Importing Libraies

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, roc_curve, roc_auc_score
```

```
import warnings
warnings.filterwarnings('ignore')
```

Data Reading

```
data = pd.read_csv('datatraining.txt', error_bad_lines=False)
data_test_1 = pd.read_csv('datatest1.txt', error_bad_lines=False)
data_test_2 = pd.read_csv('datatest2.txt', error_bad_lines=False)
```

Data Nature

Number of (rows, columns)

```
data.shape
```

```
(8143, 7)
```

```
data_test_1.shape
```

```
(2665, 7)
```

```
data_test_2.shape
```

```
(9752, 7)
```

Snippets form data

```
data.head()
```

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
1	2015-02-04 17:51:00	23.18	27.2720	426.0	721.25	0.004793	1
2	2015-02-04 17:51:59	23.15	27.2675	429.5	714.00	0.004783	1
3	2015-02-04 17:53:00	23.15	27.2450	426.0	713.50	0.004779	1
4	2015-02-04 17:54:00	23.15	27.2000	426.0	708.25	0.004772	1
5	2015-02-04 17:55:00	23.10	27.2000	426.0	704.50	0.004757	1

Data information

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8143 entries, 1 to 8143
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
```

```
 0   date           8143 non-null   object 
 1   Temperature    8143 non-null   float64
 2   Humidity       8143 non-null   float64
 3   Light          8143 non-null   float64
 4   CO2            8143 non-null   float64
 5   HumidityRatio  8143 non-null   float64
 6   Occupancy      8143 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 508.9+ KB
```

Data statistics

```
data.describe()
```

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
count	8143.000000	8143.000000	8143.000000	8143.000000	8143.000000	8143.000000
mean	20.619084	25.731507	119.519375	606.546243	0.003863	0.212330
std	1.016916	5.531211	194.755805	314.320877	0.000852	0.408982
min	19.000000	16.745000	0.000000	412.750000	0.002674	0.000000
25%	19.700000	20.200000	0.000000	439.000000	0.003078	0.000000
50%	20.390000	26.222500	0.000000	453.500000	0.003801	0.000000
75%	21.390000	30.533333	256.375000	638.833333	0.004352	0.000000
max	23.180000	39.117500	1546.333333	2028.500000	0.006476	1.000000

Count valuew of Occupancy

```
data['Occupancy'].value_counts()
```

```
0    6414
1    1729
Name: Occupancy, dtype: int64
```

```
data_test_1['Occupancy'].value_counts()
```

```
0    1693
1     972
Name: Occupancy, dtype: int64
```

```
data_test_2['Occupancy'].value_counts()
```

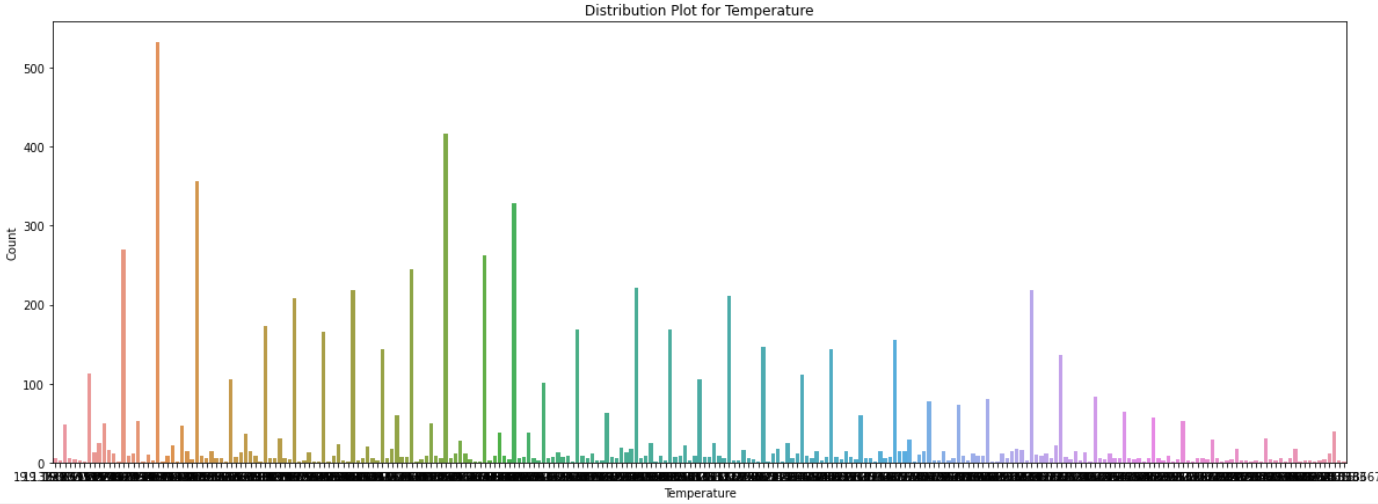
```
0    7703
1    2049
Name: Occupancy, dtype: int64
```

Drop column date

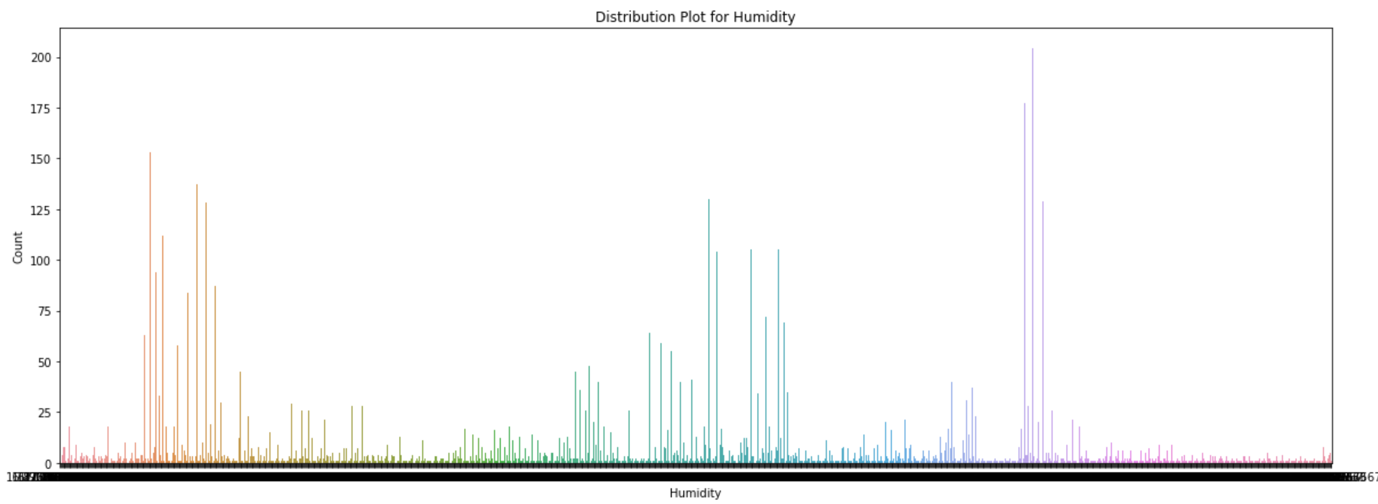
```
data.drop('date', axis=1, inplace=True)
data_test_1.drop('date', axis=1, inplace=True)
data_test_2.drop('date', axis=1, inplace=True)
```

Data Distribution

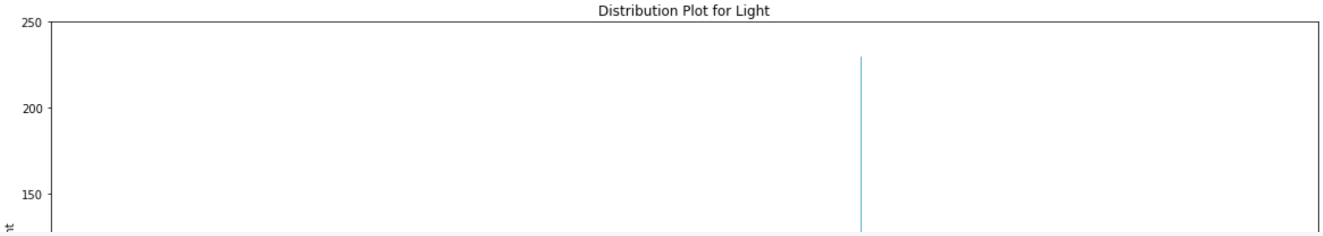
```
plt.figure(figsize=(20,7))
sns.countplot(x=data['Temperature'])
plt.title('Distribution Plot for Temperature')
plt.xlabel('Temperature')
plt.ylabel('Count')
plt.savefig('Distribution Plot for Temperature.png')
plt.show()
```



