In [1]:
```
import csv
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
```

defining column names

In [2]:
```
columns=['duration','protocol_type','service','flag','src_bytes','dst_bytes','land'
,'wrong_fragment','urgent','hot','num_failed_logins','logged_in','num_compromised','root_she
ll','su_attempted','num_root','num_file_creations','num_shells','num_access_files','num_outb
ound_cmds','is_host_login','is_guest_login','count','srv_count','serror_rate','srv_serror_ra
te','rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','srv_diff_host_rate','ds
t_host_count','dst_host_srv_count','dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_ho
st_same_src_port_rate','dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_srv_se
rror_rate','dst_host_rerror_rate','dst_host_srv_rerror_rate']
test_columns=columns
train_columns=columns+['target']
print(len(columns))
```

42

In [3]:
```
df = pd.read_csv("full.csv", names = train_columns[:])

df.head()
```

Out[3]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	ds
0	0	tcp	http	SF	215	45076	0	0	0	0	...	0	
1	0	tcp	http	SF	162	4528	0	0	0	0	...	1	
2	0	tcp	http	SF	236	1228	0	0	0	0	...	2	
3	0	tcp	http	SF	233	2032	0	0	0	0	...	3	
4	0	tcp	http	SF	239	486	0	0	0	0	...	4	

5 rows × 42 columns

finding categorical columns and then encoding them

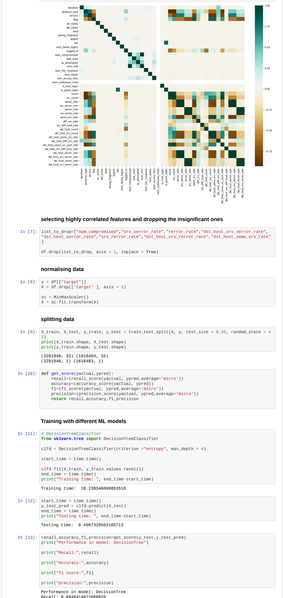
In [4]:
```
numericol_cols = df._get_numeric_data().columns

categoricol_cols = list(set(df.columns) -set(numericol_cols))

categoricol_cols.remove('target')
print(categoricol_cols)
```

['service', 'protocol_type', 'flag']

In [5]:
```
df["protocol_type"]= df["protocol_type"].astype('category')
df["protocol_type"] = df["protocol_type"].cat.codes
df["service"]= df["service"].astype('category')
df["service"] = df["service"].cat.codes
# df2["service"]
df["flag"]= df["flag"].astype('category')
df["flag"] = df["flag"].cat.codes
```

feature selection using correlation metrics

In [6]:
```
corr = df.corr()
plt.figure(figsize=(15, 12))
sns.heatmap(corr,cmap='BrBG')
plt.show()
```



selecting highly correlated features and dropping the insignificant ones

In [7]:
```
list_to_drop=['num_outbound_cmds','srv_serror_rate','rerror_rate','dst_host_srv_serror_rate',
'dst_host_serror_rate','srv_rerror_rate','dst_host_srv_rerror_rate','dst_host_same_srv_rate'
]

df.drop(list_to_drop, axis = 1, inplace = True)
```

normalising data

In [8]:
```
y = df['target']
X = df.drop('target', axis = 1)

sc = MinMaxScaler()
X = sc.fit_transform(X)
```

splitting data

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 4
2)
print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

(329349, 33) (116140, 33)
(329349,) (116140,)

In [10]:
```
def get_score(actual,pred):
    recall = recall_score(actual, pred,average='micro')
    accuracy=accuracy_score(actual, pred)
    f1=(f1_score(actual, pred,average='micro'))
    precision=precision_score(actual, pred,average='micro')
    return recall,accuracy,f1,precision
```

Training with different ML models

In [11]:
```
# DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier

clf4 = DecisionTreeClassifier(criterion ="entropy", max_depth = 4)

start_time = time.time()

clf4.fit(X_train, y_train.values.ravel())
end_time = time.time()
print("Training time: ", end_time-start_time)
```

Training time: 18.238346099853516

In [12]:
```
start_time = time.time()
y_test_pred = clf4.predict(X_test)
end_time = time.time()
print("Testing time: ", end_time-start_time)
```

Testing time: 0.4867329692840576

In [13]:
```
recall,accuracy,f1,precision=get_score(y_test,y_test_pred)
print("Performance in model: DecisionTree")

print("Recall:",recall)

print("Accuracy:",accuracy)

print("f1 score:",f1)

print("precision:",precision)
```

Performance in model: DecisionTree
Recall: 0.9949141871668926
Accuracy: 0.9949141871668926
f1 score: 0.9949141871668926
precision: 0.9949141871668926

In [14]:
```
# RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

clf7 = RandomForestClassifier(n_estimators = 20)
start_time = time.time()
clf7.fit(X_train, y_train.values.ravel())
end_time = time.time()
print("Training time: ", end_time-start_time)
```

Training time: 102.05997931993323

In [15]:
```
start_time = time.time()
y_test_pred = clf7.predict(X_test)
end_time = time.time()
print("Testing time: ", end_time-start_time)
```

