```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
df=pd.read_csv('/content/archive.zip')
df
```

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potabilit |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | NaN | 204.890455 | 20791.318981 | 7.300212 | 368.516441 | 564.308654 | 10.379783 | 86.990970 | 2.963135 | |
| **1** | 3.716080 | 129.422921 | 18630.057858 | 6.635246 | NaN | 592.885359 | 15.180013 | 56.329076 | 4.500656 | |
| **2** | 8.099124 | 224.236259 | 19909.541732 | 9.275884 | NaN | 418.606213 | 16.868637 | 66.420093 | 3.055934 | |
| **3** | 8.316766 | 214.373394 | 22018.417441 | 8.059332 | 356.886136 | 363.266516 | 18.436524 | 100.341674 | 4.628771 | |
| **4** | 9.092223 | 181.101509 | 17978.986339 | 6.546600 | 310.135738 | 398.410813 | 11.558279 | 31.997993 | 4.075075 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **3271** | 4.668102 | 193.681735 | 47580.991603 | 7.166639 | 359.948574 | 526.424171 | 13.894419 | 66.687695 | 4.435821 | |
| **3272** | 7.808856 | 193.553212 | 17329.802160 | 8.061362 | NaN | 392.449580 | 19.903225 | NaN | 2.798243 | |
| **3273** | 9.419510 | 175.762646 | 33155.578218 | 7.350233 | NaN | 432.044783 | 11.039070 | 69.845400 | 3.298875 | |
| **3274** | 5.126763 | 230.603758 | 11983.869376 | 6.303357 | NaN | 402.883113 | 11.168946 | 77.488213 | 4.708658 | |
| **3275** | 7.874671 | 195.102299 | 17404.177061 | 7.509306 | NaN | 327.459760 | 16.140368 | 78.698446 | 2.309149 | |

3276 rows × 10 columns

```
df.columns
```

```
Index(['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity',
       'Organic_carbon', 'Trihalomethanes', 'Turbidity', 'Potability'],
      dtype='object')
```

```
df.shape
```

```
(3276, 10)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          2495 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3114 non-null   float64
 8   Turbidity        3276 non-null   float64
 9   Potability       3276 non-null   int64  
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
```

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

```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
```

```
df1=df.dropna()
df1
```

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potabili |
|---|---|---|---|---|---|---|---|---|---|---|
| **3** | 8.316766 | 214.373394 | 22018.417441 | 8.059332 | 356.886136 | 363.266516 | 18.436524 | 100.341674 | 4.628771 | |
| **4** | 9.092223 | 181.101509 | 17978.986339 | 6.546600 | 310.135738 | 398.410813 | 11.558279 | 31.997993 | 4.075075 | |
| **5** | 5.584087 | 188.313324 | 28748.687739 | 7.544869 | 326.678363 | 280.467916 | 8.399735 | 54.917862 | 2.559708 | |
| **6** | 10.223862 | 248.071735 | 28749.716544 | 7.513408 | 393.663396 | 283.651634 | 13.789695 | 84.603556 | 2.672989 | |
| **7** | 8.635849 | 203.361523 | 13672.091764 | 4.563009 | 303.309771 | 474.607645 | 12.363817 | 62.798309 | 4.401425 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3267** | 8.989900 | 215.047358 | 15921.412018 | 6.297312 | 312.931022 | 390.410231 | 9.899115 | 55.069304 | 4.613843 | |
| **3268** | 6.702547 | 207.321086 | 17246.920347 | 7.708117 | 304.510230 | 329.266002 | 16.217303 | 28.878601 | 3.442983 | |
| **3269** | 11.491011 | 94.812545 | 37188.826022 | 9.263166 | 258.930600 | 439.893618 | 16.172755 | 41.558501 | 4.369264 | |
| **3270** | 6.069616 | 186.659040 | 26138.780191 | 7.747547 | 345.700257 | 415.886955 | 12.067620 | 60.419921 | 3.669712 | |
| **3271** | 4.668102 | 193.681735 | 47580.991603 | 7.166639 | 359.948574 | 526.424171 | 13.894419 | 66.687695 | 4.435821 | |

2011 rows × 10 columns

```
df1.isnull().sum()
```

```
ph                0
Hardness          0
Solids            0
Chloramines       0
Sulfate           0
Conductivity      0
Organic_carbon    0
Trihalomethanes   0
Turbidity         0
Potability        0
dtype: int64
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2011 entries, 3 to 3271
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ph               2011 non-null   float64
 1   Hardness         2011 non-null   float64
 2   Solids           2011 non-null   float64
 3   Chloramines      2011 non-null   float64
 4   Sulfate          2011 non-null   float64
 5   Conductivity     2011 non-null   float64
 6   Organic_carbon   2011 non-null   float64
 7   Trihalomethanes  2011 non-null   float64
 8   Turbidity        2011 non-null   float64
 9   Potability       2011 non-null   int64  
dtypes: float64(9), int64(1)
memory usage: 172.8 KB
```

```
df1.shape
```

```
(2011, 10)
```

```
df1.corr()
```

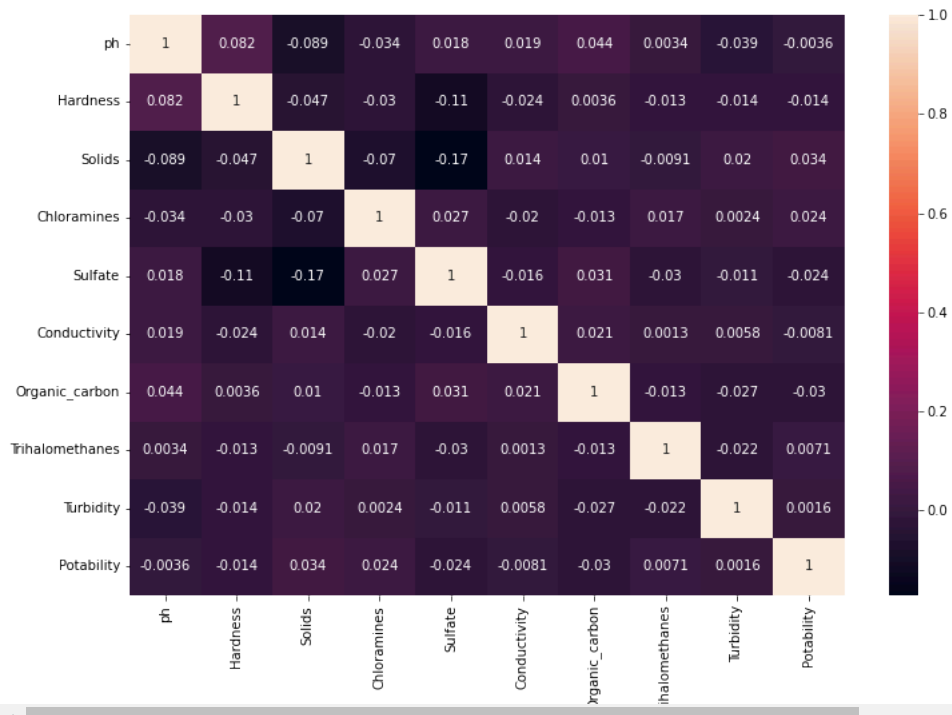
| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Pota |
|---|---|---|---|---|---|---|---|---|---|---|
| **ph** | 1.000000 | 0.108948 | -0.087615 | -0.024768 | 0.010524 | 0.014128 | 0.028375 | 0.018278 | -0.035849 | 0. |
| **Hardness** | 0.108948 | 1.000000 | -0.053269 | -0.022685 | -0.108521 | 0.011731 | 0.013224 | -0.015400 | -0.034831 | -0. |
| **Solids** | -0.087615 | -0.053269 | 1.000000 | -0.051789 | -0.162769 | -0.005198 | -0.005484 | -0.015668 | 0.019409 | 0. |
| **Chloramines** | -0.024768 | -0.022685 | -0.051789 | 1.000000 | 0.006254 | -0.028277 | -0.023808 | 0.014990 | 0.013137 | 0. |
| **Sulfate** | 0.010524 | -0.108521 | -0.162769 | 0.006254 | 1.000000 | -0.016192 | 0.026776 | -0.023347 | -0.009934 | -0. |
| **Conductivity** | 0.014128 | 0.011731 | -0.005198 | -0.028277 | -0.016192 | 1.000000 | 0.015647 | 0.004888 | 0.012495 | -0. |
| **Organic_carbon** | 0.028375 | 0.013224 | -0.005484 | -0.023808 | 0.026776 | 0.015647 | 1.000000 | -0.005667 | -0.015428 | -0. |
| **Trihalomethanes** | 0.018278 | -0.015400 | -0.015668 | 0.014990 | -0.023347 | 0.004888 | -0.005667 | 1.000000 | -0.020497 | 0. |
| **Turbidity** | -0.035849 | -0.034831 | 0.019409 | 0.013137 | -0.009934 | 0.012495 | -0.015428 | -0.020497 | 1.000000 | 0. |
| **Potability** | 0.014530 | -0.001505 | 0.040674 | 0.020784 | -0.015303 | -0.015496 | -0.015567 | 0.009244 | 0.022682 | 1 |

```
df["Potability"].value_counts()
```

```
0    1998
1    1278
Name: Potability, dtype: int64
```

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),annot=True)
```

```
<Axes: >
```




```
df1.describe()
```

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | P |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 2011.000000 | 2011.000000 | 2011.000000 | 2011.000000 | 2011.000000 | 2011.000000 | 2011.000000 | 2011.000000 | 2011.000000 | 20 |
| **mean** | 7.085990 | 195.968072 | 21917.441374 | 7.134338 | 333.224672 | 426.526409 | 14.357709 | 66.400859 | 3.969729 | |
| **std** | 1.573337 | 32.635085 | 8642.239815 | 1.584820 | 41.205172 | 80.712572 | 3.324959 | 16.077109 | 0.780346 | |
| **min** | 0.227499 | 73.492234 | 320.942611 | 1.390871 | 129.000000 | 201.619737 | 2.200000 | 8.577013 | 1.450000 | |
| **25%** | 6.089723 | 176.744938 | 15615.665390 | 6.138895 | 307.632511 | 366.680307 | 12.124105 | 55.952664 | 3.442915 | |
| **50%** | 7.027297 | 197.191839 | 20933.512750 | 7.143907 | 332.232177 | 423.455906 | 14.322019 | 66.542198 | 3.968177 | |
| **75%** | 8.052969 | 216.441070 | 27182.587067 | 8.109726 | 359.330555 | 482.373169 | 16.683049 | 77.291925 | 4.514175 | |
| **max** | 14.000000 | 317.338124 | 56488.672413 | 13.127000 | 481.030642 | 753.342620 | 27.006707 | 124.000000 | 6.494749 | |

```
df1.head()
```

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potability |
|---|---|---|---|---|---|---|---|---|---|---|
| **3** | 8.316766 | 214.373394 | 22018.417441 | 8.059332 | 356.886136 | 363.266516 | 18.436524 | 100.341674 | 4.628771 | 0 |
| **4** | 9.092223 | 181.101509 | 17978.986339 | 6.546600 | 310.135738 | 398.410813 | 11.558279 | 31.997993 | 4.075075 | 0 |
| **5** | 5.584087 | 188.313324 | 28748.687739 | 7.544869 | 326.678363 | 280.467916 | 8.399735 | 54.917862 | 2.559708 | 0 |
| **6** | 10.223862 | 248.071735 | 28749.716544 | 7.513408 | 393.663396 | 283.651634 | 13.789695 | 84.603556 | 2.672989 | 0 |
| **7** | 8.635849 | 203.361523 | 13672.091764 | 4.563009 | 303.309771 | 474.607645 | 12.363817 | 62.798309 | 4.401425 | 0 |

```
df1.tail()
```

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potabili |
|---|---|---|---|---|---|---|---|---|---|---|
| **3267** | 8.989900 | 215.047358 | 15921.412018 | 6.297312 | 312.931022 | 390.410231 | 9.899115 | 55.069304 | 4.613843 | |
| **3268** | 6.702547 | 207.321086 | 17246.920347 | 7.708117 | 304.510230 | 329.266002 | 16.217303 | 28.878601 | 3.442983 | |
| **3269** | 11.491011 | 94.812545 | 37188.826022 | 9.263166 | 258.930600 | 439.893618 | 16.172755 | 41.558501 | 4.369264 | |
| **3270** | 6.069616 | 186.659040 | 26138.780191 | 7.747547 | 345.700257 | 415.886955 | 12.067620 | 60.419921 | 3.669712 | |
| **3271** | 4.668102 | 193.681735 | 47580.991603 | 7.166639 | 359.948574 | 526.424171 | 13.894419 | 66.687695 | 4.435821 | |

```
x=df1.drop('Potability',axis=1)
y=df1["Potability"]
x.shape,y.shape
```

```
((2011, 9), (2011,))
```

## Normalization

```
scaler=StandardScaler()
x = scaler.fit_transform(x)
x
```

```
array([[ 0.7824658 ,  0.56411376,  0.01168692, ...,  1.22703167,
         2.11165179,  0.84476056],
       [ 1.27546291, -0.45565257, -0.45583491, ..., -0.84215371,
        -2.14039865,  0.13503344],
       [-0.95483488, -0.23461412,  0.7906452 , ..., -1.79234008,
        -0.7144228 , -1.80736621],
       ...,
       [ 2.8004919 , -3.10036538,  1.76750279, ...,  0.54602107,
        -1.5455849 ,  0.51212515],
       [-0.64615977, -0.28531709,  0.48857575, ..., -0.6889287 ,
        -0.3721083 , -0.3845623 ],
       [-1.53717226, -0.07007504,  2.9702871 , ..., -0.139372  ,
         0.01784567,  0.59743748]])
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
x_train,x_test,y_train,y_test
```

```
(            ph    Hardness        Solids
 1888   8.662710  173.531947  20333.079495
 301   10.049674  132.832837  11557.032038
 3027   5.290004  174.738423  28697.004827
 1919   8.458797  241.768340  29317.142440
 1587   6.539546  215.445204  28979.767601
 ...         ...         ...           ...
 2248   9.036504  199.321890  21696.003242
 2058   7.301563  177.662895  23441.267143
 2272   8.384296  223.328185  27463.654795
 1629   6.341751  139.959471  18058.320292
 2310   6.603641  206.919743  14598.025787

 [1407 rows x 3 columns],             ph    Hardness        Solids
 3103  5.967274  187.085084  30846.585474
 466   7.539882  185.825975  21575.245221
 1698  7.974065  160.224182  29477.764399
 3042  4.933106  162.184382  27771.080134
 394   8.736371  194.677687  24283.658791
 ...         ...         ...           ...
 698   6.246264  163.218038  26408.881768
 2314  5.161879  179.636409  18666.563638
 426   6.887864  173.325022  19947.924178
 1628  6.769769  178.331753  16980.258481
 2409  5.678221  143.186508  18377.008261

 [604 rows x 3 columns], 1888    0
 301     1
```

```
3027    0
1919    1
1587    1
       ..
2248    0
2058    0
2272    0
1629    1
2310    1
Name: Potability, Length: 1407, dtype: int64, 3103    0
466     0
1698    0
3042    0
394     1
       ..
698     1
2314    1
426     0
1628    1
2409    1
Name: Potability, Length: 604, dtype: int64)
```

```
x_train.shape,x_test.shape
```

```
((1407, 3), (604, 3))
```

## Logistic Regression

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
#Object of LR
model = LogisticRegression()
model
```

```
LogisticRegression()
```

```
#Training for this model
model.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
#making Prediction
pred = model.predict(x_test)
```

```
# Accuaracy checking of Logistic Regression
accuracy_lr = accuracy_score(y_test,pred)
accuracy_lr*100
```

```
60.59602649006622
```

## Decision Tree Classifier

```
features=['ph','Hardness','Solids']
x=df1[features]
y=df1['Potability']
x,y
```

```
(            ph    Hardness        Solids
3      8.316766  214.373394  22018.417441
4      9.092223  181.101509  17978.986339
5      5.584087  188.313324  28748.687739
6     10.223862  248.071735  28749.716544
7      8.635849  203.361523  13672.091764
...         ...         ...           ...
3267   8.989900  215.047358  15921.412018
3268   6.702547  207.321086  17246.920347
3269  11.491011   94.812545  37188.826022
3270   6.069616  186.659040  26138.780191
3271   4.668102  193.681735  47580.991603

[2011 rows x 3 columns], 3       0
```

```
4       0
5       0
6       0
7       0
       ..
3267    1
3268    1
3269    1
3270    1
3271    1
Name: Potability, Length: 2011, dtype: int64)
```

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
#Creating a model tree
dtree=DecisionTreeClassifier()
#Traning Decision Tree
dtree.fit(x,y)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
dtree.predict([[8.316766,214.373394,22018.417441]])
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassi
  warnings.warn(
array([0])
```

```
dtree.predict([[11.491011,94.812545,37188.826022]])
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassi
  warnings.warn(
array([1])
```

```
# Accuracy check for Decision Tree
accuracy_dt = accuracy_score(y_test,pred)
accuracy_dt*100
```

```
60.59602649006622
```

## Support Vector Machine

```
#Import svm model
from sklearn import svm
```

```
#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel
clf
```

```
SVC
SVC(kernel='linear')
```

```
#Train the model using the training sets
clf.fit(x_train,y_train)
```

```
SVC
SVC(kernel='linear')
```

```
#Predict the response for test dataset
y_pred = clf.predict(x_test)
y_pred
```

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

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

```
accuracy_svm=accuracy_score(y_test,y_pred)
accuracy_svm*100
```

```
58.94039735099338
```

## Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
# Create a Random Forect Classifier
model = RandomForestClassifier()
model
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
# Training model Forest
model.fit(x_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
#Predict the model forest
y_pred = model.predict(x_test)
y_pred
```

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

```
accuracy_rf = accuracy_score(y_test,y_pred)
accuracy_rf*100
```

```
55.29801324503312
```

Start coding or generate with AI.