] = star['Star color'].str.replace('yellow-white','5')
star['Star color'] = star['Star color'].str.replace('White-Yellow','5')
star['Star color'] = star['Star color'].str.replace('Pale yellow orange','5')
star['Star color'] = star['Star color'].str.replace('Blue','6')
star['Star color'] = star['Star color'].str.replace('Blue','6')
star['Star color'] = star['Star color'].str.replace('Orange','7')
star['Star color'] = star['Star color'].str.replace('yellowish','8')
star['Star color'] = star['Star color'].str.replace('Yelowish','8')
star['Star color'] = star['Star color'].str.replace('Yelowish','8')
star['Star color'] = star['Star color'].str.replace('Yellowish','8')
star['Star color'] = star['Star color'].str.replace('Yellowish 3','8')
star['Star color'] = star['Star color'].str.replace('yellow-3','8')
star['Star color'] = star['Star color'].str.replace('3-Yellow','8')
star['Star color'] = star['Star color'].str.replace('7-1','8')
star['Star color'] = star['Star color'].str.replace('8 3','9')
star['Star color'] = star['Star color'].str.replace('Orange-Red','9')
```

```
[157] star
```

	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
0	3068	0.002400	0.1700	16.12	0	1	M
1	3042	0.000500	0.1542	16.60	0	1	M
2	2600	0.000300	0.1020	18.70	0	1	M
3	2800	0.000200	0.1600	16.65	0	1	M
4	1939	0.000138	0.1030	20.06	0	1	M
...	...	...	...	...	...	...	...
235	38940	374830.000000	1356.0000	-9.93	5	6	O
236	30839	834042.000000	1194.0000	-10.63	5	6	O
237	8829	537493.000000	1423.0000	-10.73	5	3	A
238	9235	404940.000000	1112.0000	-11.23	5	3	A
239	37882	294903.000000	1783.0000	-7.80	5	6	O

240 rows × 7 columns

```
[158] star['Star color'] = star['Star color'].astype(int) # CONVERSION TO INT
```

```
[159] star.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Temperature (K)         240 non-null    int64
 1   Luminosity(L/Lo)        240 non-null    float64
 2   Radius(R/Ro)            240 non-null    float64
 3   Absolute magnitude(Mv)  240 non-null    float64
 4   Star type               240 non-null    int64
 5   Star color              240 non-null    int64
 6   Spectral Class          240 non-null    object
dtypes: float64(3), int64(3), object(1)
memory usage: 13.2+ KB
```

```
[160] # CLEANING SPECTRAL CLASS COLUMN
star['Spectral Class'] = star['Spectral Class'].str.replace('O','1')
star['Spectral Class'] = star['Spectral Class'].str.replace('B','2')
star['Spectral Class'] = star['Spectral Class'].str.replace('A','3')
star['Spectral Class'] = star['Spectral Class'].str.replace('F','4')
star['Spectral Class'] = star['Spectral Class'].str.replace('G','5')
star['Spectral Class'] = star['Spectral Class'].str.replace('K','6')
star['Spectral Class'] = star['Spectral Class'].str.replace('M','7')
```

```
[161] star['Spectral Class'] = star['Spectral Class'].astype(int) #CONVERSION TO INT
```

star

	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
0	3068	0.002400	0.1700	16.12	0	1	7
1	3042	0.000500	0.1542	16.60	0	1	7
2	2600	0.000300	0.1020	18.70	0	1	7
3	2800	0.000200	0.1600	16.65	0	1	7
4	1939	0.000138	0.1030	20.06	0	1	7
...	...	...	...	...	...	...	...
235	38940	374830.000000	1356.0000	-9.93	5	6	1
236	30839	834042.000000	1194.0000	-10.63	5	6	1
237	8829	537493.000000	1423.0000	-10.73	5	3	3
238	9235	404940.000000	1112.0000	-11.23	5	3	3
239	37882	294903.000000	1783.0000	-7.80	5	6	1

240 rows × 7 columns

