LOGISTIC REGRESSION on Algerian Forest Fire Dataset

Submitted By Rajan Kumar

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings

warnings.filterwarnings("ignore")
```

In [2]:

```
df = pd.read_csv(r"C:\Users\Rajan\Downloads\Algerian_forest_fires_dataset_UPDATE.csv")
df.headad(10)
```

Out[2]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	not fire
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	not fire
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	not fire
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	not fire
5	6	6	2012	31	67	14	0.0	82.6	5.8	22.2	3.1	7.0	2.5	fire
6	7	6	2012	33	54	13	0.0	88.2	9.9	30.5	6.4	10.9	7.2	fire
7	8	6	2012	30	73	15	0.0	86.6	12.1	38.3	5.6	13.5	7.1	fire
8	9	6	2012	25	88	13	0.2	52.9	7.9	38.8	0.4	10.5	0.3	not fire
9	10	6	2012	28	79	12	0.0	73.2	9.5	46.3	1.3	12.6	0.9	not fire

In [3]:

```
df.shape
```

Out[3]:

```
(244, 14)
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   day           244 non-null    int64  
 1   month         244 non-null    int64  
 2   year          244 non-null    int64  
 3   Temperature   244 non-null    int64  
 4    RH           244 non-null    int64  
 5    Ws           244 non-null    int64  
 6   Rain          244 non-null    float64
 7   FFMC          244 non-null    float64
 8   DMC           244 non-null    float64
 9   DC            244 non-null    float64
 10  ISI           244 non-null    float64
 11  BUI           244 non-null    float64
 12  FWI           244 non-null    float64
 13  Classes       244 non-null    object 
dtypes: float64(7), int64(6), object(1)
memory usage: 26.8+ KB
```

In [6]:

```
#Adding new feature/column "Region"

df["Region"] = 0
for i in range(len(df)):
    if i >= 122:
        df["Region"][i] =1
```

In [7]:

```
df.head()
```

Out[7]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	not fire
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	not fire
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	not fire
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	not fire

In [8]:

```
df.columns
```

Out[8]:

```
Index(['day', 'month', 'year', 'Temperature', ' RH', ' Ws', 'Rain ', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes  ', 'Region'],
      dtype='object')
```

In [9]:

```
df.columns=[co.strip() for co in df.columns]
df.columns
```

Out[9]:

```
Index(['day', 'month', 'year', 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes', 'Region'],
      dtype='object')
```

In [13]:

```
df['Classes'].unique()
```

Out[13]:

```
array(['not fire   ', 'fire   ', 'fire', 'fire ', 'not fire', 'not fire ',
       'not fire     ', 'not fire    '], dtype=object)
```

In [15]:

```
df['Classes'] = df.Classes.str.strip()
```

In [16]:

```
df['Classes'].unique()
```

Out[16]:

```
array(['not fire', 'fire'], dtype=object)
```

In [17]:

```
df.isnull().sum()
```

Out[17]:

```
day            0
month          0
year           0
Temperature    0
RH             0
Ws             0
Rain           0
FFMC           0
DMC            0
DC             0
ISI            0
BUI            0
FWI            0
Classes        0
Region         0
dtype: int64
```

In [19]:

```
df['Classes'] = df['Classes'].map({'fire': 1, 'not fire': 0})

df.head(10)
```

Out[19]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	0
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	0
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	0
5	6	6	2012	31	67	14	0.0	82.6	5.8	22.2	3.1	7.0	2.5	1
6	7	6	2012	33	54	13	0.0	88.2	9.9	30.5	6.4	10.9	7.2	1
7	8	6	2012	30	73	15	0.0	86.6	12.1	38.3	5.6	13.5	7.1	1
8	9	6	2012	25	88	13	0.2	52.9	7.9	38.8	0.4	10.5	0.3	0
9	10	6	2012	28	79	12	0.0	73.2	9.5	46.3	1.3	12.6	0.9	0

In [20]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   day          244 non-null    int64  
 1   month        244 non-null    int64  
 2   year         244 non-null    int64  
 3   Temperature  244 non-null    int64  
 4   RH           244 non-null    int64  
 5   Ws           244 non-null    int64  
 6   Rain         244 non-null    float64
 7   FFMC         244 non-null    float64
 8   DMC          244 non-null    float64
 9   DC           244 non-null    float64
 10  ISI          244 non-null    float64
 11  BUI          244 non-null    float64
 12  FWI          244 non-null    float64
 13  Classes      244 non-null    int64  
 14  Region       244 non-null    int64  
dtypes: float64(7), int64(8)
memory usage: 28.7 KB
```

In [21]:

```
df['date'] = pd.to_datetime(df[['day','month','year']])
df.head()
```

Out[21]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	0
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	0
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	0

In [ ]:

```
df.drop(['day','month','year'],axis =1,inplace=True)
```

In [26]:

```
df.head()
```

Out[26]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region	date
0	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0	0	2012-06-01
1	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	0	0	2012-06-02
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0	0	2012-06-03
3	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	0	0	2012-06-04
4	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	0	0	2012-06-05

In [27]:

```
df.shape
```

Out[27]:

```
(244, 13)
```

In [28]:

```
df.columns
```

Out[28]:

```
Index(['Temperature', 'RH', 'Ws', 'Rain', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI',
       'FWI', 'Classes', 'Region', 'date'],
      dtype='object')
```

