Cyber_Activity_Classifier

December 10, 2023

1 End to End Data Mining Project

1.1 Classification Problem: Predict whether a record represents "normal" (normal activities) or "attack" (attack behaviours)

Target variable: 'label'

- a) Discover and visualise the data
- b) Prepare the data for ML
- c) Select and train models
- d) Fine-tune the models
- e) Evaluate outcomes

2 Discover and visualise the data

Dataset: This dataset has nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms.

A partition from this dataset was configured as a training set and testing set. The number of records in the training set is 175,341 records and the testing set is 82,332 records from the different types, attack and normal.

2.1 Uploading and reading the data

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
import pandas as pd
import csv

#import train and test datasets

path = "/content/drive/MyDrive/UNSW_NB15_testing-set.csv"
train =  pd.read_csv(path)
```

```python
path = "/content/drive/MyDrive/UNSW_NB15_training-set.csv"
test =  pd.read_csv(path)
```

[ ]:
```python
train.shape
```

[ ]: (175341, 45)

[ ]:
```python
test.shape
```

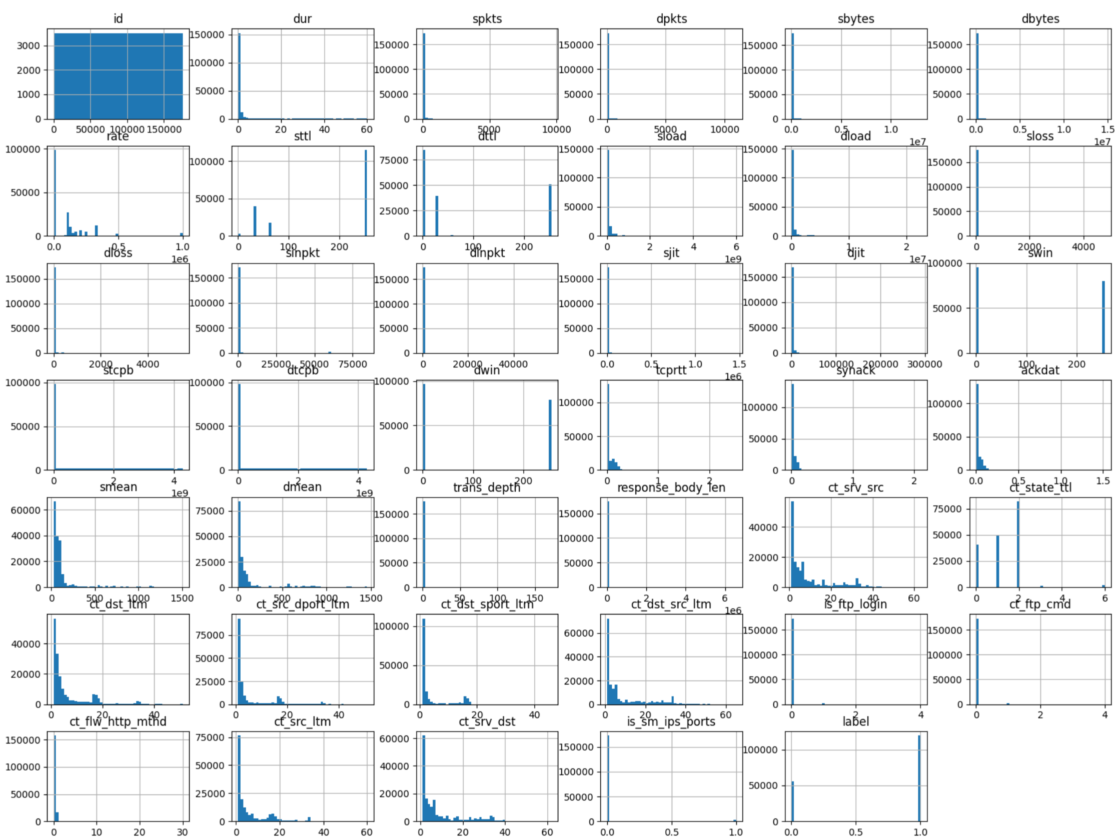
[ ]: (82332, 45)

2.2 Visualise the data

###Histograms

[ ]:
```python
%matplotlib inline
import matplotlib.pyplot as plt

# Plot the histograms for attributes
train.hist(bins = 50, figsize = (20, 15))
plt.show()
```



2.2.1 Interpretation

To start off, the x-axis represents the range of values for each attribute and is divided into bins that represents a specific range of values. The y-axis represents the frequency of values within each bin. It shows how many data points fall within the corresponding range of values on the x-axis.

1) Shape of histograms

Symmetric Distribution: The histogram is roughly symmetrical around a central value which indicates a normal distribution. This means that the data points are evenly distributed around the mean, and there are approximately an equal number of data points on both sides of the central value.

Skewed Distribution: The histogram is skewed to the left or right. A right-skewed (positive-skewed) distribution indicates that the majority of the data points are on the right side. A left-skewed (negative-skewed) distribution indicates that the majority of the data points are on the left side. An example of this is ct_srv_src.

2) Central Tendency: The central tendency of the data can be estimated from the histogram. For a symmetric distribution, the peak of the histogram corresponds to the mean, median, and mode, which are all the same in a normal distribution. For skewed distributions, these measures may differ.

3) Spread: The spread or variability of the data can be observed by looking at the width of the histogram. A wider histogram indicates higher variability, whereas a narrower histogram indicates lower variability.

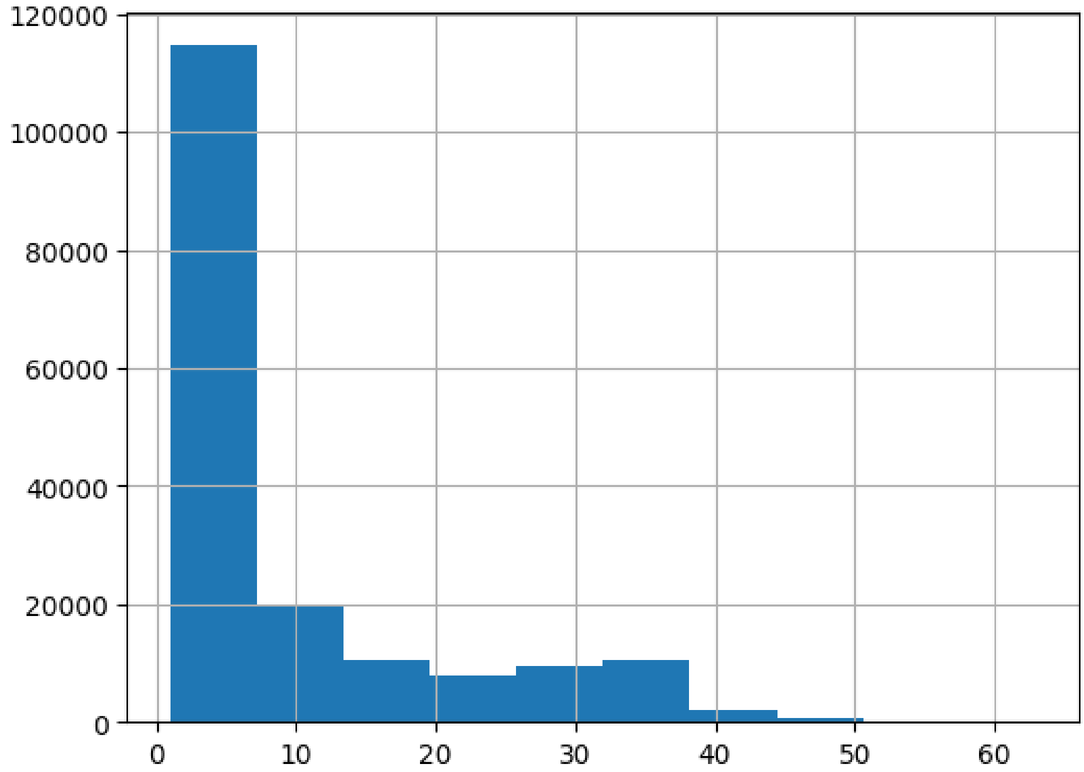
4) Outliers: Outliers, which are extreme values in the data, can also be identified from the histogram. Outliers are data points that lie far away from the bulk of the data and may appear as isolated bars far from the main distribution. For example, it is observed that there are outliers in the histogram of rate.

5) Multimodal Distribution: If there are multiple peaks in the histogram, it suggests a multimodal distribution, indicating that the data may have multiple distinct groups or categories. An example of this is ct_state_ttl.

2.2.2 Skewed distribution example

[ ]:
```
train['ct_srv_src'].hist()
```

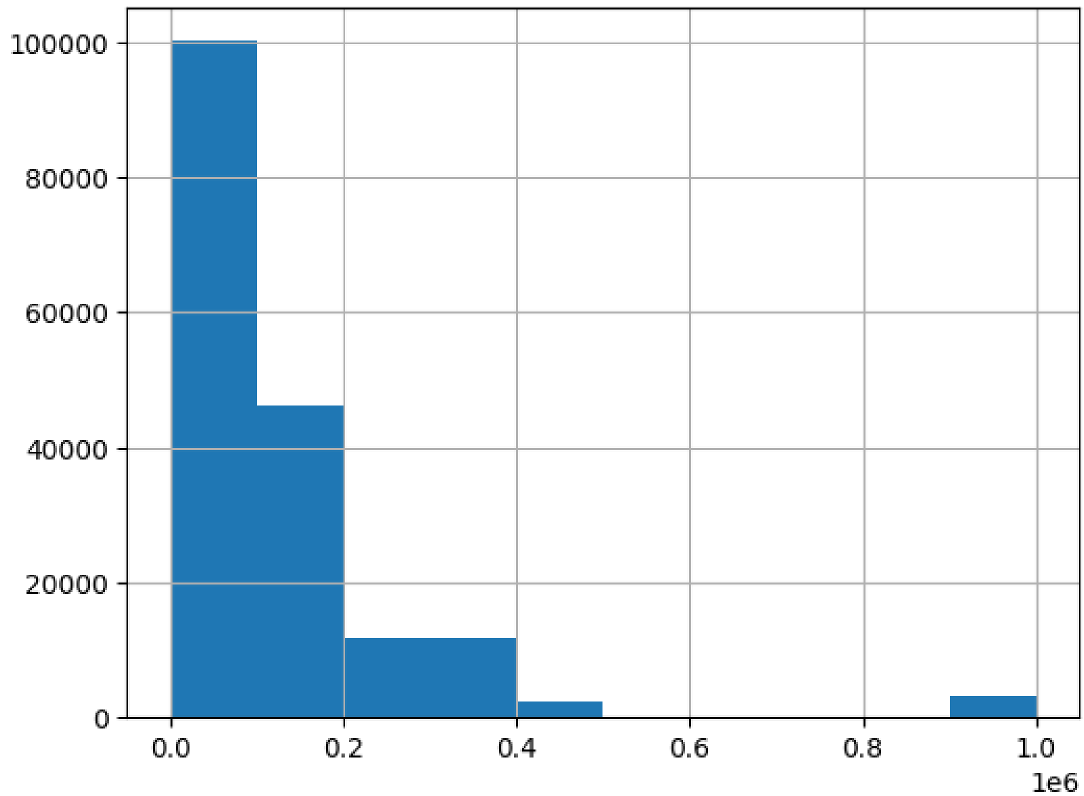
[ ]:
```
&lt;Axes: &gt;
```



2.2.3 Outliers example

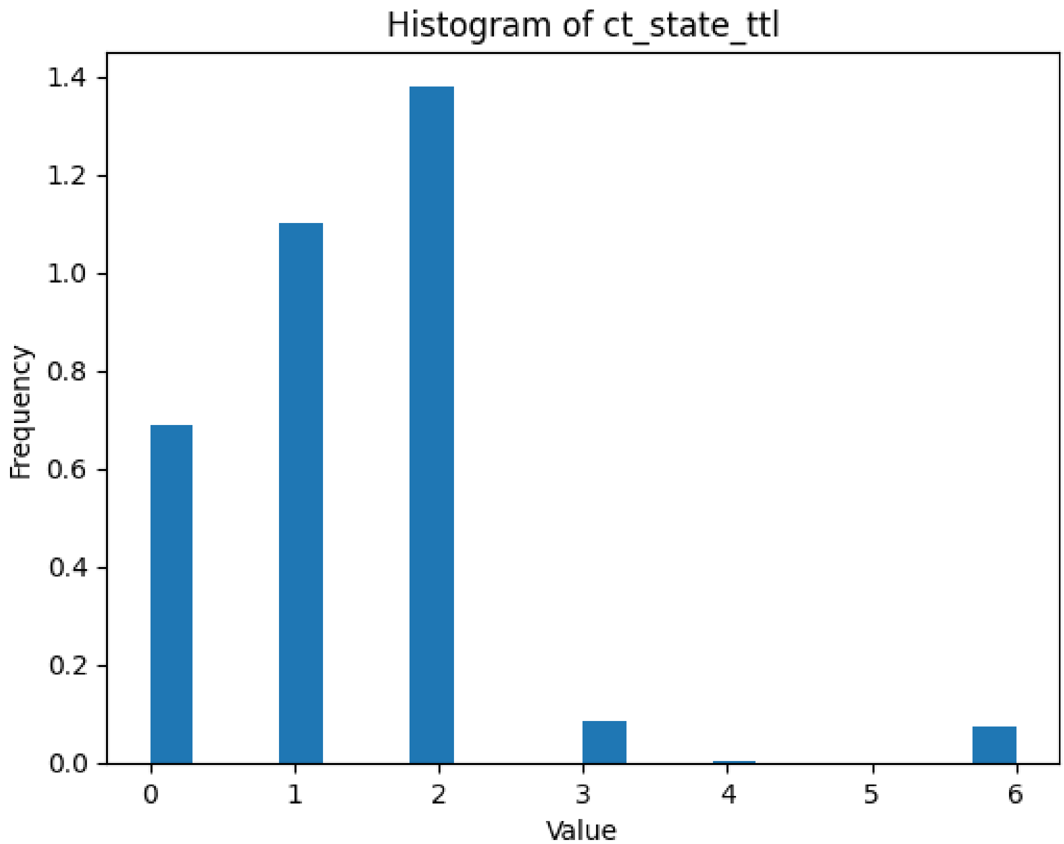
[ ]:
```
train['rate'].hist()
```

[ ]: <Axes: >



2.2.4 Multimodal distribution example

[ ]:
```
plt.hist(train['ct_state_ttl'], bins=20, range=(0, 6), density=True)
plt.title('Histogram of ct_state_ttl')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



