```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

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 = 'kdd-cup-1999-data:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F2354076%2F3966369%2Fbundle%2Farchive.zip%3FX-Goog

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(':')
```

```
    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 kdd-cup-1999-data, 21381002 bytes compressed
[==================================================] 21381002 bytes downloaded
Downloaded and uncompressed: kdd-cup-1999-data
Data source import complete.
```

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time


# reading features list

with open("../input/kdd-cup-1999-data/kddcup_names.csv", 'r') as f:
    print(f.read())
```

```
back,buffer_overflow,ftp_write,guess_passwd,imap,ipsweep,land,loadmodule,multihop,neptune,nmap,normal,perl,phf,pod,portsweep,rootkit,satan,smu
duration: continuous.
protocol_type: symbolic.
service: symbolic.
flag: symbolic.
src_bytes: continuous.
dst_bytes: continuous.
land: symbolic.
wrong_fragment: continuous.
urgent: continuous.
hot: continuous.
num_failed_logins: continuous.
logged_in: symbolic.
num_compromised: continuous.
root_shell: continuous.
su_attempted: continuous.
num_root: continuous.
num_file_creations: continuous.
num_shells: continuous.
num_access_files: continuous.
num_outbound_cmds: continuous.
is_host_login: symbolic.
is_guest_login: symbolic.
count: continuous.
srv_count: continuous.
serror_rate: continuous.
srv_serror_rate: continuous.
rerror_rate: continuous.
srv_rerror_rate: continuous.
same_srv_rate: continuous.
```

```
diff_srv_rate: continuous.
srv_diff_host_rate: continuous.
dst_host_count: continuous.
dst_host_srv_count: continuous.
dst_host_same_srv_rate: continuous.
dst_host_diff_srv_rate: continuous.
dst_host_same_src_port_rate: continuous.
dst_host_srv_diff_host_rate: continuous.
dst_host_serror_rate: continuous.
dst_host_srv_serror_rate: continuous.
dst_host_rerror_rate: continuous.
dst_host_srv_rerror_rate: continuous.
```

```
cols ="""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"""
```

```
columns =[]

for c in cols.split(', '):
    if(c.strip()):
       columns.append(c.strip())


columns.append('target')
print(len(columns))
```

```
42
```

```
with open("../input/kdd-cup-1999-data/training_attack_types.csv", 'r') as f:
    print(f.read())
```

```
back dos
buffer_overflow u2r
ftp_write r2l
guess_passwd r2l
imap r2l
ipsweep probe
land dos
loadmodule u2r
multihop r2l
neptune dos
nmap probe
perl u2r
phf r2l
pod dos
portsweep probe
rootkit u2r
satan probe
smurf dos
spy r2l
teardrop dos
warezclient r2l
warezmaster r2l
```

```
attacks_types = {

    'normal': 'normal',

'back': 'dos',

'buffer_overflow': 'u2r',

'ftp_write': 'r2l',

'guess_passwd': 'r2l',

'imap': 'r2l',

'ipsweep': 'probe',

'land': 'dos',

'loadmodule': 'u2r',

'multihop': 'r2l',

'neptune': 'dos',

'nmap': 'probe',

'perl': 'u2r',

'phf': 'r2l',

'pod': 'dos',

'portsweep': 'probe',

'rootkit': 'u2r',

'satan': 'probe',

'smurf': 'dos',

'spy': 'r2l',
```

```
'teardrop': 'dos',

'warezclient': 'r2l',

'warezmaster': 'r2l',
}
```

```
path = "../input/kdd-cup-1999-data/kddcup.data_10_percent_corrected"
df = pd.read_csv(path, names = columns)

# Adding Attack Type column
df['Attack Type'] = df.target.apply(lambda r:attacks_types[r[:-1]])
df.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate
0	0	tcp	http	SF	181	5450	0	0	0	0	...	1.0
1	0	tcp	http	SF	239	486	0	0	0	0	...	1.0
2	0	tcp	http	SF	235	1337	0	0	0	0	...	1.0
3	0	tcp	http	SF	219	1337	0	0	0	0	...	1.0
4	0	tcp	http	SF	217	2032	0	0	0	0	...	1.0

5 rows × 43 columns

```
# Finding categorical features

num_cols = df._get_numeric_data().columns
cate_cols = list(set(df.columns)-set(num_cols))
cate_cols.remove('target')
cate_cols.remove('Attack Type')

cate_cols
```