In [29]:

```
df.describe()
```

Out[29]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	
count	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	2
mean	32.172131	61.938525	15.504098	0.760656	77.887705	14.673361	49.288115	
std	3.633843	14.884200	2.810178	1.999406	14.337571	12.368039	47.619662	
min	22.000000	21.000000	6.000000	0.000000	28.600000	0.700000	6.900000	
25%	30.000000	52.000000	14.000000	0.000000	72.075000	5.800000	13.275000	
50%	32.000000	63.000000	15.000000	0.000000	83.500000	11.300000	33.100000	
75%	35.000000	73.250000	17.000000	0.500000	88.300000	20.750000	68.150000	
max	42.000000	90.000000	29.000000	16.800000	96.000000	65.900000	220.400000	

In [31]:

```
df.corr()
```

Out[31]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	
Temperature	1.000000	-0.654443	-0.278132	-0.326786	0.677491	0.483105	0.370498	0.6
RH	-0.654443	1.000000	0.236084	0.222968	-0.645658	-0.405133	-0.220330	-0.6
Ws	-0.278132	0.236084	1.000000	0.170169	-0.163255	-0.001246	0.076245	0.0
Rain	-0.326786	0.222968	0.170169	1.000000	-0.544045	-0.288548	-0.296804	-0.3
FFMC	0.677491	-0.645658	-0.163255	-0.544045	1.000000	0.602391	0.503910	0.7
DMC	0.483105	-0.405133	-0.001246	-0.288548	0.602391	1.000000	0.875358	0.6
DC	0.370498	-0.220330	0.076245	-0.296804	0.503910	0.875358	1.000000	0.5
ISI	0.605971	-0.688268	0.012245	-0.347862	0.740751	0.678355	0.503919	1.0
BUI	0.456415	-0.349685	0.030303	-0.299409	0.590251	0.982206	0.941672	0.6
FWI	0.566839	-0.580457	0.033957	-0.324755	0.691430	0.875191	0.737041	0.9
Classes	0.518119	-0.435023	-0.066529	-0.379449	0.770114	0.584188	0.507122	0.7
Region	0.273496	-0.406424	-0.176829	-0.041080	0.224680	0.191094	-0.081489	0.2

In [ ]:

