In [1]:
```
# Header file
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
# Read the file using pandas dataframe

df = pd.read_csv("DataSet.csv")
#Display the top 5 dataset from file
df.head()
```

Out[2]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urge
0	0	tcp	private	REJ	0	0	0	0	
1	0	tcp	private	REJ	0	0	0	0	
2	2	tcp	ftp_data	SF	12983	0	0	0	
3	0	icmp	eco_i	SF	20	0	0	0	
4	1	tcp	telnet	RSTO	0	15	0	0	

5 rows × 42 columns

In [3]:
```
# Data Description
df.describe()
```

Out[3]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urge
count	22544.000000	2.254400e+04	2.254400e+04	22544.000000	22544.000000	22544.0000
mean	218.859076	1.039545e+04	2.056019e+03	0.000311	0.008428	0.0007
std	1407.176612	4.727864e+05	2.121930e+04	0.017619	0.142599	0.0364
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.0000
25%	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.0000
50%	0.000000	5.400000e+01	4.600000e+01	0.000000	0.000000	0.0000
75%	0.000000	2.870000e+02	6.010000e+02	0.000000	0.000000	0.0000
max	57715.000000	6.282565e+07	1.345927e+06	1.000000	3.000000	3.0000

8 rows × 38 columns

In [4]:
```
row,col = df.shape
unique = list(df['protocol_type'].unique())
for i in range(row):
    index = unique.index(df.iloc[i,1])
    df.iloc[i,1] = index
unique = list(df['service'].unique())
for i in range(row):
    index = unique.index(df.iloc[i,2])
    df.iloc[i,2] = index
unique = list(df['flag'].unique())
for i in range(row):
    index = unique.index(df.iloc[i,3])
    df.iloc[i,3] = index
```

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

Out[5]:

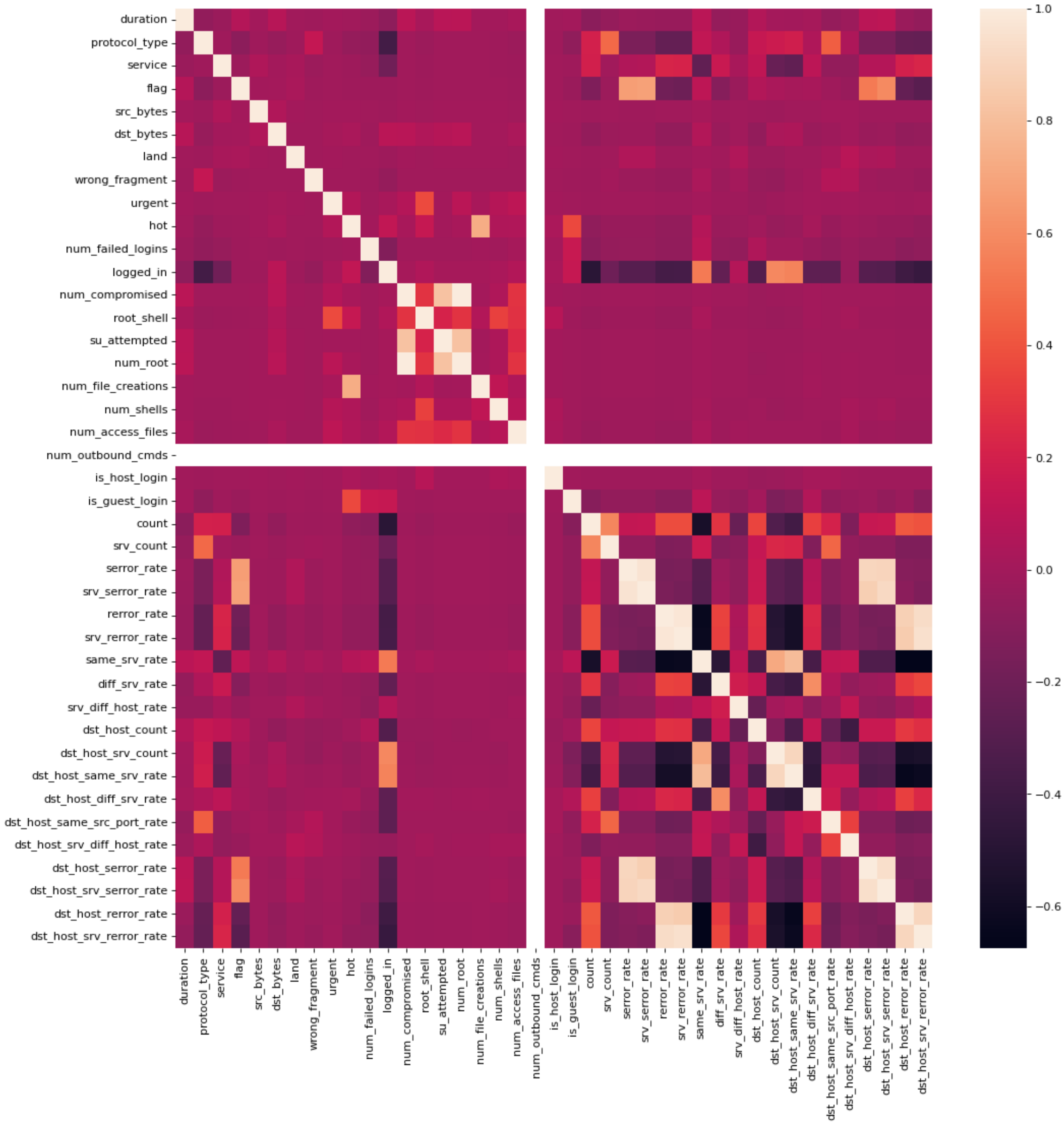
	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	2	0	1	1	12983	0	0	0	0
3	0	1	2	1	20	0	0	0	0
4	1	0	3	2	0	15	0	0	0