2.3 Exploration of the data

[ ]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 82332 entries, 0 to 82331
Data columns (total 45 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 82332 non-null  int64
 1   dur                82332 non-null  float64
 2   proto              82332 non-null  object
 3   service            82332 non-null  object
 4   state              82332 non-null  object
 5   spkts              82332 non-null  int64
 6   dpkts              82332 non-null  int64
 7   sbytes             82332 non-null  int64
 8   dbytes             82332 non-null  int64
 9   rate               82332 non-null  float64
 10  sttl               82332 non-null  int64
```

```
 11  dttl               82332 non-null  int64
 12  sload              82332 non-null  float64
 13  dload              82332 non-null  float64
 14  sloss              82332 non-null  int64
 15  dloss              82332 non-null  int64
 16  sinpkt             82332 non-null  float64
 17  dinpkt             82332 non-null  float64
 18  sjit               82332 non-null  float64
 19  djit               82332 non-null  float64
 20  swin               82332 non-null  int64
 21  stcpb              82332 non-null  int64
 22  dtcpb              82332 non-null  int64
 23  dwin               82332 non-null  int64
 24  tcprtt             82332 non-null  float64
 25  synack             82332 non-null  float64
 26  ackdat             82332 non-null  float64
 27  smean              82332 non-null  int64
 28  dmean              82332 non-null  int64
 29  trans_depth        82332 non-null  int64
 30  response_body_len  82332 non-null  int64
 31  ct_srv_src         82332 non-null  int64
 32  ct_state_ttl       82332 non-null  int64
 33  ct_dst_ltm         82332 non-null  int64
 34  ct_src_dport_ltm   82332 non-null  int64
 35  ct_dst_sport_ltm   82332 non-null  int64
 36  ct_dst_src_ltm     82332 non-null  int64
 37  is_ftp_login       82332 non-null  int64
 38  ct_ftp_cmd         82332 non-null  int64
 39  ct_flw_http_mthd   82332 non-null  int64
 40  ct_src_ltm         82332 non-null  int64
 41  ct_srv_dst         82332 non-null  int64
 42  is_sm_ips_ports    82332 non-null  int64
 43  attack_cat         82332 non-null  object
 44  label              82332 non-null  int64
dtypes: float64(11), int64(30), object(4)
memory usage: 28.3+ MB
```

[ ]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175341 entries, 0 to 175340
Data columns (total 45 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   id                 175341 non-null  int64
 1   dur                175341 non-null  float64
 2   proto              175341 non-null  object
 3   service            175341 non-null  object
```

```
 4   state              175341 non-null  object
 5   spkts              175341 non-null  int64
 6   dpkts              175341 non-null  int64
 7   sbytes             175341 non-null  int64
 8   dbytes             175341 non-null  int64
 9   rate               175341 non-null  float64
 10  sttl               175341 non-null  int64
 11  dttl               175341 non-null  int64
 12  sload              175341 non-null  float64
 13  dload              175341 non-null  float64
 14  sloss              175341 non-null  int64
 15  dloss              175341 non-null  int64