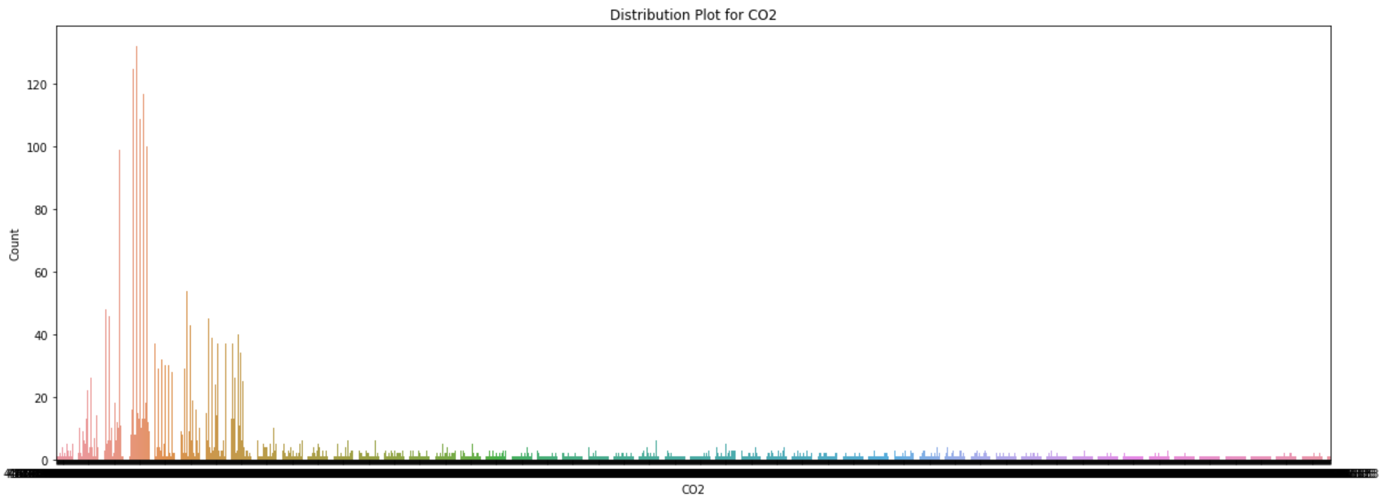

```
plt.figure(figsize=(20,7))
sns.countplot(x=data['Humidity'])
plt.title('Distribution Plot for Humidity')
plt.xlabel('Humidity')
plt.ylabel('Count')
plt.savefig('Distribution Plot for Humidity.png')
plt.show()
```



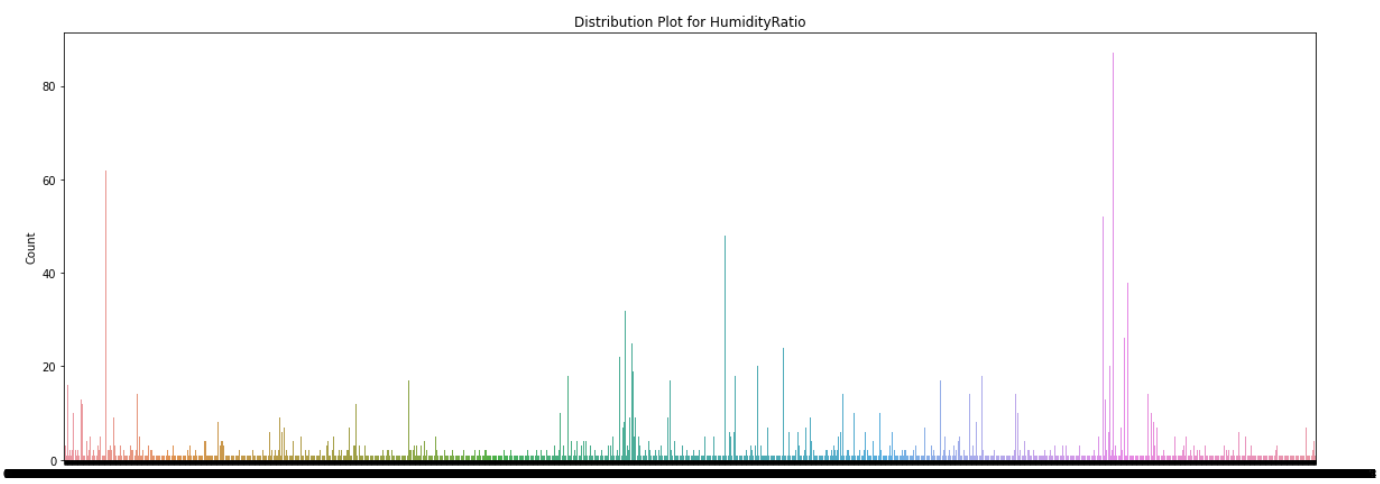

```
plt.figure(figsize=(20,7))
sns.countplot(x=data['Light'])
plt.ylim(0, 250)
plt.title('Distribution Plot for Light')
plt.xlabel('Light')
plt.ylabel('Count')
plt.savefig('Distribution Plot for Light.png')
plt.show()
```



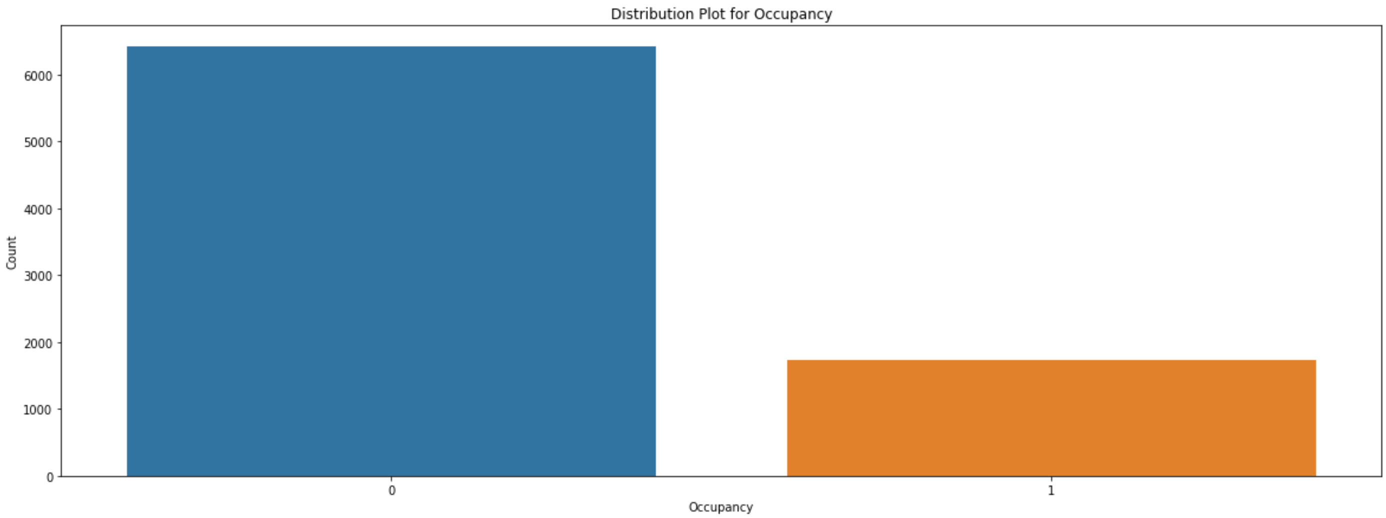

```
plt.figure(figsize=(20,7))
sns.countplot(x=data['CO2'])
plt.title('Distribution Plot for CO2')
plt.xlabel('CO2')
plt.ylabel('Count')
plt.savefig('Distribution Plot for CO2.png')
plt.show()
```




```
plt.figure(figsize=(20,7))
sns.countplot(x=data['HumidityRatio'])
plt.title('Distribution Plot for HumidityRatio')
plt.xlabel('HumidityRatio')
plt.ylabel('Count')
plt.savefig('Distribution Plot for HumidityRatio.png')
plt.show()
```

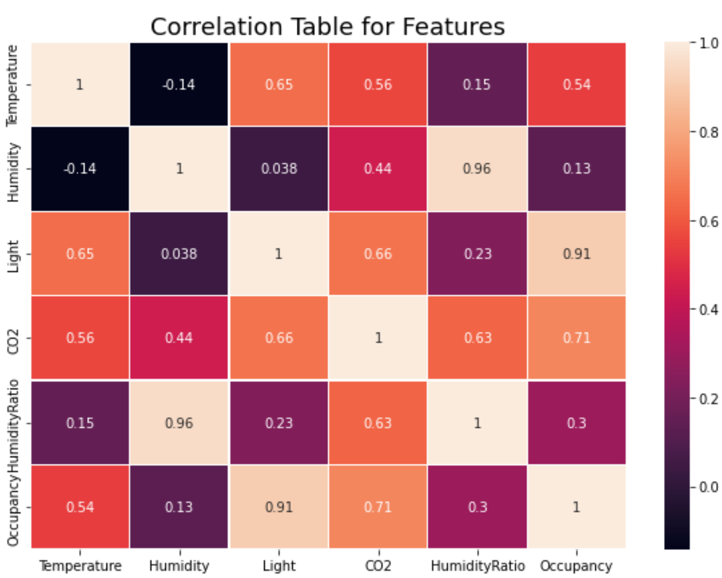


```
plt.figure(figsize=(20,7))
sns.countplot(x=data['Occupancy'])
plt.title('Distribution Plot for Occupancy')
plt.xlabel('Occupancy')
plt.ylabel('Count')
plt.savefig('Distribution Plot for Occupancy.png')
plt.show()
```