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

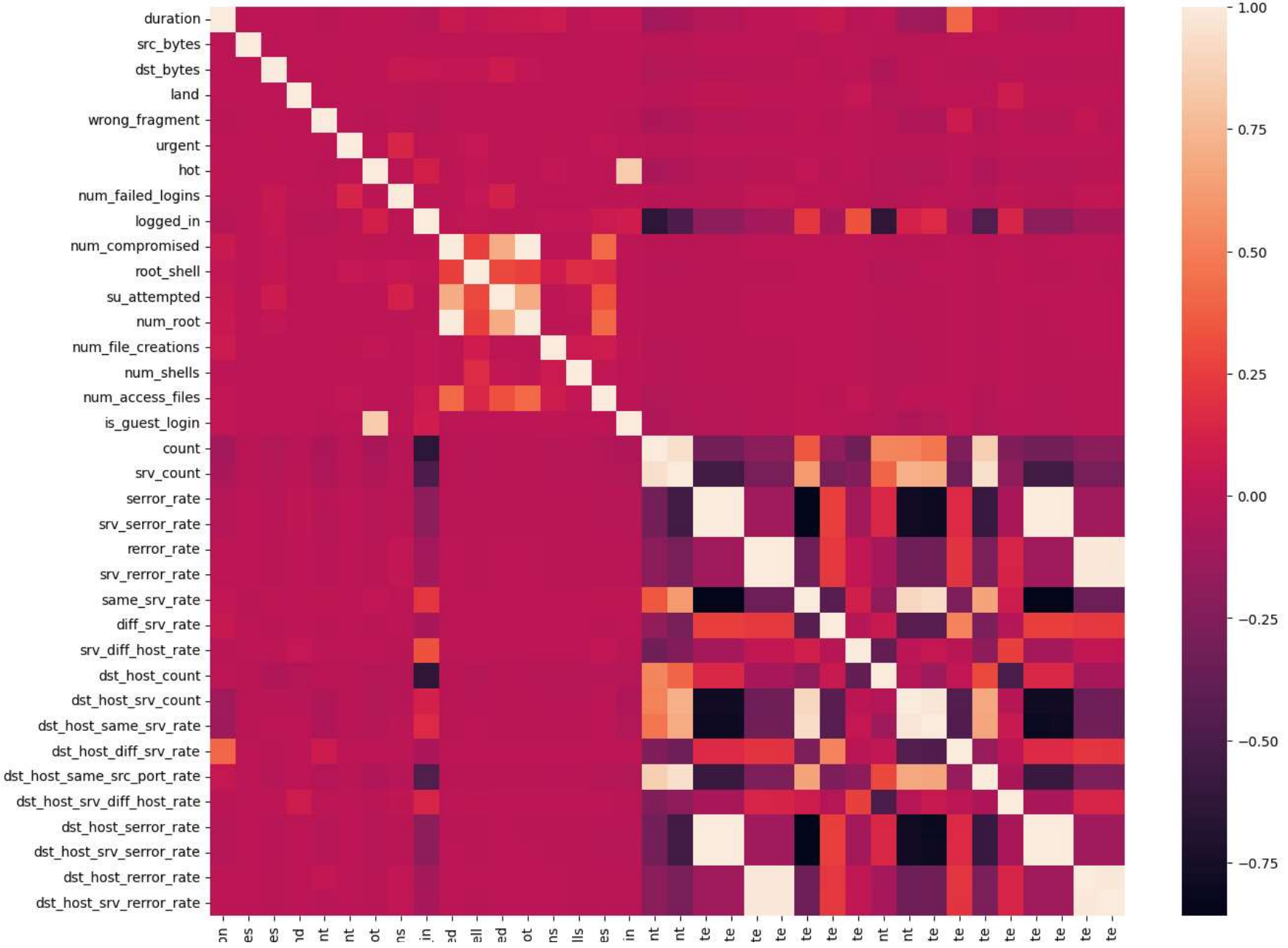
```
df = df.dropna('columns')# drop columns with NaN

df = df[[col for col in df if df[col].nunique() > 1]]# keep columns where there are more than 1 unique values

corr = df.corr()

plt.figure(figsize =(15, 12))
sns.heatmap(corr)
plt.show()
```

```
<ipython-input-8-9406440abbf9>:1: FutureWarning: In a future version of pandas all arguments of DataFrame.dropna will be keyworc
  df = df.dropna('columns')# drop columns with NaN
<ipython-input-8-9406440abbf9>:5: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future
  corr = df.corr()
```



durati
src_byt
dst_byt
lar
wrong_fragme
urge
h
num_failed_logi
logged_
num_compromise
root_sh
su_attempt
num_ro
num_file_creatio
num_shel
num_access_fil
is_guest_log
cou
srv_cou
serror_ra
srv_serror_ra
rerror_ra
srv_rerror_ra
same_srv_ra
diff_srv_ra
srv_diff_host_ra
dst_host_cou
dst_host_srv_cou
dst_host_same_srv_ra
dst_host_diff_srv_ra
dst_host_same_src_port_ra
dst_host_srv_diff_host_ra
dst_host_serror_ra
dst_host_srv_serror_ra
dst_host_rerror_ra
dst_host_srv_rerror_ra

```
#This variable is highly correlated with num_compromised and should be ignored for analysis.
#(Correlation = 0.9938277978738366)

df.drop('num_root', axis = 1, inplace = True)


# This variable is highly correlated with serror_rate and should be ignored for analysis.
#(Correlation = 0.9983615072725952)

df.drop('srv_serror_rate', axis = 1, inplace = True)


# This variable is highly correlated with rerror_rate and should be ignored for analysis.
#(Correlation = 0.9947309539817937)

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