```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Logistic Regression

```
data = pd.read_csv('Phishing attack.csv')
data
```

	id	NumDots	SubdomainLevel	PathLevel	UrlLength	NumDash	NumDashInHostnam
0	1	3	1	5	72	0	
1	2	3	1	3	144	0	
2	3	3	1	2	58	0	
3	4	3	1	6	79	1	
4	5	3	0	4	46	0	
...	...	...	...	...	...	...	
794	795	2	1	3	49	0	
795	796	2	0	2	54	0	
796	797	3	0	4	58	0	
797	798	4	1	5	94	1	
798	799	3	1	2	60	2	

799 rows × 50 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 799 entries, 0 to 798
Data columns (total 50 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   id                                  799 non-null    int64
 1   NumDots                             799 non-null    int64
 2   SubdomainLevel                      799 non-null    int64
 3   PathLevel                           799 non-null    int64
 4   UrlLength                           799 non-null    int64
 5   NumDash                             799 non-null    int64
 6   NumDashInHostname                   799 non-null    int64
 7   AtSymbol                            799 non-null    int64
 8   TildeSymbol                         799 non-null    int64
 9   NumUnderscore                       799 non-null    int64
 10  NumPercent                          799 non-null    int64
 11  NumQueryComponents                  799 non-null    int64
 12  NumAmpersand                        799 non-null    int64
 13  NumHash                             799 non-null    int64
 14  NumNumericChars                     799 non-null    int64
 15  NoHttps                             799 non-null    int64
 16  RandomString                        799 non-null    int64
 17  IpAddress                           799 non-null    int64
 18  DomainInSubdomains                  799 non-null    int64
 19  DomainInPaths                       799 non-null    int64
 20  HttpsInHostname                     799 non-null    int64
 21  HostnameLength                      799 non-null    int64
 22  PathLength                          799 non-null    int64
 23  QueryLength                         799 non-null    int64
 24  DoubleSlashInPath                   799 non-null    int64
 25  NumSensitiveWords                   799 non-null    int64
 26  EmbeddedBrandName                   799 non-null    int64
 27  PctExtHyperlinks                    799 non-null    float64
 28  PctExtResourceUrls                  799 non-null    float64
 29  ExtFavicon                          799 non-null    int64
 30  InsecureForms                       799 non-null    int64
 31  RelativeFormAction                  799 non-null    int64
 32  ExtFormAction                       799 non-null    int64
 33  AbnormalFormAction                  799 non-null    int64
```

```
 34  PctNullSelfRedirectHyperlinks       799 non-null    float64
 35  FrequentDomainNameMismatch          799 non-null    int64
 36  FakeLinkInStatusBar                 799 non-null    int64
 37  RightClickDisabled                  799 non-null    int64
 38  PopUpWindow                         799 non-null    int64
 39  SubmitInfoToEmail                   799 non-null    int64
 40  IframeOrFrame                       799 non-null    int64
 41  MissingTitle                        799 non-null    int64
 42  ImagesOnlyInForm                    799 non-null    int64
 43  SubdomainLevelRT                    799 non-null    int64
 44  UrlLengthRT                         799 non-null    int64
 45  PctExtResourceUrlsRT                799 non-null    int64
 46  AbnormalExtFormActionR              799 non-null    int64
 47  ExtMetaScriptLinkRT                 799 non-null    int64
 48  PctExtNullSelfRedirectHyperlinksRT  799 non-null    int64
 49  CLASS_LABEL                         799 non-null    int64
dtypes: float64(3), int64(47)
memory usage: 312.2 KB
```

```
# input
x = data.iloc[:, [5,6]].values
#output
y = data.iloc[:, 48]. values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size
= 0.25, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
X_train = sc_x.fit_transform(X_train)
X_test = sc_x.transform(X_test)
print (X_train[0:10, :])
```

```
[[-0.55173268 -0.36471557]
 [ 0.26027684 -0.36471557]
 [-0.55173268 -0.36471557]
 [ 0.26027684 -0.36471557]
 [-0.55173268 -0.36471557]
 [-0.55173268 -0.36471557]
 [-0.55173268 -0.36471557]
 [ 0.26027684 -0.36471557]
 [-0.55173268 -0.36471557]
 [ 0.26027684  1.48667955]]
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix : \n", cm)
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix :
 [[95  0 18]
 [ 6  2  3]
 [54  0 22]]
Accuracy :  0.595
```

Linear Regression

```
#import the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
data=pd.read_csv('Phishing attack.csv')
data
```

	id	NumDots	SubdomainLevel	PathLevel	UrlLength	NumDash	NumDashInHostnam
0	1	3	1	5	72	0	
1	2	3	1	3	144	0	
2	3	3	1	2	58	0	
3	4	3	1	6	79	1	
4	5	3	0	4	46	0	
...	...	...	...	...	...	...	.
794	795	2	1	3	49	0	
795	796	2	0	2	54	0	
796	797	3	0	4	58	0	
797	798	4	1	5	94	1	
798	799	3	1	2	60	2	

