In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv('network_security.txt', sep = "," , encoding = 'utf-8')
```

In [3]:

```
df.head()
```

Out[3]:

	0	tcp	ftp_data	SF	491	0.1	0.2	0.3	0.4	0.5	...	0.17	0.03	0.17.1	0.00.6	0.00.7
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	0.88	0.00	0.00
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	0.00	0.00	1.00
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	0.03	0.04	0.03
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	0.00	0.00	0.00
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	0.07	0.00	0.00	0.00

5 rows × 43 columns

In [4]:

```
df.tail()
```

Out[4]:

	0	tcp	ftp_data	SF	491	0.1	0.2	0.3	0.4	0.5	...	0.17	0.03	0.17.1	0.00.6	0.0
125967	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.06	0.00	0.0	
125968	8	udp	private	SF	105	145	0	0	0	0	...	0.96	0.01	0.01	0.0	
125969	0	tcp	smtp	SF	2231	384	0	0	0	0	...	0.12	0.06	0.00	0.0	
125970	0	tcp	klogin	S0	0	0	0	0	0	0	...	0.03	0.05	0.00	0.0	
125971	0	tcp	ftp_data	SF	151	0	0	0	0	0	...	0.30	0.03	0.30	0.0	

5 rows × 43 columns

In [5]:

```
df.shape
```

Out[5]:

```
(125972, 43)
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['0', 'tcp', 'ftp_data', 'SF', '491', '0.1', '0.2', '0.3', '0.4',
'0.5',
       '0.6', '0.7', '0.8', '0.9', '0.10', '0.11', '0.12', '0.13', '0.14',
       '0.15', '0.16', '0.18', '2', '2.1', '0.00', '0.00.1', '0.00.2',
       '0.00.3', '1.00', '0.00.4', '0.00.5', '150', '25', '0.17', '0.03',
       '0.17.1', '0.00.6', '0.00.7', '0.00.8', '0.05', '0.00.9', 'normal',
       '20'],
      dtype='object')
```

In [7]:

```
df.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

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

Out[8]:

```
0           0
tcp         0
ftp_data    0
SF          0
491         0
0.1         0
0.2         0
0.3         0
0.4         0
0.5         0
0.6         0
0.7         0
0.8         0
0.9         0
0.10        0
0.11        0
0.12        0
0.13        0
0.14        0
0.15        0
0.16        0
0.18        0
2           0
2.1         0
0.00        0
0.00.1      0
0.00.2      0
0.00.3      0
1.00        0
0.00.4      0
0.00.5      0
150         0
25          0
0.17        0
0.03        0
0.17.1      0
0.00.6      0
0.00.7      0
0.00.8      0
0.05        0
0.00.9      0
normal      0
20          0
dtype: int64
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 125972 entries, 0 to 125971
Data columns (total 43 columns):
 #   Column    Non-Null Count   Dtype  
---  ------    --------------   -----  
 0   0         125972 non-null  int64  
 1   tcp       125972 non-null  object 
 2   ftp_data  125972 non-null  object 
 3   SF        125972 non-null  object 
 4   491       125972 non-null  int64  
 5   0.1       125972 non-null  int64  
 6   0.2       125972 non-null  int64  
 7   0.3       125972 non-null  int64  
 8   0.4       125972 non-null  int64  
 9   0.5       125972 non-null  int64  
 10  0.6       125972 non-null  int64  
 11  0.7       125972 non-null  int64  
 12  0.8       125972 non-null  int64  
 13  0.9       125972 non-null  int64  
 14  0.10      125972 non-null  int64  
 15  0.11      125972 non-null  int64  
 16  0.12      125972 non-null  int64  
 17  0.13      125972 non-null  int64  
 18  0.14      125972 non-null  int64  
 19  0.15      125972 non-null  int64  
 20  0.16      125972 non-null  int64  
 21  0.18      125972 non-null  int64  
 22  2         125972 non-null  int64  
 23  2.1       125972 non-null  int64  
 24  0.00      125972 non-null  float64
 25  0.00.1    125972 non-null  float64
 26  0.00.2    125972 non-null  float64
 27  0.00.3    125972 non-null  float64
 28  1.00      125972 non-null  float64
 29  0.00.4    125972 non-null  float64
 30  0.00.5    125972 non-null  float64
 31  150       125972 non-null  int64  
 32  25        125972 non-null  int64  
 33  0.17      125972 non-null  float64
 34  0.03      125972 non-null  float64
 35  0.17.1    125972 non-null  float64
 36  0.00.6    125972 non-null  float64
 37  0.00.7    125972 non-null  float64
 38  0.00.8    125972 non-null  float64
 39  0.05      125972 non-null  float64
 40  0.00.9    125972 non-null  float64
 41  normal    125972 non-null  object 
 42  20        125972 non-null  int64  
dtypes: float64(15), int64(24), object(4)
memory usage: 41.3+ MB
```

In [10]:

```
df.describe()
```

Out[10]:

	0	491	0.1	0.2	0.3	
count	125972.000000	1.259720e+05	1.259720e+05	125972.000000	125972.000000	125972.000
mean	287.146929	4.556710e+04	1.977927e+04	0.000198	0.022688	0.000
std	2604.525522	5.870354e+06	4.021285e+06	0.014086	0.253531	0.014
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000
25%	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000
50%	0.000000	4.400000e+01	0.000000e+00	0.000000	0.000000	0.000
75%	0.000000	2.760000e+02	5.160000e+02	0.000000	0.000000	0.000
max	42908.000000	1.379964e+09	1.309937e+09	1.000000	3.000000	3.000

8 rows × 39 columns

In [11]:

```
df.nunique()
```

Out[11]:

```
0            2981
tcp             3
ftp_data       70
SF             11
491          3341
0.1          9326
0.2             2
0.3             3
0.4             4
0.5            28
0.6             6
0.7             2
0.8            88
0.9             2
0.10            3
0.11           82
0.12           35
0.13            3
0.14           10
0.15            1
0.16            2
0.18            2
2             512
2.1           509
0.00           89
0.00.1         86
0.00.2         82
0.00.3         62
1.00          101
0.00.4         95
0.00.5         60
150           256
25            256
0.17          101
0.03          101
0.17.1        101
0.00.6         75
0.00.7        101
0.00.8        100
0.05          101
0.00.9        101
normal         23
20             22
dtype: int64
```

