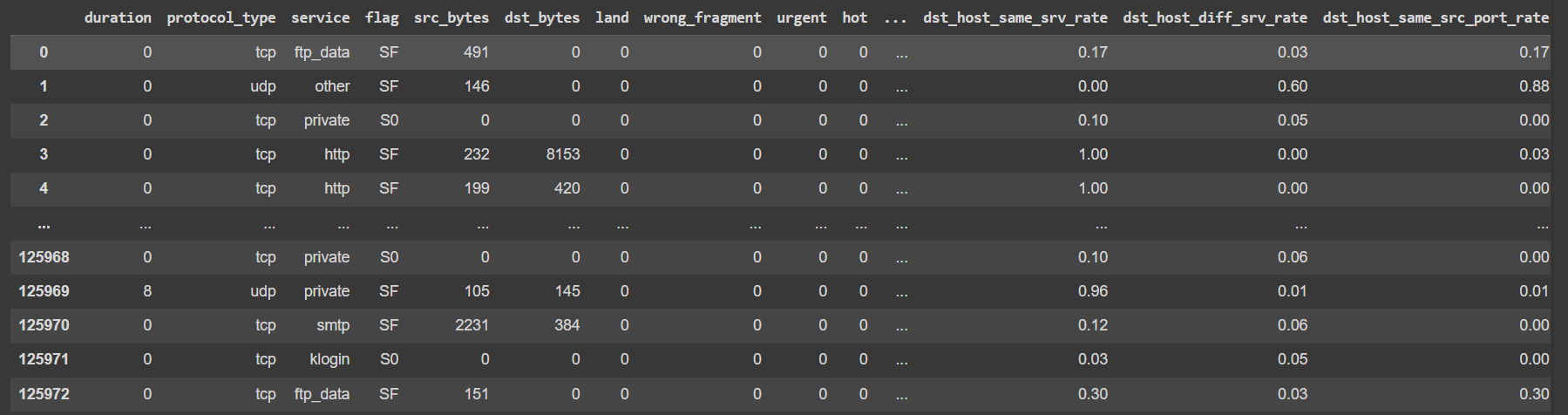
import os  
import sys  
from tempfile import NamedTemporaryFile  
from urllib.request import urlopen  
from urllib.parse import unquote, urlparse  
from urllib.error import HTTPError  
from zipfile import ZipFile  
import tarfile  
import shutil  
  
CHUNK\_SIZE = 40960  
DATA\_SOURCE\_MAPPING = 'nslkdd:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F174616%2F394223%2Fbundle%2Farchive.zip%3FX-Goog-Algorithm%3DGOOG4-RSA-SHA256%26X-Goog-Credential%3Dgcp-kaggle-com%2540kaggle-161607.iam.gserviceaccount.com%252F20240413%252Fauto%252Fstorage%252Fgoog4\_request%26X-Goog-Date%3D20240413T085322Z%26X-Goog-Expires%3D259200%26X-Goog-SignedHeaders%3Dhost%26X-Goog-Signature%'  
  
KAGGLE\_INPUT\_PATH='/kaggle/input'  
KAGGLE\_WORKING\_PATH='/kaggle/working'  
KAGGLE\_SYMLINK='kaggle'  
  
!umount /kaggle/input/ 2> /dev/null  
shutil.rmtree('/kaggle/input', ignore\_errors=True)  
os.makedirs(KAGGLE\_INPUT\_PATH, 0o777, exist\_ok=True)  
os.makedirs(KAGGLE\_WORKING\_PATH, 0o777, exist\_ok=True)  
  
try:  
 os.symlink(KAGGLE\_INPUT\_PATH, os.path.join("..", 'input'), target\_is\_directory=True)  
except FileExistsError:  
 pass  
try:  
 os.symlink(KAGGLE\_WORKING\_PATH, os.path.join("..", 'working'), target\_is\_directory=True)  
except FileExistsError:  
 pass  
  
