# Logistics Regression

## Logistic Regression on Algerian forest dataset BY Divyani Pathak

### Life Cycle of machine learning Project

1. Understand the problem statement 2. Data collection 3. Data cleaning 4. EDA 5. Preprocessing 6.Feature Engineering 7. Choose Model(statistical/ML/DL/ensemble/NLP/Recommendation/Time series/Computer vision) 8. Hyper parameter tuning 9. Multiple algorithms 10. Deployment 11. Monitoring 12. Maintenance

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence. It says that if a new observation falls in a particular class, then what is the probability that it belongs to that class.

### Advantage of Logistics Regression

### Disadvantage of Logistics Regression

## 1. Problem Statement

```
1. Title: Algerian Forest Fires Dataset
2. Source: Dr. Abdellah Chaouch [chaouch-abdellah@yahoo.fr] and Prof. Abdelaziz Touzout [touzout@yahoo.com] Bejaia University, Algeria
3. Owner: Dr. Abdellah Chaouch [chaouch-abdellah@yahoo.fr] Bejaia University, Algeria
4. Date: November 2019
5. Number of Instances: 244 instances (two regions of Algeria, each with 122 instances)
6. Number of Attributes: 14 (13 attributes + 1 class attribute)
7. Attribute Information: 1. Date : (DD/MM/YYYY) Day, month and year 2. Temperature: temperature noon (temperature max) in Celsius degrees: 22 to 42 3. RH : Relative Humidity in %: 21 to 90 4. Ws :Wind speed in km/h: 6 to 29 5. Rain: total day rain in mm: 0 to 16 6. FFMC index from the FWI system: 28.6 to 92.5 7. DMC index from the FWI system: 1.1 to 65.9 8. DC index from the FWI system: 7 to 220.4 9. ISI index from the FWI system: 0 to 18.5 10. BUI index from the FWI system: 1.1 to 68 11. FWI index from the FWI system: 0 to 31.1 12. Classes: two classes, namely Fire and not Fire
```

## 2. Data Collection

The dataset is collected from various sources of Algerian forest fire monitoring. The dataset consists of 10 columns and 244 rows.

### 2.1 Importing Required Packages

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
%matplotlib inline
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import pickle,joblib,sklearn ,sys
from sklearn import *
```

In [3]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score,roc_curve,roc_auc_score,f1_score,precision_recall_curve,plot_roc_curve,plot_precision_recall_curve
```

### 2.2 Import the dataset

In [4]:
```
df=pd.read_csv("Algerian_forest_fires_dataset_UPDATED.csv")
```

In [5]:
```
df.shape
```

Out[5]:
```
(246, 14)
```

In [6]:
```
df.head(5)
```

Out[6]:

| | day | month | year | Temperature | RH | Ws | Rain | FFMC | DMC | DC | ISI | BUI | FWI | Classes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 01 | 06 | 2012 | 29 | 57 | 18 | 0 | 65.7 | 3.4 | 7.6 | 1.3 | 3.4 | 0.5 | not fire |
| 1 | 02 | 06 | 2012 | 29 | 61 | 13 | 1.3 | 64.4 | 4.1 | 7.6 | 1 | 3.9 | 0.4 | not fire |
| 2 | 03 | 06 | 2012 | 26 | 82 | 22 | 13.1 | 47.1 | 2.5 | 7.1 | 0.3 | 2.7 | 0.1 | not fire |
| 3 | 04 | 06 | 2012 | 25 | 89 | 13 | 2.5 | 28.6 | 1.3 | 6.9 | 0 | 1.7 | 0 | not fire |
| 4 | 05 | 06 | 2012 | 28 | 77 | 16 | 0 | 64.8 | 3 | 14.2 | 1.2 | 3.1 | 0.4 | not fire |

In [7]:
```
df.tail(5)
```

Out[7]:

| | day | month | year | Temperature | RH | Ws | Rain | FFMC | DMC | DC | ISI | BUI | FWI | Classes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 241 | 26 | 09 | 2012 | 30 | 65 | 14 | 0 | 85.4 | 16 | 44.5 | 4.5 | 16.9 | 6.5 | fire |
| 242 | 27 | 09 | 2012 | 28 | 87 | 15 | 4.4 | 41.1 | 6.5 | 8 | 0.1 | 6.2 | 0 | not fire |
| 243 | 28 | 09 | 2012 | 27 | 87 | 29 | 0.5 | 45.9 | 3.5 | 7.8 | 0.4 | 3.4 | 0.2 | not fire |
| 244 | 29 | 09 | 2012 | 24 | 54 | 18 | 0.1 | 79.7 | 1.3 | 16.3 | 1.7 | 5.1 | 0.7 | not fire |
| 245 | 30 | 09 | 2012 | 24 | 64 | 15 | 0.2 | 67.3 | 3.8 | 16.6 | 1.1 | 6.3 | 0.7 | not fire |