In [12]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [13]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [14]:

```
df['tcp'].unique()
```

Out[14]:

```
array(['udp', 'tcp', 'icmp'], dtype=object)
```

In [15]:

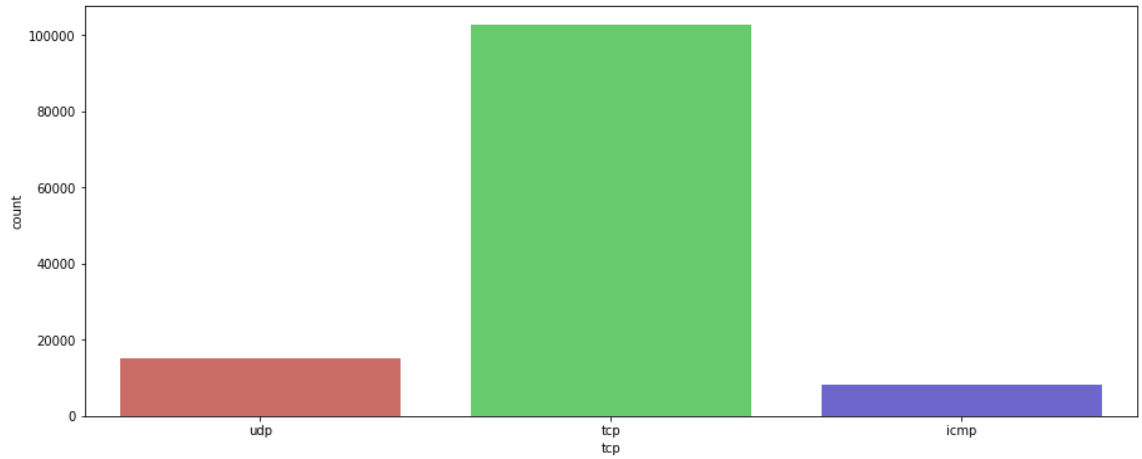
```
df['tcp'].value_counts()
```

Out[15]:

```
tcp     102688
udp      14993
icmp      8291
Name: tcp, dtype: int64
```

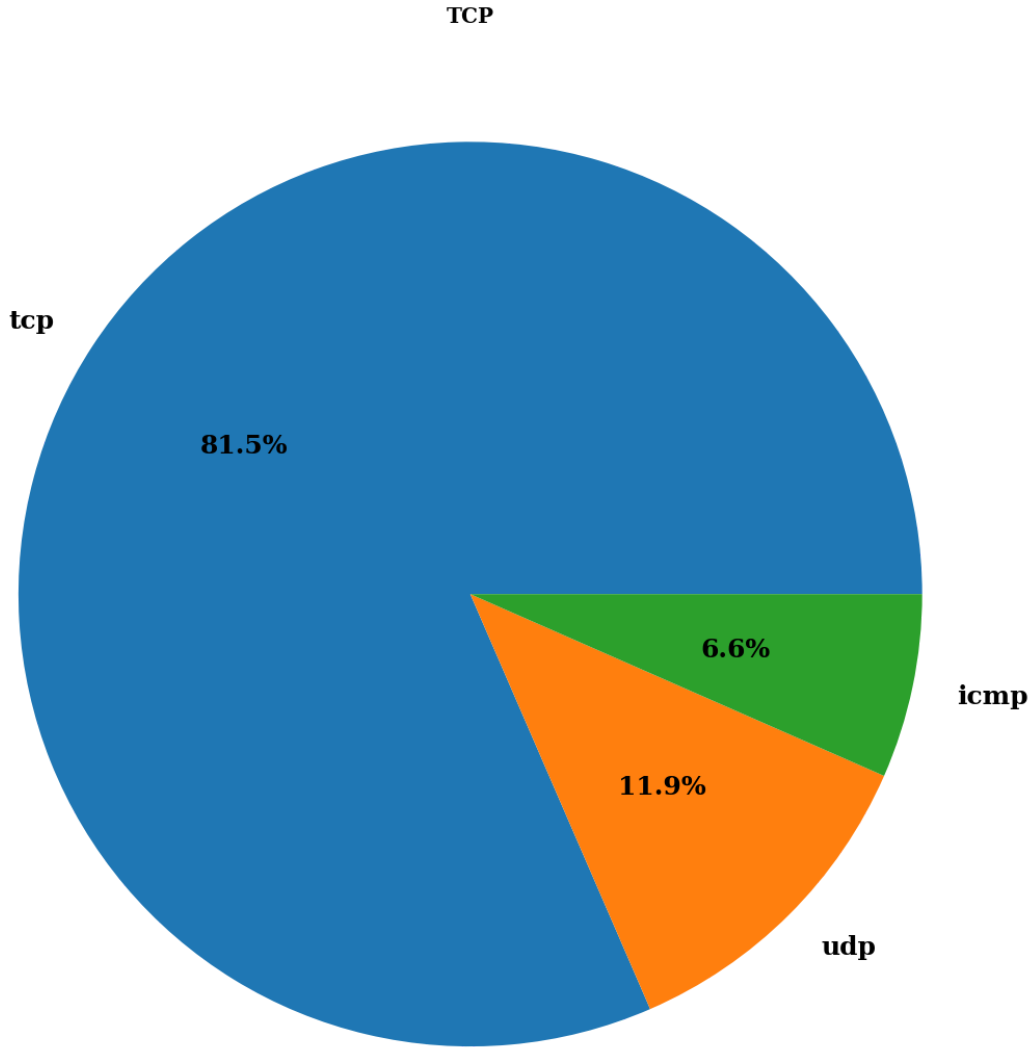
In [16]:

```
plt.figure(figsize=(15,6))
sns.countplot(df['tcp'], data = df, palette = 'hls')
plt.show()
```



In [17]:

```
plt.figure(figsize=(30,20))
plt.pie(df['tcp'].value_counts(), labels=df['tcp'].value_counts().index, autopct='%1.1f%
                                            'color': 'black',
                                            'weight': 'bold',
                                            'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('TCP', size=20, **hfont)
plt.show()
```