```
[163] star.info() # ALL STRING DATA TYPE HAVE BEEN CONVERTED
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Temperature (K)         240 non-null    int64
 1   Luminosity(L/Lo)        240 non-null    float64
 2   Radius(R/Ro)            240 non-null    float64
 3   Absolute magnitude(Mv)  240 non-null    float64
 4   Star type               240 non-null    int64
 5   Star color              240 non-null    int64
 6   Spectral Class          240 non-null    int64
dtypes: float64(3), int64(4)
memory usage: 13.2 KB
```

```
[164] # EDA

      # NUMBER OF DIFFERENT STAR TYPES IN DATA SET

      # Brown Dwarf -> Star Type = 0

      # Red Dwarf -> Star Type = 1

      # White Dwarf-> Star Type = 2

      # Main Sequence -> Star Type = 3

      # Supergiant -> Star Type = 4

      # Hypergiant -> Star Type = 5

      star['Star type'].value_counts()
```

```
0    40
1    40
2    40
3    40
4    40
5    40
Name: Star type, dtype: int64
```

```
[165] # NUMBER OF EACH SPECTRAL TYPE
      star['Spectral Class'].value_counts()
```

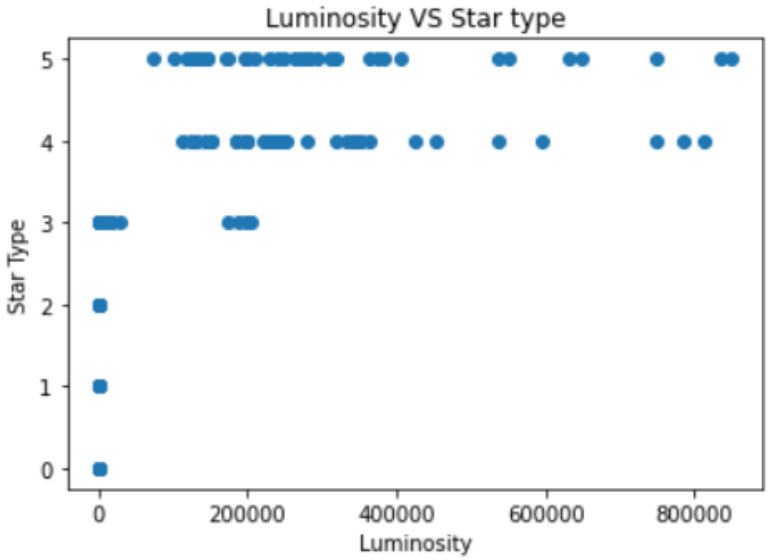
```
7    111
2     46
1     40
3     19
4     17
6      6
5      1
Name: Spectral Class, dtype: int64
```

```
[166] #3 DATA VISUALISATION
      # STAR TYPE VS LUMINOSITY(L/Lo)
      import matplotlib.pyplot as plt

      plt.scatter(star['Luminosity(L/Lo)'],star['Star type'])

      plt.title('Luminosity VS Star type')
      plt.xlabel('Luminosity')
      plt.ylabel('Star Type')
```

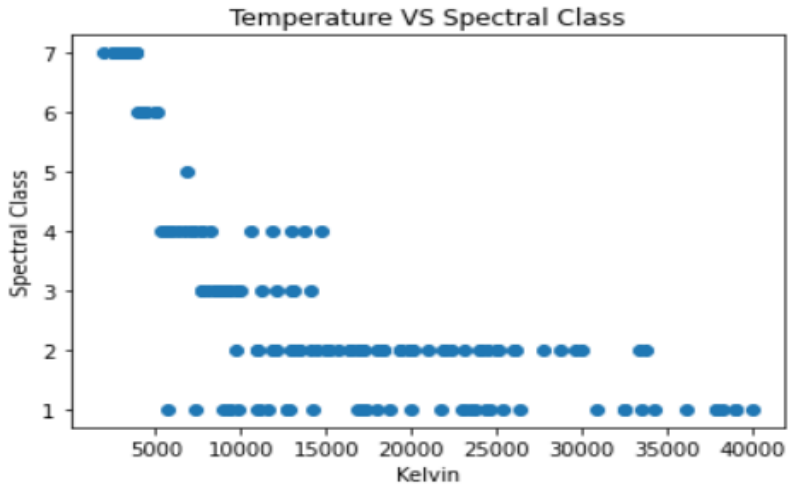
```
Text(0, 0.5, 'Star Type')
```