for data\_source\_mapping in DATA\_SOURCE\_MAPPING.split(','):  
 directory, download\_url\_encoded = data\_source\_mapping.split(':')  
 download\_url = unquote(download\_url\_encoded)  
 filename = urlparse(download\_url).path  
 destination\_path = os.path.join(KAGGLE\_INPUT\_PATH, directory)  
 try:  
 with urlopen(download\_url) as fileres, NamedTemporaryFile() as tfile:  
 total\_length = fileres.headers['content-length']  
 print(f'Downloading {directory}, {total\_length} bytes compressed')  
 dl = 0  
 data = fileres.read(CHUNK\_SIZE)  
 while len(data) > 0:  
 dl += len(data)  
 tfile.write(data)  
 done = int(50 \* dl / int(total\_length))  
 sys.stdout.write(f"\r[{'=' \* done}{' ' \* (50-done)}] {dl} bytes downloaded")  
 sys.stdout.flush()  
 data = fileres.read(CHUNK\_SIZE)  
 if filename.endswith('.zip'):  
 with ZipFile(tfile) as zfile:  
 zfile.extractall(destination\_path)  
 else:  
 with tarfile.open(tfile.name) as tarfile:  
 tarfile.extractall(destination\_path)  
 print(f'\nDownloaded and uncompressed: {directory}')  
 except HTTPError as e:  
 print(f'Failed to load (likely expired) {download\_url} to path {destination\_path}')  
 continue  
 except OSError as e:  
 print(f'Failed to load {download\_url} to path {destination\_path}')  
 continue  
  
print('Data source import complete.')

Downloading nslkdd, 14529600 bytes compressed  
[==================================================] 14529600 bytes downloaded  
Downloaded and uncompressed: nslkdd  
Data source import complete.

# Loading

#loading and importing data  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.model\_selection import train\_test\_split,StratifiedKFold,GridSearchCV  
from sklearn.linear\_model import LogisticRegression  
from sklearn.ensemble import RandomForestClassifier,VotingClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.metrics import accuracy\_score  
from sklearn.metrics import confusion\_matrix  
from sklearn.model\_selection import GridSearchCV,RandomizedSearchCV  
from sklearn.svm import SVC  
from sklearn import metrics  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.metrics import make\_scorer, roc\_auc\_score  
import scipy  
from scipy import stats  
import warnings  
warnings.filterwarnings("ignore")  
from sklearn.preprocessing import StandardScaler  
  
  
# add the column labels  
columns = (['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\_count'  
,'dst\_host\_same\_srv\_rate'  
,'dst\_host\_diff\_srv\_rate'  
,'dst\_host\_same\_src\_port\_rate'  
,'dst\_host\_srv\_diff\_host\_rate'  
,'dst\_host\_serror\_rate'  
,'dst\_host\_srv\_serror\_rate'  
,'dst\_host\_rerror\_rate'  
,'dst\_host\_srv\_rerror\_rate'  
,'attack'  
,'level'])  
  
  
df\_train=pd.read\_csv('../input/nslkdd/KDDTrain+.txt',header=None,names=columns)  
df\_test=pd.read\_csv('../input/nslkdd/KDDTest+.txt',header=None,names=columns)

df\_train

# information of our data  
df\_train.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 125973 entries, 0 to 125972  
Data columns (total 43 columns):

# Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 duration 125973 non-null int64   
 1 protocol\_type 125973 non-null object   
 2 service 125973 non-null object   
 3 flag 125973 non-null object   
 4 src\_bytes 125973 non-null int64   
 5 dst\_bytes 125973 non-null int64   
 6 land 125973 non-null int64   
 7 wrong\_fragment 125973 non-null int64   
 8 urgent 125973 non-null int64   
 9 hot 125973 non-null int64   
 10 num\_failed\_logins 125973 non-null int64   
 11 logged\_in 125973 non-null int64   
 12 num\_compromised 125973 non-null int64   
 13 root\_shell 125973 non-null int64   
 14 su\_attempted 125973 non-null int64   
 15 num\_root 125973 non-null int64   
 16 num\_file\_creations 125973 non-null int64   
 17 num\_shells 125973 non-null int64   
 18 num\_access\_files 125973 non-null int64   
 19 num\_outbound\_cmds 125973 non-null int64   
 20 is\_host\_login 125973 non-null int64   
 21 is\_guest\_login 125973 non-null int64   
 22 count 125973 non-null int64   
 23 srv\_count 125973 non-null int64   
 24 serror\_rate 125973 non-null float64  
 25 srv\_serror\_rate 125973 non-null float64  
 26 rerror\_rate 125973 non-null float64  
 27 srv\_rerror\_rate 125973 non-null float64  
 28 same\_srv\_rate 125973 non-null float64  
 29 diff\_srv\_rate 125973 non-null float64  
 30 srv\_diff\_host\_rate 125973 non-null float64  
 31 dst\_host\_count 125973 non-null int64   
 32 dst\_host\_srv\_count 125973 non-null int64   
 33 dst\_host\_same\_srv\_rate 125973 non-null float64  
 34 dst\_host\_diff\_srv\_rate 125973 non-null float64  
 35 dst\_host\_same\_src\_port\_rate 125973 non-null float64  
 36 dst\_host\_srv\_diff\_host\_rate 125973 non-null float64  
 37 dst\_host\_serror\_rate 125973 non-null float64  
 38 dst\_host\_srv\_serror\_rate 125973 non-null float64  
 39 dst\_host\_rerror\_rate 125973 non-null float64  
 40 dst\_host\_srv\_rerror\_rate 125973 non-null float64  
 41 attack 125973 non-null object   
 42 level 125973 non-null int64

dtypes: float64(15), int64(24), object(4)  
memory usage: 41.3+ MB

# Check for duplicates

print(df\_train.duplicated().sum())  
print(df\_test.duplicated().sum())