In [8]:
```
df.info()
```

Out[8]:
```
<class'pandas.core.frame.DataFrame'>
RangeIndex:246 entries,0 to 245
Data columns(total 14 columns):
 day          246 non-null  object
 month        246 non-null  object
 year         246 non-null  object
 Temperature  246 non-null  object
 RH           246 non-null  object
 Ws           246 non-null  object
 Rain         246 non-null  object
 FFMC         246 non-null  object
 DMC          246 non-null  object
 DC           246 non-null  object
 ISI          246 non-null  object
 BUI          246 non-null  object
 FWI          246 non-null  object
 Classes      245 non-null  object
dtype:object
memory usage:27.0+KB
```

## 3. Data Profiling and Data Cleaning

### 3.1 Remove unnecessay rows from datasets

In [9]:
```
df=df[:122]
```

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

Out[10]:
```
<class'pandas.core.frame.DataFrame'>
RangeIndex:122 entries,0 to 121
Data columns(total 14 columns):
 day          122 non-null  object
 month        122 non-null  object
 year         122 non-null  object
 Temperature  122 non-null  object
 RH           122 non-null  object
 Ws           122 non-null  object
 Rain         122 non-null  object
 FFMC         122 non-null  object
 DMC          122 non-null  object
 DC           122 non-null  object
 ISI          122 non-null  object
 BUI          122 non-null  object
 FWI          122 non-null  object
 Classes      122 non-null  object
dtype:object
memory usage:13.5+KB
```

### 3.2 Adding New Feature Named Region in a dataset

In [11]:
```
df['Region']=0
df['Region'].sample(5)
```

Out[11]:
```
Region
0    122
Name:count,dtype:int64
```

In [12]:
```
df.sample(5)
```

Out[12]:

| | day | month | year | Temperature | RH | Ws | Rain | FFMC | DMC | DC | ISI | BUI | FWI | Classes | Region |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 01 | 06 | 2012 | 29 | 57 | 18 | 0 | 65.7 | 3.4 | 7.6 | 1.3 | 3.4 | 0.5 | not fire | 0 |
| 1 | 02 | 06 | 2012 | 29 | 61 | 13 | 1.3 | 64.4 | 4.1 | 7.6 | 1 | 3.9 | 0.4 | not fire | 0 |
| 2 | 03 | 06 | 2012 | 26 | 82 | 22 | 13.1 | 47.1 | 2.5 | 7.1 | 0.3 | 2.7 | 0.1 | not fire | 0 |
| 3 | 04 | 06 | 2012 | 25 | 89 | 13 | 2.5 | 28.6 | 1.3 | 6.9 | 0 | 1.7 | 0 | not fire | 0 |
| 4 | 05 | 06 | 2012 | 28 | 77 | 16 | 0 | 64.8 | 3 | 14.2 | 1.2 | 3.1 | 0.4 | not fire | 0 |

### 3.3 check the columns

In [13]:
```
df.columns
```

Out[13]:
```
Index(['day','month','year','Temperature','RH','Ws','Rain','FFMC',
'DMC','DC','ISI','BUI','FWI','Classes','Region'],
dtype='object')
```

In [14]:
```
df[df['Region']==1].shape,df[df['Region']==0].shape
```

Out[14]:
```
((0,15),(122,15))
```

### 3.4 check the Categorical (feature data)

In [15]:
```
categorical=['month','day','year','Classes','Region']
numerical=['Temperature','RH','Ws','Rain','FFMC',
'DMC','DC','ISI','BUI','FWI']
```

In [16]:
```
print('We have {} categorical features : {}\n'.format(len(categorical),categorical))
print('We have {} numerical features : {}'.format(len(numerical),numerical))
```

Out[16]:
```
We have 5 categorical features : ['month','day','year','Classes','Region']

We have 10 numerical features : ['Temperature','RH','Ws','Rain','FFMC',
'DMC','DC','ISI','BUI','FWI']
```

In [17]:
```
df[categorical].head(3)
```

Out[17]:

| | month | day | year | Classes | Region |
|---|---|---|---|---|---|
| 0 | 06 | 01 | 2012 | not fire | 0 |
| 1 | 06 | 02 | 2012 | not fire | 0 |
| 2 | 06 | 03 | 2012 | not fire | 0 |

In [18]:
```
df[numerical].head(3)
```

Out[18]:

| | Temperature | RH | Ws | Rain | FFMC | DMC | DC | ISI | BUI | FWI |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 29 | 57 | 18 | 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 | 3.9 | 0.4 |
| 2 | 26 | 82 | 22 | 13.1 | 47.1 | 2.5 | 7.1 | 0.3 | 2.7 | 0.1 |

### 3.5 Changing the categorical features to 0 and 1

In [19]:
```
df['Classes']=df['Classes'].replace({'not fire':0,'fire':1})
df['month']=df['month'].replace({'06':0,'07':1,'08':2,'09':3})
df['Region']=df['Region'].replace({0:'Bejaia',1:'Sidi-Bel Abbes'})
```

In [20]:
```
df[categorical].head(3)
```

Out[20]:

| | month | day | year | Classes | Region |
|---|---|---|---|---|---|
| 0 | 0 | 01 | 2012 | 0 | Bejaia |
| 1 | 0 | 02 | 2012 | 0 | Bejaia |
| 2 | 0 | 03 | 2012 | 0 | Bejaia |

### 3.6 checking the datatype

In [21]:
```
df[numerical].dtypes
```

Out[21]:
```
Temperature  object
RH           object
Ws           object
Rain         object
FFMC         object
DMC          object
DC           object
ISI          object
BUI          object
FWI          object
dtype:object
```

In [22]:
```
for i in numerical:
print('{} : {} \n'.format(df[i].sample(5),df[i].dtypes))
```

Out[22]:
```
Temperature  object
RH           object
Ws           object
Rain         object
FFMC         object
DMC          object
DC           object
ISI          object
BUI          object
FWI          object
dtype:object
```

In [23]:
```
df[numerical].dtypes
```

Out[23]:
```
Temperature  object
RH           object
Ws           object
Rain         object
FFMC         object
DMC          object
DC           object
ISI          object
BUI          object
FWI          object
dtype:object
```

In [24]:
```
df[numerical]=df[numerical].astype(float)
```

In [25]:
```
df[numerical].dtypes
```

Out[25]:
```
Temperature  float64
RH           float64
Ws           float64
Rain         float64
FFMC         float64
DMC          float64
DC           float64
ISI          float64
BUI          float64
FWI          float64
dtype:object
```

### 3.7 checking NAN values in dataset

In [26]:
```
df[numerical].isnull()
```

Out[26]:

| | Temperature | RH | Ws | Rain | FFMC | DMC | DC | ISI | BUI | FWI |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | False | False | False | False | False | False | False | False | False | False |
| 1 | False | False | False | False | False | False | False | False | False | False |
| 2 | False | False | False | False | False | False | False | False | False | False |
| 3 | False | False | False | False | False | False | False | False | False | False |
| 4 | False | False | False | False | False | False | False | False | False | False |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 117 | False | False | False | False | False | False | False | False | False | False |
| 118 | False | False | False | False | False | False | False | False | False | False |
| 119 | False | False | False | False | False | False | False | False | False | False |
| 120 | False | False | False | False | False | False | False | False | False | False |
| 121 | False | False | False | False | False | False | False | False | False | False |

122 rows × 10 columns

In [27]:
```
df[numerical].isnull().sum
```

Out[27]:
```
Temperature  False
RH           False
Ws           False
Rain         False
FFMC         False
DMC          False
DC           False
ISI          False
BUI          False
FWI          False
dtype:object
```

### 3.8 Balance or Imbalance dataset

In [28]:
```
df['Classes'].value_counts
```

Out[28]:
```
1    70
0    52
Name:Classes,dtype:int64
```

Observation: Classes attribute is quasi balanced data so it is slightly balanced data

## 4. Statistical Analysis

### 4.1 Numerical and categorical features:

In [29]:
```
print('Numerical features:\n\n{}\n\n Categorical features:\n\n{}\n\n',
df[numerical].sample(5),df[categorical].sample(5))
```