In [18]:

```
df['ftp_data'].unique()
```

Out[18]:

```
array(['other', 'private', 'http', 'remote_job', 'ftp_data', 'name',
       'netbios_ns', 'eco_i', 'mtp', 'telnet', 'finger', 'domain_u',
       'supdup', 'uucp_path', 'Z39_50', 'smtp', 'csnet_ns', 'uucp',
       'netbios_dgm', 'urp_i', 'auth', 'domain', 'ftp', 'bgp', 'ldap',
       'ecr_i', 'gopher', 'vmnet', 'systat', 'http_443', 'efs', 'whois',
       'imap4', 'iso_tsap', 'echo', 'klogin', 'link', 'sunrpc', 'login',
       'kshell', 'sql_net', 'time', 'hostnames', 'exec', 'ntp_u',
       'discard', 'nntp', 'courier', 'ctf', 'ssh', 'daytime', 'shell',
       'netstat', 'pop_3', 'nnsp', 'IRC', 'pop_2', 'printer', 'tim_i',
       'pm_dump', 'red_i', 'netbios_ssn', 'rje', 'X11', 'urh_i',
       'http_8001', 'aol', 'http_2784', 'tftp_u', 'harvest'], dtype=objec
t)
```

In [19]:

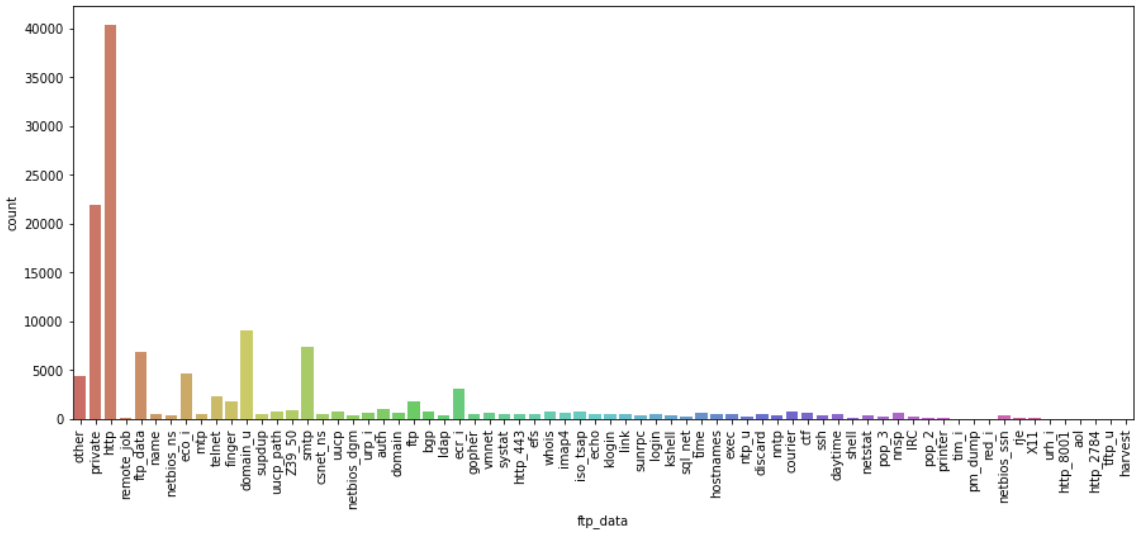
```
df['ftp_data'].value_counts()
```

Out[19]:

```
http         40338
private      21853
domain_u      9043
smtp          7313
ftp_data      6859
             ...
tftp_u           3
http_8001        2
aol              2
harvest          2
http_2784        1
Name: ftp_data, Length: 70, dtype: int64
```

In [20]:

```
plt.figure(figsize=(15,6))
sns.countplot(df['ftp_data'], data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [21]:

```
df['SF'].unique()
```

Out[21]:

```
array(['SF', 'S0', 'REJ', 'RSTR', 'SH', 'RSTO', 'S1', 'RSTOS0', 'S3',
       'S2', 'OTH'], dtype=object)
```

In [22]:

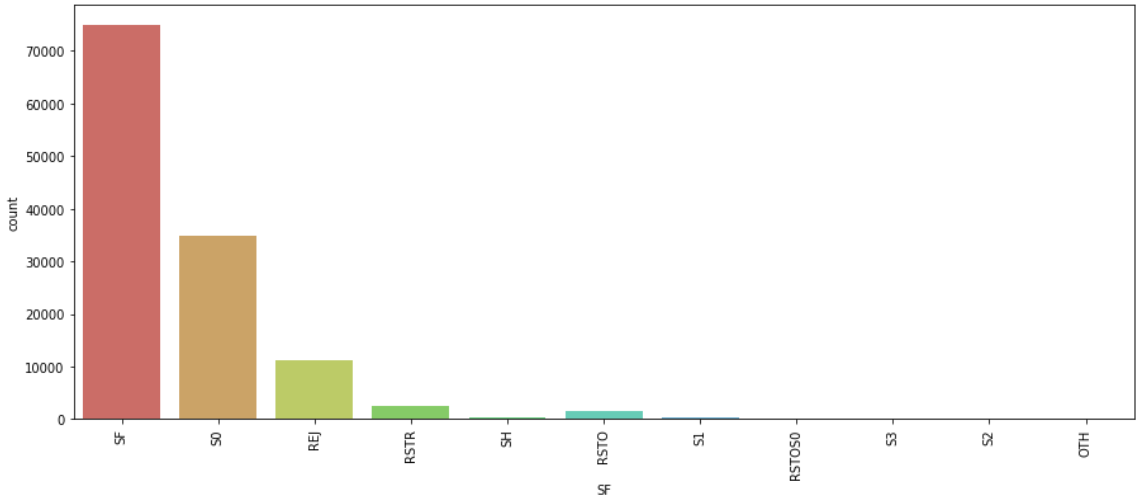
```
df['SF'].value_counts()
```

Out[22]:

```
SF        74944
S0        34851
REJ       11233
RSTR       2421
RSTO       1562
S1          365
SH          271
S2          127
RSTOS0      103
S3           49
OTH          46
Name: SF, dtype: int64
```

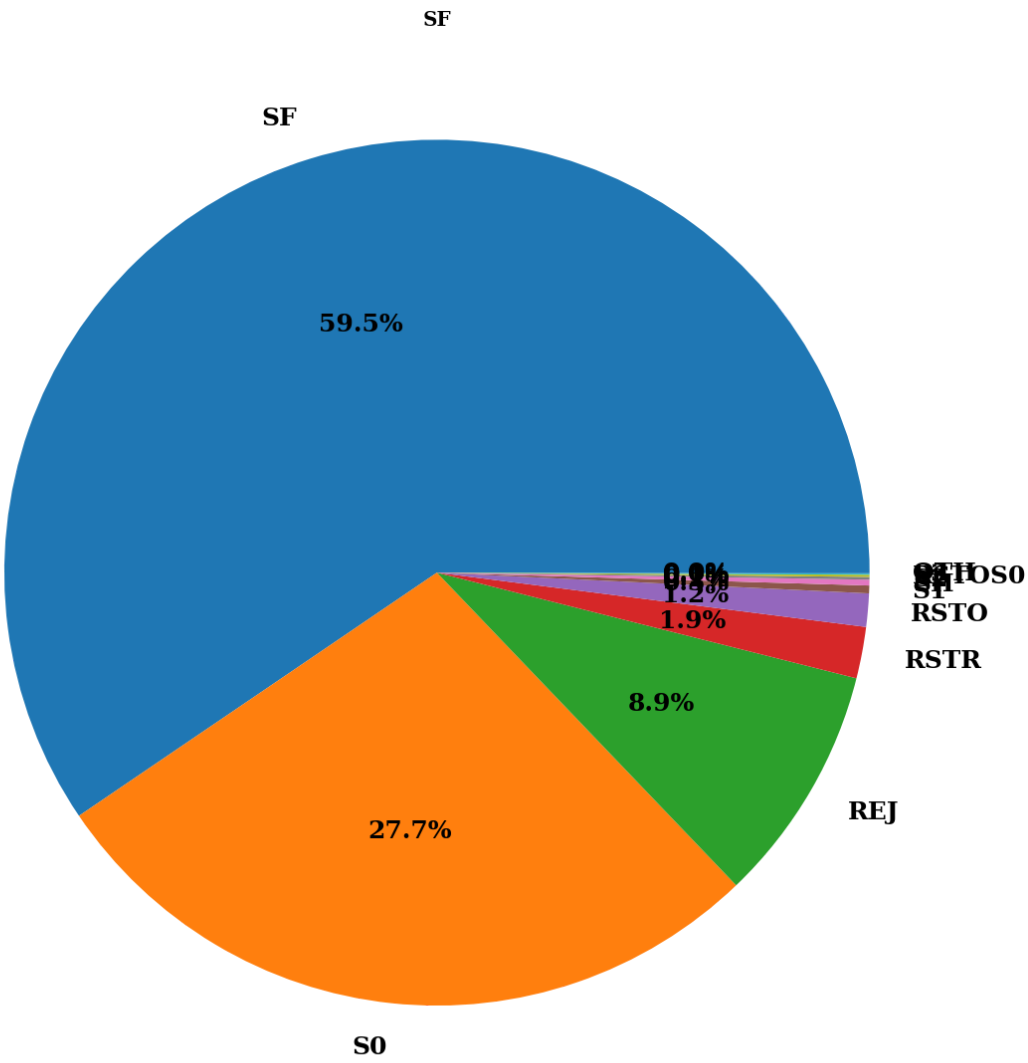
In [23]:

```
plt.figure(figsize=(15,6))
sns.countplot(df['SF'], data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [24]:

```
plt.figure(figsize=(30,20))
plt.pie(df['SF'].value_counts(), labels=df['SF'].value_counts().index, autopct='%1.1f%%'
                                        'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('SF', size=20, **hfont)
plt.show()
```



In [25]:

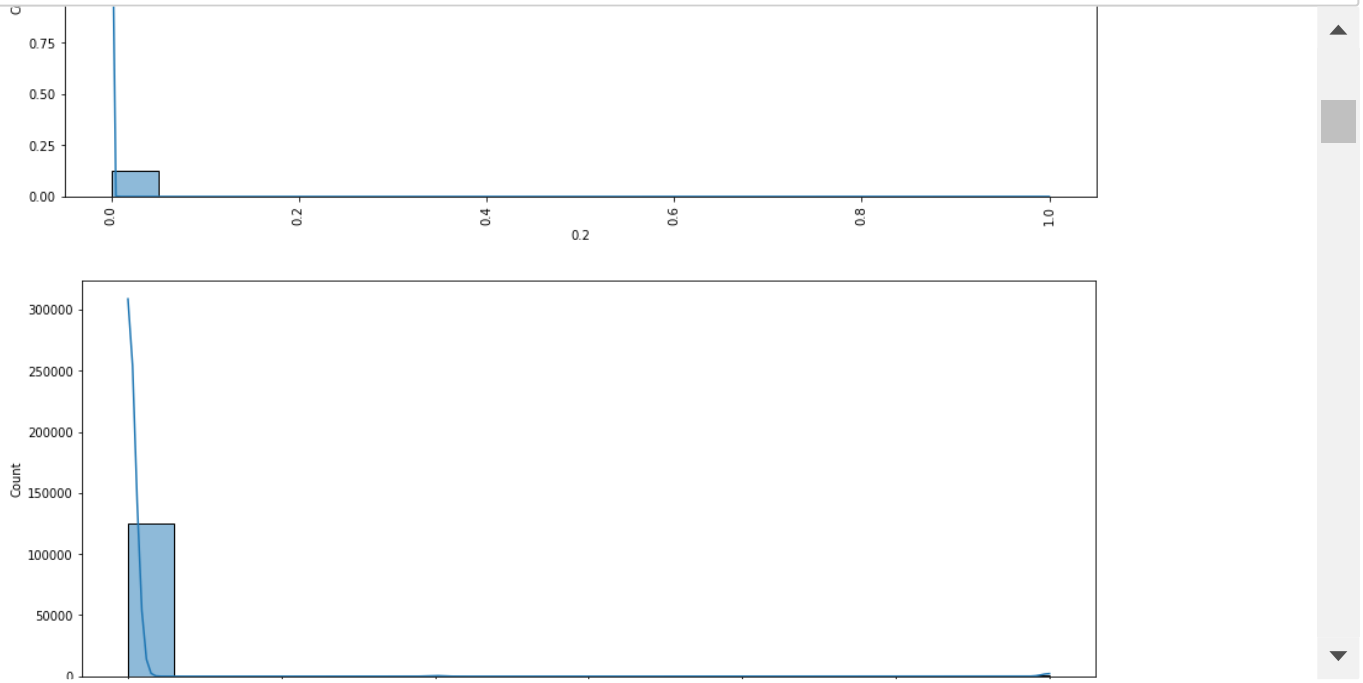
```
num_cols = df.select_dtypes(include='number')
```