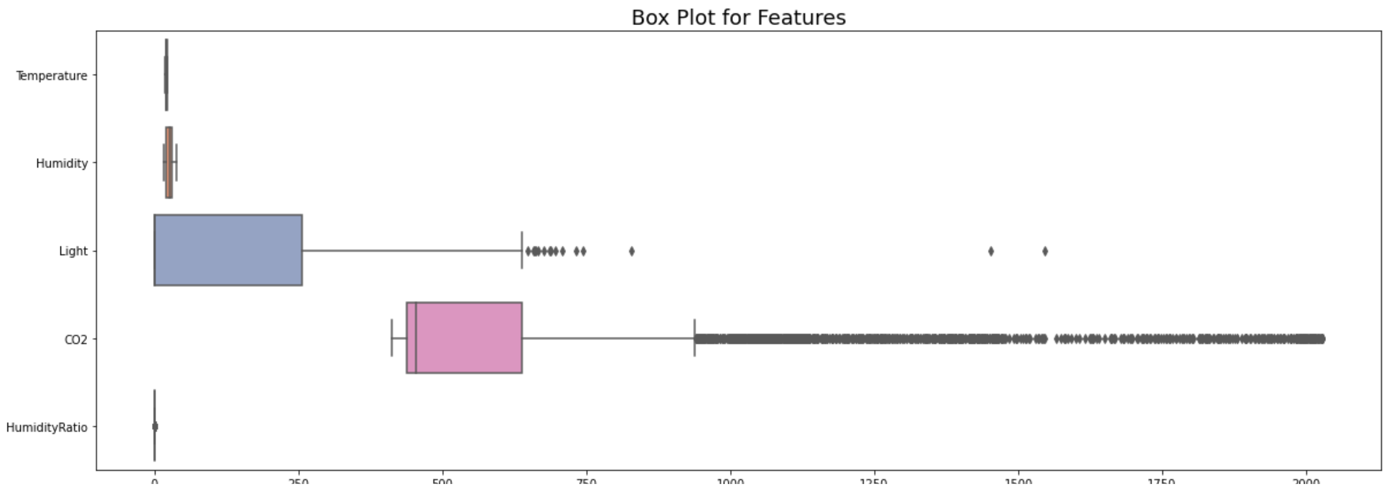
Heatmap of correlation

```
plt.figure(figsize=(10,7))
plt.title('Correlation Table for Features', fontdict={'fontsize':18})
ax = sns.heatmap(data.corr(), annot=True, linewidths=.2)
plt.savefig('Correlation Table for Feature.png')
```



Box Plot for Features

```
plt.figure(figsize=(20,7))
plt.title('Box Plot for Features', fontdict={'fontsize':18})
ax = sns.boxplot(data=data.drop(['Occupancy'], axis=1), orient="h", palette="Set2")
plt.savefig('Box Plot for Features.png')
```



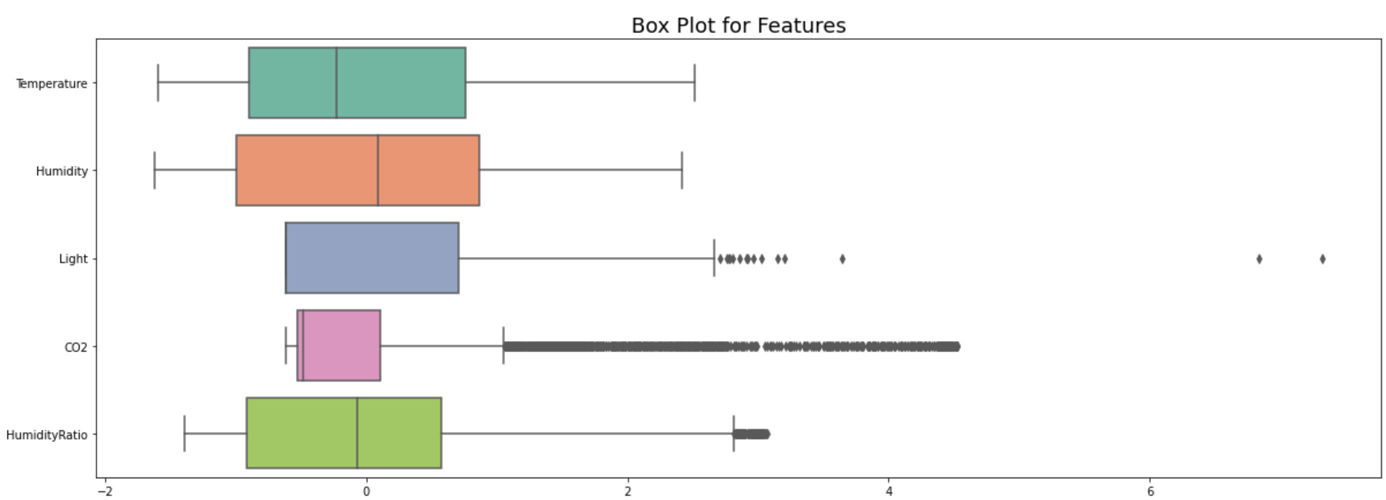
Data Scaling

Scale data using Standard Scaler

```
scaler = StandardScaler()
columns = ['Temperature', 'Humidity', 'Light', 'CO2', 'HumidityRatio']
scaler.fit(np.array(data[columns]))

data[columns] = scaler.transform(np.array(data[columns]))
data_test_1[columns] = scaler.transform(np.array(data_test_1[columns]))
data_test_2[columns] = scaler.transform(np.array(data_test_2[columns]))
```

Box Plot after Scaling

```
plt.figure(figsize=(20,7))
plt.title('Box Plot for Features', fontdict={'fontsize':18})
ax = sns.boxplot(data=data.drop(['Occupancy'], axis=1), orient="h", palette="Set2")
plt.savefig('Box Plot for Features After scaling.png')
```



Data Preprocessing

Spliting data into X, Y

```
y_values = data['Occupancy'].values
x_values = data[['Temperature', 'Humidity', 'Light',    'CO2',  'HumidityRatio']].values
```

```
x_values.shape, y_values.shape
```

```
((8143, 5), (8143,))
```

Spliting test data 1 & 2 into X,Y

```
y_test_1 = data_test_1['Occupancy'].values
x_test_1 = data_test_1[['Temperature',  'Humidity', 'Light',    'CO2',  'HumidityRatio']].values

y_test_2 = data_test_2['Occupancy'].values
x_test_2 = data_test_2[['Temperature',  'Humidity', 'Light',    'CO2',  'HumidityRatio']].values
```

```
y_test_1.shape, x_test_1.shape
```

```
((2665,), (2665, 5))
```

```
y_test_2.shape, x_test_2.shape
```

```
((9752,), (9752, 5))
```

Split data into train & test data

```
x_train, x_test, y_train, y_test = train_test_split(x_values, y_values, test_size=0.3, random_state=40)
```

```
x_train.shape, y_train.shape
```

```
((5700, 5), (5700,))
```

```
x_test.shape, y_test.shape
```

```
((2443, 5), (2443,))
```

Logistic Regression Algorithm

```
LR_Model = LogisticRegression(penalty='l2', dual=False, tol=1e-4, C=1, fit_intercept=True, intercept_scaling=1, class_weight=None, rando
                           solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)
LR_Model = LR_Model.fit(x_train, y_train)
```

```
lr_y_pred   = LR_Model.predict(x_test)
lr_y_pred_1 = LR_Model.predict(x_test_1)
lr_y_pred_2 = LR_Model.predict(x_test_2)
```

```
print('---------------------------------------------------------------------------------------')
print("{:20}\t|\t{:20}\t|\t{:20}".format('Original', 'Test 1', 'Test 2'))
print('---------------------------------------------------------------------------------------')
print("{:>15}\t|\t{:>15}\t|\t{:>15}".format(
    accuracy_score(y_test, lr_y_pred)*100,
    accuracy_score(y_test_1, lr_y_pred_1)*100,
    accuracy_score(y_test_2, lr_y_pred_2)*100)
)
print('---------------------------------------------------------------------------------------')
```

```
---------------------------------------------------------------------------------------
Original                |       Test 1                  |       Test 2
---------------------------------------------------------------------------------------
98.32173557101925       |       97.4859287054409        |       98.65668580803938
---------------------------------------------------------------------------------------
```

```
print(classification_report(y_test, lr_y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      1924
           1       0.94      0.98      0.96       519
```

```
    accuracy                           0.98      2443
   macro avg       0.97      0.98      0.98      2443
weighted avg       0.98      0.98      0.98      2443
```

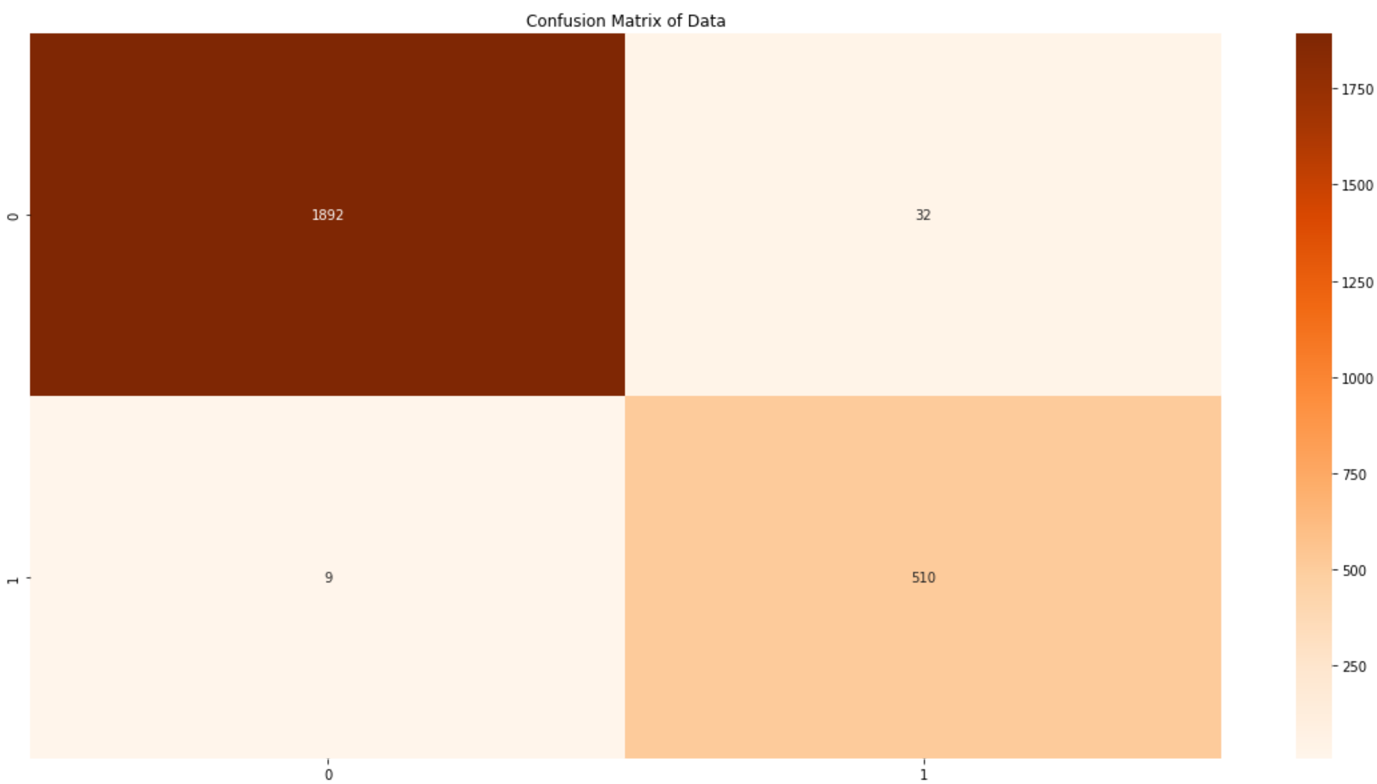
```
print(classification_report(y_test_1, lr_y_pred_1))
```

```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1693
           1       0.95      0.99      0.97       972

    accuracy                           0.97      2665
   macro avg       0.97      0.98      0.97      2665
weighted avg       0.98      0.97      0.97      2665
```

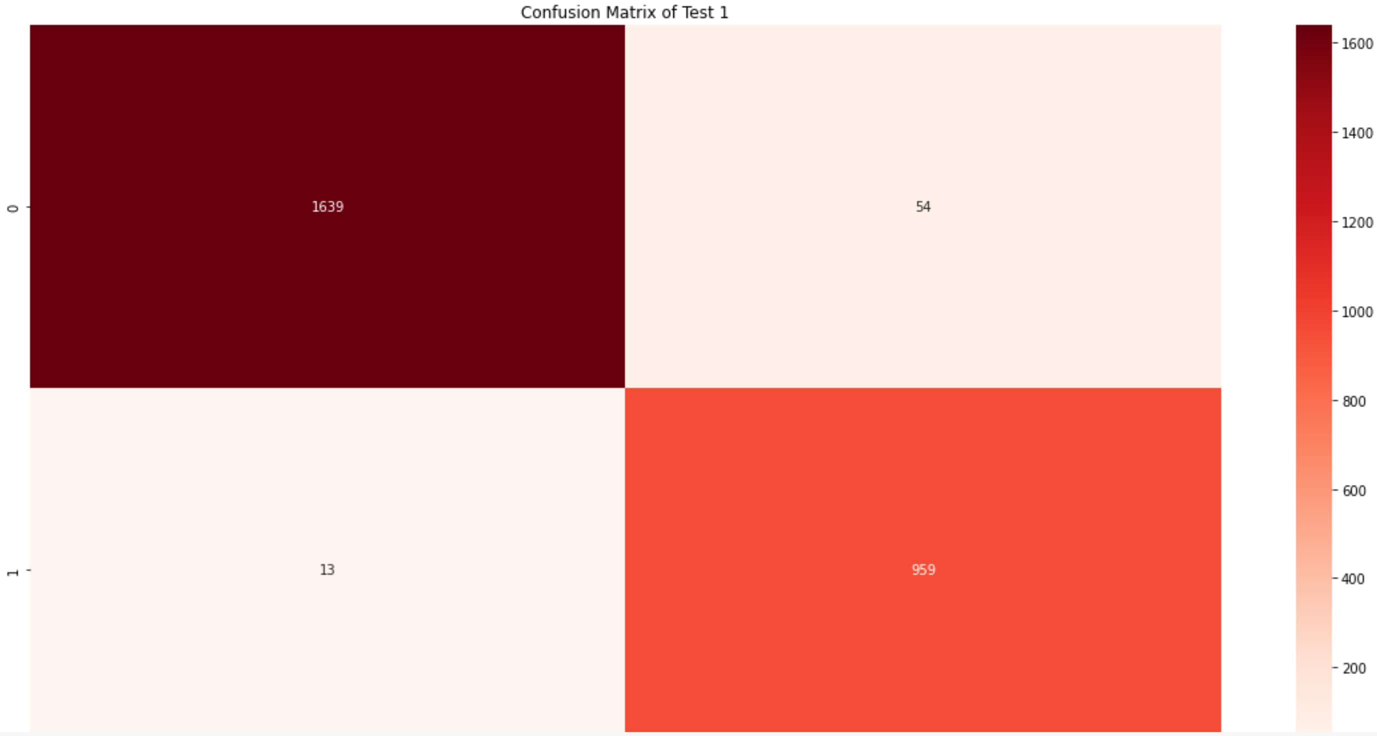
```
print(classification_report(y_test_2, lr_y_pred_2))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      7703
           1       0.97      0.96      0.97      2049

    accuracy                           0.99      9752
   macro avg       0.98      0.98      0.98      9752
weighted avg       0.99      0.99      0.99      9752
```