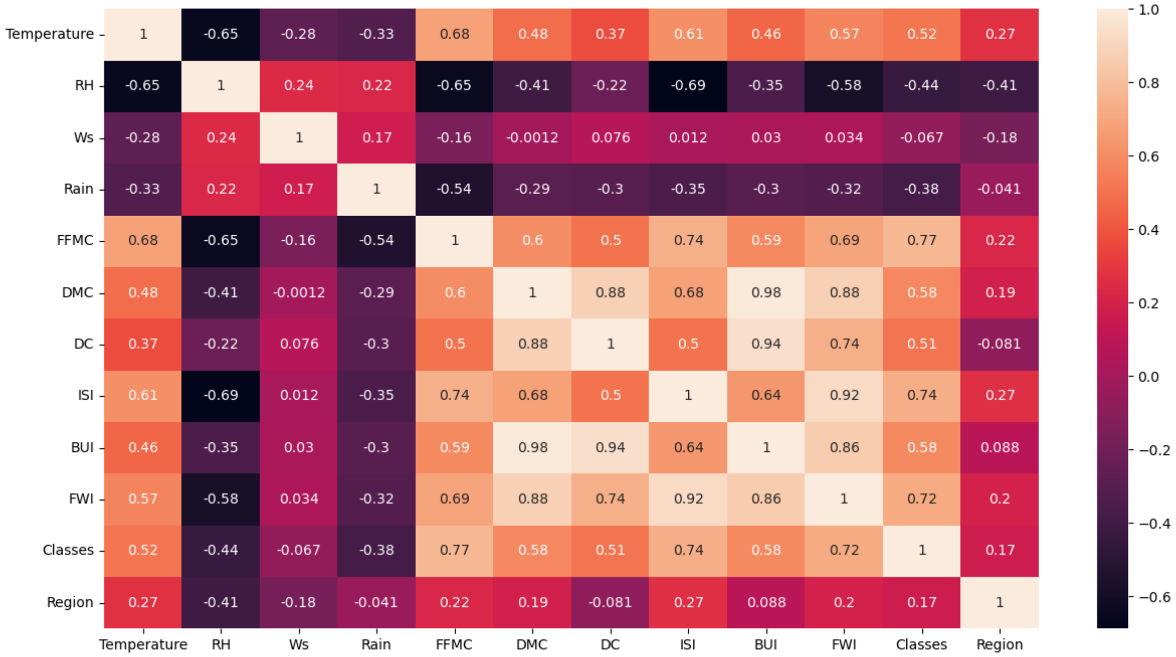
```

```

In [33]:

```
plt.figure(figsize=(15,8))
sns.heatmap(df.corr(),annot=True)
```

Out[33]:

```
<AxesSubplot:>
```



	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
Temperature	1	-0.65	-0.28	-0.33	0.68	0.48	0.37	0.61	0.46	0.57	0.52	0.27
RH	-0.65	1	0.24	0.22	-0.65	-0.41	-0.22	-0.69	-0.35	-0.58	-0.44	-0.41
Ws	-0.28	0.24	1	0.17	-0.16	-0.0012	0.076	0.012	0.03	0.034	-0.067	-0.18
Rain	-0.33	0.22	0.17	1	-0.54	-0.29	-0.3	-0.35	-0.3	-0.32	-0.38	-0.041
FFMC	0.68	-0.65	-0.16	-0.54	1	0.6	0.5	0.74	0.59	0.69	0.77	0.22
DMC	0.48	-0.41	-0.0012	-0.29	0.6	1	0.88	0.68	0.98	0.88	0.58	0.19
DC	0.37	-0.22	0.076	-0.3	0.5	0.88	1	0.5	0.94	0.74	0.51	-0.081
ISI	0.61	-0.69	0.012	-0.35	0.74	0.68	0.5	1	0.64	0.92	0.74	0.27
BUI	0.46	-0.35	0.03	-0.3	0.59	0.98	0.94	0.64	1	0.86	0.58	0.088
FWI	0.57	-0.58	0.034	-0.32	0.69	0.88	0.74	0.92	0.86	1	0.72	0.2
Classes	0.52	-0.44	-0.067	-0.38	0.77	0.58	0.51	0.74	0.58	0.72	1	0.17
Region	0.27	-0.41	-0.18	-0.041	0.22	0.19	-0.081	0.27	0.088	0.2	0.17	1

In [35]:

```
df.duplicated().sum()
```

Out[35]:

```
0
```

In [36]:

```
numeric_features = [feature for feature in df.columns if df[feature].dtype != 'O']
numeric_features
```

Out[36]:

```
['Temperature',
 'RH',
 'Ws',
 'Rain',
 'FFMC',
 'DMC',
 'DC',
 'ISI',
 'BUI',
 'FWI',
 'Classes',
 'Region',
 'date']
```

In [37]:

```
len(numeric_features)
#Here we have 13 Numeric features also termed here as Independent features
```

Out[37]:

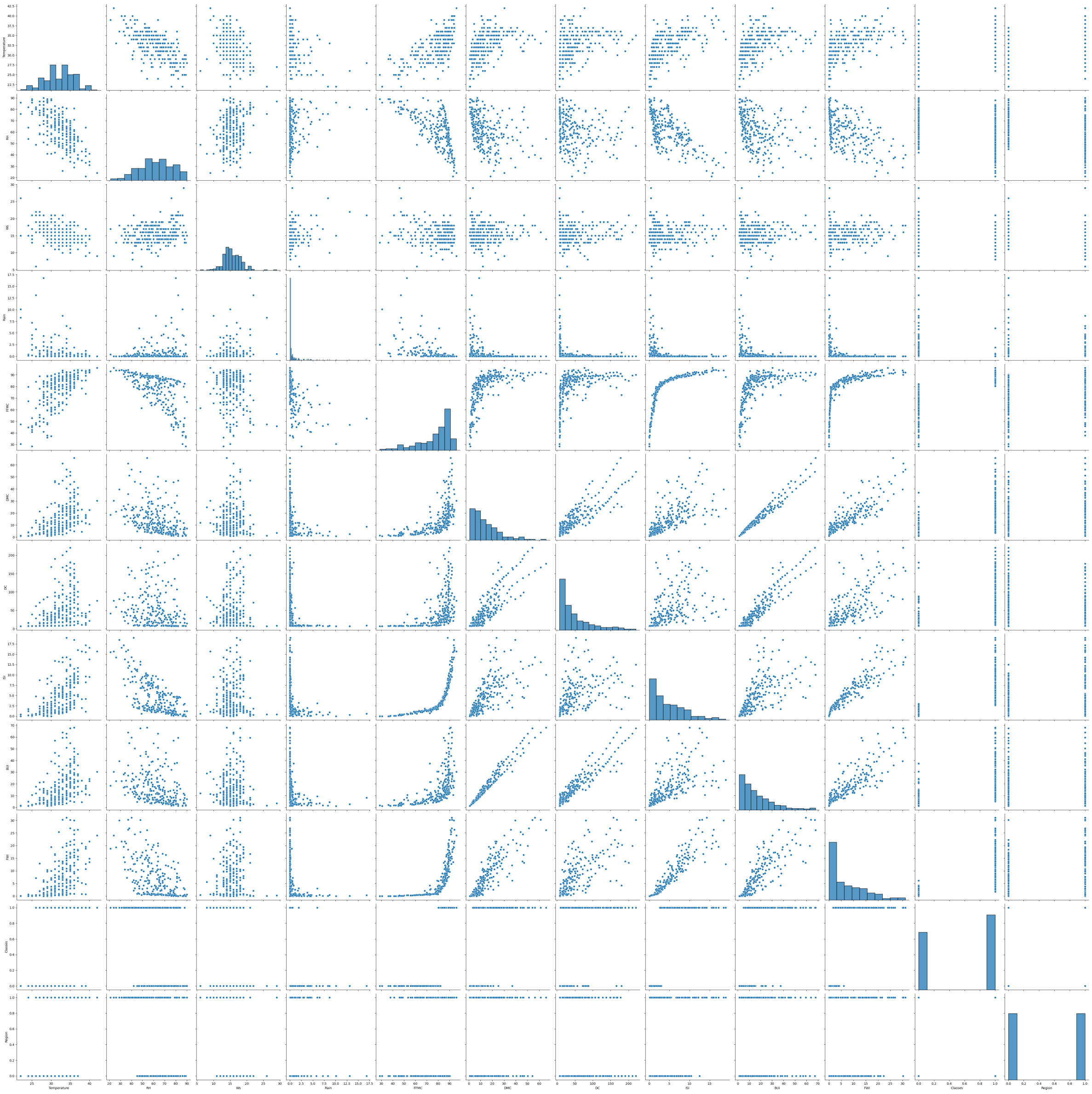
```
13
```

In [38]:

```
sns.pairplot(df,height=4)
```

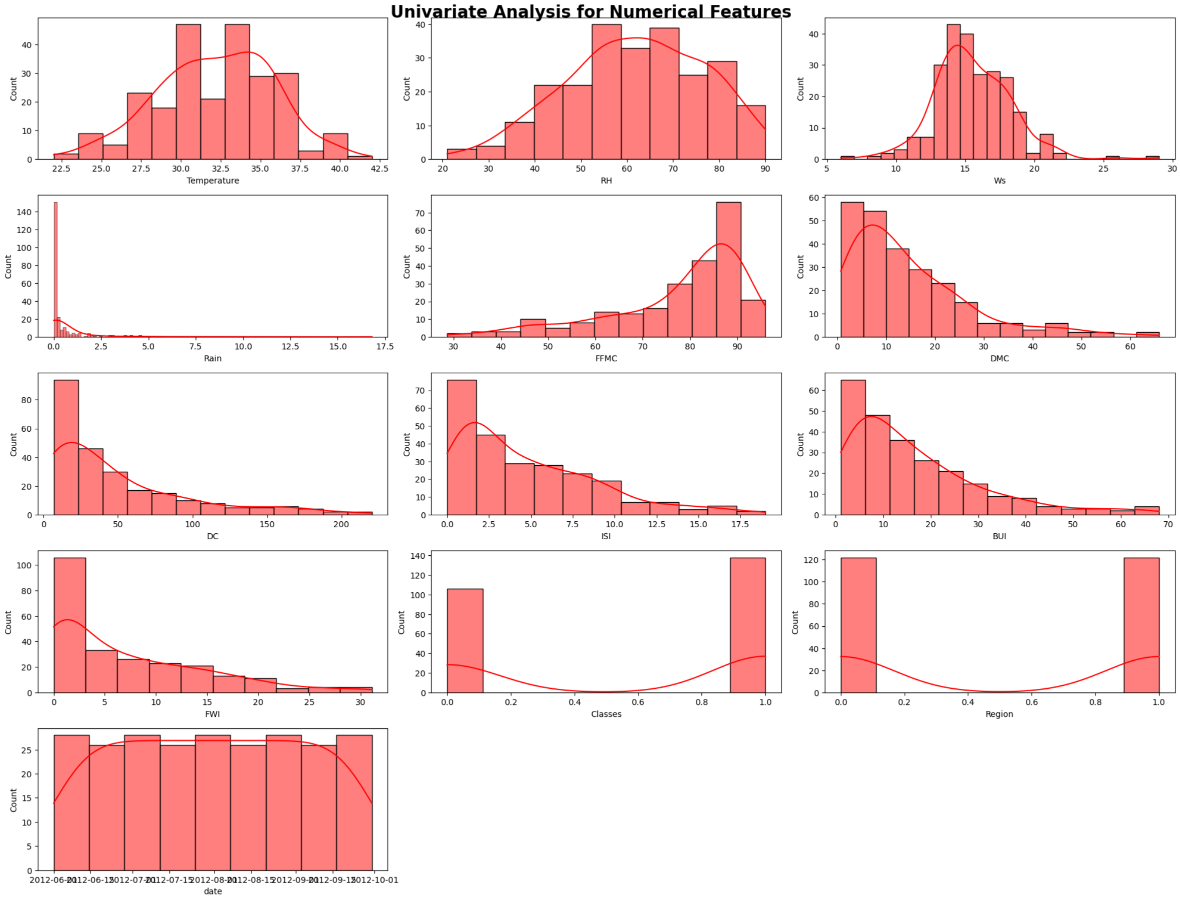
Out[38]:

```
<seaborn.axisgrid.PairGrid at 0x2524263e490>
```



In [44]:

```
plt.figure(figsize=(20,18))
plt.suptitle('Univariate Analysis for Numerical Features', fontsize=20,fontweight = 'bold')

for i in range(0, len(numeric_features)):
    plt.subplot(6,3,i+1)
    sns.histplot(x =df[numeric_features[i]],kde=True,color='r')
    plt.xlabel(numeric_features[i])
    plt.tight_layout()
```