5 rows × 42 columns

In [6]:
```
# Data Dimention
row,col= df.shape
print("Dimention of data is: row= ", row, ' column=', col)
# Columns Description
print(df.columns)
```

```
Dimention of data is: row=  22544  column= 42
Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
       'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
       'num_failed_logins', 'logged_in', 'num_compromised', 'root_s
hell',
       'su_attempted', 'num_root', 'num_file_creations', 'num_shell
s',
       'num_access_files', 'num_outbound_cmds', 'is_host_login',
       'is_guest_login', 'count', 'srv_count', 'serror_rate',
       'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_s
rv_rate',
       'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
       'dst_host_srv_count', 'dst_host_same_srv_rate',
       'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
       'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
       'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
       'dst_host_srv_rerror_rate', 'level'],
      dtype='object')
```

In [18]:
```
plt.figure(figsize=(15, 15), dpi=80)
df1 = df.drop(["level"],axis=1)
sns.heatmap(df1.corr())
plt.savefig("test2.png")
```



In [7]:
```

import random
All_index = [i for i in range(row)]
tranning_index = random.sample(range(1,row),int(row*0.8))
tranning_index.sort()
text_index = list(set(All_index) - set(tranning_index))
```

In [8]:
```
#tranning test split
df_features = df.drop(["level"],axis=1)
df_label = df["level"]
```

In [9]:

```
Tranning_features = df_features.iloc[tranning_index,:]
Tranning_label = df_label[tranning_index]
Test_features = df_features.iloc[text_index,:]
Test_label = df_label[text_index]


# use decision tree algorithm for better prediction
from sklearn import tree
from sklearn import metrics

clf = tree.DecisionTreeClassifier()
clf = clf.fit(Tranning_features,Tranning_label)
Test_predict_label = clf.predict(Test_features)
Confusion_matrix = metrics.accuracy_score(Test_label,Test_predict_lab
Recall = metrics.recall_score(Test_label,Test_predict_label,average=
Precision = metrics.precision_score(Test_label,Test_predict_label,ave
F1_score= (2*Precision*Recall)/(Precision+Recall)
print("Accuracy of the model is: ",Confusion_matrix)
print("Recall of the model is: ",Recall)
print("Precision of the model is: ",Precision)
print("F1 score of the model is: ",F1_score)
```

```
Accuracy of the model is:  0.9864715014415614
Recall of the model is:  0.9859646970081584
Precision of the model is:  0.986409159819235
F1 score of the model is:  0.9861868783351607
```

In [10]:

```
# use Logistic Regression for better prediction
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

clf = LogisticRegression(random_state=0)
clf = clf.fit(Tranning_features,Tranning_label)
Test_predict_label = clf.predict(Test_features)
Confusion_matrix = metrics.accuracy_score(Test_label,Test_predict_lab
Recall = metrics.recall_score(Test_label,Test_predict_label,average=
Precision = metrics.precision_score(Test_label,Test_predict_label,ave
F1_score= (2*Precision*Recall)/(Precision+Recall)
print("Accuracy of the model is: ",Confusion_matrix)
print("Recall of the model is: ",Recall)
print("Precision of the model is: ",Precision)
print("F1 score of the model is: ",F1_score)
```

```
Accuracy of the model is:  0.7997338656021291
Recall of the model is:  0.8107174587822488
Precision of the model is:  0.8057073147792122
F1 score of the model is:  0.8082046222551733

/home/sankar/.local/lib/python3.10/site-packages/sklearn/linear_mod
el/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as s
hown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (htt
ps://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver optio
ns:
    https://scikit-learn.org/stable/modules/linear_model.html#logis
tic-regression (https://scikit-learn.org/stable/modules/linear_mode
l.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [ ]:

```
# use SVM classifier for better prediction
from sklearn import svm
from sklearn import metrics

clf = svm.SVC(kernel='linear')
#Updated_data = np.dot(Tranning_features,Tranning_features.T)
clf = clf.fit(Tranning_features,Tranning_label)
Test_predict_label = clf.predict(Test_features)
Confusion_matrix = metrics.accuracy_score(Test_label,Test_predict_lab
Recall = metrics.recall_score(Test_label,Test_predict_label,average=
Precision = metrics.precision_score(Test_label,Test_predict_label,ave
F1_score= (2*Precision*Recall)/(Precision+Recall)
print("Accuracy of the model is: ",Confusion_matrix)
print("Recall of the model is: ",Recall)
print("Precision of the model is: ",Precision)
print("F1 score of the model is: ",F1_score)
```

In [11]:

```
# use K-nearest neighbour for better prediction
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

clf = KNeighborsClassifier (n_neighbors=5,weights="distance")
clf = clf.fit(Tranning_features,Tranning_label)
Test_predict_label = clf.predict(Test_features)
Confusion_matrix = metrics.accuracy_score(Test_label,Test_predict_lab
Recall = metrics.recall_score(Test_label,Test_predict_label,average=
Precision = metrics.precision_score(Test_label,Test_predict_label,ave
F1_score= (2*Precision*Recall)/(Precision+Recall)
print("Accuracy of the model is: ",Confusion_matrix)
print("Recall of the model is: ",Recall)
print("Precision of the model is: ",Precision)
print("F1 score of the model is: ",F1_score)
```

```
Accuracy of the model is:  0.9773785761809713
Recall of the model is:  0.9766507384010308
Precision of the model is:  0.9771497929315267
F1 score of the model is:  0.9769002019301346
```

```
In [12]: label = list(Tranning_label.unique())
         print(label)
         for i in range(len(Tranning_label)):
             print(i)
             Tranning_label[i] = label.index(Tranning_label[i])
         print(Tranning_label.head())
         # use Linear regression for better prediction
         from sklearn.linear_model import LinearRegression
         from sklearn import metrics
         clf = LinearRegression()
         clf = clf.fit(Tranning_features,Tranning_label)
         Test_predict_label = clf.predict(Test_features)
         Confusion_matrix = metrics.accuracy_score(Test_label,Test_predict_lab
         Recall = metrics.recall_score(Test_label,Test_predict_label,average=
         Precision = metrics.precision_score(Test_label,Test_predict_label,ave
         F1_score= (2*Precision*Recall)/(Precision+Recall)
         print("Accuracy of the model is: ",Confusion_matrix)
         print("Recall of the model is: ",Recall)
         print("Precision of the model is: ",Precision)
         print("F1 score of the model is: ",F1_score)
```

```
['anomaly', 'normal']
0

---------------------------------------------------------------------
--------
KeyError                                  Traceback (most recent ca
ll last)
File ~/.local/lib/python3.10/site-packages/pandas/core/indexes/bas
e.py:3652, in Index.get_loc(self, key)
   3651 try:
-> 3652     return self._engine.get_loc(casted_key)
   3653 except KeyError as err:

File ~/.local/lib/python3.10/site-packages/pandas/_libs/index.pyx:1
47, in pandas._libs.index.IndexEngine.get_loc()

File ~/.local/lib/python3.10/site-packages/pandas/_libs/index.pyx:1
76, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:2606, in pandas._libs.
hashtable.Int64HashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:2630, in pandas._libs.
hashtable.Int64HashTable.get_item()

KeyError: 0

The above exception was the direct cause of the following exceptio
n:

KeyError                                  Traceback (most recent ca
ll last)
Cell In[12], line 5
      3 for i in range(len(Tranning_label)):
      4     print(i)
----> 5     Tranning_label[i] = label.index(Tranning_label[i])
      6 print(Tranning_label.head())
      7 # use Linear regression for better prediction
```

```
File ~/.local/lib/python3.10/site-packages/pandas/core/series.py:10
07, in Series.__getitem__(self, key)
   1004     return self._values[key]
   1006 elif key_is_scalar:
-> 1007     return self._get_value(key)
   1009 if is_hashable(key):
   1010     # Otherwise index.get_value will raise InvalidIndexErro
r
   1011     try:
   1012         # For labels that don't resolve as scalars like tup
les and frozensets

File ~/.local/lib/python3.10/site-packages/pandas/core/series.py:11
16, in Series._get_value(self, label, takeable)
   1113     return self._values[label]
   1115 # Similar to Index.get_value, but we do not fall back to po
sitional
-> 1116 loc = self.index.get_loc(label)
   1118 if is_integer(loc):
   1119     return self._values[loc]

File ~/.local/lib/python3.10/site-packages/pandas/core/indexes/bas
e.py:3654, in Index.get_loc(self, key)
   3652     return self._engine.get_loc(casted_key)
   3653 except KeyError as err:
-> 3654     raise KeyError(key) from err
   3655 except TypeError:
   3656     # If we have a listlike key, _check_indexing_error will
raise
   3657     #  InvalidIndexError. Otherwise we fall through and re-
raise
   3658     #  the TypeError.
   3659     self._check_indexing_error(key)

KeyError: 0
```

```
In [ ]:
```