```
[167] #SPECTRAL CLASS VS COLOR
      plt.scatter(star['Temperature (K)'],star['Spectral Class'])

      plt.title('Temperature VS Spectral Class')
      plt.xlabel('Kelvin')
      plt.ylabel('Spectral Class')
```

```
Text(0, 0.5, 'Spectral Class')
```



```
[168] # 4. DIVIDE INTO INPUT  AND OUTPUT
      # INPUT (X) - TEMPRERATURE , LUMINOSITY , RADIUS ,ABSOLUTE MAGNITUDE,STAR COLOR
      # OUTPUT (Y) - SPECTRAL CLASS , STAR TYPE

      x = star.loc[:,['Temperature (K)','Luminosity(L/Lo)','Radius(R/Ro)','Absolute magnitude(Mv)','Star color']].values
      x
```

```
array([[ 3.06800e+03,  2.40000e-03,  1.70000e-01,  1.61200e+01,
         1.00000e+00],
       [ 3.04200e+03,  5.00000e-04,  1.54200e-01,  1.66000e+01,
         1.00000e+00],
       [ 2.60000e+03,  3.00000e-04,  1.02000e-01,  1.87000e+01,
         1.00000e+00],
       ...,
       [ 8.82900e+03,  5.37493e+05,  1.42300e+03, -1.07300e+01,
         3.00000e+00],
       [ 9.23500e+03,  4.04940e+05,  1.11200e+03, -1.12300e+01,
         3.00000e+00],
       [ 3.78820e+04,  2.94903e+05,  1.78300e+03, -7.80000e+00,
         6.00000e+00]])
```

```
[169] y = star.loc[:,['Star type','Spectral Class']].values
      y
```

```
array([[0, 7],
       [0, 7],
       [0, 7],
       [0, 7],
       [0, 7],
       [0, 7],
       [0, 7],
```

```
[170] #5.Train and Test variables - train_test_split
      from sklearn.model_selection import train_test_split
      x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 0)
```

```
[171] #6.NORMALIZATION or SCALING
      from sklearn.preprocessing import MinMaxScaler
      scaler = MinMaxScaler()
      x_train = scaler.fit_transform(x_train)
      x_test = scaler.fit_transform(x_test)
```

```
[173] #7.apply a linear regression
      from sklearn.linear_model import LinearRegression
      model = LinearRegression()
```

```
[174] #8 fitting the model
      model.fit(x_train,y_train)
```

```
LinearRegression()
```

```
[175] #9.PREDICT THE OUTPUT
      y_pred = model.predict(x_test)
      y_pred #PREDICTED OUTPUT VALUES
```

```
array([[ 3.51477754,  2.96479854],
       [ 1.30874324,  6.35996202],
       [ 4.26260971,  0.56830692],
       [ 3.71759024,  2.73621601],
       [ 3.36013474,  1.79154212],
       [ 4.1575258 ,  0.67001787],
       [ 3.21352033,  3.72071619],
       [ 1.09211234,  6.44524717],
       [ 0.9516381 ,  6.59582419],
       [ 1.82599987,  2.87121488],
       [ 1.14722758,  3.99966167],
       [ 3.35807201,  2.72827371],
       [ 4.03431498,  3.61703999],
       [ 0.83873059,  6.53751935],
```

```
[176] y_test # ACTUAL OUTPUT VALUES
```

```
array([[4, 1],
       [1, 7],
       [3, 4],
       [1, 7],
       [4, 1],
       [4, 1],
       [3, 2],
       [4, 1],
       [3, 3],
       [1, 7],
       [1, 7],
       [2, 2],
       [2, 2],
       [3, 2],
       [5, 2],
       [1, 7],
       [0, 7],
```

