Notebook Index

1- Import essential libraries

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2- Read the data and do explotary data analysis

In [2]:
```python
df = pd.read_csv('Occupancy_Estimation.csv')
df.head()
```

Out[2]:

	Date	Time	S1_Temp	S2_Temp	S3_Temp	S4_Temp	S1_Light	S2_Light	S3_Light	S4_Light	S1_Sound	S2_Sound	S3_Sound	S4_Sound	S5_CO2	S5_CO2_Slope	S6_PIR	S7_PIR	Roo
0	2017/12/22	10:49:41	24.94	24.75	24.56	25.38	121	34	53	40	0.08	0.19	0.06	0.06	390	0.769231	0	0	
1	2017/12/22	10:50:12	24.94	24.75	24.56	25.44	121	33	53	40	0.93	0.05	0.06	0.06	390	0.646154	0	0	
2	2017/12/22	10:50:42	25.00	24.75	24.50	25.44	121	34	53	40	0.43	0.11	0.08	0.06	390	0.519231	0	0	
3	2017/12/22	10:51:13	25.00	24.75	24.56	25.44	121	34	53	40	0.41	0.10	0.10	0.09	390	0.388462	0	0	
4	2017/12/22	10:51:44	25.00	24.75	24.56	25.44	121	34	54	40	0.18	0.06	0.06	0.06	390	0.253846	0	0	

In [3]:
```python
df.shape
```

Out[3]:
```
(10129, 19)
```

2.1 Check for duplicates

In [4]:
```python
df[df.duplicated()]
```

Out[4]:

Date	Time	S1_Temp	S2_Temp	S3_Temp	S4_Temp	S1_Light	S2_Light	S3_Light	S4_Light	S1_Sound	S2_Sound	S3_Sound	S4_Sound	S5_CO2	S5_CO2_Slope	S6_PIR	S7_PIR	Room_Occupa

In [5]:
```python
df.columns
```

Out[5]:
```
Index(['Date', 'Time', 'S1_Temp', 'S2_Temp', 'S3_Temp', 'S4_Temp', 'S1_Light',
       'S2_Light', 'S3_Light', 'S4_Light', 'S1_Sound', 'S2_Sound', 'S3_Sound',
       'S4_Sound', 'S5_CO2', 'S5_CO2_Slope', 'S6_PIR', 'S7_PIR',
       'Room_Occupancy_Count'],
      dtype='object')
```

2.2 Check data types and null values

In [6]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10129 entries, 0 to 10128
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Date                  10129 non-null  object
 1   Time                  10129 non-null  object
 2   S1_Temp               10129 non-null  float64
 3   S2_Temp               10129 non-null  float64
 4   S3_Temp               10129 non-null  float64
 5   S4_Temp               10129 non-null  float64
 6   S1_Light              10129 non-null  int64
 7   S2_Light              10129 non-null  int64
 8   S3_Light              10129 non-null  int64
 9   S4_Light              10129 non-null  int64
 10  S1_Sound              10129 non-null  float64
 11  S2_Sound              10129 non-null  float64
 12  S3_Sound              10129 non-null  float64
 13  S4_Sound              10129 non-null  float64
 14  S5_CO2                10129 non-null  int64
 15  S5_CO2_Slope          10129 non-null  float64
 16  S6_PIR                10129 non-null  int64
 17  S7_PIR                10129 non-null  int64
 18  Room_Occupancy_Count  10129 non-null  int64
dtypes: float64(9), int64(8), object(2)
memory usage: 1.5+ MB
```

In [7]:
```python
df.describe()
```

Out[7]:

	S1_Temp	S2_Temp	S3_Temp	S4_Temp	S1_Light	S2_Light	S3_Light	S4_Light	S1_Sound	S2_Sound	S3_Sound	S4_Sound	S5_CO2	S5_CO
count	10129.000000	10129.000000	10129.000000	10129.000000	10129.000000	10129.00000	10129.000000	10129.000000	10129.000000	10129.000000	10129.000000	10129.000000	10129.000000	10129
mean	25.454012	25.546059	25.056621	25.754125	25.445059	26.01629	34.248494	13.220259	0.168178	0.120066	0.158119	0.103840	460.860401	-0
std	0.351351	0.586325	0.427283	0.356434	51.011264	67.30417	58.400744	19.602219	0.316709	0.266503	0.413637	0.120683	199.964940	1
min	24.940000	24.750000	24.440000	24.940000	0.000000	0.00000	0.000000	0.000000	0.060000	0.040000	0.040000	0.050000	345.000000	-6
25%	25.190000	25.190000	24.690000	25.440000	0.000000	0.00000	0.000000	0.000000	0.070000	0.050000	0.060000	0.060000	355.000000	-0
50%	25.380000	25.380000	24.940000	25.750000	0.000000	0.00000	0.000000	0.000000	0.080000	0.050000	0.060000	0.080000	360.000000	0
75%	25.630000	25.630000	25.380000	26.000000	12.000000	14.00000	50.000000	22.000000	0.080000	0.060000	0.070000	0.100000	465.000000	0
max	26.380000	29.000000	26.190000	26.560000	165.000000	258.00000	280.000000	74.000000	3.880000	3.440000	3.670000	3.400000	1270.000000	8

2.3 Check for classes balance

In [8]:
```python
df.Room_Occupancy_Count.value_counts()
```

Out[8]:
```
0    8228
2     748
3     694
1     459
Name: Room_Occupancy_Count, dtype: int64
```

2.3 result: There is imbalance, we will deal with if the score are affected

2.4 Merge the data with time as DateTime and drop the columns Data and Time

In [9]:
```python
DateTimeConc = df['Date'] + ' ' + df['Time']
df['DateTime'] = pd.to_datetime(DateTimeConc)
```

In [10]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10129 entries, 0 to 10128
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Date                  10129 non-null  object
 1   Time                  10129 non-null  object
 2   S1_Temp               10129 non-null  float64
 3   S2_Temp               10129 non-null  float64
 4   S3_Temp               10129 non-null  float64
 5   S4_Temp               10129 non-null  float64
 6   S1_Light              10129 non-null  int64
 7   S2_Light              10129 non-null  int64
 8   S3_Light              10129 non-null  int64
 9   S4_Light              10129 non-null  int64
 10  S1_Sound              10129 non-null  float64
 11  S2_Sound              10129 non-null  float64
 12  S3_Sound              10129 non-null  float64
 13  S4_Sound              10129 non-null  float64
 14  S5_CO2                10129 non-null  int64
 15  S5_CO2_Slope          10129 non-null  float64
 16  S6_PIR                10129 non-null  int64
 17  S7_PIR                10129 non-null  int64
 18  Room_Occupancy_Count  10129 non-null  int64
 19  DateTime              10129 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(9), int64(8), object(2)
memory usage: 1.5+ MB
```

In [11]:
```python
df.drop(columns=['Date','Time'], inplace = True)
```

In [12]:
```python
df.columns
```

Out[12]:
```
Index(['S1_Temp', 'S2_Temp', 'S3_Temp', 'S4_Temp', 'S1_Light', 'S2_Light',
       'S3_Light', 'S4_Light', 'S1_Sound', 'S2_Sound', 'S3_Sound', 'S4_Sound',
       'S5_CO2', 'S5_CO2_Slope', 'S6_PIR', 'S7_PIR', 'Room_Occupancy_Count',
       'DateTime'],
      dtype='object')
```

2.5 Check features corralation

In [13]:
```python
df.corr()
```

Out[13]:

	S1_Temp	S2_Temp	S3_Temp	S4_Temp	S1_Light	S2_Light	S3_Light	S4_Light	S1_Sound	S2_Sound	S3_Sound	S4_Sound	S5_CO2	S5_CO2_Slope	S6_PIR	S7_PIR
S1_Temp	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S2_Temp	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S3_Temp	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S4_Temp	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S1_Light	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S2_Light	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S3_Light	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S4_Light	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S1_Sound	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S2_Sound	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S3_Sound	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S4_Sound	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]
S5_CO2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]
S5_CO2_Slope	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]
S6_PIR	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]
S7_PIR	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000
Room_Occupancy_Count	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

2.6 Aggregating features to deal with the corralation issue

In [14]:
```python
df['S1_4_Temp_Mean'] = (df['S1_Temp'] + df['S2_Temp'] + df['S3_Temp'] + df['S4_Temp']) / 4
df['S1_4_Light_Mean'] = (df['S1_Light'] + df['S2_Light'] + df['S3_Light'] + df['S4_Light']) / 4
df['S1_4_Sound_Mean'] = (df['S1_Sound'] + df['S2_Sound'] + df['S3_Sound'] + df['S4_Sound']) / 4
df['S6_7_PIR_Sum'] = df['S6_PIR'] + df['S7_PIR']
```

In [15]:
```python
#Original_Features = ['S1_Temp', 'S2_Temp', 'S3_Temp', 'S4_Temp', 'S1_Light', 'S2_Light',
#      'S3_Light', 'S4_Light', 'S1_Sound', 'S2_Sound', 'S3_Sound', 'S4_Sound',
#      'S5_CO2', 'S5_CO2_Slope', 'S6_PIR', 'S7_PIR']
Aggregated_Features = ['S1_4_Temp_Mean', 'S1_4_Light_Mean', 'S1_4_Sound_Mean',
       'S6_7_PIR_Sum', 'S5_CO2', 'S5_CO2_Slope']
```

In [16]:
```python
list_df_Agg = Aggregated_Features.copy()
list_df_Agg.insert(0, "Room_Occupancy_Count")
```

In [17]:
```python
df[list_df_Agg].describe()
```

Out[17]:

	Room_Occupancy_Count	S1_4_Temp_Mean	S1_4_Light_Mean	S1_4_Sound_Mean	S6_7_PIR_Sum	S5_CO2	S5_CO2_Slope
count	10129.000000	10129.000000	10129.000000	10129.000000	10129.000000	10129.000000	10129.000000
mean	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	460.860401	[illegible]
std	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	199.964940	[illegible]
min	0.000000	[illegible]	0.000000	[illegible]	0.000000	345.000000	[illegible]
25%	0.000000	[illegible]	0.000000	[illegible]	0.000000	355.000000	[illegible]
50%	0.000000	[illegible]	0.000000	[illegible]	0.000000	360.000000	[illegible]
75%	0.000000	[illegible]	[illegible]	[illegible]	0.000000	465.000000	[illegible]
max	3.000000	[illegible]	[illegible]	[illegible]	2.000000	1270.000000	[illegible]

2.7 Check the corralation after aggregation

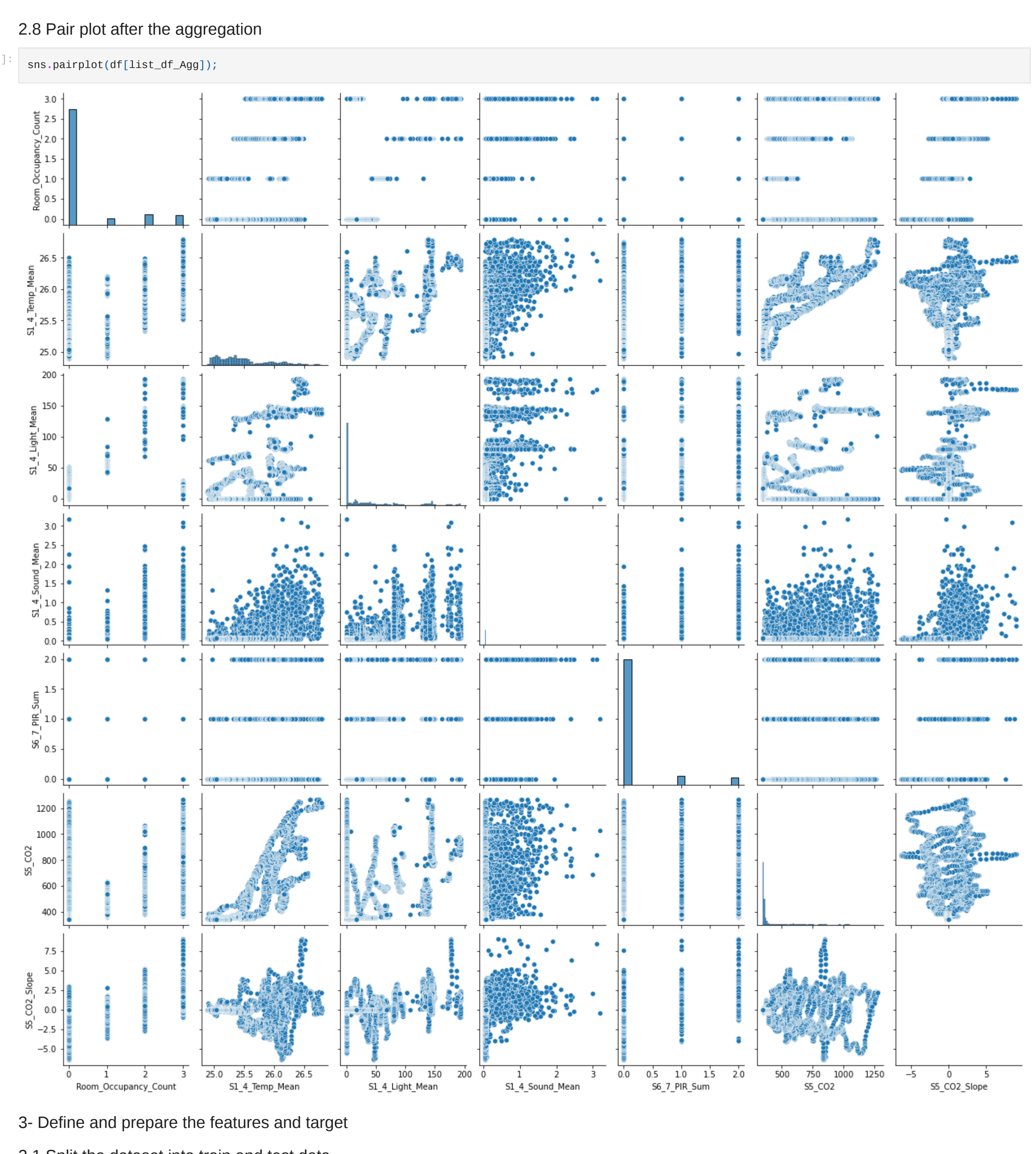
In [18]:
```python
df[list_df_Agg].corr()
```