 16  sinpkt             175341 non-null  float64
 17  dinpkt             175341 non-null  float64
 18  sjit               175341 non-null  float64
 19  djit               175341 non-null  float64
 20  swin               175341 non-null  int64
 21  stcpb              175341 non-null  int64
 22  dtcpb              175341 non-null  int64
 23  dwin               175341 non-null  int64
 24  tcprtt             175341 non-null  float64
 25  synack             175341 non-null  float64
 26  ackdat             175341 non-null  float64
 27  smean              175341 non-null  int64
 28  dmean              175341 non-null  int64
 29  trans_depth        175341 non-null  int64
 30  response_body_len  175341 non-null  int64
 31  ct_srv_src         175341 non-null  int64
 32  ct_state_ttl       175341 non-null  int64
 33  ct_dst_ltm         175341 non-null  int64
 34  ct_src_dport_ltm   175341 non-null  int64
 35  ct_dst_sport_ltm   175341 non-null  int64
 36  ct_dst_src_ltm     175341 non-null  int64
 37  is_ftp_login       175341 non-null  int64
 38  ct_ftp_cmd         175341 non-null  int64
 39  ct_flw_http_mthd   175341 non-null  int64
 40  ct_src_ltm         175341 non-null  int64
 41  ct_srv_dst         175341 non-null  int64
 42  is_sm_ips_ports    175341 non-null  int64
 43  attack_cat         175341 non-null  object
 44  label              175341 non-null  int64
dtypes: float64(11), int64(30), object(4)
memory usage: 60.2+ MB
```

Exploration of both datasets shows no missing value, and 4 ‘object’ types to encode.

Moving forward, we will first drop the ‘id’ column from both datasets.

3 Prepare the data for machine learning algorithms

[ ]:
```python
#Dropping 'id' column
columns_to_drop = ['id', 'attack_cat']
train = train.drop(columns=columns_to_drop)
test  = test.drop(columns=columns_to_drop)
```

[ ]:
```python
train.shape
```

[ ]:
```
(175341, 43)
```

[ ]:
```python
test.shape
```

[ ]:
```
(82332, 43)
```

3.1 Check for duplicate rows

3.1.1 Training data

[ ]:
```python
# Check if there is any duplicate rows and drop them
print(f"Number of duplicate rows: {train.duplicated().sum()}")
train = train.drop_duplicates(subset = None, keep = 'first', inplace = False)
```

```
Number of duplicate rows: 74072
```

[ ]:
```python
train.shape
```

[ ]:
```
(101269, 43)
```

3.1.2 Testing data

[ ]:
```python
# Check if there is any duplicate rows and drop them
print(f"Number of duplicate rows: {test.duplicated().sum()}")
test = test.drop_duplicates(subset = None, keep = 'first', inplace = False)
```

```
Number of duplicate rows: 28380
```

[ ]:
```python
test.shape
```

[ ]:
```
(53952, 43)
```

3.2 Encoding categorical variables

[ ]:
```python
# Identify the categorical columns in training set
categorical_columns = train.select_dtypes(include=['object']).columns
categorical_columns
```

```
[ ]: Index(['proto', 'service', 'state'], dtype='object')
```

```
[ ]: # Identify the categorical columns in testing set
     categorical_columns = train.select_dtypes(include=['object']).columns
     categorical_columns
```

```
[ ]: Index(['proto', 'service', 'state'], dtype='object')
```

To calculate the correlations, categorical columns need to be converted to numerical columns in the first place.