799 rows × 50 columns

```
data.info
```

```
795  796        2              0          2         54        0
796  797        3              0          4         58        0
797  798        4              1          5         94        1
798  799        3              1          2         60        2

     NumDashInHostname  AtSymbol  TildeSymbol  NumUnderscore  ...  \
0                    0         0            0              0  ...
1                    0         0            0              2  ...
2                    0         0            0              0  ...
3                    0         0            0              0  ...
4                    0         0            0              0  ...
..                 ...       ...          ...            ...  ...
794                  0         0            0              0  ...
795                  0         0            0              0  ...
796                  0         0            0              0  ...
797                  0         0            0              1  ...
798                  0         0            0              0  ...

     IframeOrFrame  MissingTitle  ImagesOnlyInForm  SubdomainLevelRT  \
0                0             0                 1                 1
1                0             0                 0                 1
2                0             0                 0                 1
3                0             0                 0                 1
4                1             0                 0                 1
..             ...           ...               ...               ...
794              0             0                 0                 1
795              0             0                 0                 1
796              1             0                 0                 1
797              0             0                 0                 1
798              0             0                 0                 1

     UrlLengthRT  PctExtResourceUrlsRT  AbnormalExtFormActionR  \
0              0                     1                       1
1             -1                     1                       1
2              0                    -1                       1
3             -1                     1                       1
4              1                    -1                       0
..           ...                   ...                     ...
794            1                     1                       1
795            0                     1                       1
796            0                     1                       1
797           -1                     1                       1
```

```
..  ...  ...  ...
794  1  1  1
795  -1  -1  1
796  1  -1  1
797  0  1  1
798  1  1  1

[799 rows x 50 columns]>
```

```
data.columns
```

```
Index(['id', 'NumDots', 'SubdomainLevel', 'PathLevel', 'UrlLength', 'NumDash',
       'NumDashInHostname', 'AtSymbol', 'TildeSymbol', 'NumUnderscore',
       'NumPercent', 'NumQueryComponents', 'NumAmpersand', 'NumHash',
       'NumNumericChars', 'NoHttps', 'RandomString', 'IpAddress',
       'DomainInSubdomains', 'DomainInPaths', 'HttpsInHostname',
       'HostnameLength', 'PathLength', 'QueryLength', 'DoubleSlashInPath',
       'NumSensitiveWords', 'EmbeddedBrandName', 'PctExtHyperlinks',
       'PctExtResourceUrls', 'ExtFavicon', 'InsecureForms',
       'RelativeFormAction', 'ExtFormAction', 'AbnormalFormAction',
       'PctNullSelfRedirectHyperlinks', 'FrequentDomainNameMismatch',
       'FakeLinkInStatusBar', 'RightClickDisabled', 'PopUpWindow',
       'SubmitInfoToEmail', 'IframeOrFrame', 'MissingTitle',
       'ImagesOnlyInForm', 'SubdomainLevelRT', 'UrlLengthRT',
       'PctExtResourceUrlsRT', 'AbnormalExtFormActionR', 'ExtMetaScriptLinkRT',
       'PctExtNullSelfRedirectHyperlinksRT', 'CLASS_LABEL'],
      dtype='object')
```

```
X=data['NumDots'].values
Y=data['UrlLength'].values
```

```
print(X.mean())
print(Y.mean())
mean_x=X.mean()
mean_y=Y.mean()
```

```
2.755944931163955
68.31414267834793
```

```
from numpy.core import numeric
n=len(X)
numer=0
denom=0
for i in range(n):
  numer=numer+((X[i]-mean_x)*(Y[i]-mean_y))
  denom=denom+(X[i]-mean_x)**2
b1=numer/denom
print(b1)
#from line equation we have y=b1*xbo
b0=mean_y-(b1*mean_x)
print(b0)
```