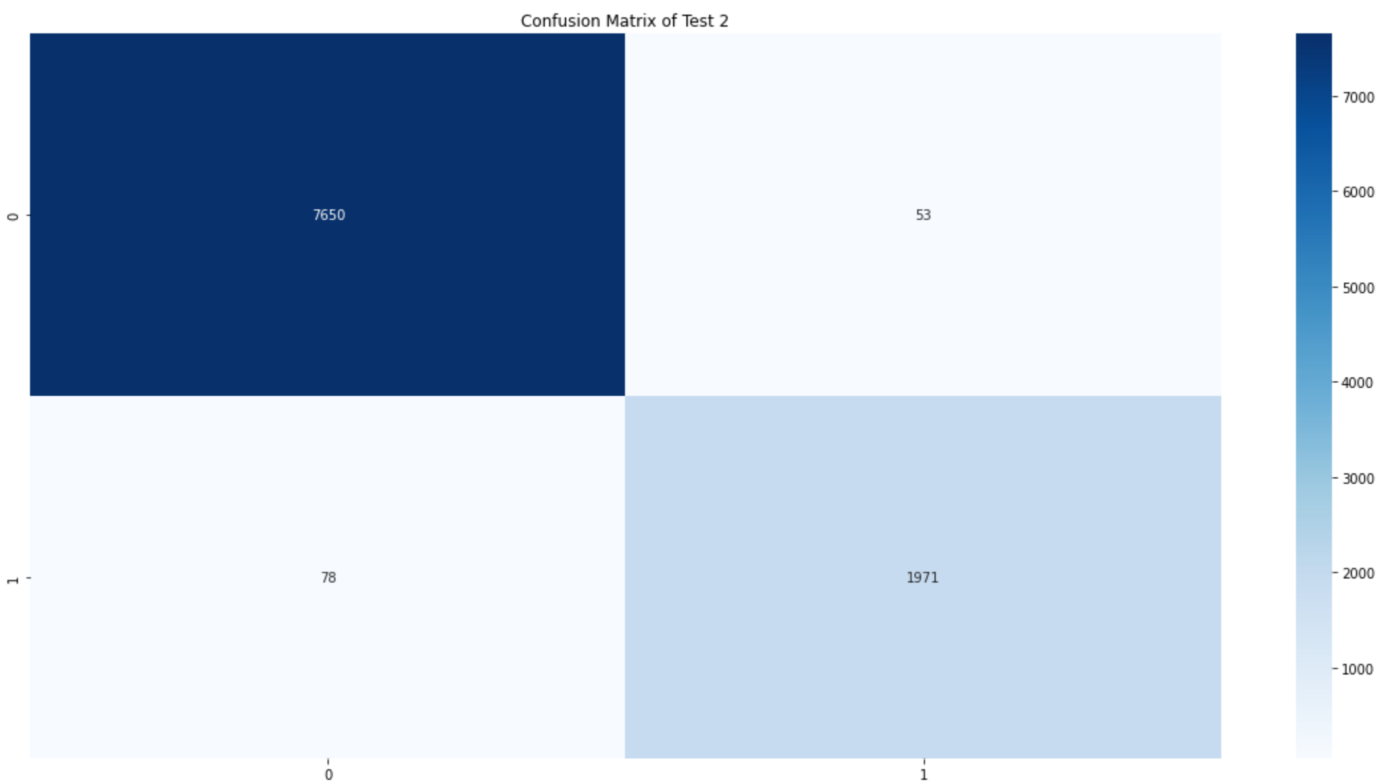
```
plt.figure(figsize=(20,10))
sns.heatmap(confusion_matrix(y_test, lr_y_pred), annot=True, cmap='Oranges', fmt="d")
plt.title('Confusion Matrix of Data')
plt.savefig('Confusion Matrix of Data Logistic Regression.png')
plt.show()
```




```
plt.figure(figsize=(20,10))
sns.heatmap(confusion_matrix(y_test_1, lr_y_pred_1), annot=True, cmap='Reds', fmt="d")
plt.title('Confusion Matrix of Test 1')
plt.savefig('Confusion Matrix of Test 1 Logistic Regression.png')
plt.show()
```

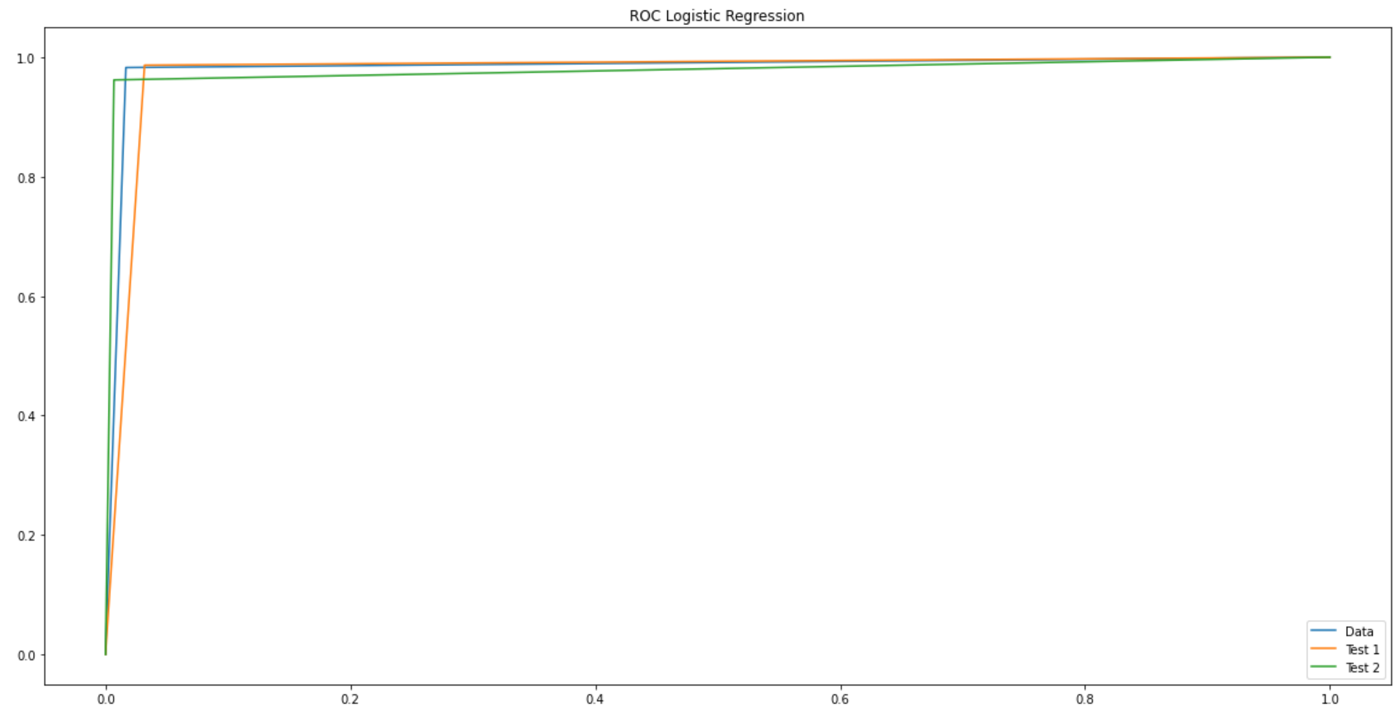



```
plt.figure(figsize=(20,10))
sns.heatmap(confusion_matrix(y_test_2, lr_y_pred_2), annot=True, cmap='Blues', fmt="d")
plt.title('Confusion Matrix of Test 2')
plt.savefig('Confusion Matrix of Test 2 Logistic Regression.png')
plt.show()
```




```
plt.figure(figsize=(20,10))

y_pred_proba = LR_Model.predict(x_test)
fpr, tpr, _ = roc_curve(y_test,  y_pred_proba)
plt.plot(fpr, tpr, label="Data")

y_pred_proba_1 = LR_Model.predict(x_test_1)
fpr, tpr, _ = roc_curve(y_test_1,  y_pred_proba_1)
```

```
plt.plot(fpr, tpr, label="Test 1")

y_pred_proba_2 = LR_Model.predict(x_test_2)
fpr, tpr, _ = roc_curve(y_test_2,  y_pred_proba_2)
plt.plot(fpr, tpr, label="Test 2")

plt.title('ROC Logistic Regression')
plt.legend(loc=4)
plt.savefig('ROC Logistic Regression.png')
plt.show()
```



Support Vector Machine Algorithm

```
LSVC_Model = LinearSVC()
LSVC_Model = LSVC_Model.fit(x_train, y_train)
```

```
svc_y_pred   = LSVC_Model.predict(x_test)
svc_y_pred_1 = LSVC_Model.predict(x_test_1)
svc_y_pred_2 = LSVC_Model.predict(x_test_2)
```

```
print('--------------------------------------------------------------------------------------')
print("{:20}\t|\t{:20}\t|\t{:20}".format('Original', 'Test 1', 'Test 2'))
print('--------------------------------------------------------------------------------------')
print("{:>15}\t|\t{:>15}\t|\t{:>15}".format(
    accuracy_score(y_test,   svc_y_pred)  *100,
    accuracy_score(y_test_1, svc_y_pred_1)*100,
    accuracy_score(y_test_2, svc_y_pred_2)*100)
)
print('--------------------------------------------------------------------------------------')
```

```
--------------------------------------------------------------------------------------
Original                |       Test 1                  |       Test 2
--------------------------------------------------------------------------------------
98.32173557101925       |       97.37335834896811       |       98.07219031993438
--------------------------------------------------------------------------------------
```

```
print(classification_report(y_test, svc_y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      1924
```

```
           1       0.94      0.98      0.96       519

    accuracy                           0.98      2443
   macro avg       0.97      0.98      0.98      2443
weighted avg       0.98      0.98      0.98      2443
```

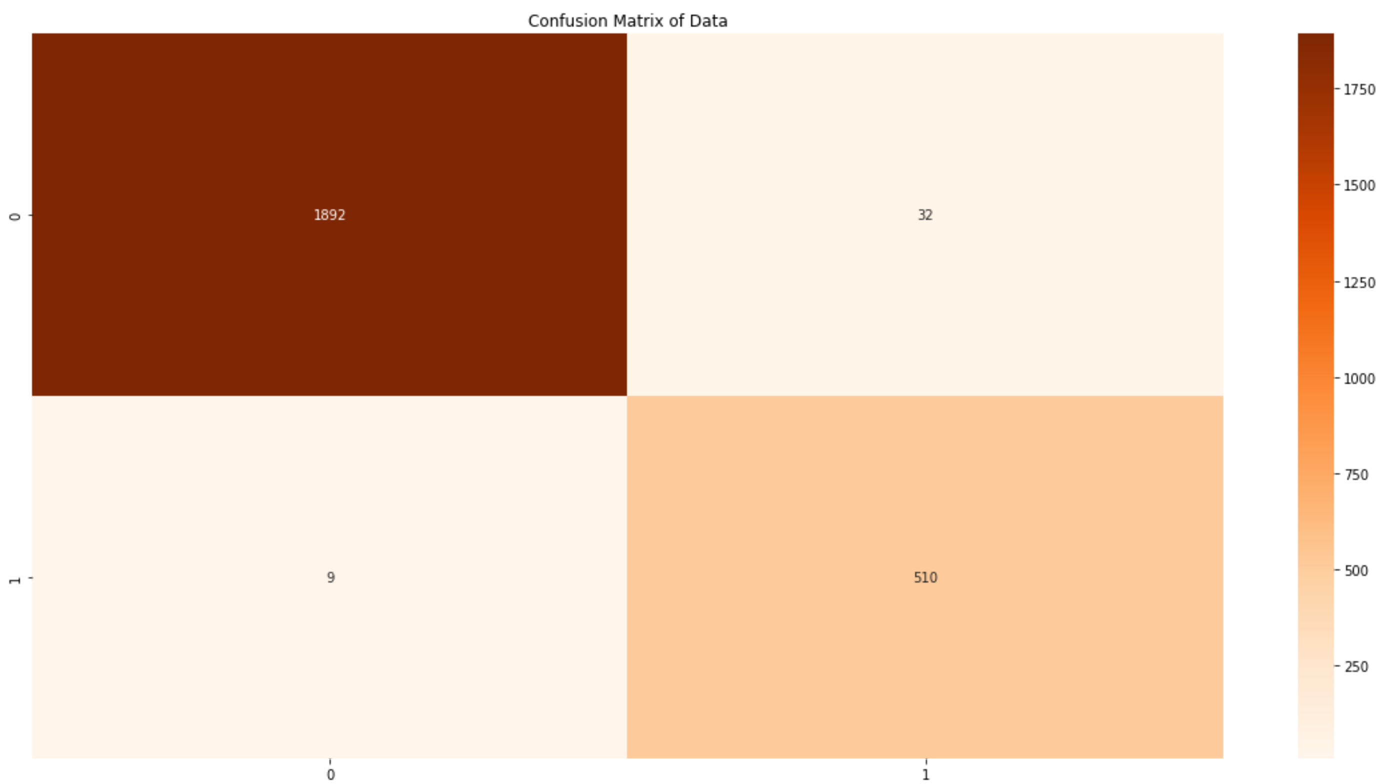
```
print(classification_report(y_test_1, svc_y_pred_1))
```

```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1693
           1       0.95      0.98      0.96       972

    accuracy                           0.97      2665
   macro avg       0.97      0.98      0.97      2665
weighted avg       0.97      0.97      0.97      2665
```

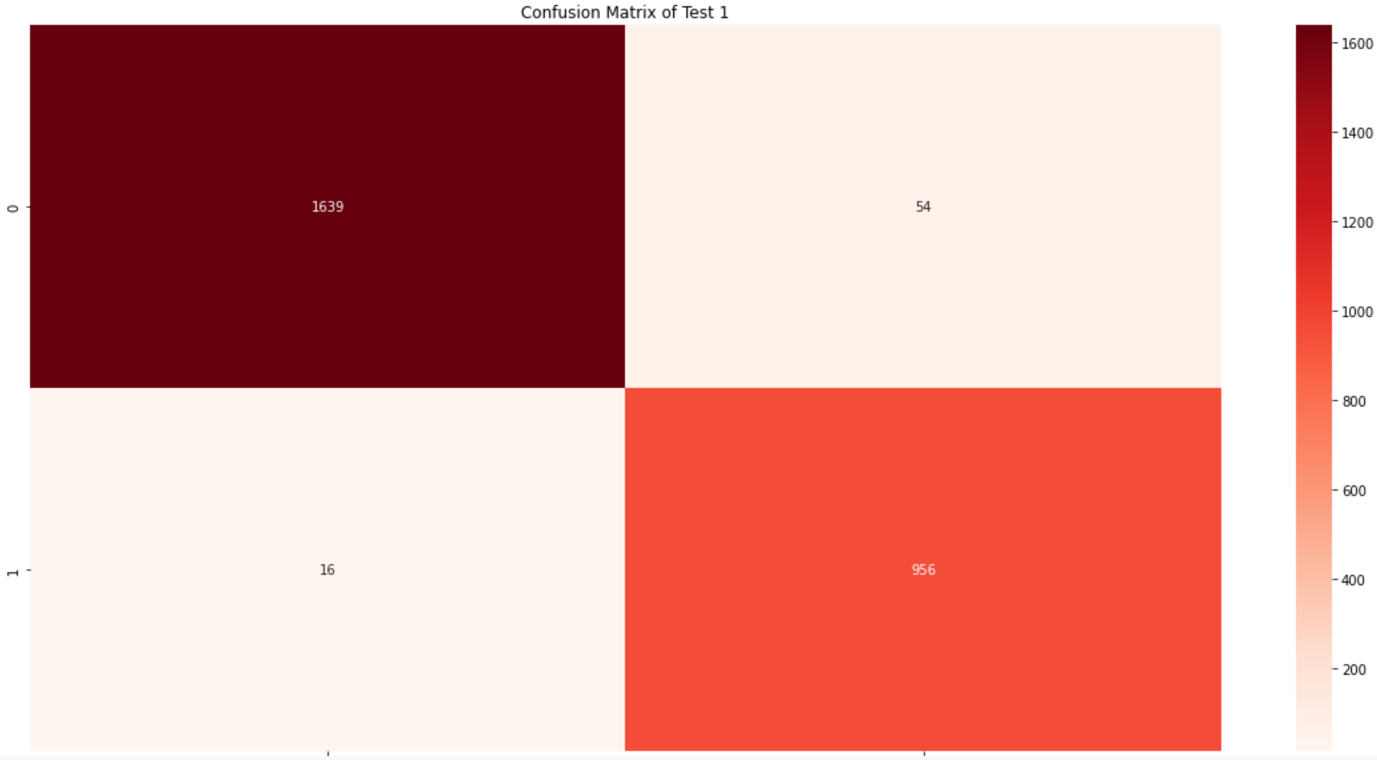
```
print(classification_report(y_test_2, svc_y_pred_2))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      7703
           1       0.96      0.94      0.95      2049

    accuracy                           0.98      9752
   macro avg       0.97      0.97      0.97      9752
weighted avg       0.98      0.98      0.98      9752
```

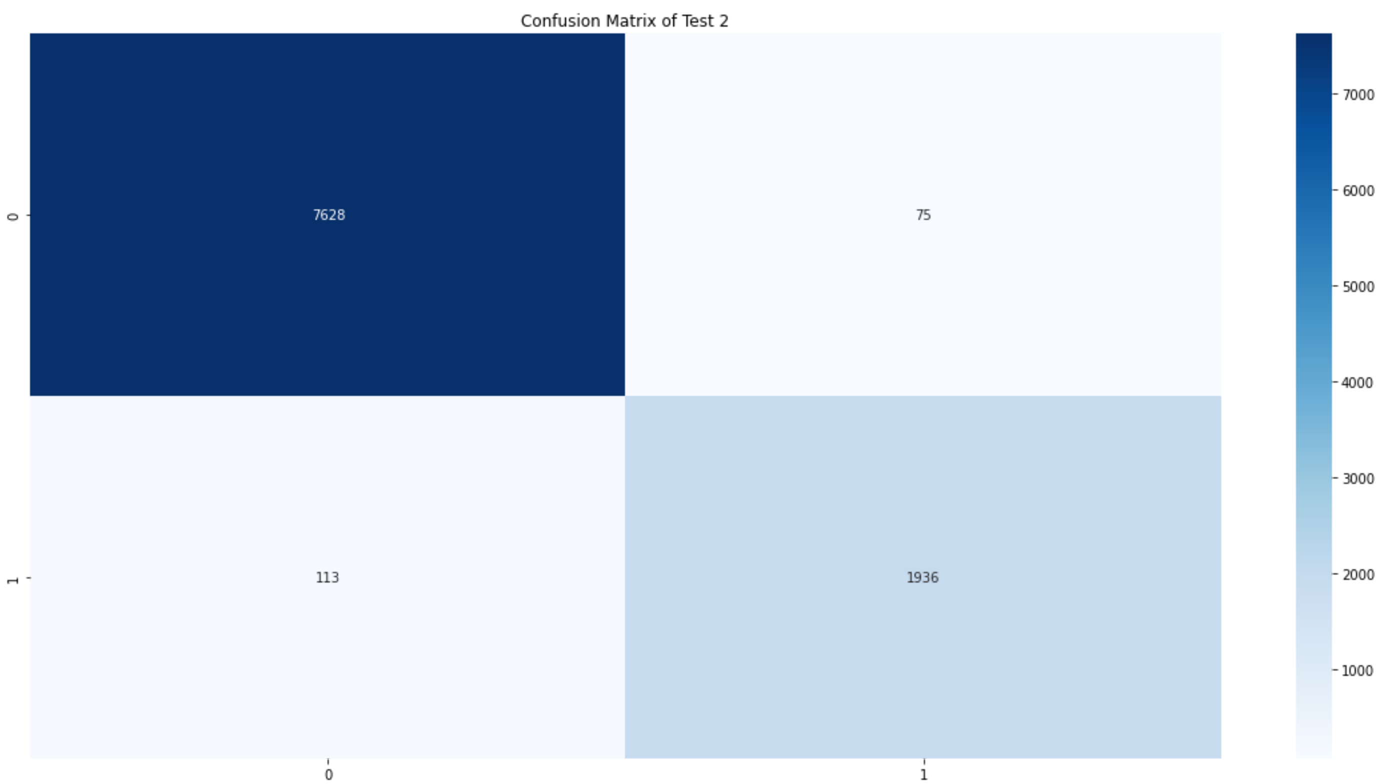
```
plt.figure(figsize=(20,10))
sns.heatmap(confusion_matrix(y_test, svc_y_pred), annot=True, cmap='Oranges', fmt="d")
plt.title('Confusion Matrix of Data')
plt.savefig('Confusion Matrix of Data Support Vector Machine.png')
plt.show()
```




```
plt.figure(figsize=(20,10))
sns.heatmap(confusion_matrix(y_test_1, svc_y_pred_1), annot=True, cmap='Reds', fmt="d")
plt.title('Confusion Matrix of Test 1')
plt.savefig('Confusion Matrix of Test 1 Support Vector Machine.png')
plt.show()
```