Out[29]:
```
Numerical features:

     Temperature  RH  Ws  Rain  FFMC  DMC    DC  ISI  BUI  FWI
0           29.0  57  18   0.0  65.7  3.4   7.6  1.3  3.4  0.5
1           29.0  61  13   1.3  64.4  4.1   7.6  1.0  3.9  0.4
2           26.0  82  22  13.1  47.1  2.5   7.1  0.3  2.7  0.1
3           25.0  89  13   2.5  28.6  1.3   6.9  0.0  1.7  0.0
4           28.0  77  16   0.0  64.8  3.0  14.2  1.2  3.1  0.4
..           ...  ..  ..   ...   ...  ...   ...  ...  ...  ...
117         30.0  65  14   0.0  85.4 16.0  44.5  4.5 16.9  6.5
118         28.0  87  15   4.4  41.1  6.5   8.0  0.1  6.2  0.0
119         27.0  87  29   0.5  45.9  3.5   7.8  0.4  3.4  0.2
120         24.0  54  18   0.1  79.7  1.3  16.3  1.7  5.1  0.7
121         24.0  64  15   0.2  67.3  3.8  16.6  1.1  6.3  0.7

[122 rows × 10 columns]

 Categorical features:

     month  day  year  Classes  Region
0        0   01  2012        0  Bejaia
1        0   02  2012        0  Bejaia
2        0   03  2012        0  Bejaia
3        0   04  2012        0  Bejaia
4        0   05  2012        0  Bejaia
..     ...  ...   ...      ...     ...
117      3   26  2012        1  Bejaia
118      3   27  2012        0  Bejaia
119      3   28  2012        0  Bejaia
120      3   29  2012        0  Bejaia
121      3   30  2012        0  Bejaia

[122 rows × 5 columns]
```

### 4.2 Uni-variant Analysis

In order to do analysis of one variable (uni) - we have the purpose to understand the distribution of values for a single variable

In [30]:
```
df.describe() # describe function of pandas dataframe
```

Out[30]:

| | Temperature | RH | Ws | Rain | FFMC | DMC | DC | ISI | BUI | FWI |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 |
| mean | 32.172131 | 62.041803 | 15.826230 | 0.762295 | 77.887705 | 14.673770 | 49.399180 | 4.774590 | 16.673770 | 7.035246 |
| std | 3.844075 | 14.844992 | 2.980114 | 2.003207 | 14.337571 | 12.368041 | 47.619392 | 4.175318 | 14.884556 | 7.476070 |
| min | 22.000000 | 21.000000 | 6.000000 | 0.000000 | 28.600000 | 1.100000 | 7.000000 | 0.000000 | 1.100000 | 0.000000 |
| 25% | 30.250000 | 52.750000 | 14.000000 | 0.000000 | 72.075000 | 3.075000 | 12.375000 | 1.400000 | 3.675000 | 0.875000 |
| 50% | 32.000000 | 63.500000 | 15.500000 | 0.300000 | 80.300000 | 9.300000 | 33.100000 | 3.900000 | 11.350000 | 4.450000 |
| 75% | 35.500000 | 73.250000 | 17.000000 | 1.500000 | 88.425000 | 19.750000 | 68.225000 | 7.250000 | 23.750000 | 11.675000 |
| max | 42.000000 | 90.000000 | 29.000000 | 16.800000 | 96.000000 | 65.900000 | 220.400000 | 19.000000 | 68.000000 | 31.100000 |

In [31]:
```
df[numerical].describe() # describe function of pandas dataframe
```

Out[31]:

| | Temperature | RH | Ws | Rain | FFMC | DMC | DC | ISI | BUI | FWI |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 | 122.000000 |
| mean | 32.172131 | 62.041803 | 15.826230 | 0.762295 | 77.887705 | 14.673770 | 49.399180 | 4.774590 | 16.673770 | 7.035246 |
| std | 3.844075 | 14.844992 | 2.980114 | 2.003207 | 14.337571 | 12.368041 | 47.619392 | 4.175318 | 14.884556 | 7.476070 |
| min | 22.000000 | 21.000000 | 6.000000 | 0.000000 | 28.600000 | 1.100000 | 7.000000 | 0.000000 | 1.100000 | 0.000000 |
| 25% | 30.250000 | 52.750000 | 14.000000 | 0.000000 | 72.075000 | 3.075000 | 12.375000 | 1.400000 | 3.675000 | 0.875000 |
| 50% | 32.000000 | 63.500000 | 15.500000 | 0.300000 | 80.300000 | 9.300000 | 33.100000 | 3.900000 | 11.350000 | 4.450000 |
| 75% | 35.500000 | 73.250000 | 17.000000 | 1.500000 | 88.425000 | 19.750000 | 68.225000 | 7.250000 | 23.750000 | 11.675000 |
| max | 42.000000 | 90.000000 | 29.000000 | 16.800000 | 96.000000 | 65.900000 | 220.400000 | 19.000000 | 68.000000 | 31.100000 |