```
[177] #10.Accuracy
      from sklearn.metrics import r2_score
      r2_score(y_test,y_pred)
```

```
0.8541063706208685
```

```
[186] #INDIVIDUAL PREDICTIONS
      a = scaler.transform([[ 3.06800e+03,  2.40000e-03,  1.70000e-01,  1.61200e+01,
              1.00000e+00]])
      model.predict(a)
      # STAR TYPE = 0.57 ~ 1 ~ RED DWARF ; SPECTRAL CLASS = 6.7 ~ 7 ~ M
```

```
array([[0.57573433, 6.70880178]])
```

```
[190] b = scaler.transform([[2.5899e+3, 2.444e-3,2.79e-1,1.61e+1,3]])
      model.predict(b)
      # STAR TYPE = 0.8 ~ 1 ~ RED DWARF ; SPECTRAL CLASS = 6.05 ~ 6 ~ K
```

```
array([[0.89792527, 6.05336886]])
```

```
[191] c = scaler.transform([[3.5899e+3, 3.444e-3,3.79e-1,2.61e+1,4]])
      model.predict(c)
      # STAR TYPE = -0.3 ~ 0 ~ BROWN DWARF ; SPECTRAL CLASS = 6.06 ~ 6 ~ K
```

```
array([[-0.35949511,  6.06445928]])
```

2) Choose any dataset of your choice and Perform Exploratory Data Analysis for Atleast 15 different facts/Conclusions.

DATASET = RESERVATION AND CANCELLATION PREDICTION (HOTEL)

DATASET LINK = https://www.kaggle.com/datasets/gauravduttakiit/reservation-cancellation-prediction?select=train__dataset.csv

```
[40] import pandas as pd
     import numpy as np
     #dataframe
     df = pd.read_csv('hotel_cancellation.csv')
     df
     #data for all hotel cancellations and factors involved
     #0-no , 1-yes , booking status = 1- cancel , 0 - not cancel
```

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal_plan	required_car_parking_space
0	2	0	1	4	0	0
1	2	1	0	2	0	0
2	1	0	1	5	0	0
3	1	0	2	4	0	0
4	2	0	0	4	1	0
...	...	...	...	...	...	...
18132	1	0	0	2	0	0
18133	2	0	0	3	0	0
18134	2	0	0	1	0	0
18135	2	0	0	3	0	0
18136	1	0	1	1	0	0

18137 rows × 18 columns

room_type_reserved	lead_time	arrival_year	arrival_month	arrival_date	market_segment_type	repeated_guest
0	118	2017	12	28	1	0
0	17	2018	4	14	1	0
0	349	2018	10	4	0	0
0	69	2018	6	12	0	0
0	11	2018	1	20	1	0
...	...	...	...	...	...	...
0	103	2018	4	19	0	0
0	129	2018	8	10	1	0
0	90	2018	7	13	1	0
0	18	2018	11	10	1	1
0	159	2018	4	9	0	0

no_of_previous_cancellations	no_of_previous_bookings_not_canceled	avg_price_per_room	no_of_special_requests	booking_status
0	0	110.80	2	0
0	0	145.00	0	1
0	0	96.67	0	1
0	0	120.00	0	1
0	0	69.50	1	0
...	...	...	...	...
0	0	115.00	0	1
0	0	88.01	1	0
0	0	105.30	0	1
0	1	123.33	1	0
0	0	65.00	0	0

✓ 0s

