Environment Setup

In [1]:
```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn
import imblearn
import sys

# Ignore warnings
import warnings
warnings.filterwarnings('ignore')

# Config
pd.set_option('display.max_columns', None)
np.set_printoptions(threshold=sys.maxsize)
np.set_printoptions(precision=3)
sns.set(style="darkgrid")
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12

print("pandas : {0}".format(pd.__version__))
print("numpy : {0}".format(np.__version__))
print("matplotlib : {0}".format(matplotlib.__version__))
print("seaborn : {0}".format(sns.__version__))
print("sklearn : {0}".format(sklearn.__version__))
print("imblearn : {0}".format(imblearn.__version__))
```

```
pandas : 1.2.4
numpy : 1.20.1
matplotlib : 3.3.4
seaborn : 0.11.1
sklearn : 0.24.1
imblearn : 0.8.0
```

Load Data

In [2]:
```
# Dataset field names
datacols = ["duration","protocol_type","service","flag","src_bytes",
    "dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins",
    "logged_in","num_compromised","root_shell","su_attempted","num_root",
    "num_file_creations","num_shells","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_srv_rate",
    "diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_coun
    "dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_por
    "dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serro
    "dst_host_rerror_rate","dst_host_srv_rerror_rate","attack", "last_flag"]

# Load NSL_KDD train dataset
dfkdd_train = pd.read_table("./NSL_KDD_dataset/KDDTrain.txt", sep=",", names
dfkdd_train = dfkdd_train.iloc[:,:-1] # removes an unwanted extra field

# Load NSL_KDD test dataset
dfkdd_test = pd.read_table("./NSL_KDD_dataset/KDDTest.txt", sep=",", names=d
dfkdd_test = dfkdd_test.iloc[:,:-1]
```

Train dataset

In [3]:
```
# View train data
dfkdd_train.head(3)

# train set dimension
print('Train set dimension: {} rows, {} columns'.format(dfkdd_train.shape[0]
```

```
Train set dimension: 125973 rows, 42 columns
```

Test dataset

In [4]:
```
# View test data
dfkdd_test.head(3)

# test set dimension
print('Test set dimension: {} rows, {} columns'.format(dfkdd_test.shape[0],
```

```
Test set dimension: 22544 rows, 42 columns
```

Data Preprocessing

In [5]:
```
mapping = {'ipsweep': 'Probe','satan': 'Probe','nmap': 'Probe','portsweep':
        'teardrop': 'DoS','pod': 'DoS','land': 'DoS','back': 'DoS','neptune'
        'udpstorm': 'DoS','apache2': 'DoS','processtable': 'DoS',
        'perl': 'U2R','loadmodule': 'U2R','rootkit': 'U2R','buffer_overflow'
        'sqlattack': 'U2R','httptunnel': 'U2R',
        'ftp_write': 'R2L','phf': 'R2L','guess_passwd': 'R2L','warezmaster':
        'spy': 'R2L','multihop': 'R2L','named': 'R2L','snmpguess': 'R2L','wo
        'xsnoop': 'R2L','xlock': 'R2L','sendmail': 'R2L',
        'normal': 'Normal'
        }
```

In [6]:
```
# Apply attack class mappings to the dataset
dfkdd_train['attack_class'] = dfkdd_train['attack'].apply(lambda v: mapping[
dfkdd_test['attack_class'] = dfkdd_test['attack'].apply(lambda v: mapping[v]
```

In [7]:
```
# Drop attack field from both train and test data
dfkdd_train.drop(['attack'], axis=1, inplace=True)
dfkdd_test.drop(['attack'], axis=1, inplace=True)
```

In [8]:
```
# View top 3 train data
dfkdd_train.head(3)
```

Out[8]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot
0	0	tcp	ftp_data	SF	491	0	0	0	0	0
1	0	udp	other	SF	146	0	0	0	0	0
2	0	tcp	private	S0	0	0	0	0	0	0

Exploratory Data Analysis

In [9]:
```
# Descriptive statistics
dfkdd_train.describe()
```

Out[9]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent
count	125973.00000	1.259730e+05	1.259730e+05	125973.000000	125973.000000	125973.000000
mean	287.14465	4.556674e+04	1.977911e+04	0.000198	0.022687	0.000111
std	2604.51531	5.870331e+06	4.021269e+06	0.014086	0.253530	0.014366
min	0.00000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000
25%	0.00000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000
50%	0.00000	4.400000e+01	0.000000e+00	0.000000	0.000000	0.000000
75%	0.00000	2.760000e+02	5.160000e+02	0.000000	0.000000	0.000000
max	42908.00000	1.379964e+09	1.309937e+09	1.000000	3.000000	3.000000

In [10]:
```
dfkdd_train['num_outbound_cmds'].value_counts()
dfkdd_test['num_outbound_cmds'].value_counts()
```

Out[10]:
```
0    22544
Name: num_outbound_cmds, dtype: int64
```

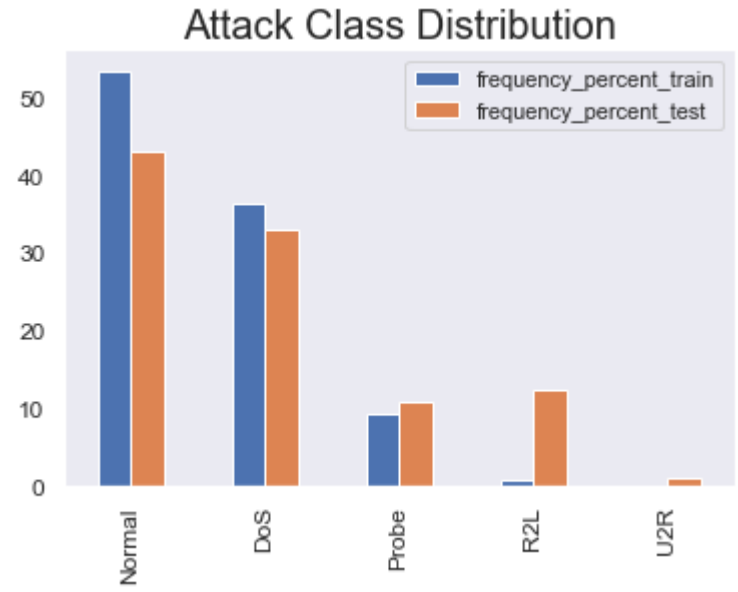
In [11]:
```
# 'num_outbound_cmds' field has all 0 values. Hence, it will be removed from
dfkdd_train.drop(['num_outbound_cmds'], axis=1, inplace=True)
dfkdd_test.drop(['num_outbound_cmds'], axis=1, inplace=True)
```