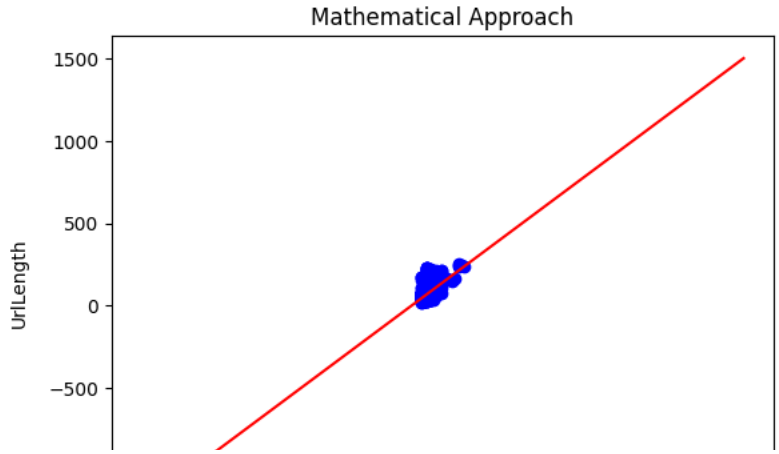
```
12.65429135945251
33.43961254879296
```

```
print("Linear Model is Y=",b1,"*x+",b0)
```

```
Linear Model is Y= 12.65429135945251 *x+ 33.43961254879296
```

```
min_x=np.min(X)-100
max_x=np.max(X)+100
x=np.linspace(min_x,max_x,1000)
y=b1*x+b0
plt.scatter(X,Y,color='blue',label="Input data")
plt.plot(x,y,color='red',label="Regression line")
plt.xlabel("ID")
plt.ylabel("UrlLength")
plt.title("Mathematical Approach")
```

```
Text(0.5, 1.0, 'Mathematical Approach')
```




```
print(X[2])
print(Y[2])
```

```
3
58
```

```
y=b1*X[2]+b0
y
```

```
71.4024866271505
```

```
ss_tot=0
ss_res=0
n=len(X)
for i in range(n):
  y_pred=b0+b1*X[i]
  ss_tot +=(Y[i]-mean_y)**2
  ss_res +=(y_pred- Y[i])**2
r2 =1-(ss_res/ss_tot)
print(r2)
print(r2*100)
```

```
0.2834062874890866
28.340628748908657
```

PolynomialRegression

```
import numpy as np
import pandas as pd
```

```
m=100
x=6*np.random.rand(m,1)-1
x
```

```
              [ 1.92898773],
              [ 4.78664456],
              [ 1.06803157],
              [ 1.94656321],
              [ 4.72157355],
              [ 0.96389685],
              [ 0.27986024],
              [ 4.41829781],
              [ 1.21876324],
              [ 4.06615948],
              [ 2.70553976],
              [ 1.86347816],
              [ 4.83210653],
              [ 2.46904964],
              [ 4.20873808],
              [ 4.81124234],
              [ 0.11955655],
              [-0.89975112],
              [-0.03758882],
              [ 3.4159352 ],
              [ 0.96744933],
              [ 2.50998343],
              [ 3.22623792],
              [ 1.21651386],
              [ 2.57928236],
              [ 3.34355006],
              [ 4.89072789],
              [ 1.75951012],
              [ 1.21887825],
              [ 1.37136899],
              [ 2.25790055],
              [-0.13384842],
              [-0.82586835],
              [ 2.4818491 ],
              [ 2.1510154 ],
              [ 4.00181089],
              [ 0.60881167],
              [ 0.03932995],
              [ 3.50014457]])
```

```
x.shape
```

```
(100, 1)
```

```
y = 0.5*-x**2+x+2+np.random.rand(m,1) #ax^2 +bx+c
y.shape
```

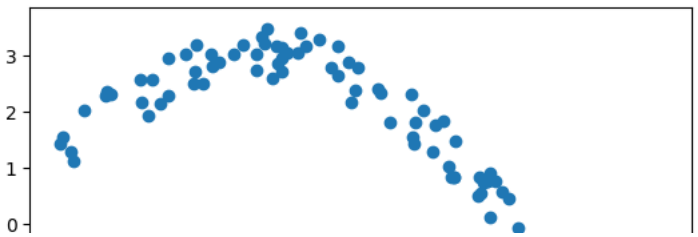
```
(100, 1)
```

```
#Ad it clearly conveys st.line will never fit the data but let's check it
from sklearn.linear_model import LinearRegression
```

```
lin_model = LinearRegression()
lin_model.fit(x,y)
```

```
▾ LinearRegression
LinearRegression()
```

```
import matplotlib.pyplot as plt
plt.scatter(x,y)
plt.show()
```



Random Forest

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

df = pd.read_csv('Phishing attack.csv')


X = df.drop('NumDots', axis=1)
y = df['NumDots']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


rfc = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier on the training set
rfc.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = rfc.predict(X_test)

# Evaluate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy*100)
```

```
Accuracy: 65.625
```

DecisionTreeClassifier

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the dataset
df = pd.read_csv('Phishing attack.csv')

# Split the dataset into training and testing sets
X = df.drop('SubdomainLevel', axis=1)
y = df['SubdomainLevel']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create a decision tree classifier
dtc = DecisionTreeClassifier(random_state=42)

# Train the classifier on the training set
dtc.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = dtc.predict(X_test)

# Evaluate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy*100)
```

```
Accuracy: 82.5
```

✓ 0s completed at 8:15 PM