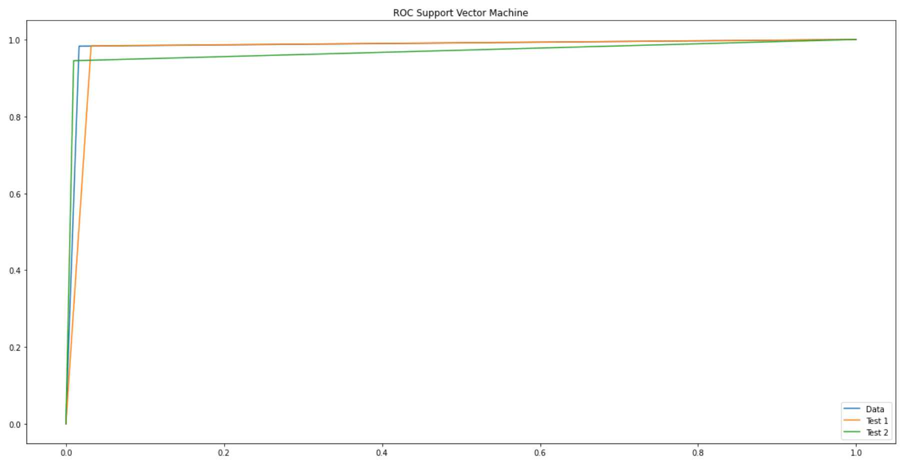



```
plt.figure(figsize=(20,10))
sns.heatmap(confusion_matrix(y_test_2, svc_y_pred_2), annot=True, cmap='Blues', fmt="d")
plt.title('Confusion Matrix of Test 2')
plt.savefig('Confusion Matrix of Test 2 Support Vector Machine.png')
plt.show()
```




```
plt.figure(figsize=(20,10))

y_pred_proba = LSVC_Model.predict(x_test)
fpr, tpr, _ = roc_curve(y_test,  y_pred_proba)
plt.plot(fpr, tpr, label="Data")

y_pred_proba_1 = LSVC_Model.predict(x_test_1)
```

```
fpr, tpr, _ = roc_curve(y_test_1,  y_pred_proba_1)
plt.plot(fpr, tpr, label="Test 1")

y_pred_proba_2 = LSVC_Model.predict(x_test_2)
fpr, tpr, _ = roc_curve(y_test_2,  y_pred_proba_2)
plt.plot(fpr, tpr, label="Test 2")

plt.title('ROC Support Vector Machine')
plt.legend(loc=4)
plt.savefig('ROC Support Vector Machine.png')
plt.show()
```



Naive Bayes Algorithm

```
GNB_Model = GaussianNB()
GNB_Model = GNB_Model.fit(x_train, y_train)
```

```
gnb_y_pred   = GNB_Model.predict(x_test)
gnb_y_pred_1 = GNB_Model.predict(x_test_1)
gnb_y_pred_2 = GNB_Model.predict(x_test_2)
```

```
print('---------------------------------------------------------------------------------------')
print("{:20}\t|\t{:20}\t|\t{:20}".format('Original', 'Test 1', 'Test 2'))
print('---------------------------------------------------------------------------------------')
print("{:>15}\t|\t{:>15}\t|\t{:>15}".format(
    accuracy_score(y_test,   gnb_y_pred)  *100,
    accuracy_score(y_test_1, gnb_y_pred_1)*100,
    accuracy_score(y_test_2, gnb_y_pred_2)*100)
)
print('---------------------------------------------------------------------------------------')
```

```
---------------------------------------------------------------------------------------
Original                |       Test 1                  |       Test 2
---------------------------------------------------------------------------------------
97.70773638968481       |       97.74859287054409       |       98.7284659557014
---------------------------------------------------------------------------------------
```

```
print(classification_report(y_test, gnb_y_pred))
```

```
                  precision    recall  f1-score   support
```

```
           0       1.00      0.97      0.99      1924
           1       0.91      0.99      0.95       519

    accuracy                           0.98      2443
   macro avg       0.95      0.98      0.97      2443
weighted avg       0.98      0.98      0.98      2443
```

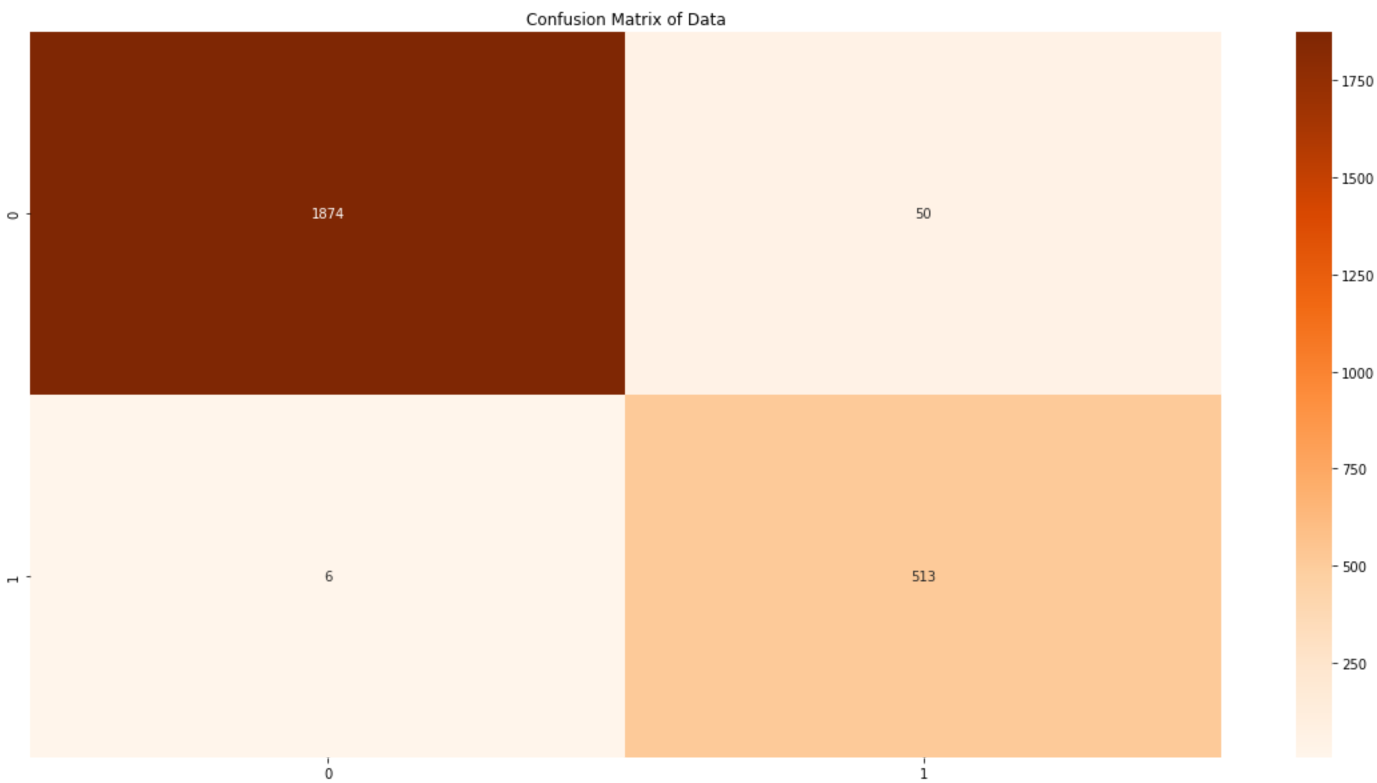
```
print(classification_report(y_test_1, gnb_y_pred_1))
```