In [12]:
```
# Attack Class Distribution
attack_class_freq_train = dfkdd_train[['attack_class']].apply(lambda x: x.va
attack_class_freq_test = dfkdd_test[['attack_class']].apply(lambda x: x.valu
attack_class_freq_train['frequency_percent_train'] = round((100 * attack_cla
attack_class_freq_test['frequency_percent_test'] = round((100 * attack_class

attack_class_dist = pd.concat([attack_class_freq_train,attack_class_freq_tes
attack_class_dist
```

Out[12]:

	attack_class	frequency_percent_train	attack_class	frequency_percent_test
Normal	67343	53.46	9711	43.08
DoS	45927	36.46	7458	33.08
Probe	11656	9.25	2421	10.74
R2L	995	0.79	2754	12.22
U2R	52	0.04	200	0.89

In [13]:
```
# Attack class bar plot
plot = attack_class_dist[['frequency_percent_train', 'frequency_percent_test
plot.set_title("Attack Class Distribution", fontsize=20);
plot.grid(color='lightgray', alpha=0.5);
```



In [14]:
```
dfkdd_train.head()
```

Out[14]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot
0	0	tcp	ftp_data	SF	491	0	0	0	0	0
1	0	udp	other	SF	146	0	0	0	0	0
2	0	tcp	private	S0	0	0	0	0	0	0
3	0	tcp	http	SF	232	8153	0	0	0	0
4	0	tcp	http	SF	199	420	0	0	0	0

Scaling Numerical Attributes

In [15]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# extract numerical attributes and scale it to have zero mean and unit varia
cols = dfkdd_train.select_dtypes(include=['float64','int64']).columns
sc_train = scaler.fit_transform(dfkdd_train.select_dtypes(include=['float64'
sc_test = scaler.fit_transform(dfkdd_test.select_dtypes(include=['float64','

# turn the result back to a dataframe
sc_traindf = pd.DataFrame(sc_train, columns = cols)
sc_testdf = pd.DataFrame(sc_test, columns = cols)
```

Encoding of Categorical Attributes

In [16]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

# extract categorical attributes from both training and test sets
cattrain = dfkdd_train.select_dtypes(include=['object']).copy()
cattest = dfkdd_test.select_dtypes(include=['object']).copy()

# encode the categorical attributes
traincat = cattrain.apply(encoder.fit_transform)
testcat = cattest.apply(encoder.fit_transform)

# separate target column from encoded data
enctrain = traincat.drop(['attack_class'], axis=1)
enctest = testcat.drop(['attack_class'], axis=1)

cat_Ytrain = traincat[['attack_class']].copy()
cat_Ytest = testcat[['attack_class']].copy()
```

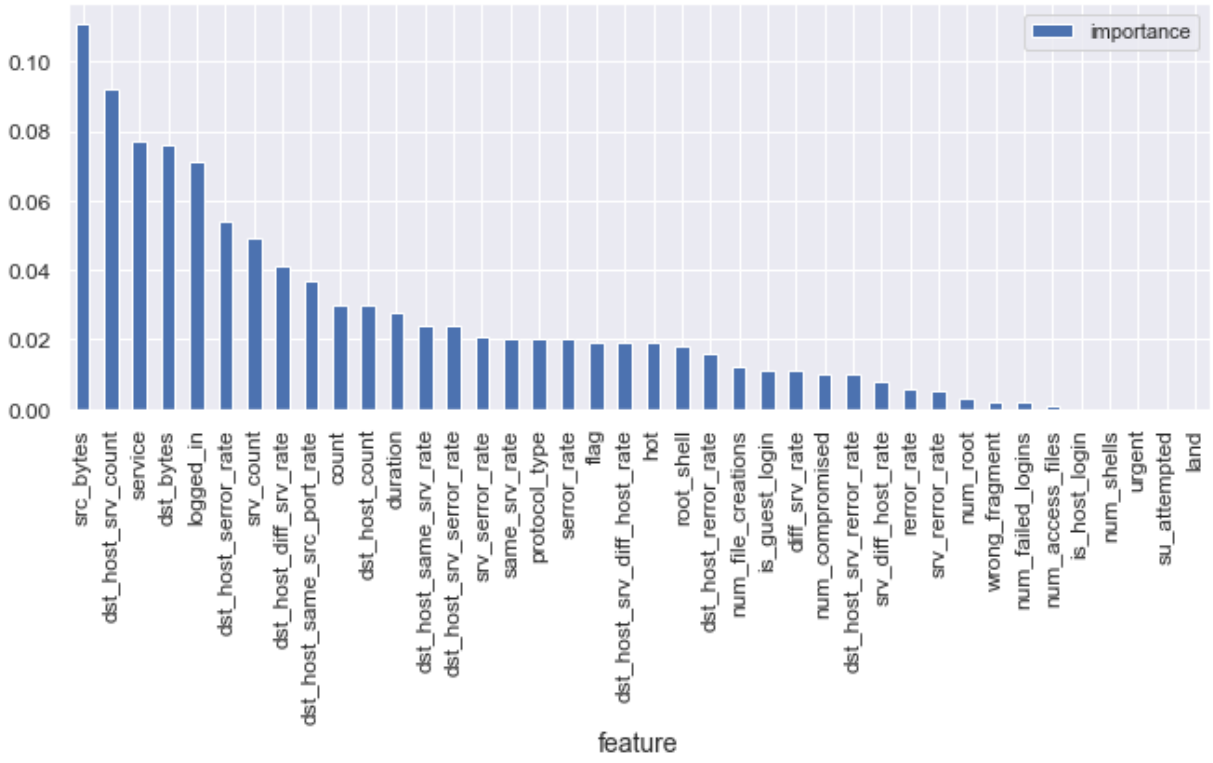
Data Sampling

In [17]:
```
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

# define columns and extract encoded train set for sampling
sc_traindf = dfkdd_train.select_dtypes(include=['float64','int64'])
refclasscol = pd.concat([sc_traindf, enctrain], axis=1).columns
refclass = np.concatenate((sc_train, enctrain.values), axis=1)
X = refclass

# reshape target column to 1D array shape
c, r = cat_Ytest.values.shape
y_test = cat_Ytest.values.reshape(c,)

c, r = cat_Ytrain.values.shape
y = cat_Ytrain.values.reshape(c,)

# apply the random over-sampling
ros = RandomOverSampler(random_state=42)
X_res, y_res = ros.fit_resample(X, y)
print('Original dataset shape {}'.format(Counter(y)))
print('Resampled dataset shape {}'.format(Counter(y_res)))
```

```
Original dataset shape Counter({1: 67343, 0: 45927, 2: 11656, 3: 995, 4: 52})
Resampled dataset shape Counter({1: 67343, 0: 67343, 3: 67343, 2: 67343, 4: 673
43})
```

Feature Selection

In [26]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier();

# fit random forest classifier on the training set
rfc.fit(X_res, y_res);
# extract important features
score = np.round(rfc.feature_importances_,3)
importances = pd.DataFrame({'feature':refclasscol,'importance':score})
importances = importances.sort_values('importance',ascending=False).set_inde
# plot importances
plt.rcParams['figure.figsize'] = (11, 4)
importances.plot.bar();
```



In [27]:
```
from sklearn.feature_selection import RFE
import itertools
rfc = RandomForestClassifier()

# create the RFE model and select 10 attributes
rfe = RFE(rfc, n_features_to_select=10)
rfe = rfe.fit(X_res, y_res)

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

In [18]:
```
selected_features = ['src_bytes','dst_bytes','logged_in','count','srv_count'
selected_features
```

Out[18]:
```
['src_bytes',
 'dst_bytes',
 'logged_in',
 'count',
 'srv_count',
 'dst_host_srv_count',
 'dst_host_diff_srv_rate',
 'dst_host_same_src_port_rate',
 'dst_host_serror_rate',
 'service']
```

Dataset Partition

In [19]:
```
# define columns to new dataframe
newcol = list(refclasscol)
newcol.append('attack_class')

# add a dimension to target
new_y_res = y_res[:, np.newaxis]

# create a dataframe from sampled data
res_arr = np.concatenate((X_res, new_y_res), axis=1)
res_df = pd.DataFrame(res_arr, columns = newcol)

# create test dataframe
reftest = pd.concat([sc_testdf, testcat], axis=1)
reftest['attack_class'] = reftest['attack_class'].astype(np.float64)
reftest['protocol_type'] = reftest['protocol_type'].astype(np.float64)
reftest['flag'] = reftest['flag'].astype(np.float64)
reftest['service'] = reftest['service'].astype(np.float64)

res_df.shape
reftest.shape
```

Out[19]:
```
(22544, 41)
```

In [20]:
```
from collections import defaultdict
classdict = defaultdict(list)

# create two-target classes (normal class and an attack class)
attacklist = [('DoS', 0.0), ('Probe', 2.0), ('R2L', 3.0), ('U2R', 4.0)]
normalclass = [('Normal', 1.0)]

def create_classdict():
    '''This function subdivides train and test dataset into two-class attack
    for j, k in normalclass:
        for i, v in attacklist:
            restrain_set = res_df.loc[(res_df['attack_class'] == k) | (res_d
            classdict[j +'_' + i].append(restrain_set)
            # test labels
            reftest_set = reftest.loc[(reftest['attack_class'] == k) | (reft
            classdict[j +'_' + i].append(reftest_set)

create_classdict()
```

In [21]:
```
for k, v in classdict.items():
    k
```

In [22]:
```
pretrain = classdict['Normal_DoS'][0]
pretest = classdict['Normal_DoS'][1]
grpclass = 'Normal_DoS'
```

Finalize data preprocessing for training

In [23]:

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(handle_unknown='ignore')

Xresdf = pretrain
newtest = pretest

Xresdfnew = Xresdf[selected_features]
Xresdfnum = Xresdfnew.drop(['service'], axis=1)
Xresdfcat = Xresdfnew[['service']].copy()

Xtest_features = newtest[selected_features]
Xtestdfnum = Xtest_features.drop(['service'], axis=1)
Xtestcat = Xtest_features[['service']].copy()


# Fit train data
enc.fit(Xresdfcat)

# Transform train data
X_train_1hotenc = enc.transform(Xresdfcat).toarray()

# Transform test data
X_test_1hotenc = enc.transform(Xtestcat).toarray()

X_train = np.concatenate((Xresdfnum.values, X_train_1hotenc), axis=1)
X_test = np.concatenate((Xtestdfnum.values, X_test_1hotenc), axis=1)

y_train = Xresdf[['attack_class']].copy()
c, r = y_train.values.shape
Y_train = y_train.values.reshape(c,)

y_test = newtest[['attack_class']].copy()
c, r = y_test.values.shape
Y_test = y_test.values.reshape(c,)
```

Train Models

In [25]:

```
from sklearn.svm import SVC
from sklearn.naive_bayes import BernoulliNB
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import VotingClassifier

# Train KNeighborsClassifier Model
KNN_Classifier = KNeighborsClassifier(n_jobs=-1)
KNN_Classifier.fit(X_train, Y_train);

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

# Train Gaussian Naive Baye Model
BNB_Classifier = BernoulliNB()
BNB_Classifier.fit(X_train, Y_train)

# Train Decision Tree Model
DTC_Classifier = tree.DecisionTreeClassifier(criterion='entropy', random_sta
DTC_Classifier.fit(X_train, Y_train);

# Train RandomForestClassifier Model
RF_Classifier = RandomForestClassifier(criterion='entropy', n_jobs=-1, rando
RF_Classifier.fit(X_train, Y_train);

# Train SVM Model
SVC_Classifier = SVC(probability=True,random_state=0)
SVC_Classifier.fit(X_train, Y_train)
print('svc done')
# Train Ensemble Model (This method combines all the individual models above
combined_model = [('Naive Baye Classifier', BNB_Classifier),
                  ('Decision Tree Classifier', DTC_Classifier),
                  ('KNeighborsClassifier', KNN_Classifier),
                  ('LogisticRegression', LGR_Classifier),
                  ('RandomForestClassifier', RF_Classifier),
                  ('SVM Classifier', SVC_Classifier),
                 ]
VotingClassifier =  VotingClassifier(estimators = combined_model,voting = 's
VotingClassifier.fit(X_train, Y_train);
```

```
svc done
```

In [32]:
```
import pickle
models = []
models.append(('SVM Classifier', SVC_Classifier))
models.append(('Naive Baye Classifier', BNB_Classifier))
models.append(('Decision Tree Classifier', DTC_Classifier))
models.append(('RandomForest Classifier', RF_Classifier))
models.append(('KNeighborsClassifier', KNN_Classifier))
models.append(('LogisticRegression', LGR_Classifier))
models.append(('VotingClassifier', VotingClassifier))
for i, v in models:
    pickle.dump(v,open(i,'wb'))
```

Evaluate Models

In [41]:

```
from sklearn import metrics

def plotHeatmap(cm):
    fig, ax = plt.subplots(figsize=(10,10))
    heatmap = sns.heatmap(cm, annot=True, linewidths=0.5, linecolor="lightgr
                          fmt='g', ax=ax, annot_kws={'size': 15}, square=Tru
    heatmap.set_xticklabels(['DoS','Normal','Probe','R2L','U2R'])
    heatmap.set_yticklabels(['DoS','Normal','Probe','R2L','U2R'])
    plt.xlabel("Predicted Group", labelpad=20)
    plt.ylabel("True Group", labelpad=20)
    plt.show()

models = []
models.append(('SVM Classifier', SVC_Classifier))
models.append(('Naive Baye Classifier', BNB_Classifier))
models.append(('Decision Tree Classifier', DTC_Classifier))
models.append(('RandomForest Classifier', RF_Classifier))
models.append(('KNeighborsClassifier', KNN_Classifier))
models.append(('LogisticRegression', LGR_Classifier))
models.append(('VotingClassifier', VotingClassifier))

for i, v in models:
    print()
    print('============================== {} {} Model Evaluation ==========='
    cross_val_score(v, X_train, Y_train, cv=5, n_jobs=-1)
    Y_pred = v.predict(X_train)
    accuracy = metrics.accuracy_score(Y_train, Y_pred)
    print()
    print ("Model Accuracy:" "\n", accuracy)
    confusion_matrix = metrics.confusion_matrix(Y_train, Y_pred)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    classification = metrics.classification_report(Y_train, Y_pred)
    print()
    print("Classification report:" "\n", classification)
    print()
```

```
============================== Normal_DoS SVM Classifier Model Evaluation ===
============================

Model Accuracy:
 0.9891376980532498

Confusion matrix:
 [[66345   998]
 [  465 66878]]

Classification report:
               precision    recall  f1-score   support

         0.0       0.99      0.99      0.99     67343
         1.0       0.99      0.99      0.99     67343

    accuracy                           0.99    134686
   macro avg       0.99      0.99      0.99    134686
```

```
weighted avg       0.99      0.99      0.99    134686



============================== Normal_DoS Naive Baye Classifier Model Evaluat
ion ==============================

Model Accuracy:
 0.9737686173767133

Confusion matrix:
 [[65346  1997]
 [ 1536 65807]]

Classification report:
               precision    recall  f1-score   support

         0.0       0.98      0.97      0.97     67343
         1.0       0.97      0.98      0.97     67343

    accuracy                           0.97    134686
   macro avg       0.97      0.97      0.97    134686
weighted avg       0.97      0.97      0.97    134686



============================== Normal_DoS Decision Tree Classifier Model Eval
uation ==============================

Model Accuracy:
 0.9875480272634127

Confusion matrix:
 [[67343     0]
 [    7 67336]]

Classification report:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     67343
         1.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_DoS RandomForest Classifier Model Evalu
ation ==============================

Model Accuracy:
 0.9933480272634127

Confusion matrix:
 [[67342     1]
 [    6 67337]]
```

```
Classification report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     67343
         1.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_DoS KNeighborsClassifier Model Evaluati
on ==============================

Model Accuracy:
 0.9877577476500898

Confusion matrix:
 [[67287    56]
 [  246 67097]]

Classification report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     67343
         1.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_DoS LogisticRegression Model Evaluation
==============================

Model Accuracy:
 0.980836909552589

Confusion matrix:
 [[65532  1811]
 [  770 66573]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.99      0.97      0.98     67343
         1.0       0.97      0.99      0.98     67343

    accuracy                           0.98    134686
   macro avg       0.98      0.98      0.98    134686
weighted avg       0.98      0.98      0.98    134686
```

```
============================== Normal_DoS VotingClassifier Model Evaluation =
==============================

Model Accuracy:
 0.998522489345589

Confusion matrix:
 [[67205   138]
 [   61 67282]]

Classification report:
               precision    recall  f1-score   support

          0.0       1.00      1.00      1.00     67343
          1.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686
```

Test Models

In [40]:
```
for i, v in models:
    print()
    print('============================== {} {} Model Test Results =========
    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 Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()

```

```
============================== Normal_DoS SVM Classifier Model Test Results ===
==========================

Model Accuracy:
 0.8366241481740346

Confusion matrix:
 [[5272 2186]
 [ 619 9092]]

Classification report:
               precision    recall  f1-score   support

         0.0       0.89      0.71      0.79      7458
         1.0       0.81      0.94      0.87      9711

    accuracy                           0.84     17169
   macro avg       0.85      0.82      0.83     17169
weighted avg       0.84      0.84      0.83     17169



============================== Normal_DoS Naive Baye Classifier Model Test Resu
lts ==============================

Model Accuracy:
 0.8336536781408352

Confusion matrix:
 [[5487 1971]
 [ 885 8826]]

Classification report:
               precision    recall  f1-score   support

         0.0       0.86      0.74      0.79      7458
         1.0       0.82      0.91      0.86      9711

    accuracy                           0.83     17169
   macro avg       0.84      0.82      0.83     17169
```

```
weighted avg       0.84      0.83      0.83     17169


============================== Normal_DoS Decision Tree Classifier Model Test R
esults ==============================

Model Accuracy:
 0.8165880365775525

Confusion matrix:
 [[5591 1867]
 [1282 8429]]

Classification report:
               precision    recall  f1-score   support

         0.0       0.81      0.75      0.78      7458
         1.0       0.82      0.87      0.84      9711

    accuracy                           0.82     17169
   macro avg       0.82      0.81      0.81     17169
weighted avg       0.82      0.82      0.82     17169


============================== Normal_DoS RandomForest Classifier Model Test Re
sults ==============================

Model Accuracy:
 0.829518317898538

Confusion matrix:
 [[5489 1969]
 [ 958 8753]]

Classification report:
               precision    recall  f1-score   support

         0.0       0.85      0.74      0.79      7458
         1.0       0.82      0.90      0.86      9711

    accuracy                           0.83     17169
   macro avg       0.83      0.82      0.82     17169
weighted avg       0.83      0.83      0.83     17169


============================== Normal_DoS KNeighborsClassifier Model Test Resul
ts ==============================

Model Accuracy:
 0.8506200710583027

Confusion matrix:
 [[5787 1671]
 [ 619 9092]]
```

```
Classification report:
              precision    recall  f1-score   support

         0.0       0.90      0.78      0.83      7458
         1.0       0.84      0.94      0.89      9711

    accuracy                           0.87     17169
   macro avg       0.87      0.86      0.86     17169
weighted avg       0.87      0.87      0.86     17169



============================== Normal_DoS LogisticRegression Model Test Results
==============================

Model Accuracy:
 0.8418661541149747

Confusion matrix:
 [[5963 1495]
 [1220 8491]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.83      0.80      0.81      7458
         1.0       0.85      0.87      0.86      9711

    accuracy                           0.84     17169
   macro avg       0.84      0.84      0.84     17169
weighted avg       0.84      0.84      0.84     17169



============================== Normal_DoS VotingClassifier Model Test Results =
==============================

Model Accuracy:
 0.8513017648086668

Confusion matrix:
 [[5604 1854]
 [ 699 9012]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.89      0.75      0.81      7458
         1.0       0.83      0.93      0.88      9711

    accuracy                           0.85     17169
   macro avg       0.86      0.84      0.85     17169
weighted avg       0.86      0.85      0.85     17169
```

In [43]:

```
grpclass = 'Normal_Probe'
KNN_ClassifierProbe = pickle.load(open("./models/binary/Probe/KNeighborsClas
LGR_ClassifierProbe = pickle.load(open("./models/binary/Probe/LogisticRegres
BNB_ClassifierProbe = pickle.load(open("./models/binary/Probe/Naive Baye Cla
DTC_ClassifierProbe = pickle.load(open("./models/binary/Probe/Decision Tree
RF_ClassifierProbe = pickle.load(open("./models/binary/Probe/RandomForest Cl
SVC_ClassifierProbe = pickle.load(open("./models/binary/Probe/SVM Classifier
EnsembleClassifierProbe = pickle.load(open("./models/binary/Probe/EnsembleCl
(X_train, Y_train, X_test, Y_test) = pickle.load(open("./models/binary/Probe
modelsProbe = []
modelsProbe.append(('SVM Classifier', SVC_ClassifierProbe))
modelsProbe.append(('Naive Baye Classifier', BNB_ClassifierProbe))
modelsProbe.append(('Decision Tree Classifier', DTC_ClassifierProbe))
modelsProbe.append(('RandomForest Classifier', RF_ClassifierProbe))
modelsProbe.append(('KNeighborsClassifier', KNN_ClassifierProbe))
modelsProbe.append(('LogisticRegression', LGR_ClassifierProbe))
modelsProbe.append(('VotingClassifier', EnsembleClassifierProbe))
```

In [44]:
```
for i, v in modelsProbe:
    print()
    print('============================== {} {} Model Evaluation ==========
    cross_val_score(v, X_train, Y_train, cv=5, n_jobs=-1)
    Y_pred = v.predict(X_train)
    accuracy = metrics.accuracy_score(Y_train, Y_pred)
    print()
    print ("Model Accuracy:" "\n", accuracy)
    confusion_matrix = metrics.confusion_matrix(Y_train, Y_pred)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    classification = metrics.classification_report(Y_train, Y_pred)
    print()
    print("Classification report:" "\n", classification)
    print()
```

```
============================== Normal_Probe SVM Classifier Model Evaluation ===
============================

Model Accuracy:
 0.9802503600968178

Confusion matrix:
 [[65325  2018]
 [  642 66701]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.99      0.97      0.98     67343
         2.0       0.97      0.99      0.98     67343

    accuracy                           0.98    134686
   macro avg       0.98      0.98      0.98    134686
weighted avg       0.98      0.98      0.98    134686



============================== Normal_Probe Naive Baye Classifier Model Evaluat
ion ==============================

Model Accuracy:
 0.9520143147765915

Confusion matrix:
 [[61583  5760]
 [  703 66640]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.99      0.91      0.95     67343
         2.0       0.92      0.99      0.95     67343

    accuracy                           0.95    134686
```

```
   macro avg       0.95      0.95      0.95    134686
weighted avg       0.95      0.95      0.95    134686



============================== Normal_Probe Decision Tree Classifier Model Eval
uation ==============================

Model Accuracy:
 0.9999925753233446

Confusion matrix:
 [[67342     1]
 [    0 67343]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     67343
         2.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_Probe RandomForest Classifier Model Evalu
ation ==============================

Model Accuracy:
 0.9999925753233446

Confusion matrix:
 [[67342     1]
 [    0 67343]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     67343
         2.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_Probe KNeighborsClassifier Model Evaluati
on ==============================

Model Accuracy:
 0.9982403516326864

Confusion matrix:
 [[67110   233]
```

```
 [    4 67339]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     67343
         2.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_Probe LogisticRegression Model Evaluation
==============================

Model Accuracy:
 0.9680441916754525

Confusion matrix:
 [[64569  2774]
 [ 1530 65813]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.98      0.96      0.97     67343
         2.0       0.96      0.98      0.97     67343

    accuracy                           0.97    134686
   macro avg       0.97      0.97      0.97    134686
weighted avg       0.97      0.97      0.97    134686



============================== Normal_Probe VotingClassifier Model Evaluation =
==============================