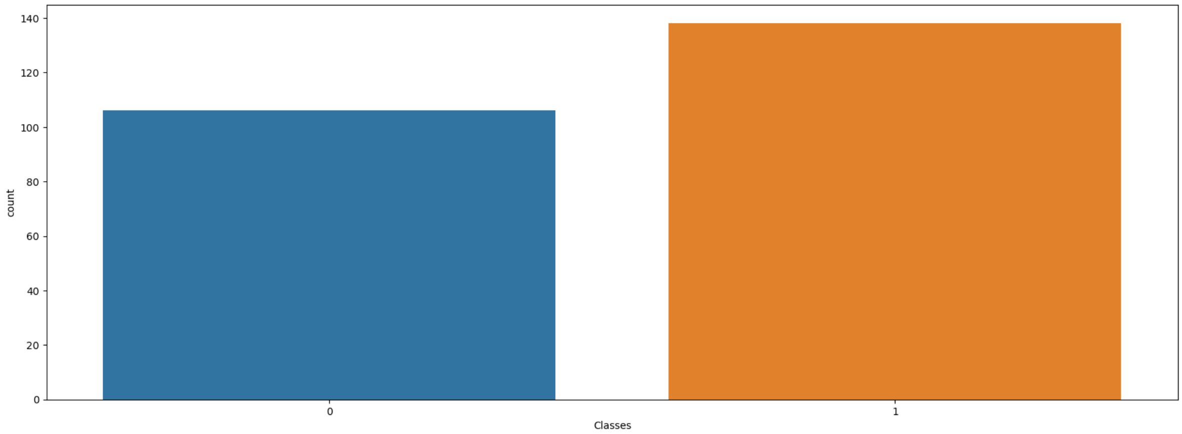


In [59]:

```
plt.figure(figsize=(20, 7))
sns.countplot(data=df, x='Classes')
```

Out[59]:

```
<AxesSubplot:xlabel='Classes', ylabel='count'>
```

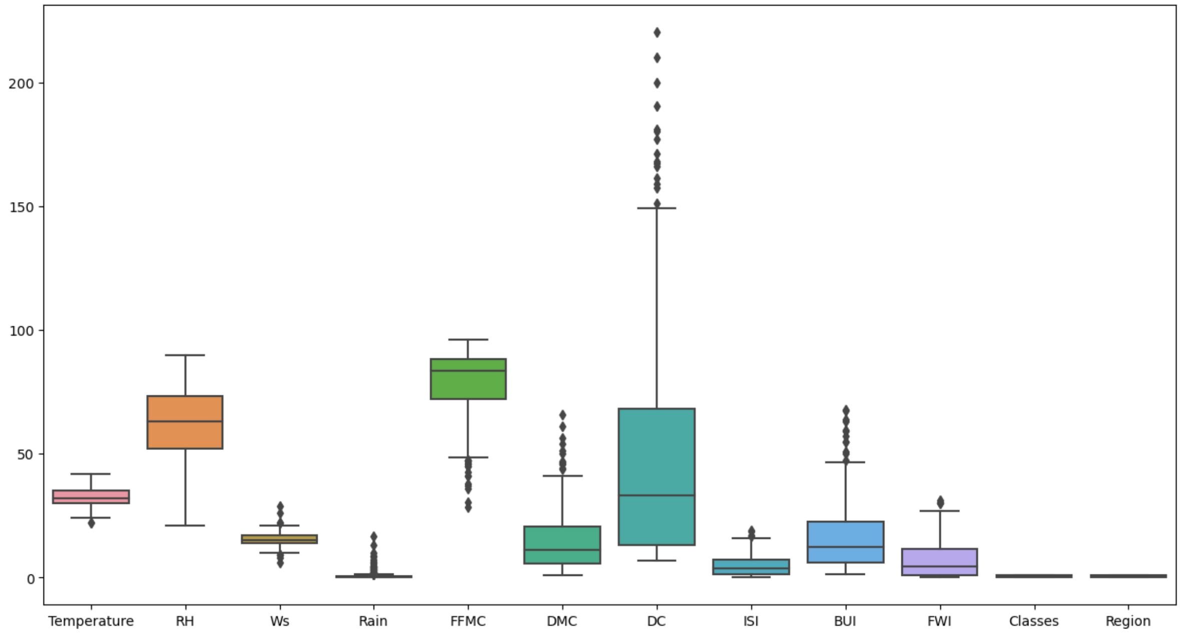


In [45]:

```
#Checking for Outliers
plt.figure(figsize=(15,8))
sns.boxplot(data=df)
```

Out[45]:

```
<AxesSubplot:>
```



In [52]:

```
#Unique data
for column in df.columns:
  print("-"*20)
  print("[" + column + "]")
  print(np.sort(df[column].unique()))
  print("-*"*10)
  print()
```

```
--------------------
[Temperature]
[22 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 42]
-*-*-*-*-*-*-*-*-*-*


--------------------
[RH]
[21 24 26 29 31 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
 76 77 78 79 80 81 82 83 84 86 87 88 89 90]
-*-*-*-*-*-*-*-*-*-*


--------------------
[Ws]
[ 6  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 26 29]
-*-*-*-*-*-*-*-*-*-*


--------------------
[Rain]
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3
```

In [60]:

```
X = df.iloc[:,[0,1,2,3,4,5,6,7,8,9]]
y = df['Classes']
```

In [61]:

```
X
```

Out[61]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI
0	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5
1	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1
3	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0
4	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5
...	...	...	...	...	...	...	...	...	...	...
239	30	65	14	0.0	85.4	16.0	44.5	4.5	16.9	6.5
240	28	87	15	4.4	41.1	6.5	8.0	0.1	6.2	0.0
241	27	87	29	0.5	45.9	3.5	7.9	0.4	3.4	0.2
242	24	54	18	0.1	79.7	4.3	15.2	1.7	5.1	0.7
243	24	64	15	0.2	67.3	3.8	16.5	1.2	4.8	0.5

244 rows × 10 columns

In [62]:

```
y
```

Out[62]:

```
0      0
1      0
2      0
3      0
4      0
      ..
239    1
240    0
241    0
242    0
243    0
Name: Classes, Length: 244, dtype: int64
```

In [65]:

```
from sklearn.model_selection import train_test_split

X_train,X_test, y_train, y_test = train_test_split(X,y,test_size=0.33, random_state=42)
```

In [67]:

```
X_train.shape
```

Out[67]:

```
(163, 10)
```

In [68]:

```
y_train.shape
```

Out[68]:

```
(163,)
```

In [69]:

```
X_test.shape
```

Out[69]:

```
(81, 10)
```

In [70]:

```
y_test.shape
```

Out[70]:

```
(81,)
```

In [71]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler
```

Out[71]:

```
StandardScaler()
```

In [72]:

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

In [73]:

```
X_train
```

Out[73]:

```
array([[ 0.00487747, -0.60257784, -1.68484146, ..., -0.80014076,
        -0.47763563, -0.8196431 ],
       [ 0.53489642,  0.14460201, -0.93856657, ...,  0.16132584,
        -0.3471914 , -0.08219052],
       [-0.260132  , -1.41768313,  2.04653297, ...,  2.13233237,
         0.09906517,  1.36540157],
       ...,
       [-1.85018883,  0.89178186,  0.5539832 , ..., -1.04050741,
        -1.01314351, -0.90158227],
       [ 0.26988695, -0.39880152,  0.18084575, ...,  0.52187581,
        -0.058841  ,  0.31384882],
       [-0.52514147,  0.9597073 ,  2.04653297, ..., -0.82417743,
        -0.9719506 , -0.87426921]])
```

In [74]:

```
X_test
```

Out[74]:

```
array([[-4.74644453e-01,  2.63611698e-01, -1.58418828e-01,
        -3.79941323e-01,  4.94929975e-01, -5.55412949e-02,
         3.16450964e-01,  1.16557214e-01,  1.04365998e-01,
         8.64078289e-02],
       [ 1.35612701e-01, -4.05747159e-01, -8.16466268e-01,
        -3.79941323e-01,  6.07329960e-01, -4.24018114e-01,
        -4.45183001e-01,  2.88214202e-01, -4.50217151e-01,
        -7.26611288e-02],
       [ 1.35612701e-01, -8.07362473e-01,  1.70604892e-01,
        -3.79941323e-01,  5.62369966e-01, -5.95402682e-01,
        -7.35656015e-01,  3.86303910e-01, -6.82542525e-01,
        -1.78707101e-01],
       [ 7.45869854e-01, -4.72683044e-01,  4.99628612e-01,
        -1.35693330e-02,  7.86337344e-03,  5.01458549e-01,
         2.26971044e+00, -6.19115592e-01,  1.02617312e+00,
        -2.44985833e-01],
       [ 2.27151274e+00, -1.27591367e+00, -1.80353743e+00,
        -3.06666925e-01,  8.92076589e-01,  6.64273887e-01,
```

In [75]:

```
X_scaled = scaler.fit_transform(X)
X_scaled
```

Out[75]:

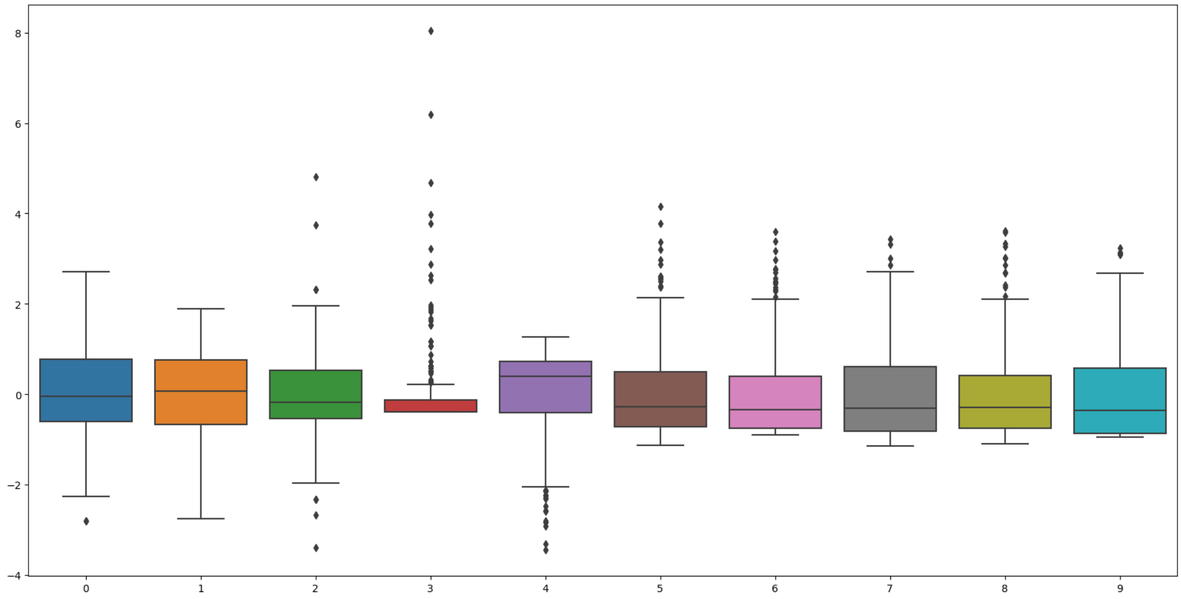
```
array([[-0.87473544, -0.33247844,  0.88999047, ..., -0.83447856,
        -0.93655635, -0.88345707],
       [-0.87473544, -0.0631847 , -0.89291326, ..., -0.90683562,
        -0.9012768 , -0.89694665],
       [-1.70200461,  1.35060746,  2.31631345, ..., -1.07566876,
        -0.98594772, -0.9374154 ],
       ...,
       [-1.42624822,  1.68722464,  4.81237868, ..., -1.05154974,
        -0.93655635, -0.92392582],
       [-2.25351739, -0.53444875,  0.88999047, ..., -0.73800248,
        -0.81660589, -0.85647791],
       [-2.25351739,  0.13878561, -0.17975177, ..., -0.85859758,
        -0.83777362, -0.88345707]])
```

In [76]:

```
plt.figure(figsize=(20, 10))
sns.boxplot(data=X_scaled)
```

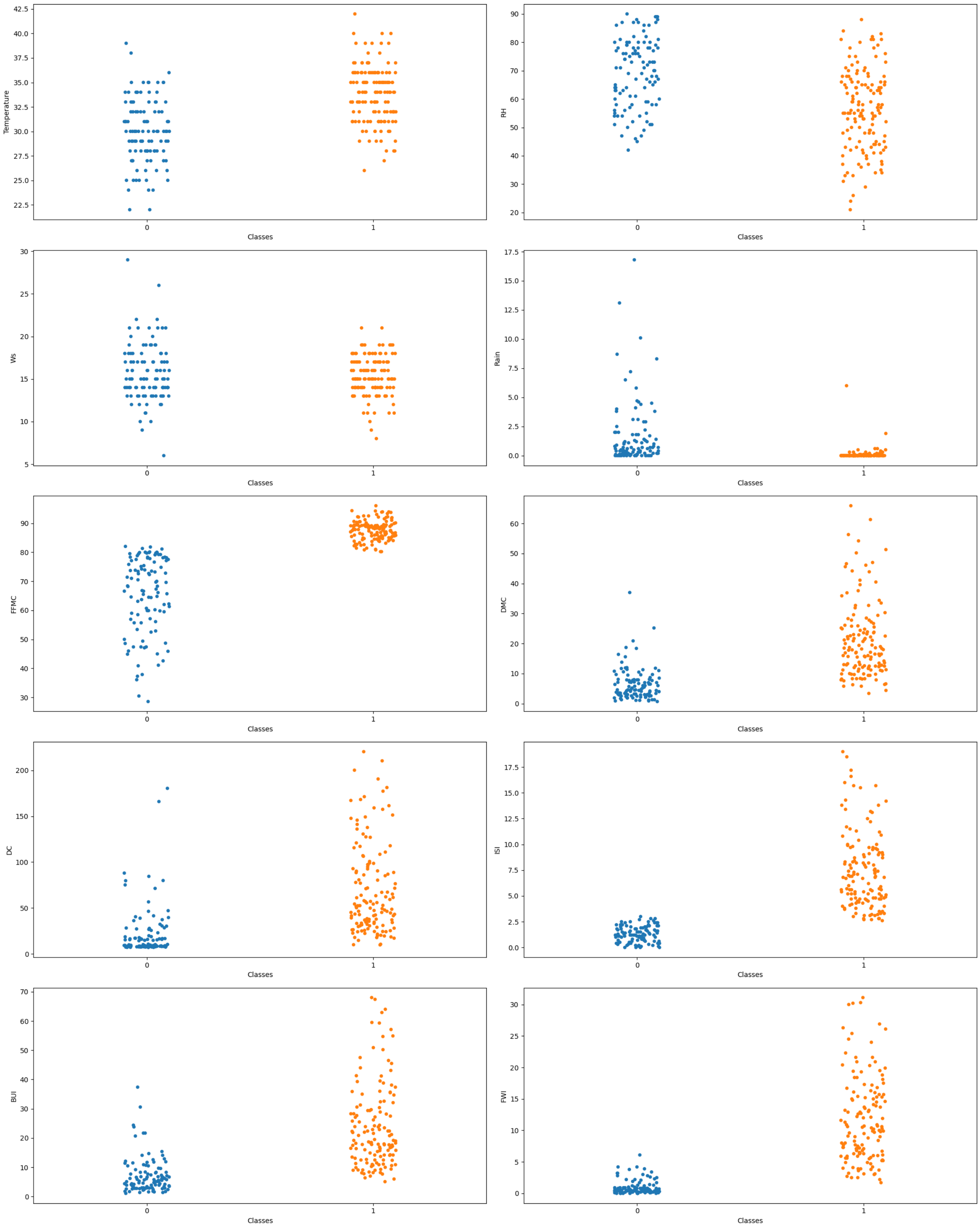
Out[76]:

```
<AxesSubplot:>
```



In [77]:

```
plt.figure(figsize=(20,25), facecolor='white')
plotnumber = 1

for column in X:
  ax = plt.subplot(5, 2, plotnumber)
  sns.stripplot(y=X[column], x=y)
  plotnumber += 1
plt.tight_layout()
```



In [78]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score
```