### 4. Numerical features analysis (uni-variant analysis) - KDE Plot and Scatter

In [32]:
```
numerical=['Temperature',
'RH',
'Ws',
'Rain',
'FFMC',
'DMC',
'DC',
'ISI',
'BUI',
'FWI']
```

KDE Plot

In [33]:
```
%matplotlib inline
for i in numerical:
sns.kdeplot(df[i],shade=True,color='r',legend=False,).set_xlabel('').set_ylabel('')
plt.suptitle('Uni-variant analysis of Numerical Features',
fontsize=20,fontweight='bold',
alpha=0.8,y=1.0)
plt.tight_legend(
labels=numerical,
loc="upper right",
bbox_to_anchor=(1, 1.5),
ncol=1,
fontsize=20,
)
```

Out[33]:

Scatter

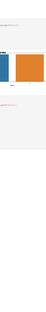
In [34]:
```
%matplotlib inline
for i in numerical:
sns.scatterplot(df[i],df['Classes'],hue=df['Classes'],legend=False,).set_xlabel('').set_ylabel('')
plt.suptitle('scatter plot with each Numerical Features to analyse feature',
fontsize=20,fontweight='bold',
alpha=0.8,y=1.0)
plt.tight_legend(
labels=['Temperature','RH','Ws','Rain','FFMC','DMC','DC','ISI','BUI','FWI'],
loc="upper right",
bbox_to_anchor=(1, 1.5),
ncol=1,
fontsize=20,
)
```

Out[34]:

Count Plot for binary category features

In [35]:
```
%matplotlib inline
for i in categorical[:-2]:
sns.countplot(df[i],hue=df['Classes'],).set_xlabel('').set_ylabel('')
plt.suptitle('Uni-variant analysis of categorical Features',
fontsize=20,fontweight='bold',
alpha=0.8,y=1.0)
plt.tight_legend(
labels=['not fire','fire'],
loc="upper right",
bbox_to_anchor=(1, 1.5),
ncol=1,
fontsize=20,
)
```

Out[35]:




Box Plot <-> to find outlier

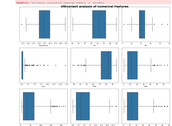
In [36]:
```
%matplotlib inline
for i in numerical:
sns.boxplot(df[i],df['Classes'],orient='h',palette='Set1',).set_xlabel('').set_ylabel('')
plt.suptitle('Bi-variant analysis of Numerical Features with respect to Target Classes',
fontsize=20,fontweight='bold',
alpha=0.8,y=1.0)
plt.tight_legend(
labels=['Temperature','RH','Ws','Rain','FFMC','DMC','DC','ISI','BUI','FWI'],
loc="upper right",
bbox_to_anchor=(1, 1.5),
ncol=1,
fontsize=20,
)
```

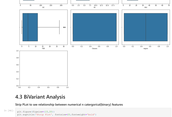
Out[36]:

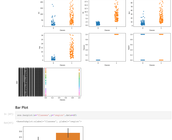
```
Supplement:
Correlation matrix of numeric variables (excluding factors and logical vars):
                temperature  co  no  nox  nhmc  ch4  co2  so2  n2o  ozone  ugm3
temperature           1.000 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00  0.00
co                    0.000 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00  0.00
no                    0.000 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00  0.00
nox                   0.000 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00  0.00  0.00
nhmc                  0.000 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00  0.00  0.00
ch4                   0.000 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00  0.00  0.00
co2                   0.000 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00  0.00  0.00
so2                   0.000 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00  0.00  0.00
n2o                   0.000 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00  0.00  0.00
ozone                 0.000 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  1.00  0.00
ugm3                  0.000 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00  1.00
Warning message:
In cor(.data[, .numeric_cols, drop = FALSE], use = "complete.obs", method = method) :
  the standard deviation is zero
```

```
Supplement:
Correlation matrix of numeric variables (excluding factors and logical vars):
                co  no  nox  nhmc  ch4  co2  so2  n2o  ozone  ugm3
co           1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00
no           0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00
nox          0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00
nhmc         0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00  0.00
ch4          0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00  0.00
co2          0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00  0.00
so2          0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00  0.00
n2o          0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00  0.00
ozone        0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00  0.00
ugm3         0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  1.00
Warning message:
In cor(.data[, .numeric_cols, drop = FALSE], use = "complete.obs", method = method) :
  the standard deviation is zero
```

```
Supplement: Correlation matrix of numeric variables:
```





```
# Remove the outliers
df_clean = df[(df['Temperature'] >= lower_bound_temp) & (df['Temperature'] <= upper_bound_temp)]
print("Shape after removing outliers:", df_clean.shape)
```

```
Shape after removing outliers: (1000, 7)
```

```
# Plot the cleaned data
plt.figure(figure=(10, 6))
sns.boxplot(x=df_clean['Type'], y=df_clean['Temperature'])
plt.title('Temperature by Type (After Outlier Removal)')
plt.show()
```

```
```

```
group_kfold = GroupKFold(n_splits=5)
for train_index, test_index in group_kfold.split(X, y, groups=groups):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
```

```
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 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 85 86 87 88 89 90 91 92 93 94 95
 96 97 98 99] TEST: [ 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153
 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189
 190 191 192 193 194 195 196 197 198 199]
TRAIN: [  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 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 85 86 87 88 89 90 91 92 93 94 95
 96 97 98 99] TEST: [ 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217
 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235
 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253
 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271
 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289
 290 291 292 293 294 295 296 297 298 299]
TRAIN: [  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 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 85 86 87 88 89 90 91 92 93 94 95
 96 97 98 99] TEST: [ 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317
 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335
 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353
 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371
 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389
 390 391 392 393 394 395 396 397 398 399]
TRAIN: [  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 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 85 86 87 88 89 90 91 92 93 94 95
 96 97 98 99] TEST: [ 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417
 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435
 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453
 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471
 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489
 490 491 492 493 494 495 496 497 498 499]
TRAIN: [  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 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 85 86 87 88 89 90 91 92 93 94 95
 96 97 98 99] TEST: [ 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517
 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535
 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553
 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571
 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589
 590 591 592 593 594 595 596 597 598 599]
```

## A. Logistics Regression with 90% accuracy

### A1 Model Training

```
from sklearn.linear_model import LogisticRegression
```

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

```
LogisticRegression
LogisticRegression()
```

```
cm = confusion_matrix(y_test, log_reg_pred)
print("Confusion Matrix:")
print(cm)
```

```
Confusion Matrix:
[[ 0 20]
 [ 0 80]]
```

### A2 Prediction

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

```
log_reg_pred
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       
```