Out[18]:

	Room_Occupancy_Count	S1_4_Temp_Mean	S1_4_Light_Mean	S1_4_Sound_Mean	S6_7_PIR_Sum	S5_CO2	S5_CO2_Slope
Room_Occupancy_Count	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S1_4_Temp_Mean	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S1_4_Light_Mean	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]	[illegible]
S1_4_Sound_Mean	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]	[illegible]
S6_7_PIR_Sum	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]	[illegible]
S5_CO2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000	[illegible]
S5_CO2_Slope	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1.000000

2.8 Pair plot after the aggregation

In [19]:
```python
sns.pairplot(df[list_df_Agg]);
```



3- Define and prepare the features and target

3.1 Split the dataset into train and test data

In [20]:
```python
from sklearn.model_selection import train_test_split

X = df[Aggregated_Features]
y = df['Room_Occupancy_Count']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

In [21]:
```python
print(X_train)
```

```
[illegible]

[8103 rows x 6 columns]
```

In [22]:
```python
print(X_test)
```

```
[illegible]

[2026 rows x 6 columns]
```

In [23]:
```python
print(y_train)
```

```
[illegible]
Name: Room_Occupancy_Count, Length: 8103, dtype: int64
```

In [24]:
```python
print(y_test)
```

```
[illegible]
Name: Room_Occupancy_Count, Length: 2026, dtype: int64
```

3.2 Scale the features (Standardization)

In [25]:
```python
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train_Scl = sc.fit_transform(X_train)
X_test_Scl = sc.transform(X_test)
```

In [26]:
```python
print(X_train_Scl)
```

```
[illegible]
```

In [27]:
```python
print(X_test_Scl)
```

```
[illegible]
```

4- Initiate the machine learning models

4.1 Logistic Regression Model with evaluation metrics and coefficients

In [28]:
```python
from sklearn.linear_model import LogisticRegression

LR = LogisticRegression(random_state = 42, max_iter = 1000)
LR.fit(X_train, y_train)

LR_SclFit = LogisticRegression(random_state = 42)
LR_SclFit.fit(X_train_Scl, y_train)
```

Out[28]:
```
LogisticRegression(random_state=42)
```

In [29]:
```python
LR_y_pred = LR.predict(X_test)
LR_SclFt_y_pred = LR_SclFit.predict(X_test_Scl)
```

In [30]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

LR_cm = confusion_matrix(y_test, LR_y_pred)
LR_SclFt_cm = confusion_matrix(y_test, LR_SclFt_y_pred)

print('LR confusion matrix')
print(LR_cm)
print('=======')
print('LR confusion matrix with standardized features')
print(LR_SclFt_cm)
print('-------')

print('LR accuracy score')
print(accuracy_score(y_test, LR_y_pred))
print('=======')
print('LR accuracy score with standardized features')
print(accuracy_score(y_test, LR_SclFt_y_pred))
print('-------')

print('LR report')
print(classification_report(y_test, LR_y_pred))
print('=======')
print('LR report with standardized features')
print(classification_report(y_test, LR_SclFt_y_pred))
```

```
LR confusion matrix
[illegible]
=======
LR confusion matrix with standardized features
[illegible]
-------
LR accuracy score
[illegible]
=======
LR accuracy score with standardized features
[illegible]
-------
LR report
              precision    recall  f1-score   support
[illegible]
=======
LR report with standardized features
              precision    recall  f1-score   support
[illegible]
```

4.1 Result: The bench mark model. it has a greate overall scores. but the individual classes (2 and 3) has a low scores

In [31]:
```python
print(LR.coef_)
print('=======')
print(LR_SclFit.coef_)
```

```
[illegible]
```

In [32]:
```python
print(LR.intercept_)
print('=======')
print(LR_SclFit.intercept_)
```

```
[illegible]
```

4.2 Gaussian naïve bayes with evaluation metrics and class probabilities

[illegible]

4.2 Result: The second best classifier out of the three classifiers. The score of individual classes are good except Class (3) has a low scores but still acceptable

4.3 XGboost with evaluation metrics and features importance plot

[illegible]

4.3 Result: The highest score out of the three models. Also the score for the indvidual classes are high

[illegible]