```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn import metrics
from imblearn.combine import SMOTEENN


Le = LabelEncoder()


cyber_data = pd.read_csv('merged_track_b_learn.csv')
print("Number of Observations in Cyber dataset:", cyber_data.shape)
cyber_data.head()
```

```
Number of Observations in Cyber dataset: (50000, 11)
```

	Unnamed: 0	idx	s_ip	s_port	d_ip	d_port	protocol	
0	0	0	210.248.110.200	47760	144.82.144.26	42718	TCP	
1	1	1	172.116.62.150	47760	38.29.1.69	39548	TCP	
2	2	2	15.164.135.69	60390	37.154.75.207	80	TCP	id=-9865%27%2
3	3	3	84.12.150.116	47760	64.89.252.59	37816	TCP	
4	4	4	95.42.196.145	53040	17.75.195.163	80	TCP(6)	GET /c

```
cyber_data = cyber_data.drop(columns=['Unnamed: 0','idx'],axis=1)


cyber_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   s_ip        50000 non-null  object
 1   s_port      50000 non-null  int64
 2   d_ip        50000 non-null  object
 3   d_port      50000 non-null  int64
 4   protocol    50000 non-null  object
```

```
 5   payload     49974 non-null  object
 6   risk        50000 non-null  int64
 7   payload_LE  50000 non-null  int64
 8   label       50000 non-null  object
dtypes: int64(4), object(5)
memory usage: 3.4+ MB
```

```
cyber_data.describe()
```

	s_port	d_port	risk	payload_LE
count	50000.000000	50000.000000	50000.00000	50000.00000
mean	36739.119080	25438.775900	1.74906	6254.52076
std	19807.334189	23460.344102	0.64254	5867.15090
min	68.000000	22.000000	1.00000	0.00000
25%	27255.500000	80.000000	1.00000	239.00000
50%	47760.000000	34107.000000	2.00000	5226.50000
75%	48673.500000	48258.000000	2.00000	12094.25000
max	65476.000000	60998.000000	3.00000	16127.00000

```
cat_col = cyber_data.dtypes[cyber_data.dtypes == "object"]
num_col = cyber_data.dtypes[cyber_data.dtypes != "object"]
#Categorical
print(cat_col)
#Numerical
print(num_col)
```

```
s_ip        object
d_ip        object
protocol    object
payload     object
label       object
dtype: object
s_port        int64
d_port        int64
risk          int64
payload_LE    int64
dtype: object
```

```
cyber_data.isnull().sum()
```

```
s_ip          0
s_port        0
d_ip          0
d_port        0
protocol      0
```

```
payload       26
risk           0
payload_LE     0
label          0
dtype: int64
```

```
cyber_data['payload'].fillna(0, inplace = True)
```

```
cyber_data.isnull().sum()
```

```
s_ip          0
s_port        0
d_ip          0
d_port        0
protocol      0
payload       0
risk          0
payload_LE    0
label         0
dtype: int64
```

```
for col in list(cat_col.index):
    print(f"--------------------{col.title()}-------------------------")
    total= cyber_data[col].value_counts()
    percent = cyber_data[col].value_counts() / cyber_data.shape[0]
    df = pd.concat([total,percent],keys = ['total','percent'],axis = 1)
    print(df)
    print('\n')
```

```
142.215.174.2        1  0.00002
206.136.40.227       1  0.00002
211.126.163.217      1  0.00002
113.192.4.211        1  0.00002

[49998 rows x 2 columns]


--------------------D_Ip-------------------------
                total  percent
61.246.2.210        2  0.00004
144.82.144.26       1  0.00002
50.54.50.248        1  0.00002
153.250.66.166      1  0.00002
71.223.88.77        1  0.00002
...               ...      ...
70.14.234.118       1  0.00002
64.240.6.187        1  0.00002
40.156.106.17       1  0.00002
153.34.143.130      1  0.00002
95.42.130.186       1  0.00002

[49999 rows x 2 columns]
```

```
--------------------Protocol-------------------------
           total  percent
TCP        25288  0.50576
TCP( 6 )   24656  0.49312
UDP( 17 )     52  0.00104
UDP            4  0.00008


--------------------Payload-------------------------
                                                  total  percent
......(                                           15818  0.31636
.........4.........                                2736  0.05472
POST /wsman?PSVersion=7.1.3 HTTP/1.1\r\nConnect...   90  0.00180
HTTP/1.1 403 Forbidden\r\nDate: Thu, 28 Jul 202...   90  0.00180
HTTP/1.1 403 Forbidden\r\nDate: Thu, 28 Jul 202...   86  0.00172
...                                                ...      ...
GET /?id=-4835%60%29%20WHERE%203564%3D3564%20UN...    1  0.00002
GET /vulnerabilities/sqli/?id=1%27%3BSELECT%20P...    1  0.00002
GET /?id=user%20UNION%20ALL%20SELECT%20NULL%2CN...    1  0.00002
GET /manager/html/upload?org.apache.catalina.fi...    1  0.00002
GET /?id=user%29%29%20UNION%20ALL%20SELECT%20NU...    1  0.00002

[16128 rows x 2 columns]


--------------------Label-------------------------
                  total  percent
2_exploit         19690  0.39380
4_unknown         16089  0.32178
3_post            11718  0.23436
1_reconnaissance   2503  0.05006
```

```
cyber_data['payload'].nunique()
```

```
16128
```

```
from pyparsing import replace_with
unknown_payload = ['......(','.........4......... ','.........4......... ','.........4.....

#Replace
cyber_data['payload'].replace(to_replace = unknown_payload, value = 'un_payload' , inplace =
```

```
cyber_data['payload'].value_counts()
```

```
font-size: 16px;\n}\n\nh2, h3, h4 {\n  font-weight: 200;\n}\n\nh2 {\n  font-size:
28px;\n}\n\n.jumbotron {\n  margin-bottom: 0;\n  color: #333;\n  background:
rgb(212,212,221); /* Old browsers */\n  background: radial-gradient(ellipse at

center top, rgba(255,255,255,1) 0%,rgba(174,174,183,1) 100%); /* W3C
```

```
*/\n}\n\n.jumbotron h1 {\n  font-size: 128px;\n  font-weight: 700;\n  color:
white;\n  text-shadow: 0px 2px 0px #abc,\n               0px 4px 10px
rgba(0,0,0,0.15),\n               0px 5px 2px rgba(0,0,0,0.1),\n               0px
6px 30px rgba(0,0,0,0.1);\n}\n\n.jumbotron p {\n  font-size: 28px;\n  font-weight:
100;\n}\n\n.main {\n   background: white;\n   color: #234;\n   border-top: 1px solid
rgba(0,0,0,0.12);\n   padding-top: 30px;\n   padding-bottom: 40px;\n}\n\n.footer {\n
border-top: 1px solid rgba(255,255,255,0.2);\n   padding-top: 30px;\n}\n\n    -->
</style>\n</head>\n<body>\n  <div class="jumbotron text-center">\n    <div
class="container">\n   .  <h1>Testing 123..</h1>\n  ..<p class="lead">This page is
used to test the proper operation of the <a href="http://apache.org">Apache HTTP
server</a> after it has been installed. If you can read this page it means that this
site is working properly. This server is powered by <a
href="http://centos.org">CentOS</a>.</p>\n..</div>\n  </div>\n  <div class="main">\n
<div class="container">\n       <div class="row">\n  ...<div class="col-sm-6">\n
...<h2>Just visiting?</h2>\n...  ..<p class="lead">The website you just visited is
either experiencing problems or is undergoing routine maintenance.</p>\n  .....<p>If
you would like to let the administrators of this website know that you've seen this
page instead of the page you expected, you should send them e-mail. In general, mail
sent to the name "webmaster" and directed to the website's domain should reach the
appropriate person.</p>\n  .....<p>For example, if you experienced problems while
visiting www.example.com, you should send e-mail to "webmaster@example.com".</p>\n.
...</div>\n  ....<div class="col-s       86
HTTP/1.1 403 Forbidden\r\nDate: Thu, 28 Jul 2022 08:20:39 GMT\r\nServer:
Apache/2.4.6 (CentOS)\r\nLast-Modified: Thu, 16 Oct 2014 13:20:58 GMT\r\nETag:
"1321-5058a1e728280"\r\nAccept-Ranges: bytes\r\nContent-Length: 4897\r\nConnection:
close\r\nContent-Type: text/html; charset=UTF-8\r\n\r\n<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"><html>
<head>\n<meta http-equiv="content-type" content="text/html; charset=UTF-8">\n..
<title>Apache HTTP Server Test Page powered by CentOS</title>\n..<meta http-
equiv="Content-Type" content="text/html; charset=UTF-8">\n\n    <!-- Bootstrap -->\n
<link href="/noindex/css/bootstrap.min.css" rel="stylesheet">\n    <link
rel="stylesheet" href="noindex/css/open-sans.css" type="text/css" />\n\n<style
type="text/css"><!--.. \n\nbody {\n  font-family: "Open Sans", Helvetica, sans-
serif;\n  font-weight: 100;\n  color: #ccc;\n  background: rgba(10, 24, 55, 1);\n
font-size: 16px;\n}\n\nh2, h3, h4 {\n  font-weight: 200;\n}\n\nh2 {\n  font-size:
28px;\n}\n\n.jumbotron {\n  margin-bottom: 0;\n  color: #333;\n  background:
rgb(212,212,221); /* Old browsers */\n  background: radial-gradient(ellipse at
center top, rgba(255,255,255,1) 0%,rgba(174,174,183,1) 100%); /* W3C
*/\n}\n\n.jumbotron h1 {\n  font-size: 128px;\n  font-weight: 700;\n  color:
white;\n  text-shadow: 0px 2px 0px #abc,\n               0px 4px 10px
rgba(0,0,0,0.15),\n               0px 5px 2px rgba(0,0,0,0.1),\n               0px
6px 30px rgba(0,0,0,0.1);\n}\n\n.jumbotron p {\n  font-size: 28px;\n  font-weight:
100;\n}\n\n.main {\n   background: white;\n   color: #234;\n   border-top: 1px solid
rgba(0,0,0,0.12);\n   padding-top: 30px;\n   padding-bottom: 40px;\n}\n\n.footer {\n
border-top: 1px solid rgba(255,255,255,0.2);\n   padding-top: 30px;\n}\n\n    -->
</style>\n</head>\n<body>\n  <div class="jumbotron text-center">\n    <div
class="container">\n   .  <h1>Testing 123..</h1>\n  ..<p class="lead">This page is
used to test the proper operation of the <a href="http://apache.org">Apache HTTP
server</a> after it has been installed. If you can read this page it means that this
site is working properly. This server is powered by <a
href="http://centos.org">CentOS</a>.</p>\n..</div>\n  </div>\n  <div class="main">\n
<div class="container">\n       <div class="row">\n  ...<div class="col-sm-6">\n
...<h2>Just visiting?</h2>\n...  ..<p class="lead">The website you just visited is
```

```
for col in list(cat_col.index):
    print(f"--------------------{col.title()}-------------------------")
    total= cyber_data[col].value_counts()
    percent = cyber_data[col].value_counts() / cyber_data.shape[0]
    df = pd.concat([total,percent],keys = ['total','percent'],axis = 1)
    print(df)
    print('\n')
```

```
142.215.174.2        1  0.00002
206.136.40.227       1  0.00002
211.126.163.217      1  0.00002
113.192.4.211        1  0.00002

[49998 rows x 2 columns]


--------------------D_Ip-------------------------
                total  percent
61.246.2.210        2  0.00004
144.82.144.26       1  0.00002
50.54.50.248        1  0.00002
153.250.66.166      1  0.00002
71.223.88.77        1  0.00002
...               ...      ...
70.14.234.118       1  0.00002
64.240.6.187        1  0.00002
40.156.106.17       1  0.00002
153.34.143.130      1  0.00002
95.42.130.186       1  0.00002

[49999 rows x 2 columns]


--------------------Protocol-------------------------
           total  percent
TCP        25288  0.50576
TCP( 6 )   24656  0.49312
UDP( 17 )     52  0.00104
UDP            4  0.00008


--------------------Payload-------------------------
                                                    total  percent
un_payload                                          18554  0.37108
POST /wsman?PSVersion=7.1.3 HTTP/1.1\r\nConnect...     90  0.00180
HTTP/1.1 403 Forbidden\r\nDate: Thu, 28 Jul 202...     90  0.00180

HTTP/1.1 403 Forbidden\r\nDate: Thu, 28 Jul 202...     86  0.00172
HTTP/1.1 403 Forbidden\r\nDate: Thu, 28 Jul 202...     85  0.00170
...                                                   ...      ...
GET /?id=-4835%60%29%20WHERE%203564%3D3564%20UN...      1  0.00002
GET /vulnerabilities/sqli/?id=1%27%3BSELECT%20P...      1  0.00002
GET /?id=user%20UNION%20ALL%20SELECT%20NULL%2CN...      1  0.00002
GET /manager/html/upload?org.apache.catalina.fi...      1  0.00002
GET /?id=user%29%29%20UNION%20ALL%20SELECT%20NU...      1  0.00002

[16137 rows x 2 columns]
```

```
[16127 rows x 2 columns]


--------------------Label-------------------------
                  total  percent
2_exploit         19690  0.39380
4_unknown         16089  0.32178
3_post            11718  0.23436
1_reconnaissance   2503  0.05006
```

```
source_ip = cyber_data['s_ip'].nunique()
destination_ip = cyber_data['d_ip'].nunique()
source_port = cyber_data['s_port'].nunique()
dest_port = cyber_data['d_port'].nunique()
```

```
print(f'unique s_ip: {source_ip}, Unique Dest_ip: {destination_ip}, Source_Port: {source_port
```

```
unique s_ip: 49998, Unique Dest_ip: 49999, Source_Port: 9176, Destination_port: 13067
```

```
# cyber_data.info()
```

```
cyber_data = cyber_data.drop(columns=['s_ip','d_ip'],axis=1)
```

```
cyber_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   s_port      50000 non-null  int64 
 1   d_port      50000 non-null  int64 
 2   protocol    50000 non-null  object
 3   payload     50000 non-null  object
 4   risk        50000 non-null  int64 
 5   payload_LE  50000 non-null  int64 
 6   label       50000 non-null  object
dtypes: int64(4), object(3)
memory usage: 2.7+ MB
```

```
cyber_data['label'].value_counts()
```

```
2_exploit           19690
4_unknown           16089
3_post              11718
1_reconnaissance     2503
Name: label, dtype: int64
```

```
cyber_data['protocol'].value_counts()
```

```
TCP          25288
TCP( 6 )     24656
UDP( 17 )       52
UDP              4
Name: protocol, dtype: int64
```

```
cyber_data['payload'].value_counts()
```

```
serif;\n  font-weight: 100;\n  color: #ccc;\n  background: rgba(10, 24, 55, 1);\n
font-size: 16px;\n}\n\nh2, h3, h4 {\n  font-weight: 200;\n}\n\nh2 {\n  font-size:
28px;\n}\n\n.jumbotron {\n  margin-bottom: 0;\n  color: #333;\n  background:
rgb(212,212,221); /* Old browsers */\n  background: radial-gradient(ellipse at
center top, rgba(255,255,255,1) 0%,rgba(174,174,183,1) 100%); /* W3C
*/\n}\n\n.jumbotron h1 {\n  font-size: 128px;\n  font-weight: 700;\n  color:
white;\n  text-shadow: 0px 2px 0px #abc,\n               0px 4px 10px
rgba(0,0,0,0.15),\n               0px 5px 2px rgba(0,0,0,0.1),\n               0px
6px 30px rgba(0,0,0,0.1);\n}\n\n.jumbotron p {\n  font-size: 28px;\n  font-weight:
100;\n}\n\n.main {\n   background: white;\n   color: #234;\n   border-top: 1px solid
rgba(0,0,0,0.12);\n   padding-top: 30px;\n   padding-bottom: 40px;\n}\n\n.footer {\n
border-top: 1px solid rgba(255,255,255,0.2);\n   padding-top: 30px;\n}\n\n    -->
</style>\n</head>\n<body>\n  <div class="jumbotron text-center">\n    <div
class="container">\n   .  <h1>Testing 123..</h1>\n  ..<p class="lead">This page is
used to test the proper operation of the <a href="http://apache.org">Apache HTTP
server</a> after it has been installed. If you can read this page it means that this
site is working properly. This server is powered by <a
href="http://centos.org">CentOS</a>.</p>\n..</div>\n  </div>\n  <div class="main">\n
<div class="container">\n       <div class="row">\n  ...<div class="col-sm-6">\n
...<h2>Just visiting?</h2>\n...  ..<p class="lead">The website you just visited is
either experiencing problems or is undergoing routine maintenance.</p>\n  .....<p>If
you would like to let the administrators of this website know that you've seen this
page instead of the page you expected, you should send them e-mail. In general, mail
sent to the name "webmaster" and directed to the website's domain should reach the
appropriate person.</p>\n  .....<p>For example, if you experienced problems while
visiting www.example.com, you should send e-mail to "webmaster@example.com".</p>\n.
...</div>\n  ....<div class="col-s       86
HTTP/1.1 403 Forbidden\r\nDate: Thu, 28 Jul 2022 08:20:39 GMT\r\nServer:
Apache/2.4.6 (CentOS)\r\nLast-Modified: Thu, 16 Oct 2014 13:20:58 GMT\r\nETag:
"1321-5058a1e728280"\r\nAccept-Ranges: bytes\r\nContent-Length: 4897\r\nConnection:
close\r\nContent-Type: text/html; charset=UTF-8\r\n\r\n<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"><html>
<head>\n<meta http-equiv="content-type" content="text/html; charset=UTF-8">\n..
<title>Apache HTTP Server Test Page powered by CentOS</title>\n..<meta http-
equiv="Content-Type" content="text/html; charset=UTF-8">\n\n    <!-- Bootstrap -->\n
<link href="/noindex/css/bootstrap.min.css" rel="stylesheet">\n    <link
rel="stylesheet" href="noindex/css/open-sans.css" type="text/css" />\n\n<style
type="text/css"><!--.. \n\nbody {\n  font-family: "Open Sans", Helvetica, sans-
serif;\n  font-weight: 100;\n  color: #ccc;\n  background: rgba(10, 24, 55, 1);\n
font-size: 16px;\n}\n\nh2, h3, h4 {\n  font-weight: 200;\n}\n\nh2 {\n  font-size:
28px;\n}\n\n.jumbotron {\n  margin-bottom: 0;\n  color: #333;\n  background:
rgb(212,212,221); /* Old browsers */\n  background: radial-gradient(ellipse at
center top, rgba(255,255,255,1) 0%,rgba(174,174,183,1) 100%); /* W3C
*/\n}\n\n.jumbotron h1 {\n  font-size: 128px;\n  font-weight: 700;\n  color:
white;\n  text-shadow: 0px 2px 0px #abc,\n               0px 4px 10px
```

```
rgba(0,0,0,0.15),\n                0px 5px 2px rgba(0,0,0,0.1),\n                0px
6px 30px rgba(0,0,0,0.1);\n}\n\n.jumbotron p {\n  font-size: 28px;\n  font-weight:
100;\n}\n\n.main {\n   background: white;\n   color: #234;\n   border-top: 1px solid
rgba(0,0,0,0.12);\n   padding-top: 30px;\n   padding-bottom: 40px;\n}\n\n.footer {\n
border-top: 1px solid rgba(255,255,255,0.2);\n   padding-top: 30px;\n}\n\n    -->
</style>\n</head>\n<body>\n  <div class="jumbotron text-center">\n    <div
class="container">\n   .  <h1>Testing 123..</h1>\n  ..<p class="lead">This page is
used to test the proper operation of the <a href="http://apache.org">Apache HTTP
server</a> after it has been installed. If you can read this page it means that this
site is working properly. This server is powered by <a
href="http://centos.org">CentOS</a>.</p>\n..</div>\n  </div>\n  <div class="main">\n
<div class="container">\n       <div class="row">\n  ...<div class="col-sm-6">\n
```

```
# xtra = []
# for i in cyber_data['payload']:

#   # cyber_data['payload'] = Le.fit_transform(cyber_data['payload'])
```

```
cyber_data['label'] = Le.fit_transform(cyber_data['label'])
cyber_data['protocol'] = Le.fit_transform(cyber_data['protocol'])
# cyber_data['payload'] = Le.fit_transform(cyber_data['payload'])
```

```
cyber_data['label'].value_counts()
```

```
1    19690
3    16089
2    11718
0     2503
Name: label, dtype: int64
```

```
cyber_data['protocol'].value_counts()
```

```
0    25288
1    24656
3       52
2        4
Name: protocol, dtype: int64
```

```
cyber_data['risk'].value_counts()
```

```
2    26209
1    18169
```

```
3     5622
```

```
cyber_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   s_port      50000 non-null  int64
 1   d_port      50000 non-null  int64
 2   protocol    50000 non-null  int64
 3   payload     50000 non-null  object
 4   risk        50000 non-null  int64
 5   payload_LE  50000 non-null  int64
 6   label       50000 non-null  int64
dtypes: int64(6), object(1)
memory usage: 2.7+ MB
```

```
feature_cols = ['s_port', 'd_port', 'protocol','payload_LE']
X = cyber_data[feature_cols] # Features
y = cyber_data.label # Target variable
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=1)
```

```
# print(X_train)
# print(y_train)
```

Logistic Regression

```
# logreg = LogisticRegression(random_state=16)
# logreg.fit(X_train, y_train)

# y_pred = logreg.predict(X_test)
# print(y_test)
# print("Predicted")
# print(y_pred)
```

```
# cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
# cnf_matrix

# from sklearn.metrics import classification_report
# # target_names = ['0', '1']
# print(classification_report(y_test, y_pred))
```

```
# from sklearn.metrics import accuracy_score
```

```
# predictions= logreg.predict(X_test)
# ac = accuracy_score(y_test,predictions)
# print(ac)



# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=1)
# logreg = LogisticRegression(random_state=16)
# logreg.fit(X_train, y_train)

# y_pred = logreg.predict(X_test)
# print(y_test)
# print("Predicted")
# print(y_pred)

# cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
# cnf_matrix

# from sklearn.metrics import classification_report
# # target_names = ['0', '1']
# print(classification_report(y_test, y_pred))

# predictions= logreg.predict(X_test)
# ac = accuracy_score(y_test,predictions)
# print(ac)


from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.metrics import silhouette_score
import scipy.cluster.hierarchy as shc
from sklearn.cluster import AgglomerativeClustering
from sklearn.decomposition import PCA
from scipy.cluster.hierarchy import dendrogram, linkage
```

KNN MODEL By elbow method

```
clf = KNeighborsClassifier(n_neighbors=3)
# clf.fit(features, label)
clf.fit(X_train,y_train)
clf.score(X_test,y_test)
```

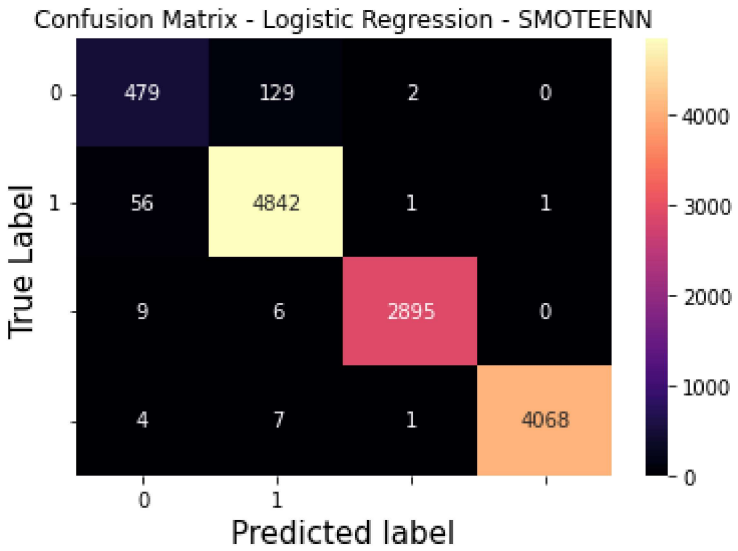
```
0.98272
```

```
print(y_test)
y_predict = clf.predict(X_test)
print(y_predict)
```

```
26247    2
```

```
35067    1
34590    3
16668    1
12196    2
        ..
42763    3
26781    1
28306    3
7425     1
45705    3
Name: label, Length: 12500, dtype: int64
[2 1 3 ... 3 1 3]
```

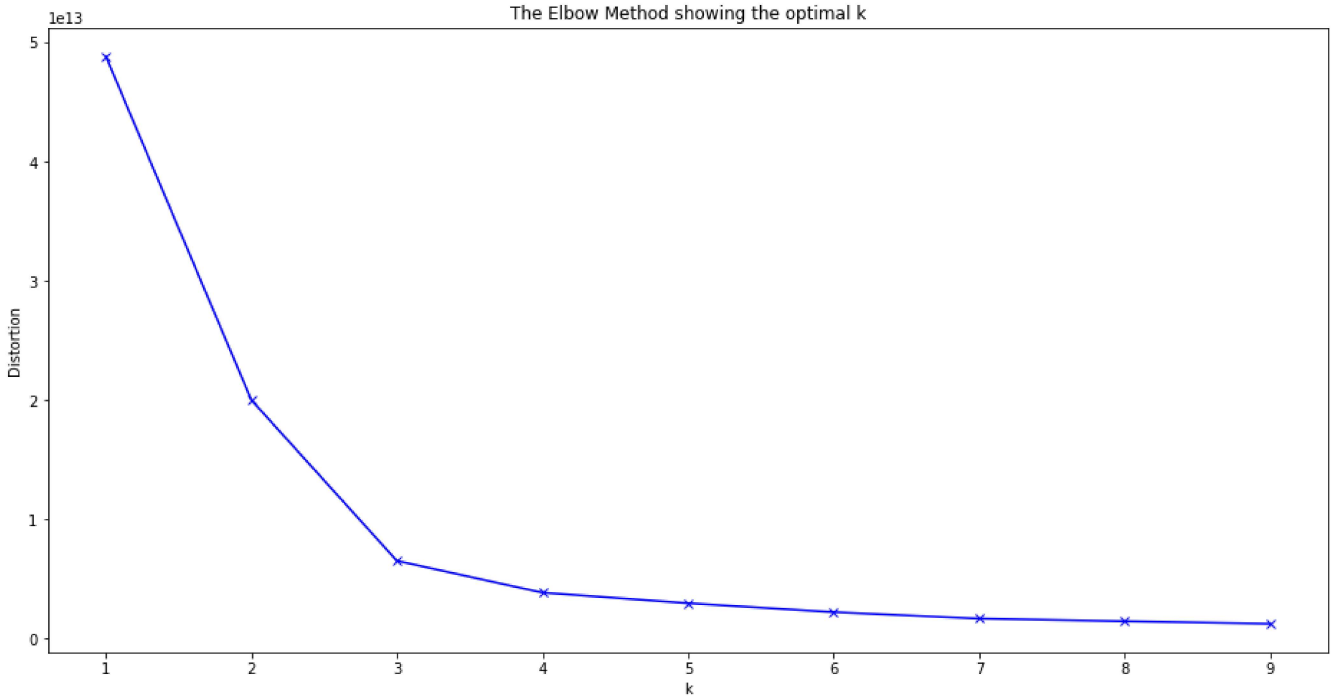
```
cnf_matrix = metrics.confusion_matrix(y_test, y_predict)
cnf_matrix
matrix_df = pd.DataFrame(cnf_matrix)

ax = plt.axes()
sns.heatmap(matrix_df, annot=True, fmt="g", ax=ax, cmap="magma")
ax.set_title('Confusion Matrix - Logistic Regression - SMOTEENN')
ax.set_xlabel("Predicted label", fontsize =15)
ax.set_xticklabels(['0', '1'])
ax.set_ylabel("True Label", fontsize=15)
ax.set_yticklabels(['0', '1'], rotation = 0)
plt.show()
```




```
cyber_data = cyber_data.drop(columns=['payload'],axis=1)
```

```
distortions = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(cyber_data)
    distortions.append(kmeanModel.inertia_)
```

```
plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```




```
clf = KNeighborsClassifier(n_neighbors=2)
# clf.fit(features, label)
clf.fit(X_train,y_train)
clf.score(X_test,y_test)
```

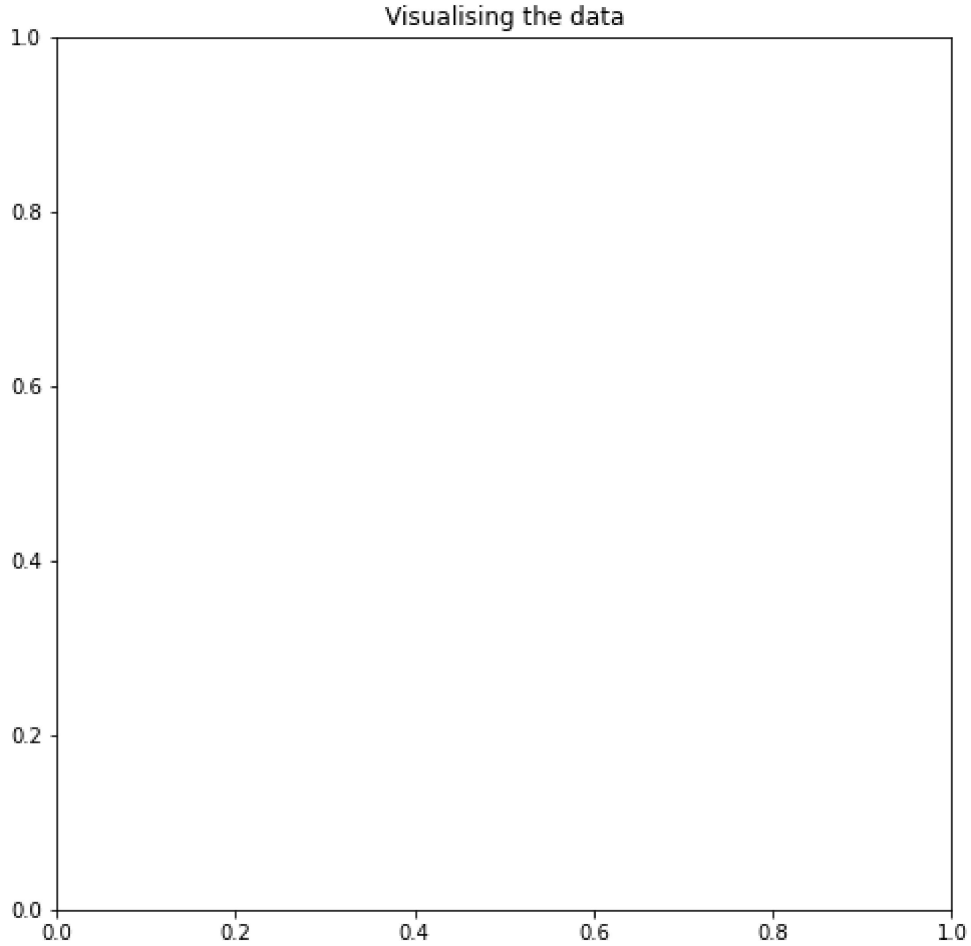
```
0.98144
```

Silhouette Method

```
scaler = StandardScaler()     # Scaling the data so that all the features become comparable
X_scaled = scaler.fit_transform(X)

X_normalized = normalize(X_scaled)
```

```
X_normalized = pd.DataFrame(X_normalized)

pca = PCA(n_components = 2)
X_principal = pca.fit_transform(X_normalized)
X_principal = pd.DataFrame(X_principal)
X_principal.columns = ['P1', 'P2']
plt.figure(figsize =(8, 8))
plt.title('Visualising the data')
# Dendrogram = shc.dendrogram((shc.linkage(X_principal, method ='ward')))
```

```
Text(0.5, 1.0, 'Visualising the data')
```




```
ac2 = AgglomerativeClustering(n_clusters = 2)
ac3 = AgglomerativeClustering(n_clusters = 3)
ac4 = AgglomerativeClustering(n_clusters = 4)
ac5 = AgglomerativeClustering(n_clusters = 5)
ac6 = AgglomerativeClustering(n_clusters = 6)
k = [2, 3, 4, 5, 6]

# Appending the silhouette scores of the different models to the list
silhouette_scores = []
silhouette_scores.append(
        silhouette_score(X_principal, ac2.fit_predict(X_principal)))
silhouette_scores.append(
        silhouette_score(X_principal, ac3.fit_predict(X_principal)))
```

```
    silhouette_scores.append(
            silhouette_score(X_principal, ac4.fit_predict(X_principal)))
    silhouette_scores.append(
            silhouette_score(X_principal, ac5.fit_predict(X_principal)))
    silhouette_scores.append(
            silhouette_score(X_principal, ac6.fit_predict(X_principal)))

    # Plotting a bar graph to compare the results
    plt.bar(k, silhouette_scores)
    plt.xlabel('Number of clusters', fontsize = 20)
    plt.ylabel('S(i)', fontsize = 20)
    plt.show()
```

```
    from typing import Tuple
    import matplotlib.pyplot as plt
    import seaborn as sns
    import pandas as pd

    def show_correlation_graph(cyber_data: pd.DataFrame, figsize: Tuple):
        plt.figure(figsize=figsize)
        corrmat = cyber_data.corr()
        top_corr_features = corrmat.index
        sns.heatmap(cyber_data[top_corr_features].corr(), annot=True, cmap="RdYlGn")
        plt.show()

    show_correlation_graph(df,(10,10))
```

Colab paid products - Cancel contracts here

12s completed at 02:19