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.98      1693
           1       0.95      0.99      0.97       972

    accuracy                           0.98      2665
   macro avg       0.97      0.98      0.98      2665
weighted avg       0.98      0.98      0.98      2665
```

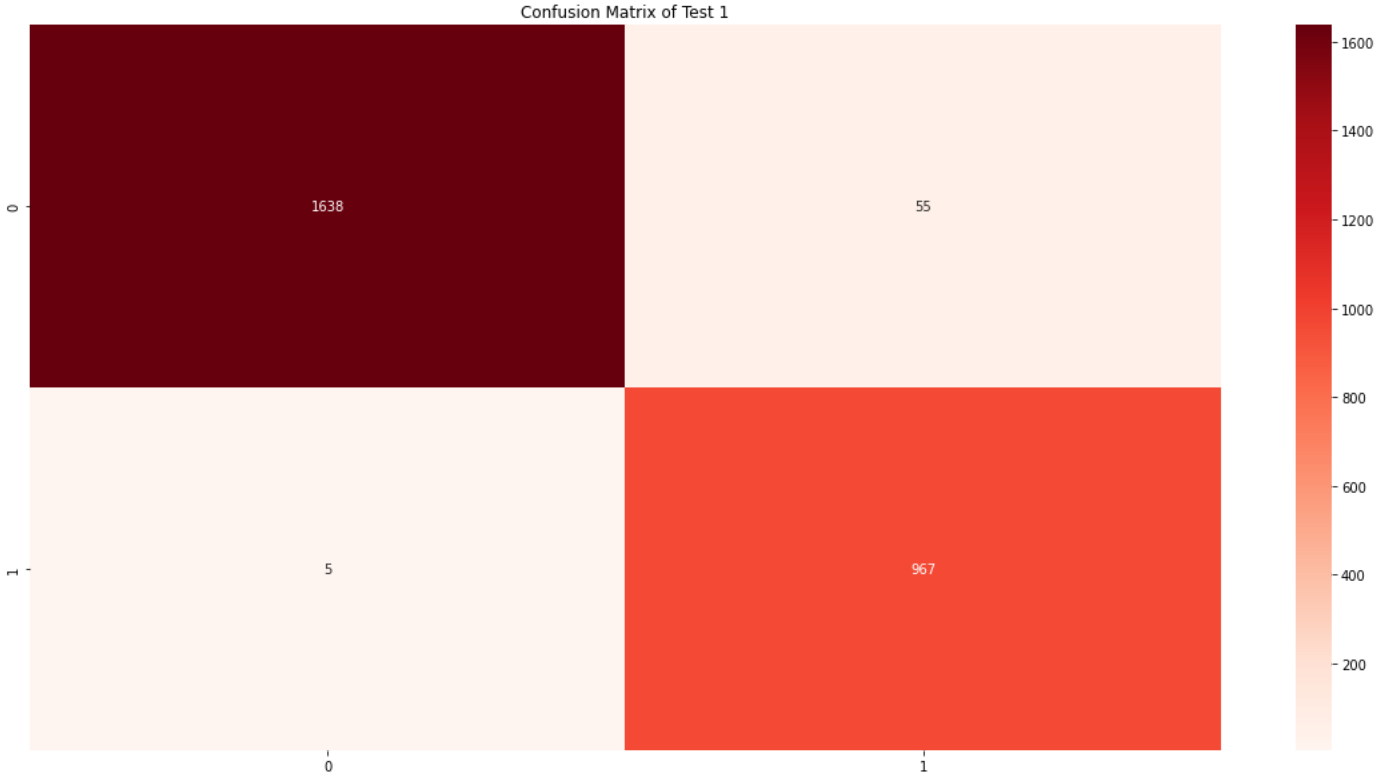
```
print(classification_report(y_test_2, gnb_y_pred_2))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      7703
           1       0.95      0.99      0.97      2049

    accuracy                           0.99      9752
   macro avg       0.97      0.99      0.98      9752
weighted avg       0.99      0.99      0.99      9752
```

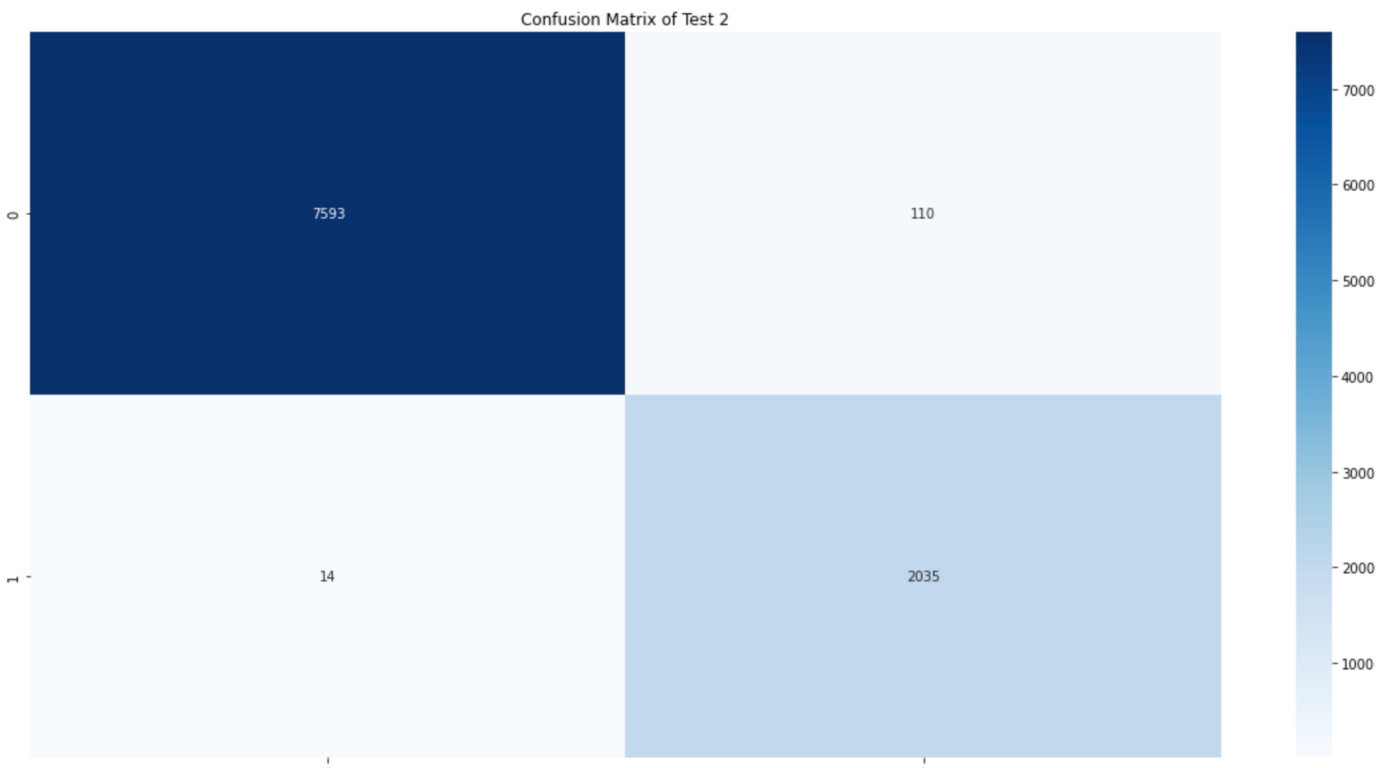
```
plt.figure(figsize=(20,10))
sns.heatmap(confusion_matrix(y_test, gnb_y_pred), annot=True, cmap='Oranges', fmt="d")
plt.title('Confusion Matrix of Data')
plt.savefig('Confusion Matrix of Data Naive Bayes.png')
plt.show()
```




```
plt.figure(figsize=(20,10))
sns.heatmap(confusion_matrix(y_test_1, gnb_y_pred_1), annot=True, cmap='Reds', fmt="d")
plt.title('Confusion Matrix of Test 1')
plt.savefig('Confusion Matrix of Test 1 Naive Bayes.png')
plt.show()
```




```
plt.figure(figsize=(20,10))
sns.heatmap(confusion_matrix(y_test_2, gnb_y_pred_2), annot=True, cmap='Blues', fmt="d")
plt.title('Confusion Matrix of Test 2')
plt.savefig('Confusion Matrix of Test 2 Naive Bayes.png')
plt.show()
```




```
plt.figure(figsize=(20,10))

y_pred_proba = GNB_Model.predict(x_test)
fpr, tpr, _ = roc_curve(y_test,  y_pred_proba)
plt.plot(fpr, tpr, label="Data")
```

```
y_pred_proba_1 = GNB_Model.predict(x_test_1)
fpr, tpr, _ = roc_curve(y_test_1,  y_pred_proba_1)
plt.plot(fpr, tpr, label="Test 1")

y_pred_proba_2 = GNB_Model.predict(x_test_2)
fpr, tpr, _ = roc_curve(y_test_2,  y_pred_proba_2)
plt.plot(fpr, tpr, label="Test 2")

plt.title('ROC Naive Bayes')
plt.legend(loc=4)
plt.savefig('ROC Naive Bayes.png')
plt.show()
```