```
[35] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18137 entries, 0 to 18136
Data columns (total 18 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   no_of_adults                          18137 non-null  int64
 1   no_of_children                        18137 non-null  int64
 2   no_of_weekend_nights                  18137 non-null  int64
 3   no_of_week_nights                     18137 non-null  int64
 4   type_of_meal_plan                     18137 non-null  int64
 5   required_car_parking_space            18137 non-null  int64
 6   room_type_reserved                    18137 non-null  int64
 7   lead_time                             18137 non-null  int64
 8   arrival_year                          18137 non-null  int64
 9   arrival_month                         18137 non-null  int64
 10  arrival_date                          18137 non-null  int64
 11  market_segment_type                   18137 non-null  int64
 12  repeated_guest                        18137 non-null  int64
 13  no_of_previous_cancellations          18137 non-null  int64
 14  no_of_previous_bookings_not_canceled  18137 non-null  int64
 15  avg_price_per_room                    18137 non-null  float64
 16  no_of_special_requests                18137 non-null  int64
 17  booking_status                        18137 non-null  int64
dtypes: float64(1), int64(17)
memory usage: 2.5 MB
```

```
[50] #EDA 1
     # FIND % OF ROOMS CANCELLED AND BOOKED ON SAME DAY
     # We will use Lead time column which is Number of days between the date of booking and the arrival date
     a = np.sum(df['lead_time']==0)
     print("no. of bookings on same day as arrival :",a)
     b = np.sum((df['lead_time']==0)&(df['booking_status']==1))/643 * 100
     print("% of booking cancelled on same day as arival :",b)
```

```
no. of bookings on same day as arrival : 643
% of booking cancelled on same day as arival : 5.132192846034215
```

```
[52] #EDA 2
     #FIND % OF CANCELLED HOTELS BOOKED BY SINGLE PERSON and COUPLES
     #WE WILL USE no_of_adults column
     print(np.sum(df['no_of_adults']==1))
     print(np.sum(df['no_of_adults']==2))
```

```
3809
13104
```

```
[57] b = np.sum((df['no_of_adults']==1)&(df['booking_status']==1))
     print("% of one adult room cancelled :", b/3809 * 100)
     c = np.sum((df['no_of_adults']==2)&(df['booking_status']==1))
     print("% of two adult room cancelled :", c/13104 * 100)
```

```
% of one adult room cancelled : 23.418220005250724
% of two adult room cancelled : 34.99694749694749
```

```
[59] #EDA 3
     # FIND number OF CANCELLED ROOMS BY room PRICE
     very_cheap = np.sum((df['avg_price_per_room']<=50)&(df['booking_status']==1))
     moderate = np.sum((df['avg_price_per_room']>50)&(df['avg_price_per_room']<=125)&(df['booking_status']==1))
     expensive = np.sum((df['avg_price_per_room']>125)&(df['avg_price_per_room']<=175)&(df['booking_status']==1))
     very_expensive = np.sum((df['avg_price_per_room']>175)&(df['booking_status']==1))
     print("very cheap rooms cancelled :" , very_cheap)
     print("moderate rooms cancelled :" , moderate)
     print("expensive rooms cancelled :" , expensive)
     print("very expensive rooms cancelled :",  very_expensive)
```

```
very cheap rooms cancelled : 27
moderate rooms cancelled : 4359
expensive rooms cancelled : 1308
very expensive rooms cancelled : 248
```

```
[60] #EDA 4
     #FIND NUMBER OF ROOMS BOOKED FOR EACH MONTH
     df.groupby('arrival_month').size()
```

```
arrival_month
1      501
2      853
3     1176
4     1386
5     1305
6     1607
7     1422
8     1868
9     2301
10    2678
11    1505
12    1535
dtype: int64
```

```
[25] #EDA 5
     #FIND NUMBER % OF ROOMS CANCELLED BY REPEATING AND NON REPEATING GUEST WHO HAD MADE SPECIAL REQUESTS
     df.groupby('repeated_guest').size()
```

```
repeated_guest
0    17663
1      475
dtype: int64
```

```
[64] var1 = np.sum((df['repeated_guest'] == 1) &(df['no_of_special_requests']>1)&(df['booking_status']==1))
     print("% of repeating guest who cancelled and requested special service:",var1/475 * 100)
     var2 = np.sum((df['repeated_guest'] == 0) &(df['no_of_special_requests']==0)&(df['booking_status']==1))
     print("% of non repeating guest who cancelled and requested special service:",var2/17663 * 100)
```