Testing time: 10.290551185607666

In [16]:
```
recall,accuracy,f1,precision=get_score(y_test,y_test_pred)
print("Performance in model: RandomForestClassifier")

print("Recall:",recall)

print("Accuracy:",accuracy)

print("f1 score:",f1)

print("precision:",precision)
```

Performance in model: RandomForestClassifier
Recall: 0.9998975265479438
Accuracy: 0.9998975265479438
f1 score: 0.9998975265479438
precision: 0.9998975265479438

In [17]:
```
# LogisticRegression
from sklearn.linear_model import LogisticRegression

clf1 = LogisticRegression(max_iter = 1000000)
start_time = time.time()
clf1.fit(X_train, y_train.values.ravel())
end_time = time.time()
print("Training time: ", end_time-start_time)
```

Training time: 2332.4567892313385

In [18]:
```
start_time = time.time()
y_test_pred = clf1.predict(X_test)
end_time = time.time()
print("Testing time: ", end_time-start_time)
```

Testing time: 1.1773362159729004

In [19]:
```
recall,accuracy,f1,precision=get_score(y_test,y_test_pred)
print("Performance in model: LogisticRegression")

print("Recall:",recall)

print("Accuracy:",accuracy)

print("f1 score:",f1)

print("precision:",precision)
```

Performance in model: LogisticRegression
Recall: 0.9900575704487168
Accuracy: 0.9900575704487168
f1 score: 0.9900575704487168
precision: 0.9900575704487168

Selecting the best model of the above based on train-test scores - Random Forest classifier

Training RFC on the complete data

In [21]:
```
# RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

clf= RandomForestClassifier(n_estimators = 50)
start_time = time.time()
clf.fit(X, y.values.ravel())
# clf.fit(X_train, y_train.values.ravel())
end_time = time.time()
print("Training time: ", end_time-start_time)
```

Training time: 181.05042990875244

Reading test data, data cleaning, encoding and feature selection

In [22]:
```
df2=pd.read_csv("test.csv",names = test_columns)
# df2

df2.drops('index', axis = 1, inplace = True)
df2.head()
```

/opt/anaconda3/envs/ri/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3147: DtypeWarning: Columns (1,2,3,6,11,20,21,34,35,37,38,39,40,41) have mixed types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)

Out[22]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_count	dst_h
0	0	tcp	ftp_data	SF	105	146	0.0	0.0	0.0	0.0	...	255	
1	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255	
2	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255	
3	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255	
4	0	udp	private	SF	105	146	0.0	0.0	0.0	0.0	...	255	

5 rows × 41 columns

In [23]:
```
df2=df2.iloc[:]
df2.drop(list_to_drop, axis = 1, inplace = True)

df2["flag"]= df2["flag"].astype('category')
df2["flag"] = df2["flag"].cat.codes

df2["service"]= df2["service"].astype('category')
df2["service"] = df2["service"].cat.codes

df2["protocol_type"]= df2["protocol_type"].astype('category')
df2["protocol_type"] = df2["protocol_type"].cat.codes
```

In [24]:
```
df2_x=sc.fit_transform(df2)
```

Predicting on the test data

In [25]:
```
start_time = time.time()
y_test_pred = clf7.predict(df2_x)
end_time = time.time()
print("Testing time: ", end_time-start_time)
```

Testing time: 3.4897217941284

In [26]:
```
df2=pd.DataFrame(y_test_pred)
df2
```

Out[26]:

	0
0	normal.
1	normal.
2	normal.
3	normal.
4	normal.
...	...
311029	normal.
311030	normal.
311031	normal.
311032	normal.
311033	normal.

311034 rows × 1 columns

In [27]:
```
df3=pd.DataFrame([["normal."]])
df3
df3=pd.concat([df3,df2])
df3
# df3
```

Out[27]:

	0
0	normal.
0	normal.
1	normal.
2	normal.
3	normal.
...	...
311029	normal.
311030	normal.
311031	normal.
311032	normal.
311033	normal.

311034 rows × 1 columns

In [28]:
```
df3.insert(0, "index", range(311029), True)
df3
```

Out[28]:

	index	0
0	0	normal.
0	1	normal.
1	2	normal.
2	3	normal.
3	4	normal.
...	...	...
311029	311024	normal.
311030	311025	normal.
311031	311026	normal.
311032	311027	normal.
311033	311028	normal.

311034 rows × 2 columns

In [29]:
```
df3=df3.rename(columns={0:"target"},inplace=False)
df3
```

Out[29]:

	index	target
0	0	normal.
0	1	normal.
1	2	normal.
2	3	normal.
3	4	normal.
...	...	...
311029	311024	normal.
311030	311025	normal.
311031	311026	normal.
311032	311027	normal.
311033	311028	normal.

311034 rows × 2 columns

In [ ]:

In [30]:
```
# df7=pd.read_csv("testlabel2.csv")
df7.to_csv("testlabel4.csv",index=False)
```

In [31]:
```
set(df7["target"])
```

Out[31]:
```
{'back.',
 'ipsweep.',
 'land.',
 'neptune.',
 'nmap.',
 'normal.',
 'pod.',
 'portsweep.',
 'satan.',
 'smurf.',
 'teardrop.',
 'warezclient.',
 'warezmaster.'}
```

In [ ]: