AUTOMATIC WATERING SYSTEM - SOFTWARE PART

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

```
df=pd.read_csv('TARP.csv')
df.head()
```

	Soil Moisture	Temperature	Soil Humidity	Time	Air temperature (C)	Wind speed (Km/h)	Air humidity (%)	Wind gust (Km/h)	Pressur (KPa
0	54	22	70	21	19.52	2.13	55.04	6.30	101.5
1	12	20	40	104	19.49	2.01	55.17	10.46	101.5
2	34	26	35	62	19.47	1.90	55.30	14.63	101.5
3	7	44	44	93	19.54	2.28	54.20	16.08	101.5
4	50	38	23	92	19.61	2.66	53.09	17.52	101.5

```
#Finding Null Values

df.isnull()
```

	Soil Moisture	Temperature	Soil Humidity	Time	Air temperature (C)	Wind speed (Km/h)	Air humidity (%)	Wi gu (Km/

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

```
Soil Moisture              0
Temperature                0
 Soil Humidity             0
Time                       0
Air temperature (C)    76005
Wind speed (Km/h)      76005
Air humidity (%)       76005
Wind gust (Km/h)       76005
Pressure (KPa)         76005
ph                     97800
rainfall               97800
N                      97800
P                      97800
K                      97800
Status                     0
dtype: int64
```

100000 rows × 15 columns

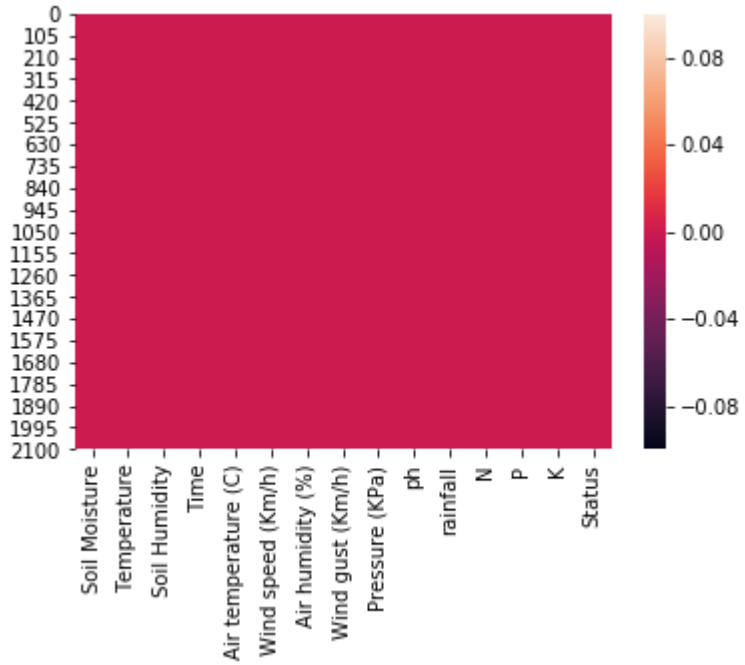
```
#Removing Null Values

df=df.dropna()
df
```

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

```
Soil Moisture          0
Temperature            0
 Soil Humidity         0
Time                   0
Air temperature (C)    0
Wind speed (Km/h)      0
Air humidity (%)       0
Wind gust (Km/h)       0
Pressure (KPa)         0
ph                     0
rainfall               0
N                      0
P                      0
K                      0
Status                 0
dtype: int64
```

```
#Heatmap for NullValues

sns.heatmap(df.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28f81330148>
```

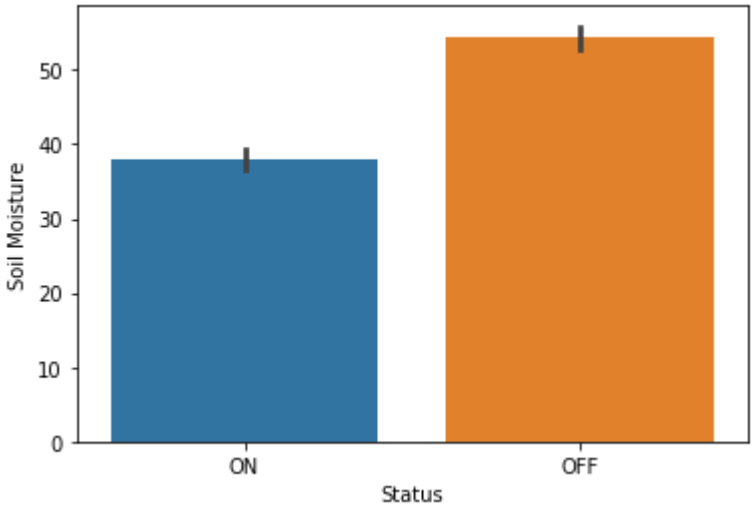



```
#CountPlot for TargetAttribute(Motor ON and OFF)

sns.countplot(df['Status'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28f8179c908>
```


```
#Barplot

sns.barplot(df['Status'],df['Soil Moisture'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28f8180a708>
```



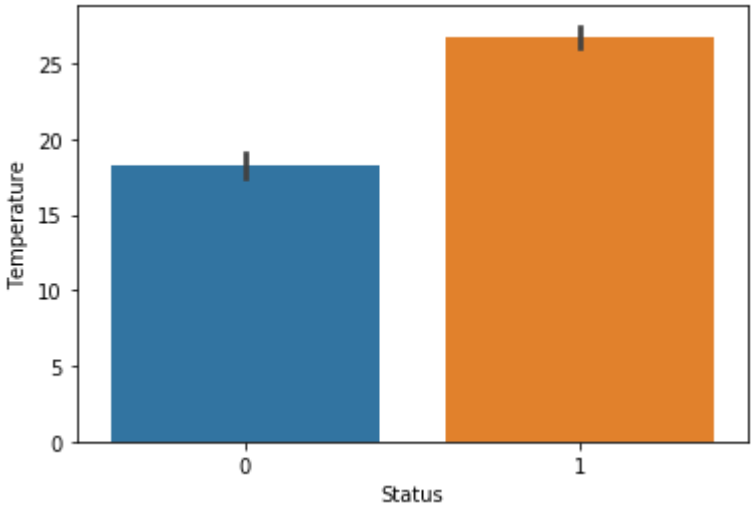

```
df.update(df['Status'].replace({'ON': 1,'OFF':0}))
df.head()
```

	Soil Moisture	Temperature	Soil Humidity	Time	Air temperature (C)	Wind speed (Km/h)	Air humidity (%)	Wind gust (Km/h)	P
0	54	22	70	21	19.52	2.13	55.04	6.30	
1	12	20	40	104	19.49	2.01	55.17	10.46	
2	34	26	35	62	19.47	1.90	55.30	14.63	
3	7	44	44	93	19.54	2.28	54.20	16.08	
4	50	38	23	92	19.61	2.66	53.09	17.52	

```
#Barplot
```

```
sns.barplot(df['Status'],df['Temperature'])
```

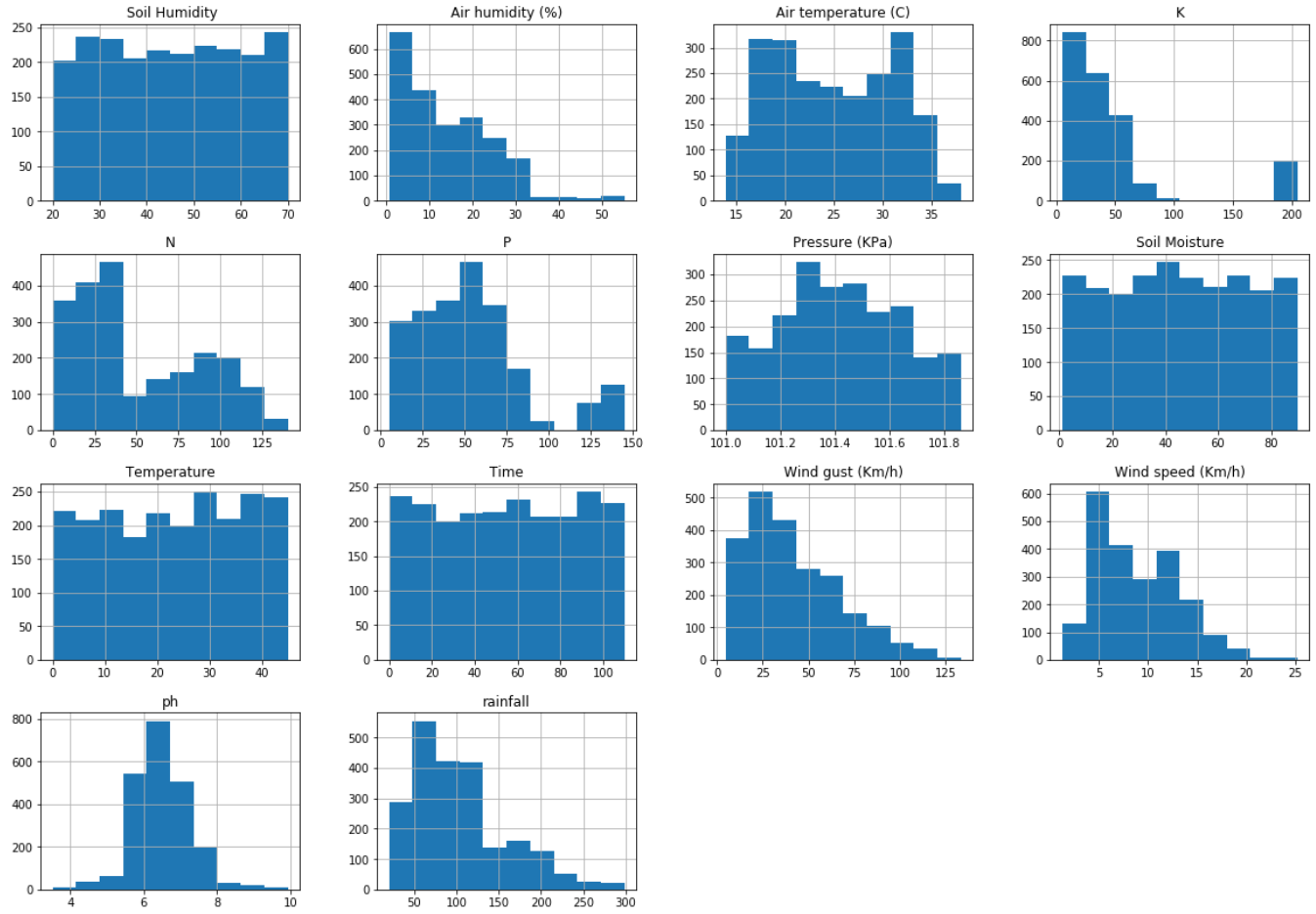
```
<matplotlib.axes._subplots.AxesSubplot at 0x28f818711c8>
```




```
#Correlation

df.corr()
```

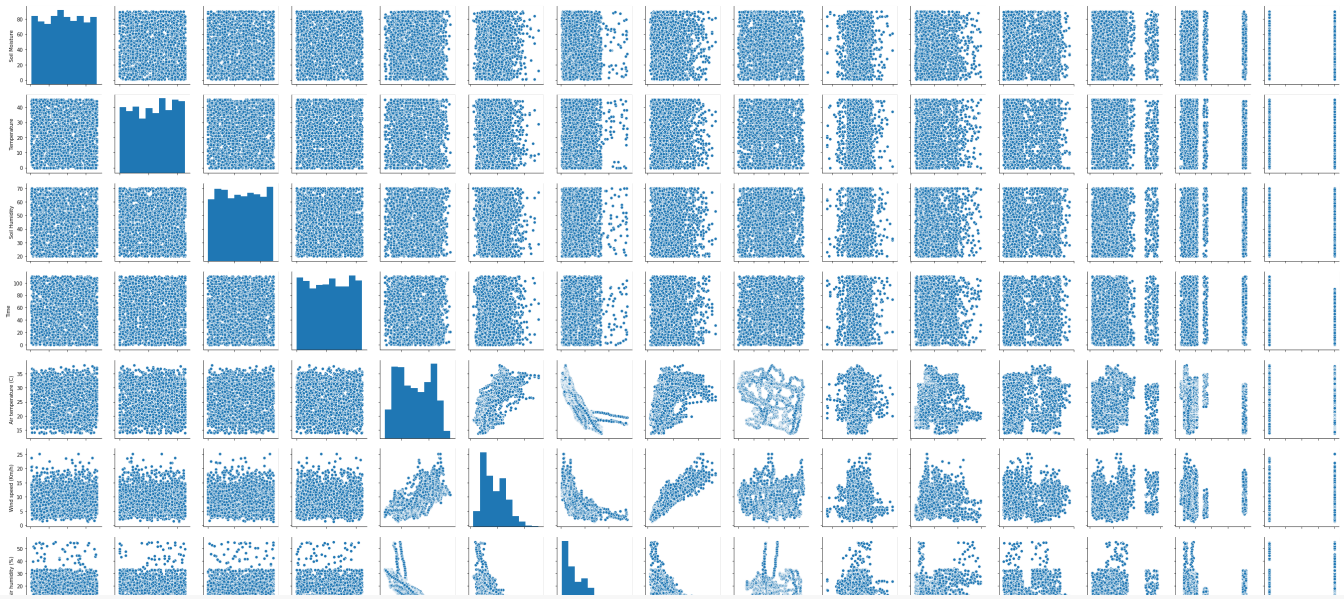
	Soil Moisture	Temperature	Soil Humidity	Time	Air temperature (C)	Wind speed (Km/h)	hu

```
df.hist(figsize=(20,14))
plt.show()
```

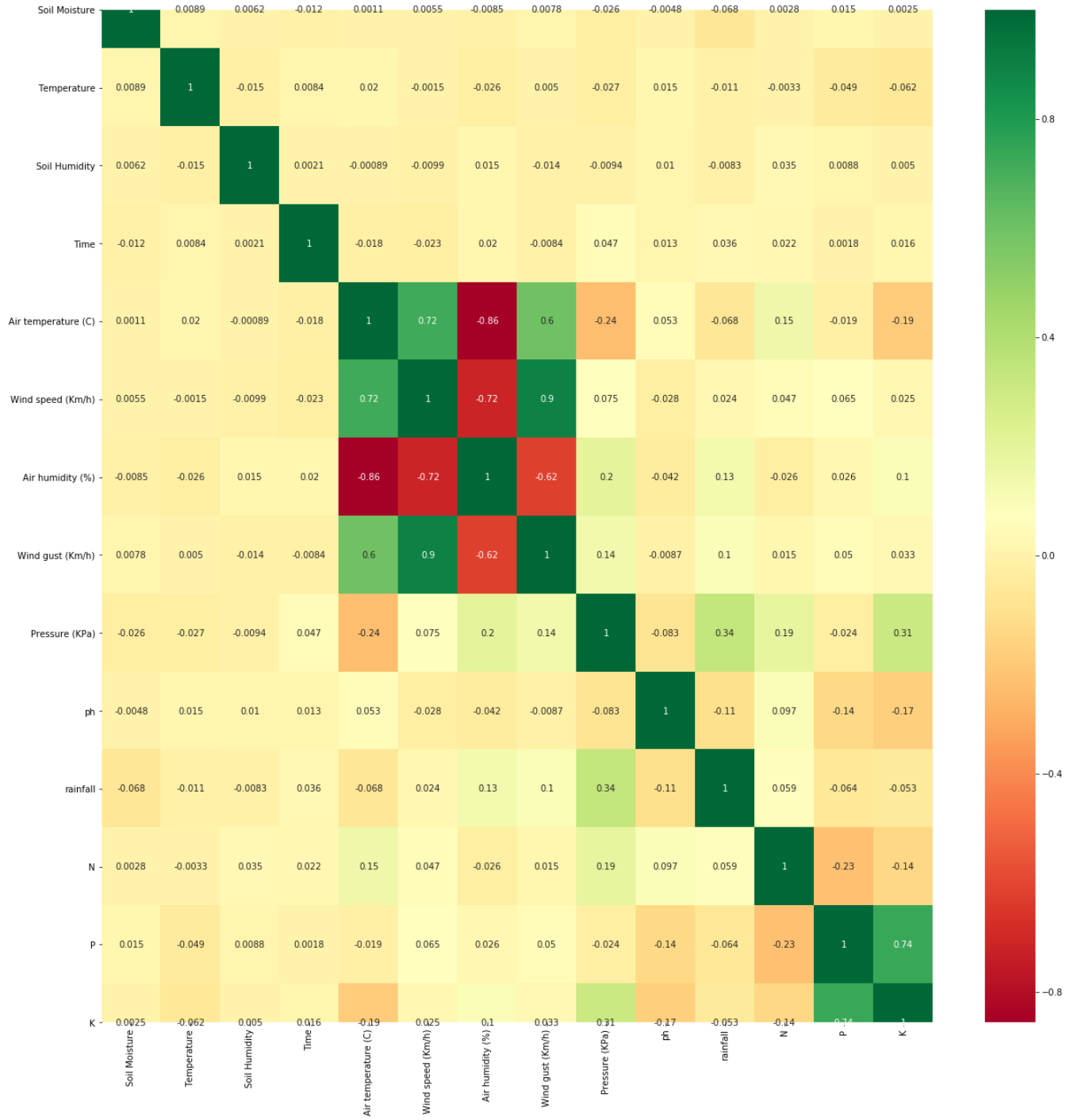


```
sns.pairplot(data=df)
```

```
<seaborn.axisgrid.PairGrid at 0x28f818dee88>
```



```
corrmat=df.corr()
top_corr_features=corrmat.index
plt.figure(figsize=(21,21))
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```




```
#Data Splitting

x=df.drop("Status",axis=1)
x.head()
```

	Soil Moisture	Temperature	Soil Humidity	Time	Air temperature (C)	Wind speed (Km/h)	Air humidity (%)	Wind gust (Km/h)	P

```
y=df[["Status"]]
y.head()
```

	Status
0	1
1	0
2	1
3	0
4	0

```
#TRAIN TEST DATA

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=5)
```

```
X_train.head()
```

	Soil Moisture	Temperature	Soil Humidity	Time	Air temperature (C)	Wind speed (Km/h)	Air humidity (%)	Wind gust (Km/h)
807	89	10	51	77	27.96	10.27	2.53	43.64
2120	65	12	24	26	20.29	5.46	22.45	28.24
812	56	20	45	62	26.62	7.12	3.91	32.64
1442	41	33	25	22	20.16	4.74	22.69	20.70
1051	65	21	24	35	31.97	14.65	3.74	85.71

```
X_test.head()
```

	Soil Temperature	Soil Time	Air temperature	Wind speed	Air humidity	Wind gust

```
y_train.head()
```

	Status
807	0
2120	0
812	1
1442	1
1051	0

```
y_test.head()
```

	Status
1270	0
1481	1
1832	1
293	1
1307	1

```
#Build the model

#1.Linear Regression()
#2.DecisionTreeRegressor()
#3.RandomForestRegressor()
#4.ScalarVectorMachine()
```

```
#1.LinearRegression()

from sklearn.linear_model import LinearRegression
l=LinearRegression()
l.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
Lin=l.predict(X_test)
Lin
```

```
array([[ 0.10787547],
       [ 0.28597437],
```

```
[ 0.42656369],
[ 0.76436112],
[ 0.13201558],
[ 0.27714688],
[ 0.55066428],
[ 0.68622592],
[ 0.51415193],
[ 0.2223561 ],
[ 1.07688013],
[ 0.09808055],
[ 0.3188968 ],
[ 0.75115913],
[ 0.45086959],
[ 0.4881762 ],
[ 1.02392238],
[ 0.50544166],
[ 0.56746982],
[ 0.4259001 ],
[ 0.34958226],
[ 0.12575588],
[ 0.3168269 ],
[ 0.29958091],
[ 0.5365357 ],
[ 0.27237835],
[ 0.27010241],
[ 0.03768203],
[ 0.844108  ],
[ 0.65064871],
[ 0.81076145],
[ 0.87087242],
[ 0.31466207],
[ 0.52244266],
[ 0.07901778],
[ 0.94411755],
[ 0.08921269],
[ 0.92351533],
[ 0.53202569],
[ 0.53529244],
[ 0.18639856],
[ 0.37067524],
[ 0.9935421 ],
[ 0.41751734],
[ 0.64431864],
[ 0.37197692],
[ 0.92085291],
[ 0.7195558 ],
[ 0.47916369],
[ 1.12745661],
[ 0.72800415],
[ 0.67746448],
[ 0.96908386],
[ 0.70991076],
[ 0.75330692],
[ 0.19118097],
[ 0.37967618],
[ 0.10787785],
```

```
y_test
```

	Status
1270	0
1481	1
1832	1
293	1
1307	1
...	...
1983	0
1011	0
1557	0
1900	0
2077	1

660 rows × 1 columns

```
#2.DecisionTreeRegressor()