In [26]:

```
obj_cols = df.select_dtypes(include='object')
```

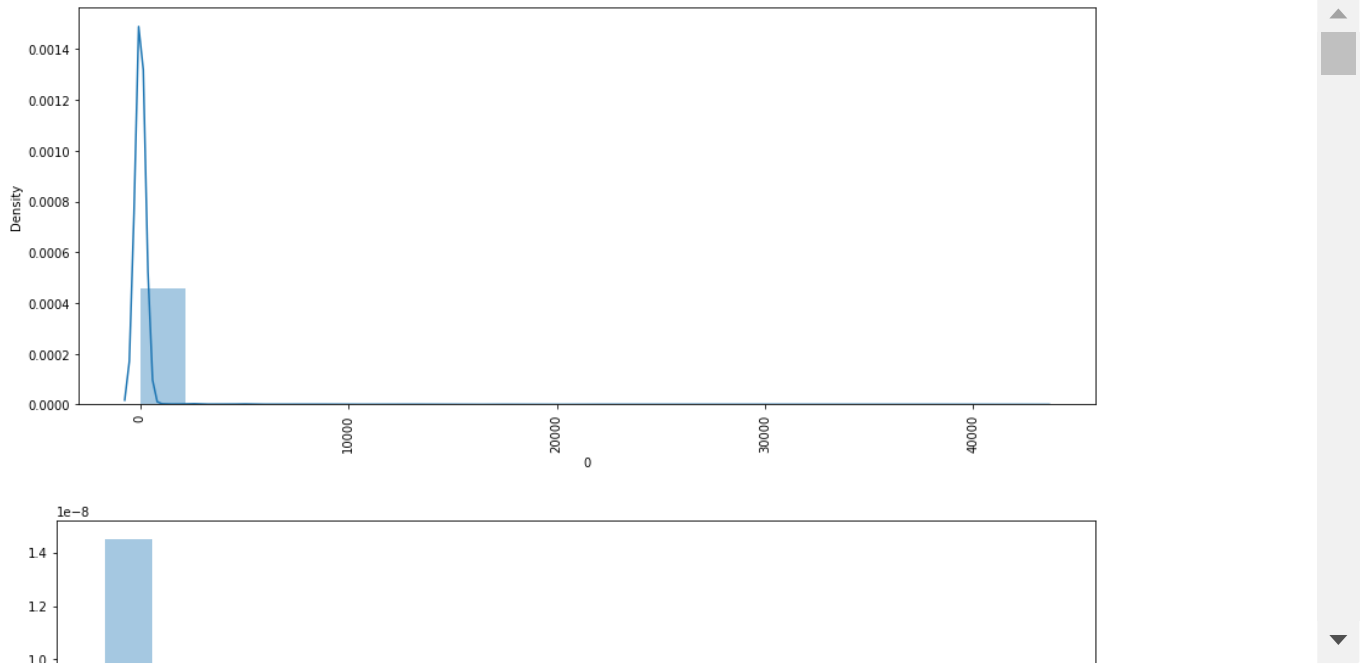
In [27]:

```
for i in num_cols.columns:
    plt.figure(figsize=(15,6))
    sns.histplot(num_cols[i], kde = True, bins = 20, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



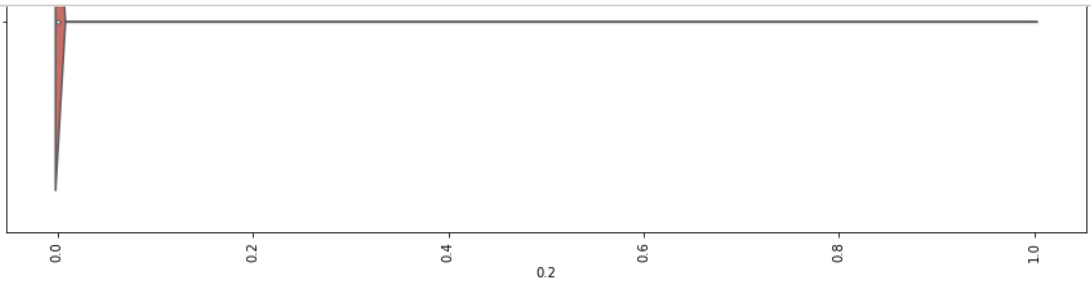
In [28]:

```
for i in num_cols.columns:
    plt.figure(figsize=(15,6))
    sns.distplot(num_cols[i], kde = True, bins = 20)
    plt.xticks(rotation = 90)
    plt.show()
```



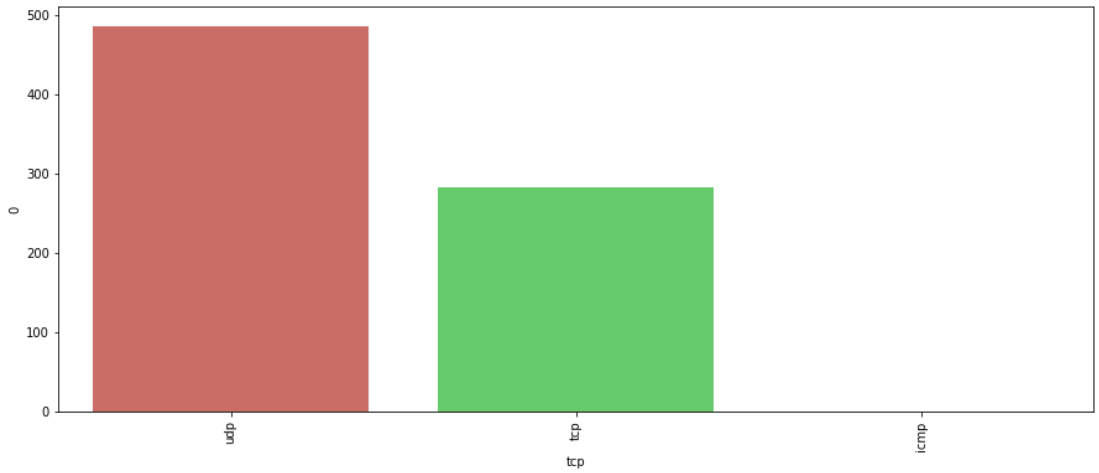
In [29]:

```
for i in num_cols.columns:
    plt.figure(figsize=(15,6))
    sns.violinplot(num_cols[i], palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



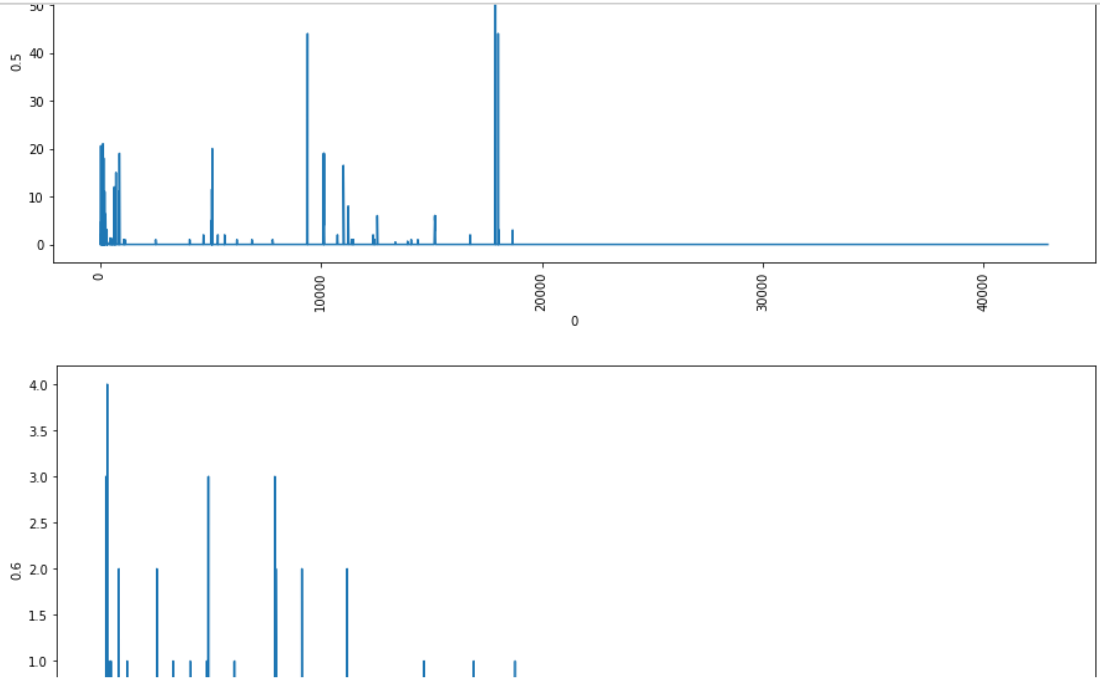
In [30]:

```
for i in obj_cols.columns:
    for j in num_cols.columns:
        plt.figure(figsize=(15,6))
        sns.barplot(x = obj_cols[i], y = num_cols[j], ci = None,
                    palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```



In [31]:

```
for i in num_cols.columns:
    for j in num_cols.columns:
        plt.figure(figsize=(15,6))
        sns.lineplot(x = num_cols[i], y = num_cols[j], ci = None,
                     palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```



In [32]:

```
Columns = (['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','
            'num_failed_logins','logged_in','num_compromised','root_shell','su_attempted
            'num_shells','num_access_files','num_outbound_cmds','is_host_login','is_gues
            'serror_rate','srv_serror_rate','rerror_rate','srv_rerror_rate','same_srv_ra
            'dst_host_count','dst_host_srv_count','dst_host_same_srv_rate','dst_host_dif
            'dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_srv_serror_ra
            'dst_host_srv_rerror_rate','attack','level'])
```

In [33]:

```
df.columns = Columns
```

In [34]:

```
df
```

Out[34]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment
0	0	udp	other	SF	146	0	0	0
1	0	tcp	private	S0	0	0	0	0
2	0	tcp	http	SF	232	8153	0	0
3	0	tcp	http	SF	199	420	0	0
4	0	tcp	private	REJ	0	0	0	0
...	...	...	...	...	...	...	...	...
125967	0	tcp	private	S0	0	0	0	0
125968	8	udp	private	SF	105	145	0	0
125969	0	tcp	smtp	SF	2231	384	0	0
125970	0	tcp	klogin	S0	0	0	0	0
125971	0	tcp	ftp_data	SF	151	0	0	0

125972 rows × 43 columns

In [35]:

```
df.max()
```

Out[35]:

```
duration                          42908
protocol_type                       udp
service                           whois
flag                                 SH
src_bytes                    1379963888
dst_bytes                    1309937401
land                                  1
wrong_fragment                        3
urgent                                3
hot                                  77
num_failed_logins                     5
logged_in                             1
num_compromised                    7479
root_shell                            1
su_attempted                          2
num_root                           7468
num_file_creations                   43
num_shells                            2
num_access_files                      9
num_outbound_cmds                     0
is_host_login                         1
is_guest_login                        1
count                               511
srv_count                           511
serror_rate                         1.0
srv_serror_rate                     1.0
rerror_rate                         1.0
srv_rerror_rate                     1.0
same_srv_rate                       1.0
diff_srv_rate                       1.0
srv_diff_host_rate                  1.0
dst_host_count                      255
dst_host_srv_count                  255
dst_host_same_srv_rate              1.0
dst_host_diff_srv_rate              1.0
dst_host_same_src_port_rate         1.0
dst_host_srv_diff_host_rate         1.0
dst_host_serror_rate                1.0
dst_host_srv_serror_rate            1.0
dst_host_rerror_rate                1.0
dst_host_srv_rerror_rate            1.0
attack                       warezmaster
level                                21
dtype: object
```

In [36]:

```
df['attack'].unique()
```

Out[36]:

```
array(['normal', 'neptune', 'warezclient', 'ipsweep', 'portsweep',
       'teardrop', 'nmap', 'satan', 'smurf', 'pod', 'back',
       'guess_passwd', 'ftp_write', 'multihop', 'rootkit',
       'buffer_overflow', 'imap', 'warezmaster', 'phf', 'land',
       'loadmodule', 'spy', 'perl'], dtype=object)
```

In [37]:

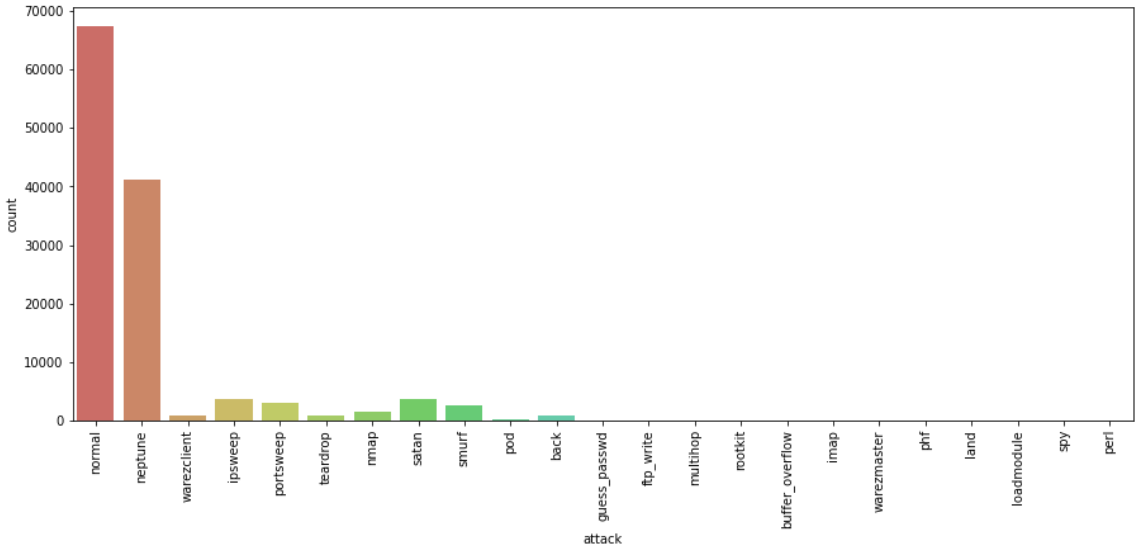
```
df['attack'].value_counts()
```

Out[37]:

```
normal             67342
neptune            41214
satan               3633
ipsweep             3599
portsweep           2931
smurf               2646
nmap                1493
back                 956
teardrop             892
warezclient          890
pod                  201
guess_passwd          53
buffer_overflow       30
warezmaster           20
land                  18
imap                  11
rootkit               10
loadmodule             9
ftp_write              8
multihop               7
phf                    4
perl                   3
spy                    2
Name: attack, dtype: int64
```

In [38]:

```
plt.figure(figsize=(15,6))
sns.countplot(df['attack'], data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [39]:

```
Trained_attack = df.attack.map(lambda a: 0 if a == 'normal' else 1)
```

In [40]:

```
df['attack_state'] = Trained_attack
```

In [41]:

```
df['attack_state'].unique()
```

Out[41]:

```
array([0, 1], dtype=int64)
```

In [42]:

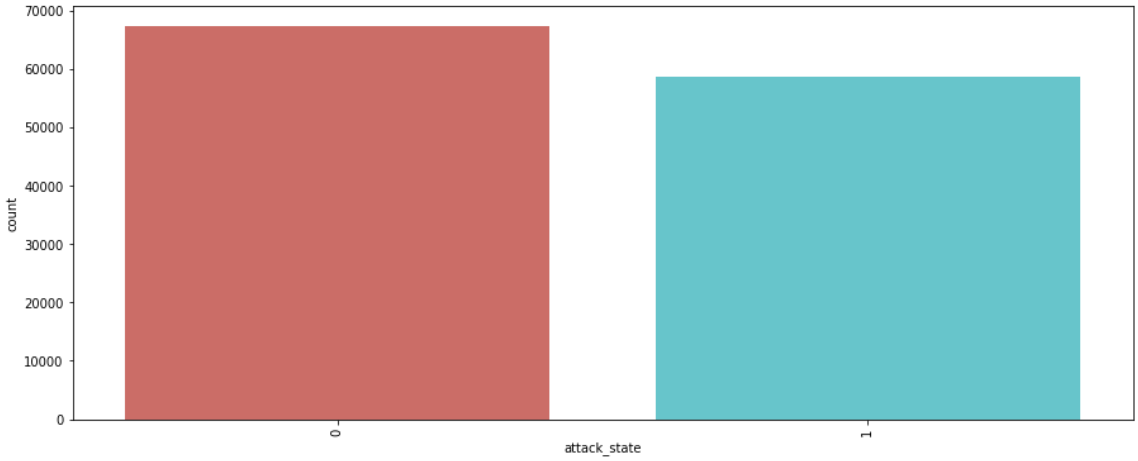
```
df['attack_state'].value_counts()
```

Out[42]:

```
0    67342
1    58630
Name: attack_state, dtype: int64
```

In [43]:

```
plt.figure(figsize=(15,6))
sns.countplot(df['attack_state'], data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [44]:

```
df = pd.get_dummies(df,columns=['protocol_type','service','flag'],prefix="",prefix_sep="
```

In [45]:

```
from sklearn.preprocessing import LabelEncoder
```

In [46]:

```
LE = LabelEncoder()
attack_LE= LabelEncoder()
df['attack'] = attack_LE.fit_transform(df["attack"])
```

In [47]:

```
from sklearn.model_selection import train_test_split
```

In [48]:

```
X = df.drop(['attack', 'level', 'attack_state'], axis = 1)
y = df['attack_state']
```

In [49]:

```
X_train,X_test,Y_train,Y_test = train_test_split(X, y, test_size= 0.25 ,
                                                 random_state=42)
```

In [50]:

```
from sklearn.preprocessing import RobustScaler
```

In [51]:

```
Ro_scaler = RobustScaler()
X_train = Ro_scaler.fit_transform(X_train)
X_test= Ro_scaler.transform(X_test)
```

In [52]:

```
import statsmodels.api as sm
```

In [53]:

```
A = sm.add_constant(X_train)
Est1 = sm.GLM(Y_train, A)
Est2 = Est1.fit()
Est2.summary()
```

Out[53]:

Generalized Linear Model Regression Results

Dep. Variable:	attack_state	**No. Observations:**	94479
Model:	GLM	**Df Residuals:**	94361
Model Family:	Gaussian	**Df Model:**	117
Link Function:	identity	**Scale:**	0.030102
Method:	IRLS	**Log-Likelihood:**	31488.
Date:	Fri, 24 Mar 2023	**Deviance:**	2840.4
Time:	19:22:05	**Pearson chi2:**	2.84e+03
No. Iterations:	3	**Pseudo R-squ. (CS):**	0.9993
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**