[0]: 0
[1]: 1
[2]: 2
[3]: 3
[4]: 4
[5]: 5
[6]: 6
[7]: 7
[8]: 8
[9]: 9
[10]: 10
[11]: 11
[12]: 12
[13]: 13
[14]: 14
[15]: 15
[16]: 16
[17]: 17
[18]: 18
[19]: 19
[20]: 20
[21]: 21
[22]: 22
[23]: 23
[24]: 24
[25]: 25
[26]: 26
[27]: 27
[28]: 28
[29]: 29
[30]: 30
[31]: 31
[32]: 32
[33]: 33
[34]: 34
[35]: 35
[36]: 36
[37]: 37
[38]: 38
[39]: 39
[40]: 40
[41]: 41
[42]: 42
[43]: 43
[44]: 44
[45]: 45
[46]: 46
[47]: 47
[48]: 48
[49]: 49
[50]: 50
[51]: 51
[52]: 52
[53]: 53
[54]: 54
[55]: 55
[56]: 56
[57]: 57
[58]: 58
[59]: 59
[60]: 60
[61]: 61
[62]: 62
[63]: 63
[64]: 64
[65]: 65
[66]: 66
[67]: 67
[68]: 68
[69]: 69
[70]: 70
[71]: 71
[72]: 72
[73]: 73
[74]: 74
[75]: 75
[76]: 76
[77]: 77
[78]: 78
[79]: 79
[80]: 80
[81]: 81
[82]: 82
[83]: 83
[84]: 84
[85]: 85
[86]: 86
[87]: 87
[88]: 88
[89]: 89
[90]: 90
[91]: 91
[92]: 92
[93]: 93
[94]: 94
[95]: 95
[96]: 96
[97]: 97
[98]: 98
[99]: 99
[100]: 100
[101]: 101
[102]: 102
[103]: 103
[104]: 104
[105]: 105
[106]: 106
[107]: 107
[108]: 108
[109]: 109
[110]: 110
[111]: 111
[112]: 112
[113]: 113
[114]: 114
[115]: 115
[116]: 116
[117]: 117
[118]: 118
[119]: 119
[120]: 120
[121]: 121
[122]: 122
[123]: 123
[124]: 124
[125]: 125
[126]: 126
[127]: 127
[128]: 128
[129]: 129
[130]: 130
[131]: 131
[132]: 132
[133]: 133
[134]: 134
[135]: 135
[136]: 136
[137]: 137
[138]: 138
[139]: 139
[140]: 140
[141]: 141
[142]: 142
[143]: 143
[144]: 144
[145]: 145
[146]: 146
[147]: 147
[148]: 148
[149]: 149
[150]: 150
[151]: 151
[152]: 152
[153]: 153
[154]: 154
[155]: 155
[156]: 156
[157]: 157
[158]: 158
[159]: 159
[160]: 160
[161]: 161
[162]: 162
[163]: 163
[164]: 164
[165]: 165
[166]: 166
[167]: 167
[168]: 168
[169]: 169
[170]: 170
[171]: 171
[172]: 172
[173]: 173
[174]: 174
[175]: 175
[176]: 176
[177]: 177
[178]: 178
[179]: 179
[180]: 180
[181]: 181
[182]: 182
[183]: 183
[184]: 184
[185]: 185
[186]: 186
[187]: 187
[188]: 188
[189]: 189
[190]: 190
[191]: 191
[192]: 192
[193]: 193
[194]: 194
[195]: 195
[196]: 196
[197]: 197
[198]: 198
[199]: 199
[200]: 200
[201]: 201
[202]: 202
[203]: 203
[204]: 204
[205]: 205
[206]: 206
[207]: 207
[208]: 208
[209]: 209
[210]: 210
[211]: 211
[212]: 212
[213]: 213
[214]: 214
[215]: 215
[216]: 216
[217]: 217
[218]: 218
[219]: 219
[220]: 220
[221]: 221
[222]: 222
[223]: 223
[224]: 224
[225]: 225
[226]: 226
[227]: 227
[228]: 228
[229]: 229
[230]: 230
[231]: 231
[232]: 232
[233]: 233
[234]: 234
[235]: 235
[236]: 236
[237]: 237
[238]: 238
[239]: 239
[240]: 240
[241]: 241
[242]: 242
[243]: 243
[244]: 244
[245]: 245
[246]: 246
[247]: 247
[248]: 248
[249]: 249
[250]: 250
[251]: 251
[252]: 252
[253]: 253
[254]: 254
[255]: 255
[256]: 256
[257]: 257
[258]: 258
[259]: 259
[260]: 260
[261]: 261
[262]: 262
[263]: 263
[264]: 264
[265]: 265
[266]: 266
[267]: 267
[268]: 268
[269]: 269
[270]: 270
[271]: 271
[272]: 272
[273]: 273
[274]: 274
[275]: 275
[276]: 276
[277]: 277
[278]: 278
[279]: 279
[280]: 280
[281]: 281
[282]: 282
[283]: 283
[284]: 284
[285]: 285
[286]: 286
[287]: 287
[288]: 288
[289]: 289
[290]: 290
[291]: 291
[292]: 292
[293]: 293
[294]: 294
[295]: 295
[296]: 296
[297]: 297
[298]: 298
[299]: 299
[300]: 300
[301]: 301
[302]: 302
[303]: 303
[304]: 304
[305]: 305
[306]: 306
[307]: 307
[308]: 308
[309]: 309
[310]: 310
[311]: 311
[312]: 312
[313]: 313
[314]: 314
[315]: 315
[316]: 316
[317]: 317
[318]: 318
[319]: 319
[320]: 320
[321]: 321
[322]: 322
[323]: 323
[324]: 324
[325]: 325
[326]: 326
[327]: 327
[328]: 328
[329]: 329
[330]: 330
[331]: 331
[332]: 332
[333]: 333
[334]: 334
[335]: 335
[336]: 336
