Cyber Security Mitigation and Response AI

Dataset Source: https://www.kaggle.com/sampadab17/network-intrusion-detection (https://www.kaggle.com/sampadab17/network-intrusion-detection)

In [75]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sweetviz as sv
from scipy import stats
import seaborn as sns
```

In [76]:

```
test_cs = pd.read_csv('test_data.csv')
train_cs = pd.read_csv('train_data.csv')
```

In [77]:

```
train_cs
```

Out[77]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	u
0	0	tcp	ftp_data	SF	491	0	0	0	
1	0	udp	other	SF	146	0	0	0	
2	0	tcp	private	S0	0	0	0	0	
3	0	tcp	http	SF	232	8153	0	0	
4	0	tcp	http	SF	199	420	0	0	
...	...	...	...	...	...	...	...	...	
25187	0	tcp	exec	RSTO	0	0	0	0	
25188	0	tcp	ftp_data	SF	334	0	0	0	
25189	0	tcp	private	REJ	0	0	0	0	
25190	0	tcp	nnsp	S0	0	0	0	0	
25191	0	tcp	finger	S0	0	0	0	0	

25192 rows × 42 columns

In [78]:

```
train_cs.describe()
```

Out[78]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent
count	25192.000000	2.519200e+04	2.519200e+04	25192.000000	25192.000000	25192.00000
mean	305.054104	2.433063e+04	3.491847e+03	0.000079	0.023738	0.00004
std	2686.555640	2.410805e+06	8.883072e+04	0.008910	0.260221	0.00630
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.00000
25%	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.00000
50%	0.000000	4.400000e+01	0.000000e+00	0.000000	0.000000	0.00000
75%	0.000000	2.790000e+02	5.302500e+02	0.000000	0.000000	0.00000
max	42862.000000	3.817091e+08	5.151385e+06	1.000000	3.000000	1.00000

8 rows × 38 columns

In [79]:

```
train_cs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25192 entries, 0 to 25191
Data columns (total 42 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   duration                     25192 non-null  int64  
 1   protocol_type                25192 non-null  object 
 2   service                      25192 non-null  object 
 3   flag                         25192 non-null  object 
 4   src_bytes                    25192 non-null  int64  
 5   dst_bytes                    25192 non-null  int64  
 6   land                         25192 non-null  int64  
 7   wrong_fragment               25192 non-null  int64  
 8   urgent                       25192 non-null  int64  
 9   hot                          25192 non-null  int64  
 10  num_failed_logins            25192 non-null  int64  
 11  logged_in                    25192 non-null  int64  
 12  num_compromised              25192 non-null  int64  
 13  root_shell                   25192 non-null  int64  
 14  su_attempted                 25192 non-null  int64  
 15  num_root                     25192 non-null  int64  
 16  num_file_creations           25192 non-null  int64  
 17  num_shells                   25192 non-null  int64  
 18  num_access_files             25192 non-null  int64  
 19  num_outbound_cmds            25192 non-null  int64  
 20  is_host_login                25192 non-null  int64  
 21  is_guest_login               25192 non-null  int64  
 22  count                        25192 non-null  int64  
 23  srv_count                    25192 non-null  int64  
 24  serror_rate                  25192 non-null  float64
 25  srv_serror_rate              25192 non-null  float64
 26  rerror_rate                  25192 non-null  float64
 27  srv_rerror_rate              25192 non-null  float64
 28  same_srv_rate                25192 non-null  float64
 29  diff_srv_rate                25192 non-null  float64
 30  srv_diff_host_rate           25192 non-null  float64
 31  dst_host_count               25192 non-null  int64  
 32  dst_host_srv_count           25192 non-null  int64  
 33  dst_host_same_srv_rate       25192 non-null  float64
 34  dst_host_diff_srv_rate       25192 non-null  float64
 35  dst_host_same_src_port_rate  25192 non-null  float64
 36  dst_host_srv_diff_host_rate  25192 non-null  float64
 37  dst_host_serror_rate         25192 non-null  float64
 38  dst_host_srv_serror_rate     25192 non-null  float64
 39  dst_host_rerror_rate         25192 non-null  float64
 40  dst_host_srv_rerror_rate     25192 non-null  float64
 41  class                        25192 non-null  object 
dtypes: float64(15), int64(23), object(4)
memory usage: 8.1+ MB
```

In [80]:

```
#demotrain = sv.analyze(train_cs)
#demotrain.show_html()
```

In [81]:

```
train_cs.isnull().sum()
```

Out[81]:

```
duration                       0
protocol_type                  0
service                        0
flag                           0
src_bytes                      0
dst_bytes                      0
land                           0
wrong_fragment                 0
urgent                         0
hot                            0
num_failed_logins              0
logged_in                      0
num_compromised                0
root_shell                     0
su_attempted                   0
num_root                       0
num_file_creations             0
num_shells                     0
num_access_files               0
num_outbound_cmds              0
is_host_login                  0
is_guest_login                 0
count                          0
srv_count                      0
serror_rate                    0
srv_serror_rate                0
rerror_rate                    0
srv_rerror_rate                0
same_srv_rate                  0
diff_srv_rate                  0
srv_diff_host_rate             0
dst_host_count                 0
dst_host_srv_count             0
dst_host_same_srv_rate         0
dst_host_diff_srv_rate         0
dst_host_same_src_port_rate    0
dst_host_srv_diff_host_rate    0
dst_host_serror_rate           0
dst_host_srv_serror_rate       0
dst_host_rerror_rate           0
dst_host_srv_rerror_rate       0
class                          0
dtype: int64
```

In [82]:

```
train_cs.duplicated().sum()
```

Out[82]:

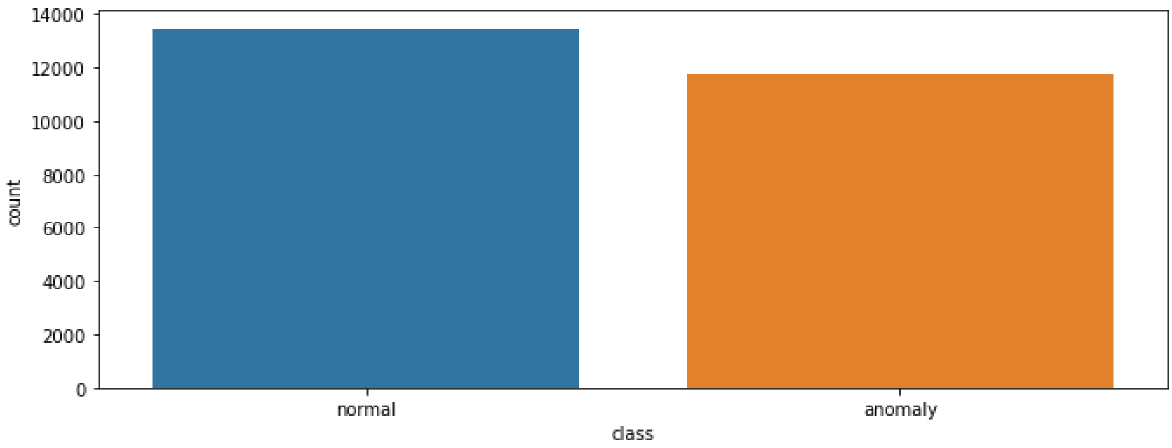
```
0
```

In [83]:

```
sns.countplot(data=train_cs, x='class')
```

Out[83]:

```
<AxesSubplot:xlabel='class', ylabel='count'>
```



In [84]:

```
(train_cs['class'].values == 'anomaly').sum()
```

Out[84]:

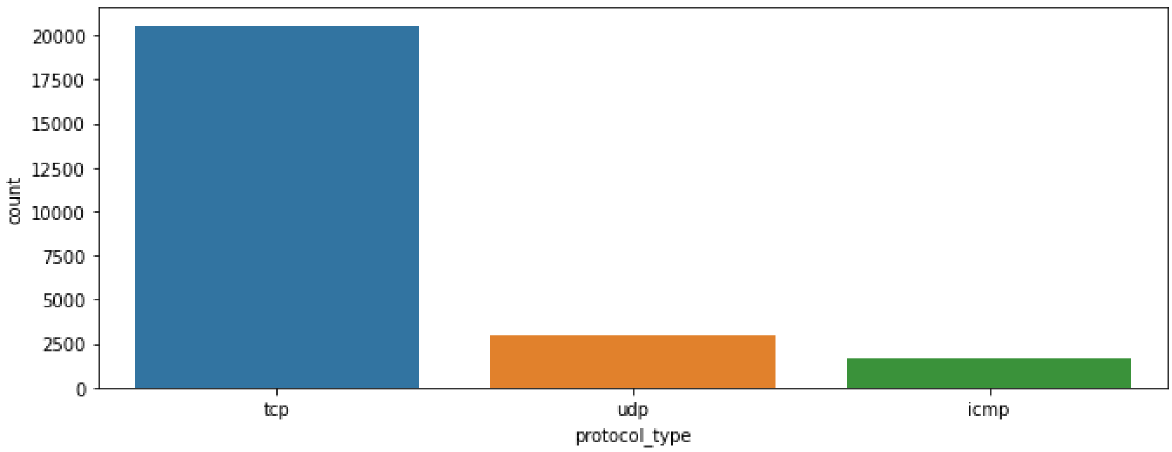
```
11743
```

In [85]:

```
sns.countplot(data=train_cs, x='protocol_type')
```

Out[85]:

```
<AxesSubplot:xlabel='protocol_type', ylabel='count'>
```



In [86]:

```
train_cs['service'].unique()
```

Out[86]:

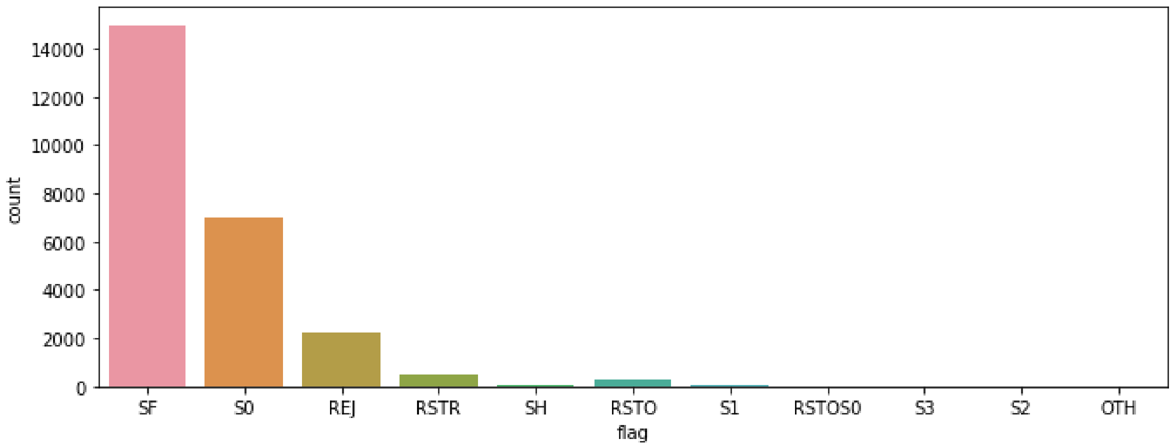
```
array(['ftp_data', 'other', 'private', 'http', 'remote_job', 'name',
       'netbios_ns', 'eco_i', 'mtp', 'telnet', 'finger', 'domain_u',
       'supdup', 'uucp_path', 'Z39_50', 'smtp', 'csnet_ns', 'uucp',
       'netbios_dgm', 'urp_i', 'auth', 'domain', 'ftp', 'bgp', 'ldap',
       'ecr_i', 'gopher', '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'], dtype=object)
```

In [87]:

```
sns.countplot(data=train_cs, x='flag')
```

Out[87]:

```
<AxesSubplot:xlabel='flag', ylabel='count'>
```



In [88]:

```
nor_data = train_cs.loc[train_cs['class'] == 'normal']
```

In [89]:

```
nor_data #normal data distribution
```

Out[89]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urg
0	0	tcp	ftp_data	SF	491	0	0	0	
1	0	udp	other	SF	146	0	0	0	
3	0	tcp	http	SF	232	8153	0	0	
4	0	tcp	http	SF	199	420	0	0	
12	0	tcp	http	SF	287	2251	0	0	
...	...	...	...	...	...	...	...	...	
25176	0	tcp	ftp_data	SF	748	0	0	0	
25177	0	tcp	http	SF	293	2486	0	0	
25184	29	tcp	ftp	SF	329	1063	0	0	
25185	1	tcp	smtp	SF	2896	333	0	0	
25186	0	tcp	http	S1	339	14600	0	0	

13449 rows × 42 columns

In [90]:

```
ano_data = train_cs.loc[train_cs['class'] == 'anomaly']
```

In [91]:

```
ano_data #anomaly data distribution
```

Out[91]:

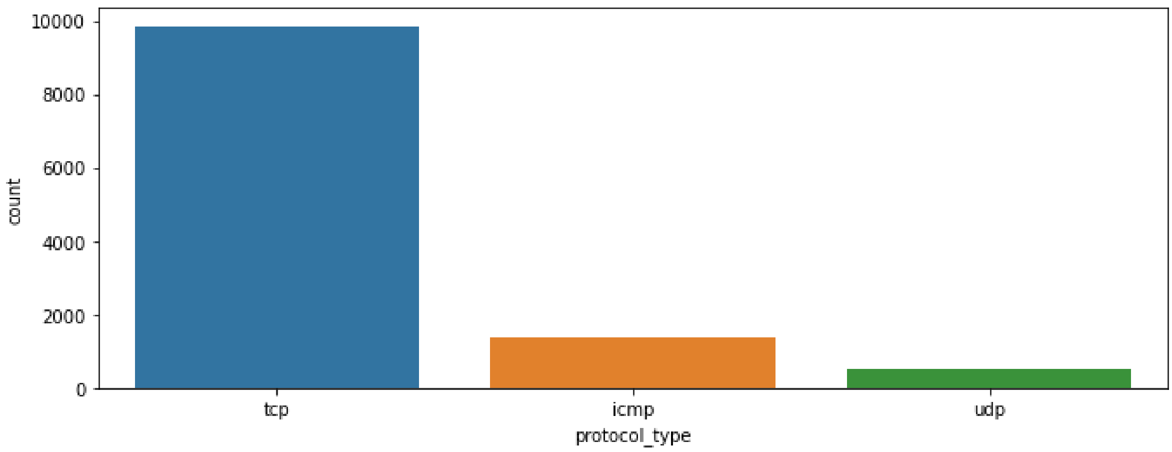
e	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_se
5	0.00	0.00	1.0	
7	0.00	0.00	0.0	
5	0.00	0.00	1.0	
7	0.00	0.00	1.0	
5	0.00	0.00	1.0	
..	...	...	...	
6	0.00	0.00	0.0	
0	1.00	0.18	0.0	
7	0.00	0.00	0.0	
6	0.00	0.00	1.0	
3	0.01	0.00	1.0	

In [92]:

```
sns.countplot(data=ano_data, x='protocol_type')
```

Out[92]:

```
<AxesSubplot:xlabel='protocol_type', ylabel='count'>
```



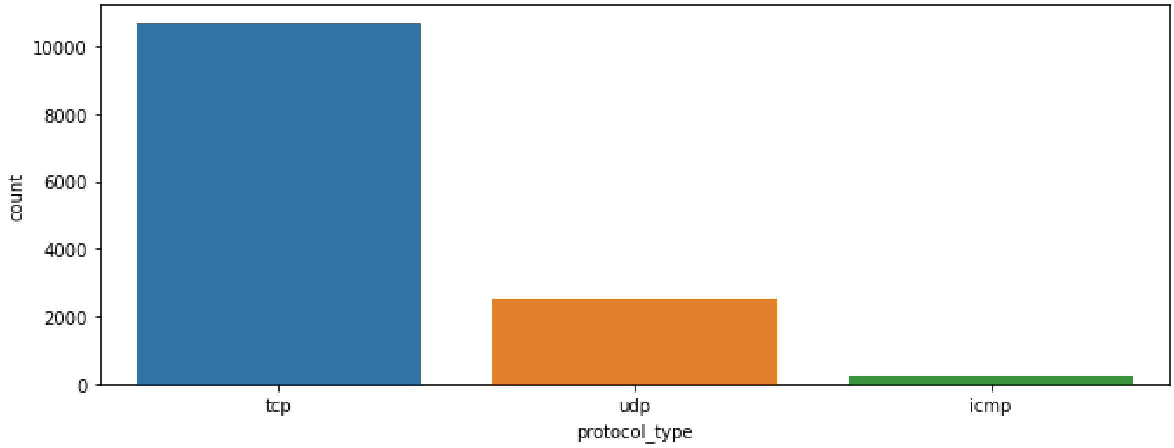
In anomaly data tcp and icmp protocols have higher usages

In [93]:

```
sns.countplot(data=nor_data, x='protocol_type')
```

Out[93]:

```
<AxesSubplot:xlabel='protocol_type', ylabel='count'>
```



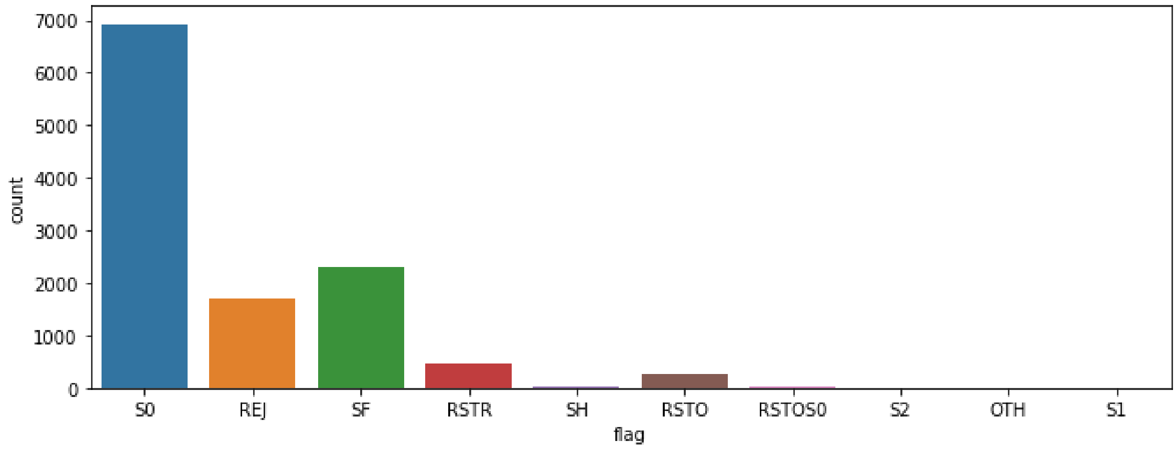
In normal distribution tcp and udp protocols have higher usage

In [94]:

```
sns.countplot(data=ano_data, x='flag')
```

Out[94]:

```
<AxesSubplot:xlabel='flag', ylabel='count'>
```

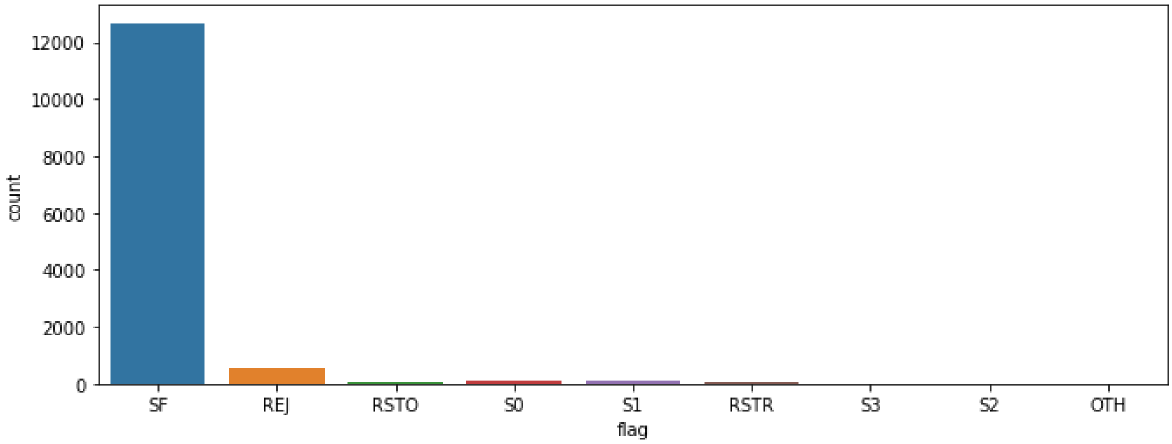


In [95]:

```
sns.countplot(data=nor_data, x='flag')
```

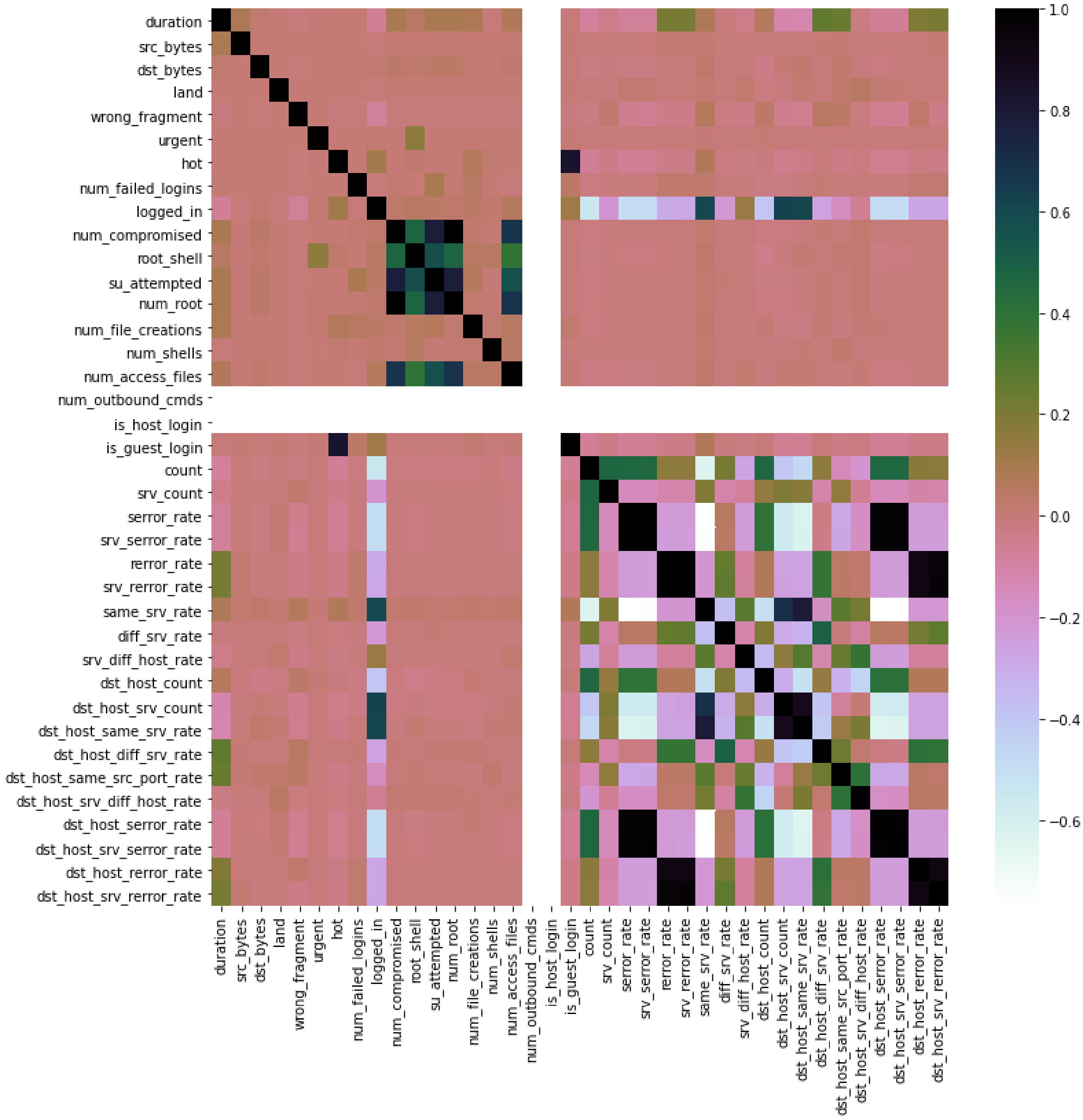
Out[95]:

```
<AxesSubplot:xlabel='flag', ylabel='count'>
```



In [96]:

```
corr = train_cs.corr()
plt.figure(figsize=(12,12))
sns.heatmap(corr, cmap='cubehelix_r');
```



In [97]:

```
train_cs.drop(['num_outbound_cmds', 'is_host_login'], axis=1, inplace=True)
test_cs.drop(['num_outbound_cmds', 'is_host_login'], axis=1, inplace=True)
```

Separating numerical and object columns

Combining all Numerical Columns

In [98]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

cols = train_cs.select_dtypes(include=['float64','int64']).columns
sc_train = scaler.fit_transform(train_cs.select_dtypes(include=['float64','int64']))
sc_test = scaler.fit_transform(test_cs.select_dtypes(include=['float64','int64']))

train_num = pd.DataFrame(sc_train, columns = cols)
test_num = pd.DataFrame(sc_test, columns = cols)
```

Finding Object type Columns

In [99]:

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

# extract object attributes from both train and test sets
train_obj = train_cs.select_dtypes(include=['object']).copy()
test_obj = test_cs.select_dtypes(include=['object']).copy()

# encode the object attributes
train_a = train_obj.apply(encoder.fit_transform)
test_a = test_obj.apply(encoder.fit_transform)

# separate 'class' column from encoded data
train_noclass = train_a.drop(['class'], axis=1)
Ytrain_class = train_a[['class']].copy()
```

In [100]:

```
train_x = pd.concat([train_num,train_noclass],axis=1)
train_y = train_cs['class']
train_x.shape
```

Out[100]:

```
(25192, 39)
```

In [101]:

```
test_ = pd.concat([test_num,test_a],axis=1)
test_.shape
```

Out[101]:

```
(22544, 39)
```

Feature Selection

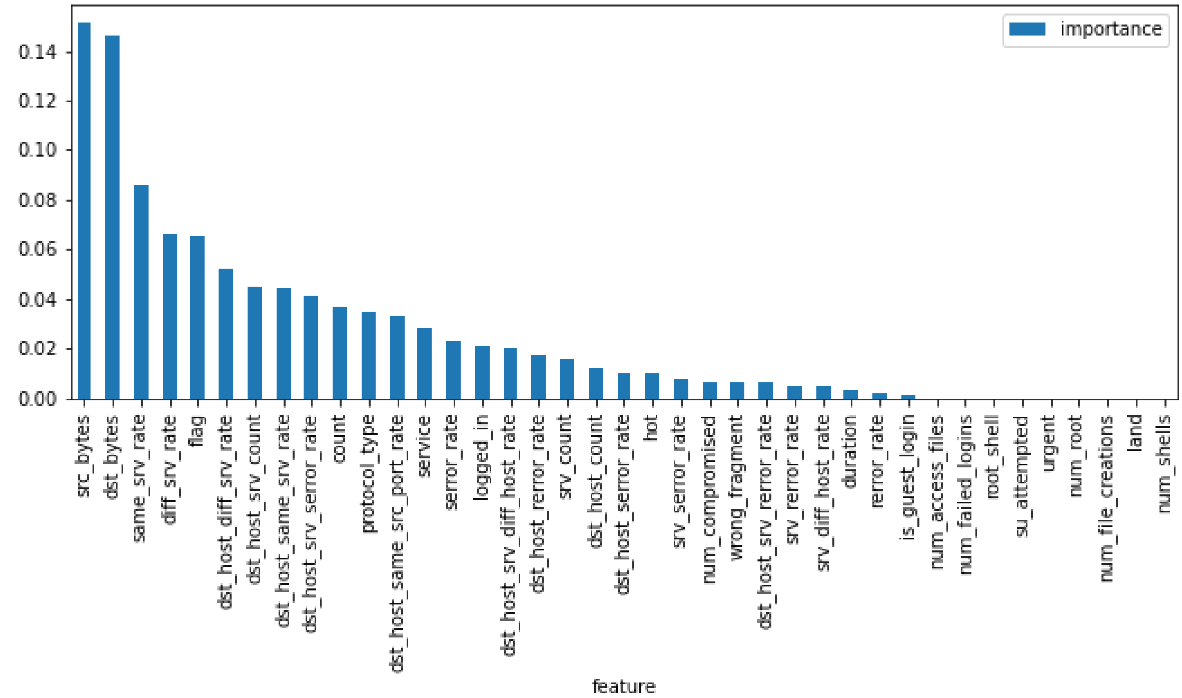
In [102]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier();

# fit random forest classifier on the training set
rfc.fit(train_x, train_y);

# extract important features
score = np.round(rfc.feature_importances_,3)
imp = pd.DataFrame({'feature':train_x.columns,'importance':score})
imp = imp.sort_values('importance',ascending=False).set_index('feature')

# plot importances
plt.rcParams['figure.figsize'] = (11, 4)
imp.plot.bar();
```



In [103]:

```
from sklearn.feature_selection import RFE
import itertools
rfc = RandomForestClassifier()

# create the RFE model and select 15 attributes
rfe = RFE(rfc, n_features_to_select=15)
rfe = rfe.fit(train_x, train_y)

# summarize the selection of the attributes
feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), train_x.columns)
features = [v for i, v in feature_map if i==True]

features
```

Out[103]:

```
['src_bytes',
 'dst_bytes',
 'logged_in',
 'count',
 'srv_count',
 'same_srv_rate',
 'diff_srv_rate',
 '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',
 'protocol_type',
 'service',
 'flag']
```

In [104]:

```
train_feature = train_x[features]
train_feature
```

Out[104]:

rv_rate	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_po
349282	-0.813985	-0.779157	-0.280673	0.
490836	-1.030895	-1.157831	2.764403	2.
042773	-0.804947	-0.935081	-0.173828	-0.
349282	1.264742	1.069663	-0.440940	-0.
349282	1.264742	1.069663	-0.440940	-0.
...	...	...	...	
042773	-0.976667	-1.091006	-0.120406	-0.
349282	-0.687453	1.069663	-0.440940	2.
042773	-0.922440	-1.046456	-0.066984	-0.
013235	-0.859174	-0.979631	-0.120406	-0.
266804	-0.597074	-0.734607	-0.280673	-0.

Defining Train and Test

In [105]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(train_feature, train_y, test_size = 0.2
```

Applying Logistic Regression

In [106]:

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import cross_val_score
```

In [107]:

```
LGR_Classifier = LogisticRegression(n_jobs=-1, random_state=0)
LGR_Classifier.fit(X_train, Y_train);
```

In [122]:

```
LGR_Classifier.coef_
```

Out[122]:

```
array([[ 9.81364175e-02, -3.65668863e-01, -7.22050813e-02,
        -1.65244595e-03, -1.31498884e+00, -9.28785294e-02,
        -5.90453266e-01, -2.05501508e-02,  1.91271626e-01,
        -4.50808136e-01, -7.36916857e-02,  3.45145878e-01,
         3.01212944e-01,  3.97918465e-02,  1.18981959e-01,
         2.18297502e-02,  3.97770604e-01, -1.48222618e+00,
         3.98018259e-01, -4.21418384e-01, -1.31829496e+00,
        -4.56260289e-01, -1.13789835e+00,  1.19844649e+00,
         3.55295807e-01, -2.63479517e-01, -9.60747615e-01,
         1.70019090e+00, -1.20766337e+00, -3.01584356e-01,
        -8.48080088e-01, -3.32264621e-01, -5.11430962e-01,
        -9.89170359e-01, -3.09202230e-01,  1.07031839e-01,
         1.60210425e+00,  4.85916741e-03, -3.65035186e-01]])
```

Evaluating Model

In [108]:

```
models = []
models.append(('LogisticRegression', LGR_Classifier))

for i, v in models:
    scores = cross_val_score(v, X_train, Y_train, cv=10)
    accuracy = metrics.accuracy_score(Y_train, v.predict(X_train))
    confusion_matrix = metrics.confusion_matrix(Y_train, v.predict(X_train))
    classification = metrics.classification_report(Y_train, v.predict(X_train))
    print()
    print('============================== {} Model Evaluation ============================
    print()
    print ("Cross Validation Mean Score:" "\n", scores.mean())
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
```

```
============================== LogisticRegression Model Evaluation =========
=====================

Cross Validation Mean Score:
 0.943632391371936

Model Accuracy:
 0.9439504604636393

Confusion matrix:
 [[8162  652]
 [ 407 9673]]

Classification report:
               precision    recall  f1-score   support

     anomaly       0.95      0.93      0.94      8814
      normal       0.94      0.96      0.95     10080

    accuracy                           0.94     18894
   macro avg       0.94      0.94      0.94     18894
weighted avg       0.94      0.94      0.94     18894
```

Validating Model

In [109]:

```
for i, v in models:
    accuracy = metrics.accuracy_score(Y_test, v.predict(X_test))
    confusion_matrix = metrics.confusion_matrix(Y_test, v.predict(X_test))
    classification = metrics.classification_report(Y_test, v.predict(X_test))
    print()
    print('============================== {} Model Test Results ==========================
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
```

```
============================== LogisticRegression Model Test Results =======
=======================

Model Accuracy:
 0.9402985074626866

Confusion matrix:
 [[2703  226]
 [ 150 3219]]

Classification report:
               precision    recall  f1-score   support

     anomaly       0.95      0.92      0.93      2929
      normal       0.93      0.96      0.94      3369

    accuracy                           0.94      6298
   macro avg       0.94      0.94      0.94      6298
weighted avg       0.94      0.94      0.94      6298
```

Another approach to improve model's accuracy rate

In [110]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(train_x, train_y, test_size = 0.25, ran
```

In [111]:

```
LGR_Classifier = LogisticRegression(n_jobs=-1, random_state=0)
LGR_Classifier.fit(X_train, Y_train);
```

Evaluation Model

In [112]:

```
models = []
models.append(('LogisticRegression', LGR_Classifier))

for i, v in models:
    scores = cross_val_score(v, X_train, Y_train, cv=10)
    accuracy = metrics.accuracy_score(Y_train, v.predict(X_train))
    confusion_matrix = metrics.confusion_matrix(Y_train, v.predict(X_train))
    classification = metrics.classification_report(Y_train, v.predict(X_train))
    print()
    print('============================== {} Model Evaluation ============================
    print()
    print ("Cross Validation Mean Score:" "\n", scores.mean())
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
```

```
============================== LogisticRegression Model Evaluation =========
=====================

Cross Validation Mean Score:
 0.9547470036776546

Model Accuracy:
 0.9556472954377051

Confusion matrix:
 [[8298  516]
 [ 322 9758]]

Classification report:
               precision    recall  f1-score   support

     anomaly       0.96      0.94      0.95      8814
      normal       0.95      0.97      0.96     10080

    accuracy                           0.96     18894
   macro avg       0.96      0.95      0.96     18894
weighted avg       0.96      0.96      0.96     18894
```

Validation Model

In [113]:

```
for i, v in models:
    accuracy = metrics.accuracy_score(Y_test, v.predict(X_test))
    confusion_matrix = metrics.confusion_matrix(Y_test, v.predict(X_test))
    classification = metrics.classification_report(Y_test, v.predict(X_test))
    print()
    print('============================== {} Model Test Results ==========================
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
```

```
============================== LogisticRegression Model Test Results =======
=======================

Model Accuracy:
 0.9537948555096856

Confusion matrix:
 [[2754  175]
 [ 116 3253]]

Classification report:
               precision    recall  f1-score   support

     anomaly       0.96      0.94      0.95      2929
      normal       0.95      0.97      0.96      3369

    accuracy                           0.95      6298
   macro avg       0.95      0.95      0.95      6298
weighted avg       0.95      0.95      0.95      6298
```

Predicting normal and anamoly behaviour on Test dataset

In [114]:

```
pred_log = LGR_Classifier.predict(test_)
```

In [115]:

```
pred_log
```

Out[115]:

```
array(['anomaly', 'anomaly', 'normal', ..., 'normal', 'normal', 'anomaly'],
      dtype=object)
```

In [116]:

```
test_cs
```

Out[116]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment
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
...	...	...	...	...	...	...	...	...
22539	0	tcp	smtp	SF	794	333	0	0
22540	0	tcp	http	SF	317	938	0	0
22541	0	tcp	http	SF	54540	8314	0	0
22542	0	udp	domain_u	SF	42	42	0	0
22543	0	tcp	sunrpc	REJ	0	0	0	0

22544 rows × 39 columns

In [117]:

```
pred_df = pd.DataFrame(pred_log, columns = ['class'])
test_output = pd.concat([test_cs, pred_df],axis=1)
```

In [118]:

```
test_output
```

Out[118]:

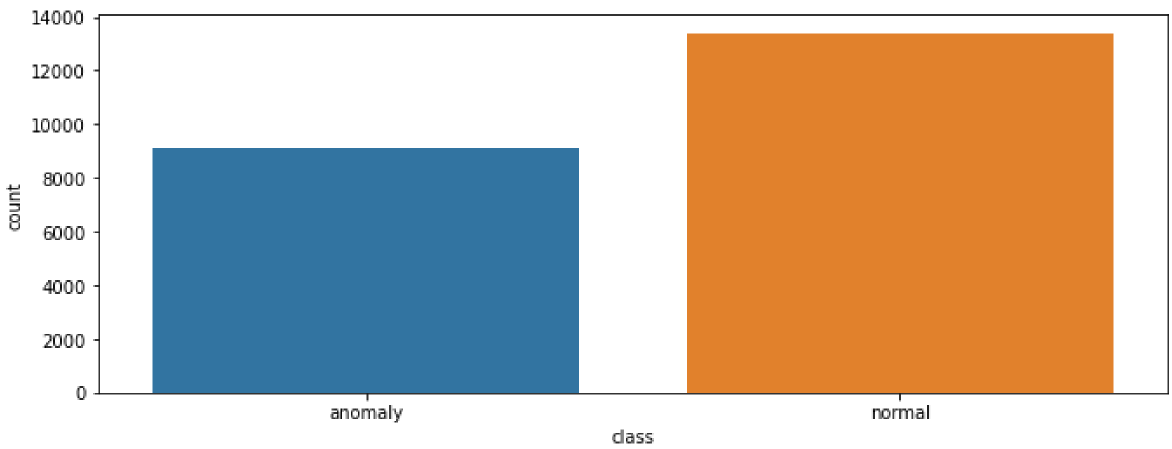
e	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_se
6	0.00	0.00	0.00	
6	0.00	0.00	0.00	
4	0.61	0.02	0.00	
0	1.00	0.28	0.00	
7	0.03	0.02	0.00	
..	...	...	...	
6	0.01	0.01	0.01	
0	0.01	0.01	0.01	
0	0.00	0.00	0.00	
1	0.00	0.00	0.00	
3	0.00	0.00	0.00	

In [119]:

```
sns.countplot(data = test_output, x = 'class')
```

Out[119]:

```
<AxesSubplot:xlabel='class', ylabel='count'>
```



In [ ]:

```
```