In [54]:

```
from sklearn.linear_model import LogisticRegression
```

In [55]:

```
LR= LogisticRegression()
LR.fit(X_train , Y_train)
```

Out[55]:

▾ LogisticRegression

LogisticRegression()

In [56]:

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [57]:

```
y_pred = LR.predict(X_test)
```

In [58]:

```
print("Accuracy:", accuracy_score(Y_test, y_pred))
```

```
Accuracy: 0.8534594989362716
```

In [60]:

```
 cm = confusion_matrix(Y_test, y_pred)
```

In [61]:

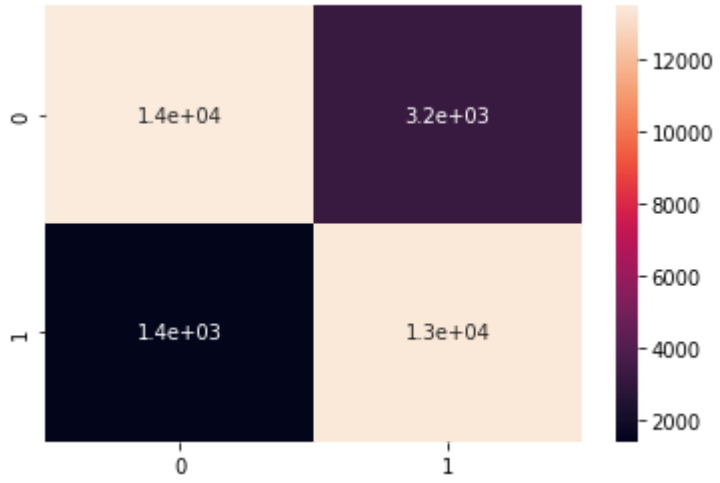
```
cm
```

Out[61]:

```
array([[13508,  3212],
       [ 1403, 13370]], dtype=int64)
```

In [62]:

```
sns.heatmap(cm, annot = True)
plt.show()
```



In [63]:

```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.81      0.85     16720
           1       0.81      0.91      0.85     14773

    accuracy                           0.85     31493
   macro avg       0.86      0.86      0.85     31493
weighted avg       0.86      0.85      0.85     31493
```

In [64]:

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, Y_train)
```

Out[64]:

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [65]:

```
y_pred = dt.predict(X_test)
```

In [66]:

```
print("Accuracy:", accuracy_score(Y_test, y_pred))
```

```
Accuracy: 0.9983170863366463
```

In [67]:

```
cm = confusion_matrix(Y_test, y_pred)
```

In [68]:

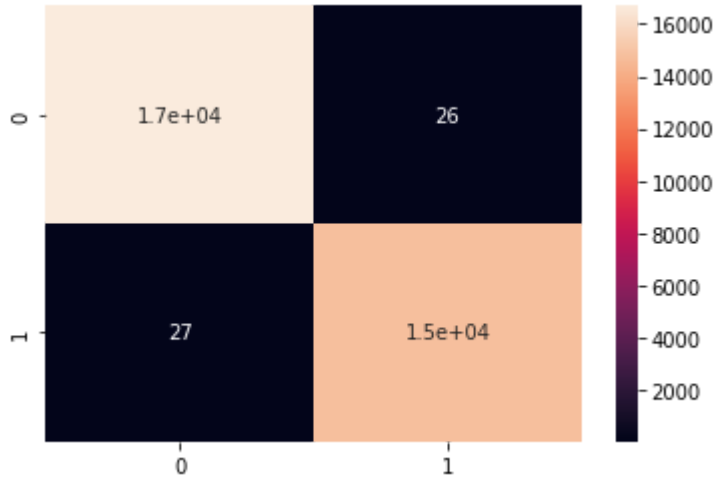
```
cm
```

Out[68]:

```
array([[16694,    26],
       [   27, 14746]], dtype=int64)
```

In [69]:

```
sns.heatmap(cm, annot = True)
plt.show()
```



In [70]:

```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16720
           1       1.00      1.00      1.00     14773

    accuracy                           1.00     31493
   macro avg       1.00      1.00      1.00     31493
weighted avg       1.00      1.00      1.00     31493
```

In [78]:

```
from sklearn.naive_bayes import GaussianNB
```

In [79]:

```
gnb = GaussianNB()
gnb.fit(X_train, Y_train)
```

Out[79]:

▾ GaussianNB

GaussianNB()

In [80]:

```
y_pred = gnb.predict(X_test)
```

In [81]:

```
accuracy = accuracy_score(Y_test, y_pred)
print('Accuracy:', accuracy)
```

```
Accuracy: 0.9175372304956657
```

In [82]:

```
cm = confusion_matrix(Y_test, y_pred)
```

In [83]:

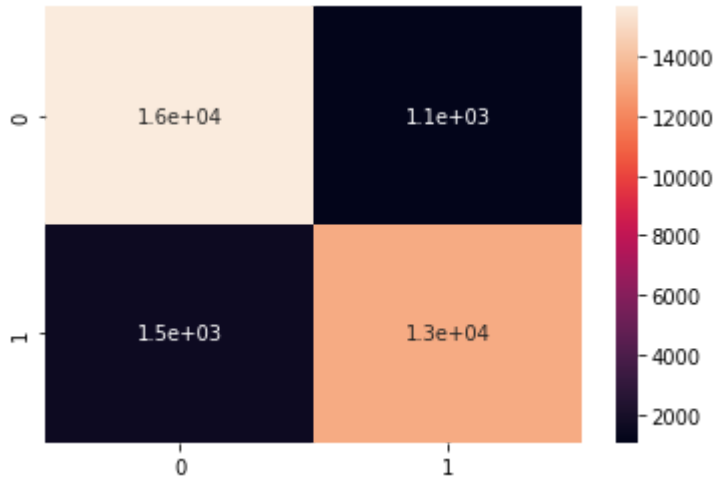
```
cm
```

Out[83]:

```
array([[15663,  1057],
       [ 1540, 13233]], dtype=int64)
```

In [84]:

```
sns.heatmap(cm, annot = True)
plt.show()
```



In [85]:

```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.94      0.92     16720
           1       0.93      0.90      0.91     14773

    accuracy                           0.92     31493
   macro avg       0.92      0.92      0.92     31493
weighted avg       0.92      0.92      0.92     31493
```