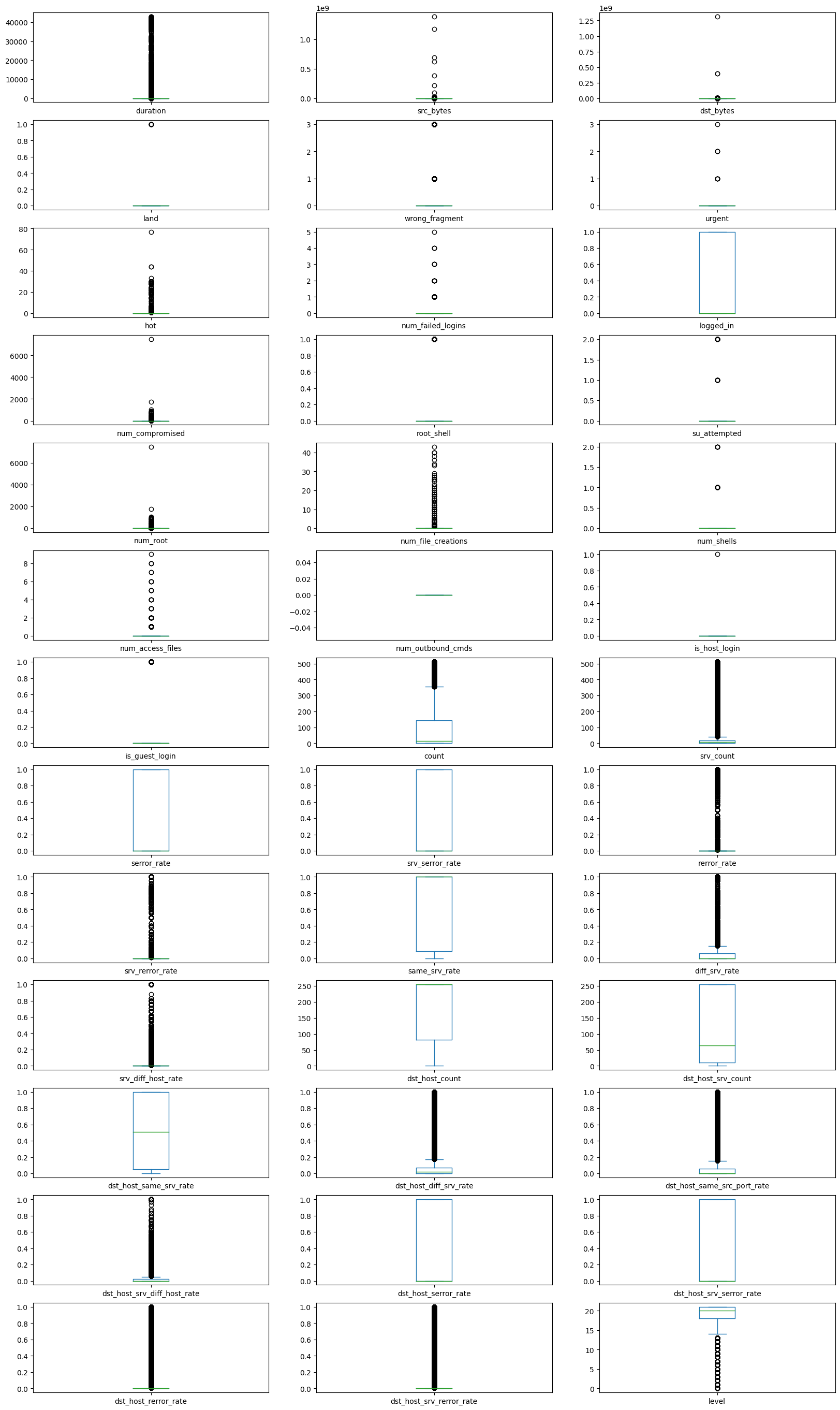
0  
0

df\_train.isnull().sum()

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  
attack 0  
level 0  
dtype: int64

# checking for outliers

#Boxplot of all columns to see outliers  
df\_train.plot(kind='box',subplots=True,layout=(15,3),figsize=(20,40))  
plt.show()



df\_train['attack'].value\_counts()

attack  
normal 67343  
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: count, dtype: int64

# I will convert other abnormal classes to one class  
  
df\_train["binary\_attack"]=df\_train.attack.map(lambda a: "normal" if a == 'normal' else "abnormal")  
df\_train.drop('attack',axis=1,inplace=True)  
  
df\_test["binary\_attack"]=df\_test.attack.map(lambda a: "normal" if a == 'normal' else "abnormal")  
df\_test.drop('attack',axis=1,inplace=True)

df\_train.select\_dtypes(['object']).columns

Index(['protocol\_type', 'service', 'flag', 'binary\_attack'], dtype='object')

# Encoding

# Label Encoder  
from sklearn import preprocessing  
le=preprocessing.LabelEncoder()  
clm=['protocol\_type', 'service', 'flag', 'binary\_attack']  
for x in clm:  
 df\_train[x]=le.fit\_transform(df\_train[x])  
 df\_test[x]=le.fit\_transform(df\_test[x])

# Spliting

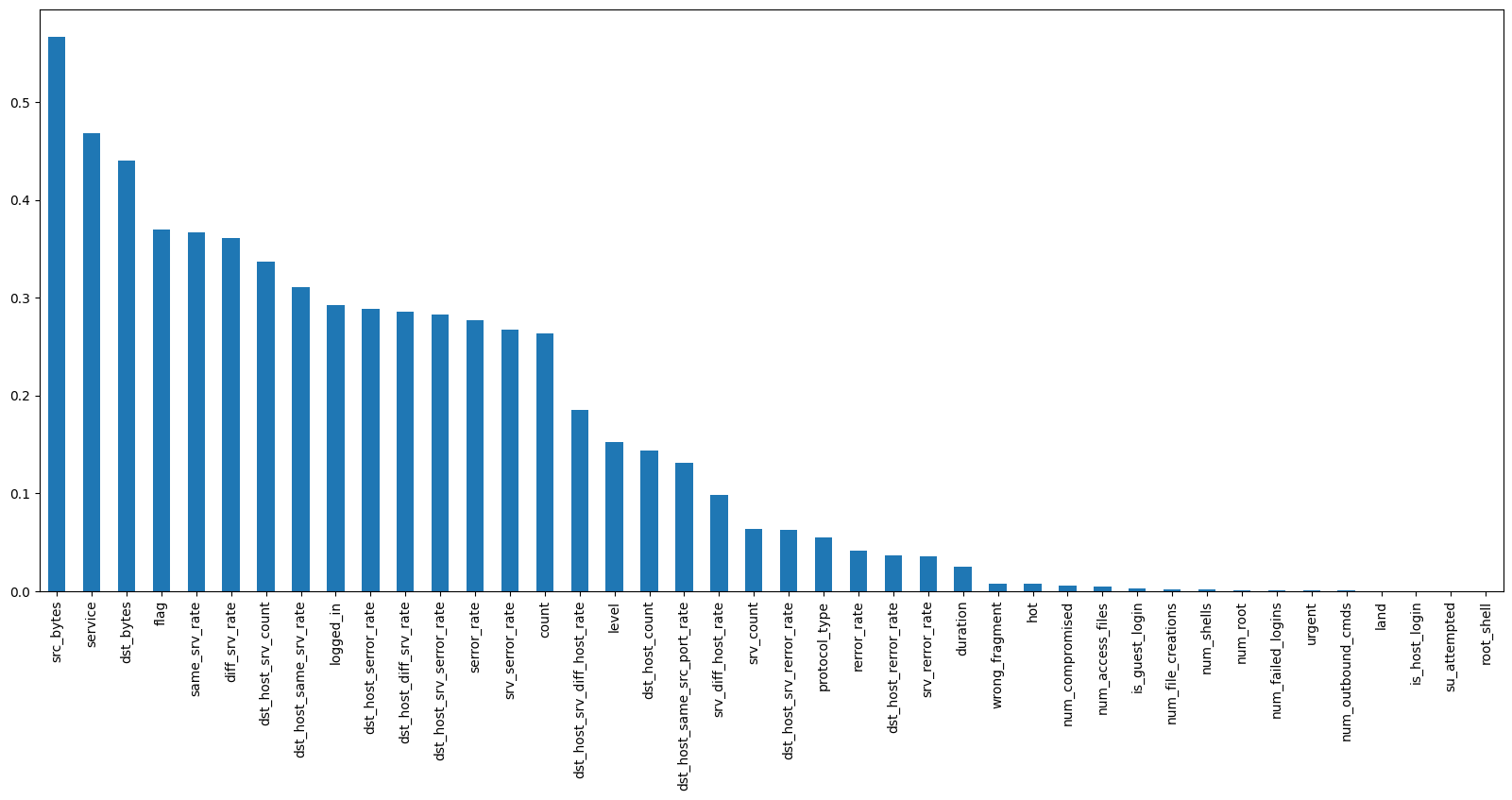
#Spliting the data  
  
x\_train=df\_train.drop('binary\_attack',axis=1)  
y\_train=df\_train["binary\_attack"]  
  
x\_test=df\_test.drop('binary\_attack',axis=1)  
y\_test=df\_test["binary\_attack"]

# Feature Engineering

from sklearn.feature\_selection import mutual\_info\_classif  
mutual\_info = mutual\_info\_classif(x\_train, y\_train)  
mutual\_info = pd.Series(mutual\_info)  
mutual\_info.index = x\_train.columns  
mutual\_info.sort\_values(ascending=False)

src\_bytes 0.566350  
service 0.468260  
dst\_bytes 0.439951  
flag 0.369316  
same\_srv\_rate 0.367211  
diff\_srv\_rate 0.360573  
dst\_host\_srv\_count 0.336736  
dst\_host\_same\_srv\_rate 0.310676  
logged\_in 0.291961  
dst\_host\_serror\_rate 0.288241  
dst\_host\_diff\_srv\_rate 0.285672  
dst\_host\_srv\_serror\_rate 0.282648  
serror\_rate 0.277372  
srv\_serror\_rate 0.267642  
count 0.263547  
dst\_host\_srv\_diff\_host\_rate 0.185727  
level 0.152463  
dst\_host\_count 0.143691  
dst\_host\_same\_src\_port\_rate 0.131663  
srv\_diff\_host\_rate 0.098324  
srv\_count 0.063181  
dst\_host\_srv\_rerror\_rate 0.062184  
protocol\_type 0.055352  
rerror\_rate 0.041890  
dst\_host\_rerror\_rate 0.036344  
srv\_rerror\_rate 0.035635  
duration 0.024879  
wrong\_fragment 0.007604  
hot 0.007384  
num\_compromised 0.005702  
num\_access\_files 0.004444  
is\_guest\_login 0.003038  
num\_file\_creations 0.001993  
num\_shells 0.001881  
num\_root 0.001138  
num\_failed\_logins 0.000907  
urgent 0.000590  
num\_outbound\_cmds 0.000403  
land 0.000247  
is\_host\_login 0.000000  
su\_attempted 0.000000  
root\_shell 0.000000  
dtype: float64

mutual\_info.sort\_values(ascending=False).plot.bar(figsize=(20, 8));

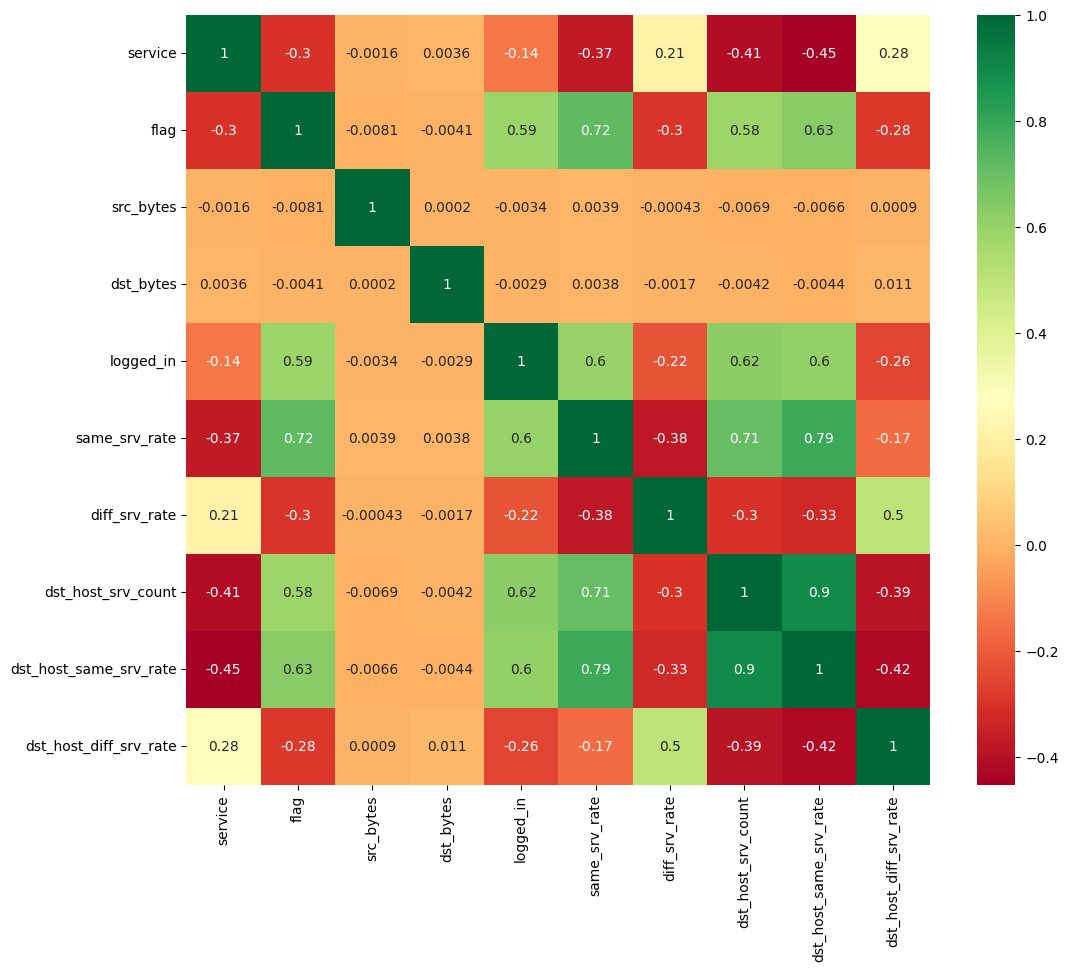


# I will choose 20 features to select  
from sklearn.feature\_selection import SelectKBest  
sel\_five\_cols = SelectKBest(mutual\_info\_classif, k=20)  
sel\_five\_cols.fit(x\_train, y\_train)  
x\_train.columns[sel\_five\_cols.get\_support()]

Index(['service', 'flag', 'src\_bytes', 'dst\_bytes', 'logged\_in', 'count',  
 'serror\_rate', 'srv\_serror\_rate', 'same\_srv\_rate', 'diff\_srv\_rate',  
 'srv\_diff\_host\_rate', 'dst\_host\_count', 'dst\_host\_srv\_count',  
 'dst\_host\_same\_srv\_rate', 'dst\_host\_diff\_srv\_rate',  
 'dst\_host\_same\_src\_port\_rate', 'dst\_host\_srv\_diff\_host\_rate',  
 'dst\_host\_serror\_rate', 'dst\_host\_srv\_serror\_rate', 'level'],  
 dtype='object')

col=['service', 'flag', 'src\_bytes', 'dst\_bytes', 'logged\_in',  
 'same\_srv\_rate', 'diff\_srv\_rate', 'dst\_host\_srv\_count',  
 'dst\_host\_same\_srv\_rate', 'dst\_host\_diff\_srv\_rate']  
x\_train=x\_train[col]  
x\_test=x\_test[col]

plt.figure(figsize=(12,10))  
p=sns.heatmap(x\_train.corr(), annot=True,cmap ='RdYlGn')



# Scaling

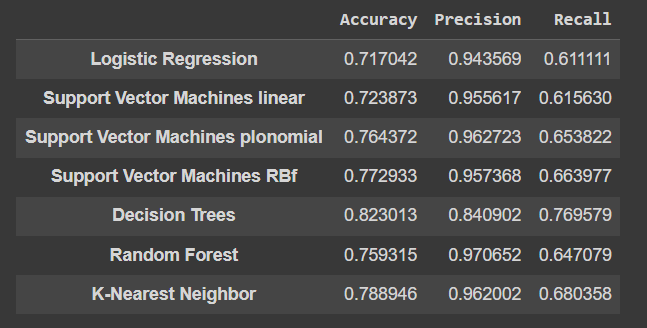
from sklearn.preprocessing import MinMaxScaler  
scaler = MinMaxScaler()  
x\_train= scaler.fit\_transform(x\_train)  
x\_test= scaler.fit\_transform(x\_test)

# Binary classification models

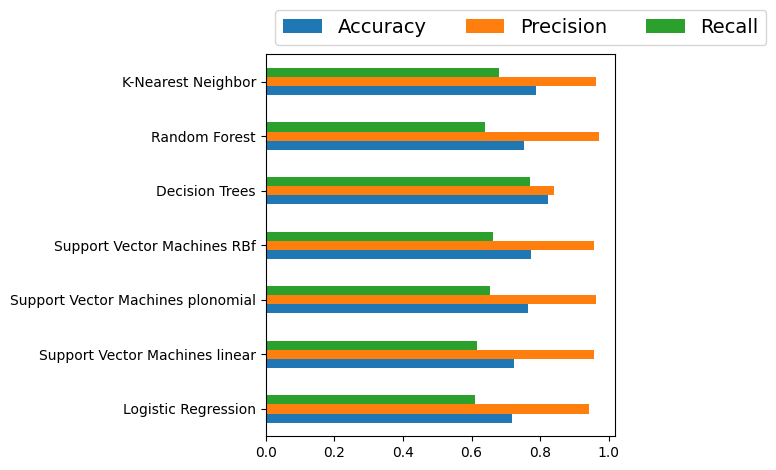
# we'll initialize each model and store it by name in a dictionary  
models = {}  
  
# Logistic Regression  
from sklearn.linear\_model import LogisticRegression  
models['Logistic Regression'] = LogisticRegression()  
  
# Support Vector Machines  
from sklearn.svm import LinearSVC  
models['Support Vector Machines linear'] = LinearSVC()  
models['Support Vector Machines plonomial'] = SVC(kernel='poly')  
models['Support Vector Machines RBf'] = SVC(C=100.0)  
  
# Decision Trees  
from sklearn.tree import DecisionTreeClassifier  
models['Decision Trees'] = DecisionTreeClassifier(max\_depth=3)  
  
# Random Forest  
from sklearn.ensemble import RandomForestClassifier  
models['Random Forest'] = RandomForestClassifier()  
  
# K-Nearest Neighbors  
from sklearn.neighbors import KNeighborsClassifier  
models['K-Nearest Neighbor'] = KNeighborsClassifier(n\_neighbors=20)

# we'll loop over each one, train it by calling .fit(), make predictions, calculate metrics, and store each result in a dictionary.  
  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score  
  
accuracy, precision, recall = {}, {}, {}  
  
for key in models.keys():  
  
 # Fit the classifier  
 models[key].fit(x\_train, y\_train)  
  
 # Make predictions  
 predictions = models[key].predict(x\_test)  
  
 # Calculate metrics  
 accuracy[key] = accuracy\_score(predictions, y\_test)  
 precision[key] = precision\_score(predictions, y\_test)  
 recall[key] = recall\_score(predictions, y\_test)

df\_model = pd.DataFrame(index=models.keys(), columns=['Accuracy', 'Precision', 'Recall'])  
df\_model['Accuracy'] = accuracy.values()  
df\_model['Precision'] = precision.values()  
df\_model['Recall'] = recall.values()  
  
df\_model



ax = df\_model.plot.barh()  
ax.legend(  
 ncol=len(models.keys()),  
 bbox\_to\_anchor=(0, 1),  
 loc='lower left',  
 prop={'size': 14}  
)  
plt.tight\_layout()

[](https://www.kaggleusercontent.com/kf/105881659/eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..JcEgiIXSkPIHERgsJfQ_ig.Ticmp8kpJklg2OsVm7RpHx9cryeebqi79MoP1RZvctqu0wejS81UK8S7RFjXu8tPqRmB_8zoErMMgCLm3KF42V1eMCCeDKnMIDRlRvqyb_LeEHJ83NgejAXVkdaomYbVDm1tPkkJ9b9y1qLh1jPaXLpWUt5Z9WLX30Cig1ZV3k25iM47N40IPuGp1_ow7zJ2sSXanqBSM1WZK4FS8FlesuWlzWO-zF9weyGpxXVexy0XZqRzfDaDW3TByHnL_hY7OlWwqLfAmDj95cUJFHRqp0Vs_-5haqoYoYWdGmT5Z2tSPdToEn3tVjpjyDs7qVRm1Mo1uCOKcY6NjeSfV4cV20hH3zr-73YZrNC5aZKYb5y_-53zzSGdXzeCoBsERjM_tOYIhAYYxKUAt4Dm5Oiicwfe8IUAeoAOKpQbYmIOucvuSjBoCKWGaDkeyFDNdT8pxNtMYxVUofMy5wa6fL0H0W7IUPq0P6JwmJcEzMYjsu054azxLHMkBGBNPQHSAlLPNuGAwnTl-v2PDKR3vYvDFNNv4Y2tnUU9JTuM1mVWkJWWglLNbNha2HTSTOTIsYzozg3G7SiRO-doVzpvQIj5h46nqT-_CnMBYDhVjDZak0g8ykxG63mHl81O9FUV_uJvowj-U5ZgWT3-zuUxICRl2o8an8tOziTPKRmhqNOFgBjrNHYcB97-I__UNZyFocZI.tKvX0tRdHhCCg5_eFT55UQ/__results___files/__results___32_0.png)

from sklearn.metrics import plot\_roc\_curve

for key in models.keys():

plot\_roc\_curve( models[key], x\_test, y\_test)