Model Accuracy:
 0.9726623405550688

Confusion matrix:
 [[64329  3014]
 [  668 66675]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.99      0.96      0.97     67343
         2.0       0.96      0.99      0.97     67343

    accuracy                           0.97    134686
   macro avg       0.97      0.97      0.97    134686
weighted avg       0.97      0.97      0.97    134686
```

In [45]:
```
for i, v in modelsProbe:
    print()
    print('============================== {} {} Model Test Results =========
    Y_pred = v.predict(X_test)
    accuracy = metrics.accuracy_score(Y_test, Y_pred)
    confusion_matrix = metrics.confusion_matrix(Y_test, Y_pred)
    classification = metrics.classification_report(Y_test, Y_pred)
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
```

```
============================== Normal_Probe SVM Classifier Model Test Results =
==============================

Model Accuracy:
 0.859874711506759

Confusion matrix:
 [[8801  910]
 [ 790 1631]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.92      0.91      0.91      9711
         2.0       0.64      0.67      0.66      2421

    accuracy                           0.86     12132
   macro avg       0.78      0.79      0.78     12132
weighted avg       0.86      0.86      0.86     12132



============================== Normal_Probe Naive Baye Classifier Model Test Re
sults ==============================

Model Accuracy:
 0.827316188592153

Confusion matrix:
 [[8133 1578]
 [ 517 1904]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.94      0.84      0.89      9711
         2.0       0.55      0.79      0.65      2421

    accuracy                           0.83     12132
   macro avg       0.74      0.81      0.77     12132
```

```
weighted avg       0.86      0.83      0.84     12132


============================== Normal_Probe Decision Tree Classifier Model Test
Results ==============================

Model Accuracy:
 0.801763930102209

Confusion matrix:
 [[7658 2053]
 [ 352 2069]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.96      0.79      0.86      9711
         2.0       0.50      0.85      0.63      2421

    accuracy                           0.80     12132
   macro avg       0.73      0.82      0.75     12132
weighted avg       0.87      0.80      0.82     12132


============================== Normal_Probe RandomForest Classifier Model Test
Results ==============================

Model Accuracy:
 0.8205572040883614

Confusion matrix:
 [[8519 1192]
 [ 985 1436]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.90      0.88      0.89      9711
         2.0       0.55      0.59      0.57      2421

    accuracy                           0.82     12132
   macro avg       0.72      0.74      0.73     12132
weighted avg       0.83      0.82      0.82     12132


============================== Normal_Probe KNeighborsClassifier Model Test Res
ults ==============================

Model Accuracy:
 0.8552588196505111

Confusion matrix:
 [[9024  687]
 [1069 1352]]
```

```
Classification report:
              precision    recall  f1-score   support

         1.0       0.89      0.93      0.91      9711
         2.0       0.66      0.56      0.61      2421

    accuracy                           0.86     12132
   macro avg       0.78      0.74      0.76     12132
weighted avg       0.85      0.86      0.85     12132



============================== Normal_Probe LogisticRegression Model Test Resul
ts ==============================

Model Accuracy:
 0.8402571711177053

Confusion matrix:
 [[8370 1341]
 [ 597 1824]]

Classification report:
              precision    recall  f1-score   support

         1.0       0.93      0.86      0.90      9711
         2.0       0.58      0.75      0.65      2421

    accuracy                           0.84     12132
   macro avg       0.75      0.81      0.77     12132
weighted avg       0.86      0.84      0.85     12132



============================== Normal_Probe VotingClassifier Model Test Results
==============================

Model Accuracy:
 0.8781734256511704

Confusion matrix:
 [[8766  945]
 [ 533 1888]]

Classification report:
              precision    recall  f1-score   support

         1.0       0.94      0.90      0.92      9711
         2.0       0.67      0.78      0.72      2421

    accuracy                           0.88     12132
   macro avg       0.80      0.84      0.82     12132
weighted avg       0.89      0.88      0.88     12132
```

Load the trained models for the other attack gourps and evaluate them

In [46]:

```
grpclass = 'Normal_R2L'
KNN_ClassifierR2L = pickle.load(open("./models/binary/R2L/KNeighborsClassifi
LGR_ClassifierR2L = pickle.load(open("./models/binary/R2L/LogisticRegression
BNB_ClassifierR2L = pickle.load(open("./models/binary/R2L/Naive Baye Classif
DTC_ClassifierR2L = pickle.load(open("./models/binary/R2L/Decision Tree Clas
RF_ClassifierR2L = pickle.load(open("./models/binary/R2L/RandomForest Classi
SVC_ClassifierR2L = pickle.load(open("./models/binary/R2L/SVM ClassifierR2L"
EnsembleClassifierR2L = pickle.load(open("./models/binary/R2L/EnsembleClassi
(X_train, Y_train, X_test, Y_test) = pickle.load(open("./models/binary/R2L/d
modelsR2L = []
modelsR2L.append(('SVM Classifier', SVC_ClassifierR2L))
modelsR2L.append(('Naive Baye Classifier', BNB_ClassifierR2L))
modelsR2L.append(('Decision Tree Classifier', DTC_ClassifierR2L))
modelsR2L.append(('RandomForest Classifier', RF_ClassifierR2L))
modelsR2L.append(('KNeighborsClassifier', KNN_ClassifierR2L))
modelsR2L.append(('LogisticRegression', LGR_ClassifierR2L))
modelsR2L.append(('VotingClassifier', EnsembleClassifierR2L))
```

In [47]:
```
for i, v in modelsR2L:
    print()
    print('============================== {} {} Model Evaluation ==========
    cross_val_score(v, X_train, Y_train, cv=5, n_jobs=-1)
    Y_pred = v.predict(X_train)
    accuracy = metrics.accuracy_score(Y_train, Y_pred)
    print()
    print ("Model Accuracy:" "\n", accuracy)
    confusion_matrix = metrics.confusion_matrix(Y_train, Y_pred)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    classification = metrics.classification_report(Y_train, Y_pred)
    print()
    print("Classification report:" "\n", classification)
    print()
```

```
============================== Normal_R2L SVM Classifier Model Evaluation =====
=========================

Model Accuracy:
 0.977547777794277

Confusion matrix:
 [[64972  2371]
 [  653 66690]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.99      0.96      0.98     67343
         3.0       0.97      0.99      0.98     67343

    accuracy                           0.98    134686
   macro avg       0.98      0.98      0.98    134686
weighted avg       0.98      0.98      0.98    134686



============================== Normal_R2L Naive Baye Classifier Model Evaluatio
n ==============================

Model Accuracy:
 0.9458889565359428

Confusion matrix:
 [[60320  7023]
 [  265 67078]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      0.90      0.94     67343
         3.0       0.91      1.00      0.95     67343

    accuracy                           0.95    134686
```

```
   macro avg       0.95      0.95      0.95    134686
weighted avg       0.95      0.95      0.95    134686



============================== Normal_R2L Decision Tree Classifier Model Evalua
tion ==============================

Model Accuracy:
 0.9559889565359428

Confusion matrix:
 [[67343     0]
 [    0 67343]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     67343
         3.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_R2L RandomForest Classifier Model Evaluat
ion ==============================

Model Accuracy:
 0.99935174578343434

Confusion matrix:
 [[67343     0]
 [    0 67343]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     67343
         3.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_R2L KNeighborsClassifier Model Evaluation
==============================

Model Accuracy:
 0.9983517217825164

Confusion matrix:
 [[67121   222]
```

```
 [    0 67343]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     67343
         3.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_R2L LogisticRegression Model Evaluation =
==============================

Model Accuracy:
 0.9655495003192611

Confusion matrix:
 [[63638  3705]
 [  935 66408]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.99      0.94      0.96     67343
         3.0       0.95      0.99      0.97     67343

    accuracy                           0.97    134686
   macro avg       0.97      0.97      0.97    134686
weighted avg       0.97      0.97      0.97    134686



============================== Normal_R2L VotingClassifier Model Evaluation ===
============================

Model Accuracy:
 0.9978097203866771

Confusion matrix:
 [[67048   295]
 [    0 67343]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     67343
         3.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686
```

In [48]:
```
for i, v in modelsR2L:
    print()
    print('============================== {} {} Model Test Results =========
    Y_pred = v.predict(X_test)
    accuracy = metrics.accuracy_score(Y_test, Y_pred)
    confusion_matrix = metrics.confusion_matrix(Y_test, Y_pred)
    classification = metrics.classification_report(Y_test, Y_pred)
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
```

```
============================== Normal_R2L SVM Classifier Model Test Results =
==============================

Model Accuracy:
 0.7994384275972723

Confusion matrix:
 [[9707    4]
 [2496  258]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.80      1.00      0.89      9711
         3.0       0.98      0.09      0.17      2754

    accuracy                           0.80     12465
   macro avg       0.89      0.55      0.53     12465
weighted avg       0.84      0.80      0.73     12465



============================== Normal_R2L Naive Baye Classifier Model Test Re
sults ==============================

Model Accuracy:
 0.7591656638588047

Confusion matrix:
 [[8963  748]
 [2254  500]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.80      0.92      0.86      9711
         3.0       0.40      0.18      0.25      2754

    accuracy                           0.76     12465
   macro avg       0.60      0.55      0.55     12465
```

```
weighted avg       0.71      0.76      0.72     12465



============================== Normal_R2L Decision Tree Classifier Model Test
Results ==============================

Model Accuracy:
 0.7820296831127156

Confusion matrix:
 [[9082  629]
 [2088  666]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.81      0.94      0.87      9711
         3.0       0.51      0.24      0.33      2754

    accuracy                           0.78     12465
   macro avg       0.66      0.59      0.60     12465
weighted avg       0.75      0.78      0.75     12465



============================== Normal_R2L RandomForest Classifier Model Test
Results ==============================

Model Accuracy:
 0.7790613718411552

Confusion matrix:
 [[9711    0]
 [2754    0]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.78      1.00      0.88      9711
         3.0       0.00      0.00      0.00      2754

    accuracy                           0.78     12465
   macro avg       0.39      0.50      0.44     12465
weighted avg       0.61      0.78      0.68     12465



============================== Normal_R2L KNeighborsClassifier Model Test Res
ults ==============================

Model Accuracy:
 0.7869233854793422

Confusion matrix:
 [[9650   61]
 [2595  159]]
```

```
Classification report:
               precision    recall  f1-score   support

         1.0       0.79      0.99      0.88      9711
         3.0       0.72      0.06      0.11      2754

    accuracy                           0.79     12465
   macro avg       0.76      0.53      0.49     12465
weighted avg       0.77      0.79      0.71     12465



============================== Normal_R2L LogisticRegression Model Test Resul
ts ==============================

Model Accuracy:
 0.7985559566787004

Confusion matrix:
 [[9542  169]
 [2342  412]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.80      0.98      0.88      9711
         3.0       0.71      0.15      0.25      2754

    accuracy                           0.80     12465
   macro avg       0.76      0.57      0.57     12465
weighted avg       0.78      0.80      0.74     12465



============================== Normal_R2L VotingClassifier Model Test Results
==============================

Model Accuracy:
 0.8087820296831128

Confusion matrix:
 [[9705    6]
 [2567  187]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.79      1.00      0.88      9711
         3.0       0.97      0.07      0.13      2754

    accuracy                           0.79     12465
   macro avg       0.88      0.53      0.50     12465
weighted avg       0.83      0.79      0.72     12465
```

In [49]:

```
grpclass = 'Normal_U2R'
KNN_ClassifierU2R = pickle.load(open("./models/binary/U2R/KNeighborsClassifi
LGR_ClassifierU2R = pickle.load(open("./models/binary/U2R/LogisticRegression
BNB_ClassifierU2R = pickle.load(open("./models/binary/U2R/Naive Baye Classif
DTC_ClassifierU2R = pickle.load(open("./models/binary/U2R/Decision Tree Clas
RF_ClassifierU2R = pickle.load(open("./models/binary/U2R/RandomForest Classi
SVC_ClassifierU2R = pickle.load(open("./models/binary/U2R/SVM ClassifierU2R"
EnsembleClassifierU2R = pickle.load(open("./models/binary/U2R/EnsembleClassi
(X_train, Y_train, X_test, Y_test) = pickle.load(open("./models/binary/U2R/d
modelsU2R = []
modelsU2R.append(('SVM Classifier', SVC_ClassifierU2R))
modelsU2R.append(('Naive Baye Classifier', BNB_ClassifierU2R))
modelsU2R.append(('Decision Tree Classifier', DTC_ClassifierU2R))
modelsU2R.append(('RandomForest Classifier', RF_ClassifierU2R))
modelsU2R.append(('KNeighborsClassifier', KNN_ClassifierU2R))
modelsU2R.append(('LogisticRegression', LGR_ClassifierU2R))
modelsU2R.append(('VotingClassifier', EnsembleClassifierU2R))
```

In [50]:
```
for i, v in modelsU2R:
    print()
    print('============================== {} {} Model Evaluation ==========
    cross_val_score(v, X_train, Y_train, cv=5, n_jobs=-1)
    Y_pred = v.predict(X_train)
    accuracy = metrics.accuracy_score(Y_train, Y_pred)
    print()
    print ("Model Accuracy:" "\n", accuracy)
    confusion_matrix = metrics.confusion_matrix(Y_train, Y_pred)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    classification = metrics.classification_report(Y_train, Y_pred)
    print()
    print("Classification report:" "\n", classification)
    print()
```

```
============================== Normal_U2R SVM Classifier Model Evaluation ===
============================

Model Accuracy:
 0.9952482069405877

Confusion matrix:
 [[66703   640]
 [    0 67343]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      0.99      1.00     67343
         4.0       0.99      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_U2R Naive Baye Classifier Model Evaluat
ion ==============================

Model Accuracy:
 0.9332150334852917

Confusion matrix:
 [[59684  7659]
 [ 1336 66007]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.98      0.89      0.93     67343
         4.0       0.90      0.98      0.94     67343

    accuracy                           0.93    134686
```

```
   macro avg       0.94      0.93      0.93    134686
weighted avg       0.94      0.93      0.93    134686



============================== Normal_U2R Decision Tree Classifier Model Eval
uation ==============================

Model Accuracy:
 0.9999925753233446

Confusion matrix:
 [[67342     1]
 [    0 67343]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     67343
         4.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_U2R RandomForest Classifier Model Evalu
ation ==============================

Model Accuracy:
 0.9999925753233446

Confusion matrix:
 [[67342     1]
 [    0 67343]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     67343
         4.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_U2R KNeighborsClassifier Model Evaluati
on ==============================

Model Accuracy:
 0.9969139168139228

Confusion matrix:
 [[67291    52]
```

```
 [    0 67343]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     67343
         4.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686



============================== Normal_U2R LogisticRegression Model Evaluation
==============================

Model Accuracy:
 0.9584886328200407

Confusion matrix:
 [[64355  2988]
 [ 2603 64740]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.96      0.96      0.96     67343
         4.0       0.96      0.96      0.96     67343

    accuracy                           0.96    134686
   macro avg       0.96      0.96      0.96    134686
weighted avg       0.96      0.96      0.96    134686



============================== Normal_U2R VotingClassifier Model Evaluation =
==============================

Model Accuracy:
 0.9992352583045009

Confusion matrix:
 [[67240   103]
 [    0 67343]]

Classification report:
               precision    recall  f1-score   support

         1.0       1.00      1.00      1.00     67343
         4.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686
```

In [51]:
```
for i, v in modelsU2R:
    print()
    print('============================== {} {} Model Test Results =========
    Y_pred = v.predict(X_test)
    accuracy = metrics.accuracy_score(Y_test, Y_pred)
    confusion_matrix = metrics.confusion_matrix(Y_test, Y_pred)
    classification = metrics.classification_report(Y_test, Y_pred)
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
```

```
============================== Normal_U2R SVM Classifier Model Test Results =
==============================

Model Accuracy:
 0.9783069316920593

Confusion matrix:
 [[9692   19]
 [ 196    4]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.98      1.00      0.99      9711
         4.0       0.17      0.02      0.04       200

    accuracy                           0.98      9911
   macro avg       0.58      0.51      0.51      9911
weighted avg       0.96      0.98      0.97      9911



============================== Normal_U2R Naive Baye Classifier Model Test Re
sults ==============================

Model Accuracy:
 0.9103016849964686

Confusion matrix:
 [[8982  729]
 [ 160   40]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.98      0.92      0.95      9711
         4.0       0.05      0.20      0.08       200

    accuracy                           0.91      9911
   macro avg       0.52      0.56      0.52      9911
```

```
weighted avg       0.96      0.91      0.94      9911



============================== Normal_U2R Decision Tree Classifier Model Test
Results ==============================

Model Accuracy:
 0.9798204015740086

Confusion matrix:
 [[9711    0]
 [ 200    0]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.98      1.00      0.99      9711
         4.0       0.00      0.00      0.00       200

    accuracy                           0.98      9911
   macro avg       0.49      0.50      0.49      9911
weighted avg       0.96      0.98      0.97      9911



============================== Normal_U2R RandomForest Classifier Model Test
Results ==============================

Model Accuracy:
 0.9798204015740086

Confusion matrix:
 [[9711    0]
 [ 200    0]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.98      1.00      0.99      9711
         4.0       0.00      0.00      0.00       200

    accuracy                           0.98      9911
   macro avg       0.49      0.50      0.49      9911
weighted avg       0.96      0.98      0.97      9911



============================== Normal_U2R KNeighborsClassifier Model Test Res
ults ==============================

Model Accuracy:
 0.9794168096054888

Confusion matrix:
 [[9707    4]
 [ 200    0]]
```

```
Classification report:
               precision    recall  f1-score   support

         1.0       0.98      1.00      0.99      9711
         4.0       0.00      0.00      0.00       200

    accuracy                           0.98      9911
   macro avg       0.49      0.50      0.49      9911
weighted avg       0.96      0.98      0.97      9911



============================== Normal_U2R LogisticRegression Model Test Resul
ts ==============================

Model Accuracy:
 0.9797195035818788

Confusion matrix:
 [[9708    3]
 [ 198    2]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.98      1.00      0.99      9711
         4.0       0.40      0.01      0.02       200

    accuracy                           0.98      9911
   macro avg       0.69      0.50      0.50      9911
weighted avg       0.97      0.98      0.97      9911



============================== Normal_U2R VotingClassifier Model Test Results
==============================

Model Accuracy:
 0.9896204015740086

Confusion matrix:
 [[9711    0]
 [ 200    0]]

Classification report:
               precision    recall  f1-score   support

         1.0       0.98      1.00      0.99      9711
         4.0       0.00      0.00      0.00       200

    accuracy                           0.98      9911
   macro avg       0.49      0.50      0.49      9911
weighted avg       0.96      0.98      0.97      9911
```

In [ ]:
```
1
```