In [80]:

```
vif = pd.DataFrame()
vif["vif"] = [variance_inflation_factor(X_scaled,i) for i in range(X_scaled.shape[1])]
vif["Features"] = X.columns

#let's check the values
vif
```

Out[80]:

	vif	Features
0	2.319892	Temperature
1	2.763251	RH
2	1.281528	Ws
3	1.539725	Rain
4	4.109607	FFMC
5	76.208113	DMC
6	24.548675	DC
7	22.931108	ISI
8	170.409228	BUI
9	40.407207	FWI

In [83]:

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
```

In [84]:

```
log_reg.fit(X_train,y_train)
```

Out[84]:

```
LogisticRegression()
```

In [87]:

```
y_pred = log_reg.predict(X_test)
y_pred
```

Out[87]:

```
array([1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1], dtype=int64)
```

In [91]:

```
#Confusion Matrix
conf_mat = confusion_matrix(y_test, y_pred)
conf_mat
```

Out[91]:

```
array([[28,  0],
       [ 6, 47]], dtype=int64)
```

In [92]:

```
true_positive = conf_mat[0][0]
false_positive = conf_mat[0][1]
false_negative = conf_mat[1][0]
true_negative = conf_mat[1][1]
```

In [93]:

```
#Formulae for accuracy

Accuracy = (true_positive + true_negative) / (true_positive +false_positive + false_negativ
Accuracy
```

Out[93]:

```
0.9259259259259259
```

In [94]:

```
# Precison
Precision = true_positive/(true_positive+false_positive)
Precision
```

Out[94]:

```
1.0
```

In [95]:

```
# Recall
Recall = true_positive/(true_positive+false_negative)
Recall
```

Out[95]:

```
0.8235294117647058
```

In [96]:

```
# F1 Score
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
F1_Score
```

Out[96]:

```
0.9032258064516129
```

In [97]:

```
# Area Under Curve
areauc = roc_auc_score(y_test, y_pred)
areauc
```

Out[97]:

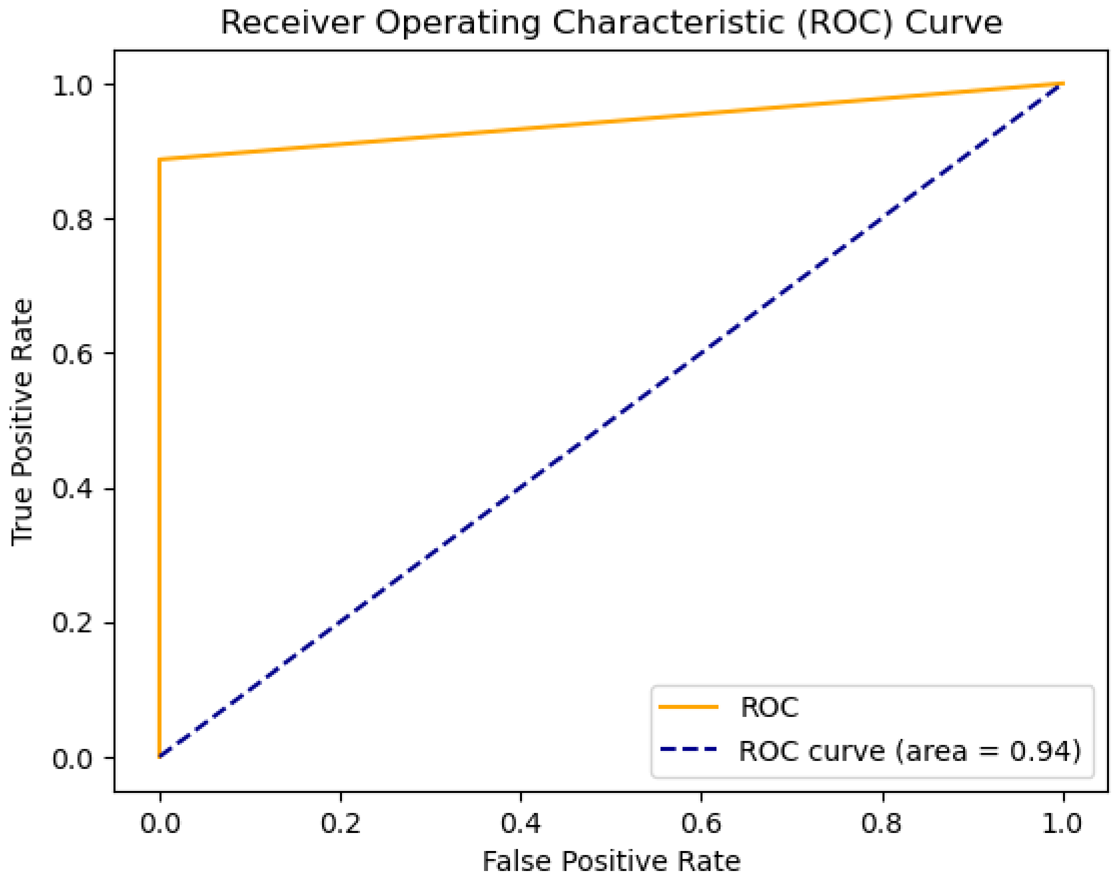
```
0.9433962264150944
```

In [98]:

```
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
```

In [100]:

```
plt.plot(fpr, tpr, color='orange', label='ROC')
plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--',label='ROC curve (area = %0.2f)'
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```



In [ ]:

```
```