[337]: 337
[338]: 338
[339]: 339
[340]: 340
[341]: 341
[342]: 342
[343]: 343
[344]: 344
[345]: 345
[346]: 346
[347]: 347
[348]: 348
[349]: 349
[350]: 350
[351]: 351
[352]: 352
[353]: 353
[354]: 354
[355]: 355
[356]: 356
[357]: 357
[358]: 358
[359]: 359
[360]: 360
[361]: 361
[362]: 362
[363]: 363
[364]: 364
[365]: 365
[366]: 366
[367]: 367
[368]: 368
[369]: 369
[370]: 370
[371]: 371
[372]: 372
[373]: 373
[374]: 374
[375]: 375
[376]: 376
[377]: 377
[378]: 378
[379]: 379
[380]: 380
[381]: 381
[382]: 382
[383]: 383
[384]: 384
[385]: 385
[386]: 386
[387]: 387
[388]: 388
[389]: 389
[390]: 390
[391]: 391
[392]: 392
[393]: 393
[394]: 394
[395]: 395
[396]: 396
[397]: 397
[398]: 398
[399]: 399
[400]: 400
[401]: 401
[402]: 402
[403]: 403
[404]: 404
[405]: 405
[406]: 406
[407]: 407
[408]: 408
[409]: 409
[410]: 410
[411]: 411
[412]: 412
[413]: 413
[414]: 414
[415]: 415
[416]: 416
[417]: 417
[418]: 418
[419]: 419
[420]: 420
[421]: 421
[422]: 422
[423]: 423
[424]: 424
[425]: 425
[426]: 426
[427]: 427
[428]: 428
[429]: 429
[430]: 430
[431]: 431
[432]: 432
[433]: 433
[434]: 434
[435]: 435
[436]: 436
[437]: 437
[438]: 438
[439]: 439
[440]: 440
[441]: 441
[442]: 442
[443]: 443
[444]: 444
[445]: 445
[446]: 446
[447]: 447
[448]: 448
[449]: 449
[450]: 450
[451]: 451
[452]: 452
[453]: 453
[454]: 454
[455]: 455
[456]: 456
[457]: 457
[458]: 458
[459]: 459
[460]: 460
[461]: 461
[462]: 462
[463]: 463
[464]: 464
[465]: 465
[466]: 466
[467]: 467
[468]: 468
[469]: 469
[470]: 470
[471]: 471
[472]: 472
[473]: 473
[474]: 474
[475]: 475
[476]: 476
[477]: 477
[478]: 478
[479]: 479
[480]: 480
[481]: 481
[482]: 482
[483]: 483
[484]: 484
[485]: 485
[486]: 486
[487]: 487
[488]: 488
[489]: 489
[490]: 490
[491]: 491
[492]: 492
[493]: 493
[494]: 494
[495]: 495
[496]: 496
[497]: 497
[498]: 498
[499]: 499
[500]: 500
[501]: 501
[502]: 502
[503]: 503
[504]: 504
[505]: 505
[506]: 506
[507]: 507
[508]: 508
[509]: 509
[510]: 510
[511]: 511
[512]: 512
[513]: 513
[514]: 514
[515]: 515
[516]: 516
[517]: 517
[518]: 518
[519]: 519
[520]: 520
[521]: 521
[522]: 522
[523]: 523
[524]: 524
[525]: 525
[526]: 526
[527]: 527
[528]: 528
[529]: 529
[530]: 530
[531]: 531
[532]: 532
[533]: 533
[534]: 534
[535]: 535
[536]: 536
[537]: 537
[538]: 538
[539]: 539
[540]: 540
[541]: 541
[542]: 542
[543]: 543
[544]: 544
[545]: 545
[546]: 546
[547]: 547
[548]: 548
[549]: 549
[550]: 550
[551]: 551
[552]: 552
[553]: 553
[554]: 554
[555]: 555
[556]: 556
[557]: 557
[558]: 558
[559]: 559
[560]: 560
[561]: 561
[562]: 562
[563]: 563
[564]: 564
[565]: 565
[566]: 566
[567]: 567
[568]: 568
[569]: 569
[570]: 570
[571]: 571
[572]: 572
[573]: 573
[574]: 574
[575]: 575
[576]: 576
[577]: 577
[578]: 578
[579]: 579
[580]: 580
[581]: 581
[582]: 582
[583]: 583
[584]: 584
[585]: 585
[586]: 586
[587]: 587
[588]: 588
[589]: 589
[590]: 590
[591]: 591
[592]: 592
[593]: 593
[594]: 594
[595]: 595
[596]: 596
[597]: 597
[598]: 598
[599]: 599
[600]: 600
[601]: 601
[602]: 602
[603]: 603
[604]: 604
[605]: 605
[606]: 606
[607]: 607
[608]: 608
[609]: 609
[610]: 610
[611]: 611
[612]: 612
[613]: 613
[614]: 614

## 8.4 Model Train

```
[615]: clf = LogisticRegression()
clf.fit(X_train, y_train)
```

```
[616]: LogisticRegression()
```

```
[617]: log_reg_ov_ros_score=clf.score(X_test, y_test) *100
print("Accuracy",log_reg_ov_ros_score)
```

```
[618]: Accuracy 96.0
```

## 8.5 Prediction

```
[619]: log_reg_ov_ros_pred=clf.predict(X_test)
print(log_reg_ov_ros_pred)
```

```
[620]: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```