```
% of repeating guest who cancelled and requested special service: 0.0
% of non repeating guest who cancelled and requested special service: 24.07292079488196
```

```
[75] #EDA 6
     # CAR PARKING REQUIRED BY GUESTS
     df.groupby('required_car_parking_space').size()
```

```
required_car_parking_space
0    17563
1      574
dtype: int64
```

[69]
```
#EDA 7
# MEAL PLANS BOOKED IN COMBINATION OF BREAKFAST , LUNCH , DINNER
df.groupby('type_of_meal_plan').size()
```

```
type_of_meal_plan
0    13979
1     2543
2     1612
3        3
dtype: int64
```

[76]
```
#EDA 8
# FIND % OF ROOMS BOOKED FOR CHILDREN CANCELLED
e = np.sum(df['no_of_children']>0)
print("no. of families with children :",e)
d = np.sum((df['no_of_children']>0)&(df['booking_status']==1))/1370 * 100
print("% of cancelled rooms for children :",d)
```

```
no. of families with children : 1370
% of cancelled rooms for children : 37.81021897810219
```

[74]
```
#EDA 9
# FAMILIES WITH KIDS WHO PRE BOOKED MEALS
d = np.sum((df['no_of_children']>0)&(df['type_of_meal_plan']>0))
print("% of families with kids which pre ordered meals",c/1370 * 100)
```

```
% of families with kids which pre ordered meals 0.20845858958045094
```

```
[84] #EDA 10
     # CANCELLATION STATUS FOR HOTELS BOOKED FOR WEEKEND NIGHTS
     # 0 - NOT CANCELLED , 1 - CANCELLED
     df.groupby(['no_of_weekend_nights','booking_status']).size()
```

```
no_of_weekend_nights  booking_status
0                     0                5896
                      1                2548
1                     0                3256
                      1                1729
2                     0                2977
                      1                1558
3                     0                  44
                      1                  40
4                     0                  17
                      1                  39
5                     0                   3
                      1                  16
6                     0                   2
                      1                  11
7                     1                   1
dtype: int64
```

```
[92] #EDA 11
     # % OF SPECIAL REQUESTS FOR WEEKEND NIGHTS
     f = np.sum(df['no_of_weekend_nights']>0)
     print('bookings on weekend nights : ',f)
     g = np.sum((df['no_of_weekend_nights']>0) & (df['no_of_special_requests']>0))
     print('% of bookings for weekend nights with special requests:',g/9693 * 100)
```

```
bookings on weekend nights :  9693
% of bookings for weekend nights with special requests: 47.87991333952337
```

```
[96] #EDA 12
     #CHEAPEST ROOM PURCHASED AT
     h = np.min(df['avg_price_per_room'])
     print("cheapest room price : ",h)
```

```
cheapest room price :  0.0
```

```
[97] #EDA 13
     # MOST EXPENSIVE ROOM PURCHASED AT
     h = np.max(df['avg_price_per_room'])
     print("most expensive room price : ",h)
```

```
most expensive room price :  540.0
```

```
[97] #EDA 13
     # MOST EXPENSIVE ROOM PURCHASED AT
     h = np.max(df['avg_price_per_room'])
     print("most expensive room price : ",h)
```

```
most expensive room price :  540.0
```

```
[99] #EDA 14
     # NO. OF GUEST WHO CANCELLED MORE THAN ONCE
     i = np.sum((df['booking_status']==1)&(df['no_of_previous_cancellations']>1))
     print("number of guests who cancelled more than once : " , i)
```

```
number of guests who cancelled more than once :  3
```

```
[100] #EDA 15
      # NO. OF GUESTS WHO BOOKED AFTER CANCELLING BEFORE
      j = np.sum((df['booking_status']==0)&(df['no_of_previous_cancellations']>1))
      print("number of guests who booked after cancelling before : " , j)
```

```
number of guests who booked after cancelling before :  67
```

END OF REPORT