from sklearn.tree import DecisionTreeRegressor
d=DecisionTreeRegressor()
d.fit(X_train,y_train)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

```
Dec_tree=d.predict(X_test)
Dec_tree
```

```
array([0., 1., 1., 1., 0., 0., 0., 1., 0., 0., 1., 0., 1., 1., 0., 1., 1.,
       1., 1., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 1., 1.,
       0., 1., 0., 1., 0., 1., 0., 0., 1., 1., 0., 0., 1., 1., 0., 1., 1.,
       1., 1., 1., 1., 1., 1., 0., 1., 0., 0., 1., 1., 0., 1., 1., 0., 0.,
       0., 0., 0., 1., 0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0.,
       0., 1., 1., 1., 1., 0., 1., 1., 0., 1., 0., 0., 0., 1., 1., 0., 0.,
       0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       0., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 0., 0., 0., 1., 0., 1.,
       1., 0., 1., 1., 1., 0., 1., 1., 0., 1., 0., 0., 0., 1., 0., 0., 1.,
       0., 0., 1., 0., 0., 0., 0., 0., 1., 1., 0., 1., 0., 1., 0., 1., 1.,
```

```
1., 0., 1., 0., 1., 1., 1., 1., 1., 0., 0., 0., 1., 1., 1., 1., 0.,
1., 0., 0., 0., 0., 0., 1., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1.,
0., 1., 1., 0., 0., 1., 1., 1., 0., 0., 1., 1., 0., 0., 1., 1., 1.,
0., 0., 1., 0., 0., 0., 1., 1., 0., 0., 1., 0., 1., 1., 0., 1., 0.,
0., 0., 1., 1., 1., 0., 1., 0., 1., 1., 1., 0., 0., 0., 1., 0., 0.,
0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1.,
1., 0., 0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0., 1.,
1., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 0., 1., 0., 0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 0.,
1., 0., 1., 1., 0., 0., 1., 1., 0., 0., 1., 0., 0., 0., 0., 1., 1.,
0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1.,
0., 0., 0., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1., 1., 1., 0., 0.,
1., 0., 1., 1., 0., 1., 0., 0., 1., 1., 1., 0., 0., 0., 0., 1., 0.,
0., 0., 1., 1., 0., 1., 0., 0., 0., 0., 0., 0., 1., 1., 0., 1., 0.,
1., 1., 0., 1., 1., 1., 1., 0., 1., 0., 0., 1., 1., 1., 1., 0., 1.,
0., 1., 1., 0., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 1., 0., 0.,
0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 0., 0., 0., 1., 1.,
1., 1., 1., 0., 1., 0., 0., 0., 1., 1., 1., 0., 0., 0., 1., 1., 0.,
1., 0., 0., 1., 0., 1., 0., 0., 1., 1., 0., 0., 1., 0., 0., 1., 1.,
1., 1., 1., 1., 0., 1., 0., 0., 0., 1., 1., 0., 1., 0., 0., 1., 1.,
1., 1., 1., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 0., 1., 0., 0.,
0., 0., 0., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1.,
0., 0., 0., 0., 0., 1., 0., 1., 1., 0., 0., 0., 0., 0., 0., 1., 0.,
1., 0., 0., 1., 0., 1., 0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 1.,
1., 0., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 1., 0.,
1., 0., 1., 0., 1., 0., 0., 0., 0., 1., 0., 1., 1., 1., 1., 0., 1.,
0., 0., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 0., 0.,
0., 0., 1., 1., 0., 0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 0., 0.,
0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 1.])
```

```
y_test
```

	Status
1270	0
1481	1
1832	1
293	1
1307	1
...	...
1983	0
1011	0
1557	0
1900	0
2077	1

660 rows × 1 columns

```
# Random forest

from sklearn.ensemble import RandomForestRegressor
r=RandomForestRegressor()
r.fit(X_train,y_train)
```

```
C:\Users\pc\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: T
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\pc\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: DataConversionWarning:
  """
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```

```
Random_for=r.predict(X_test)
Random_for
```

```
array([0.1, 0.6, 1. , 1. , 0.3, 0.3, 0. , 0.9, 0.5, 0.4, 1. , 0.2, 1. ,
       0.8, 0.1, 1. , 1. , 0.7, 0.5, 0. , 0.9, 0. , 0.3, 0. , 0.8, 0.1,
       0. , 0. , 0. , 0.1, 0.9, 0. , 1. , 0.9, 0. , 1. , 0. , 1. , 0. ,
       0.5, 0. , 0.3, 1. , 1. , 0. , 0. , 1. , 1. , 0. , 1. , 0.8, 1. ,
       1. , 1. , 0.9, 0.2, 0.7, 0.1, 0.9, 0. , 0. , 0.8, 0.2, 0. , 1. ,
       1. , 0. , 0. , 0. , 0.3, 0. , 0.9, 0. , 1. , 1. , 1. , 0.6, 0. ,
       1. , 0.8, 0.3, 0.4, 1. , 1. , 0. , 0. , 0.2, 1. , 0.7, 0.3, 0.1,
       1. , 0.8, 0.1, 0.9, 0. , 0. , 0. , 1. , 1. , 0. , 0. , 0. , 1. ,
       0. , 0.8, 1. , 0.5, 1. , 0.9, 0.9, 0.8, 1. , 1. , 1. , 1. , 1. ,
       0.8, 0.9, 0.2, 0.9, 0.9, 0.8, 0.9, 0.3, 1. , 1. , 0. , 0.7, 0.6,
       0. , 0. , 0.6, 0.9, 0. , 0.7, 0.5, 0. , 1. , 1. , 0.9, 0.1, 1. ,
       0.8, 0. , 0.6, 0. , 0. , 0.6, 1. , 0.5, 0. , 0.5, 0.1, 0.4, 1. ,
       0.2, 0.6, 0. , 0. , 0.3, 0.8, 0.9, 0.1, 1. , 0. , 1. , 0. , 1. ,
       1. , 0.8, 0. , 1. , 0. , 1. , 1. , 1. , 1. , 0.6, 0. , 0. , 0. ,
       1. , 1. , 0.9, 1. , 0. , 1. , 0. , 0. , 0.9, 0. , 0.2, 0.7, 1. ,
       0. , 0. , 1. , 0. , 1. , 0.9, 1. , 0.1, 1. , 0. , 0.9, 0.9, 0. ,
       0. , 0.9, 0.4, 1. , 0. , 0.1, 0.7, 0.5, 0. , 0.5, 0.7, 1. , 1. ,
       0. , 0. , 1. , 0.4, 0.6, 0. , 1. , 1. , 0. , 0. , 1. , 0. , 0.9,
       0.7, 0. , 1. , 0. , 0. , 0. , 0.9, 1. , 0.6, 0. , 0.9, 0.7, 0.8,
       0.8, 0.7, 0.4, 0. , 0.3, 1. , 0. , 1. , 0. , 1. , 0.4, 0.8, 0.9,
       1. , 0. , 0.2, 0.3, 0.2, 0. , 0.8, 0.9, 0.8, 1. , 0. , 1. , 0.8,
       0. , 0. , 0.2, 1. , 0.9, 0. , 0.6, 1. , 1. , 1. , 1. , 1. , 0. ,
       0. , 0. , 0.9, 1. , 1. , 0. , 0. , 0. , 0.9, 0.9, 1. , 0.9, 0.1,
       0.4, 1. , 1. , 1. , 1. , 0.9, 0.9, 1. , 1. , 0.9, 0.4, 0.8, 0.4,
       0. , 1. , 0. , 0.9, 0.3, 0.4, 1. , 0. , 0.6, 0.1, 0.6, 0.7, 0. ,
       1. , 1. , 0. , 0.1, 0.7, 1. , 0. , 1. , 1. , 0. , 0. , 0. , 0.5,
       0.5, 0.9, 0. , 0. , 0.4, 1. , 0. , 0. , 0.8, 1. , 1. , 0.8, 0. ,
       0.9, 1. , 1. , 0.8, 0.9, 0.5, 0. , 0. , 0.1, 1. , 1. , 1. , 0. ,
       0.8, 1. , 1. , 0.3, 0. , 1. , 1. , 1. , 0. , 0. , 1. , 0.1, 0.5,
       1. , 0.1, 0.9, 0. , 0. , 0.9, 0.7, 1. , 0.1, 0. , 0.2, 0. , 1. ,
       0. , 0. , 0. , 1. , 0.8, 0. , 1. , 0.7, 0.1, 0. , 0. , 0.2, 0. ,
```

```
0.5, 1. , 0. , 0.9, 0.2, 1. , 0.7, 0.1, 1. , 1. , 0.9, 1. , 0. ,
1. , 0. , 0. , 0.8, 1. , 0.7, 1. , 0.1, 1. , 0. , 0.8, 1. , 0. ,
0. , 1. , 0. , 1. , 0. , 0.9, 0.9, 0. , 1. , 0. , 1. , 0. , 0. ,
0.1, 0.4, 1. , 1. , 1. , 1. , 0.1, 1. , 0.6, 0.8, 0.5, 0.8, 0.1,
0. , 0. , 1. , 1. , 1. , 1. , 0.8, 0. , 1. , 0.1, 0. , 0. , 1. ,
0.9, 1. , 0. , 0. , 0. , 1. , 0.8, 0.1, 1. , 0. , 0.2, 1. , 0. ,
0.8, 0. , 0. , 0.8, 0.8, 0.5, 0.2, 1. , 0. , 0.2, 0.9, 1. , 0.3,
1. , 0.8, 0.9, 0. , 1. , 0. , 0. , 0. , 0.8, 1. , 0. , 1. , 0. ,
0.1, 1. , 0.8, 1. , 1. , 1. , 0. , 0. , 1. , 0.9, 1. , 1. , 0. ,
0. , 0. , 0. , 0. , 0.8, 0. , 0. , 0. , 0. , 0.1, 1. , 0.7, 1. ,
0.9, 0.5, 0.9, 0.5, 0. , 1. , 1. , 1. , 0. , 1. , 0.6, 0. , 0.2,
0.1, 0.3, 0. , 1. , 0.5, 1. , 1. , 0.9, 0. , 0. , 0.2, 0. , 0. ,
0.7, 0. , 0.9, 0. , 0. , 1. , 0. , 0.8, 0. , 0.6, 1. , 0.8, 0. ,
0. , 1. , 1. , 0.8, 1. , 0.6, 1. , 0. , 0. , 0. , 1. , 0.8, 1. ,
1. , 1. , 0. , 0. , 0. , 0. , 0. , 0. , 0.9, 0.2, 1. , 0. , 0.9,
0. , 1. , 0.7, 0.1, 0. , 0. , 0.8, 0. , 1. , 0.9, 0.9, 0.9, 0.1,
1. , 0. , 0. , 0. , 1. , 0.2, 1. , 0. , 1. , 0.8, 1. , 1. , 1. ,
0.9, 0. , 0.2, 0.4, 0. , 0. , 0. , 1. , 1. , 0.3, 0.6, 0.2, 0. ,
0.5, 0.7, 0.4, 0.9, 0.8, 1. , 0.9, 0. , 0. , 0.4, 0. , 0. , 0.8,
0.7, 1. , 0.6, 0. , 0.3, 0. , 0. , 0.2, 0. , 0.9])
```

```
y_test
```

	Status
1270	0
1481	1
1832	1
293	1
1307	1
...	...
1983	0
1011	0
1557	0
1900	0
2077	1

660 rows × 1 columns

```
#Support Vector Regressor

from sklearn.svm import SVR
s=SVR()
s.fit(X_train,y_train)
```

```
s_v_m=s.predict(X_test)
s_v_m
```

```
C:\Users\pc\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversi
  y = column_or_1d(y, warn=True)
C:\Users\pc\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The d
  "avoid this warning.", FutureWarning)
array([0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428541, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53426628, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428262, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428565,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
       0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
```

```
0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563,
0.53428563, 0.53428563, 0.53428563, 0.53428563, 0.53428563
```

```
y_test
```

	Status
1270	0
1481	1
1832	1
293	1
1307	1
...	...
1983	0
1011	0
1557	0
1900	0
2077	1

660 rows × 1 columns

```
#Finding whiuch is the Best model by comparing the mean square erro

from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score,accuracy_score
```

```
print("Mean Square error: ")
print("1.Linear Regression           :",mean_squared_error(Lin,y_test))
print("2.Decision Tree Regression    :",mean_squared_error(Dec_tree,y_test))
print("3.Random Forest Regressor     :",mean_squared_error(Random_for,y_test))
print("4.Support Vector Classifier   :",mean_squared_error(s_v_m,y_test))
```

```
Mean Square error:
1.Linear Regression           : 0.18709691363581482
2.Decision Tree Regression    : 0.06212121212121212
3.Random Forest Regressor     : 0.059121212121212116
4.Support Vector Classifier   : 0.250136566139686
```

```
print("R2_Score: ")
print("1.Linear Regression           :",r2_score(Lin,y_test))
print("2.Decision Tree Regression    :",r2_score(Dec_tree,y_test))
print("3.Random Forest Regressor     :",r2_score(Random_for,y_test))
```

```
print("4.Support Vector Classifier   :",r2_score(s_v_m,y_test))
```

```
R2_Score:
1.Linear Regression           : -1.4537559757397722
2.Decision Tree Regression    : 0.750688692543694
3.Random Forest Regressor     : 0.684251721748393
4.Support Vector Classifier   : -429314862923.2707
```

```
# The best is Decision Tree Regression Because of HIGH R2_Score and LOW MAE
#We predict our Further Data Using Decision Tree

x1=df[['Soil Moisture','Temperature']]
x1.head()
```

	Soil Moisture	Temperature
0	54	22
1	12	20
2	34	26
3	7	44
4	50	38

```
y1=df['Status']
y1.head()
```

```
0    1
1    0
2    1
3    0
4    0
Name: Status, dtype: object
```

```
datast=pd.read_csv("soil_moisture.csv")
datast.head()
```

	Soil Moisture	Temperature
0	50	23
1	25	32
2	55	21
3	15	30
4	87	22

```
#TRAIN TEST DATA
from sklearn.model_selection import train_test_split
X1_train,X1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.3,random_state=5)
```

```
from sklearn.tree import DecisionTreeRegressor
d=DecisionTreeRegressor()
d.fit(X1_train,y1_train)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

```
Dec_tree=d.predict(datast)
Dec_tree
```

```
array([0., 1., 1., 1., 0., 0., 1., 0., 1., 0.])
```

```
d.predict([[18,25]])
```

```
array([0.])
```

```
d.predict([[40,20]])
```

```
array([1.])
```

```
```