In the list above, one-hot encoding will be applied to categorical attributes as they are nominal and do not have an inherent order.

When performing one-hot encoding on categorical variables, it can introduce different sets of columns in the training and testing datasets, leading to a mismatch in the number of features.

To resolve this issue, I ensure that the same set of one-hot encoding columns is applied to both the training and testing datasets.

3.2.1 'proto' attributes

```
[ ]: train['proto'].value_counts()
```

```
[ ]: tcp     76119
     udp     22984
     arp       633
     unas      474
     ospf      196
             …
     ip          5
     tlsp        5
     cbt         5
     ggp         4
     rtp         1
     Name: proto, Length: 133, dtype: int64
```

```
[ ]: test['proto'].value_counts()
```

```
[ ]: tcp       40527
     udp       12523
     arp         298
     unas        199
     ospf         74
               …
     narp          2
     rvd           2
     i-nlsp        2
     mhrp          2
```

```
ib              2
Name: proto, Length: 131, dtype: int64
```

Training set and testing set has different number of unique values for the 'proto' attirbute, applying simple One Hot Encoding will lead to addition of a lot of dimensions, and a mismatch between the two sets for training.

The strategy here would be to encode the top two values: 'tcp' and 'udp', and the rest as 'others'

```
[ ]: # Define a custom function to categorize protocols
def categorize_protocol(proto):
    if proto == 'tcp':
        return 'tcp'
    elif proto == 'udp':
        return 'udp'
    else:
        return 'other'
```

Encode 'proto' in training set

```
[ ]: # Apply the custom function to create the "category" column
train['category'] = train['proto'].apply(categorize_protocol)

# Perform one-hot encoding
one_hot_encoded = pd.get_dummies(train['category'], prefix='proto',␣
 ↪prefix_sep='_')

# Concatenate the one-hot encoded columns with the original DataFrame
train = pd.concat([train, one_hot_encoded], axis=1)

# Drop the original "proto" and "category
train.drop(['proto', 'category'], axis=1, inplace=True)
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101269 entries, 0 to 175337
Data columns (total 45 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   dur                101269 non-null  float64
 1   service            101269 non-null  object
 2   state              101269 non-null  object
 3   spkts              101269 non-null  int64
 4   dpkts              101269 non-null  int64
 5   sbytes             101269 non-null  int64
 6   dbytes             101269 non-null  int64
 7   rate               101269 non-null  float64
 8   sttl               101269 non-null  int64
```

```
 9   dttl               101269 non-null  int64
 10  sload              101269 non-null  float64
 11  dload              101269 non-null  float64
 12  sloss              101269 non-null  int64
 13  dloss              101269 non-null  int64
 14  sinpkt             101269 non-null  float64
 15  dinpkt             101269 non-null  float64
 16  sjit               101269 non-null  float64
 17  djit               101269 non-null  float64
 18  swin               101269 non-null  int64
 19  stcpb              101269 non-null  int64
 20  dtcpb              101269 non-null  int64
 21  dwin               101269 non-null  int64
 22  tcprtt             101269 non-null  float64
 23  synack             101269 non-null  float64
 24  ackdat             101269 non-null  float64
 25  smean              101269 non-null  int64
 26  dmean              101269 non-null  int64
 27  trans_depth        101269 non-null  int64
 28  response_body_len  101269 non-null  int64
 29  ct_srv_src         101269 non-null  int64
 30  ct_state_ttl       101269 non-null  int64
 31  ct_dst_ltm         101269 non-null  int64
 32  ct_src_dport_ltm   101269 non-null  int64
 33  ct_dst_sport_ltm   101269 non-null  int64
 34  ct_dst_src_ltm     101269 non-null  int64
 35  is_ftp_login       101269 non-null  int64
 36  ct_ftp_cmd         101269 non-null  int64
 37  ct_flw_http_mthd   101269 non-null  int64
 38  ct_src_ltm         101269 non-null  int64
 39  ct_srv_dst         101269 non-null  int64
 40  is_sm_ips_ports    101269 non-null  int64
 41  label              101269 non-null  int64
 42  proto_other        101269 non-null  uint8
 43  proto_tcp          101269 non-null  uint8
 44  proto_udp          101269 non-null  uint8
dtypes: float64(11), int64(29), object(2), uint8(3)
memory usage: 33.5+ MB
```

Encode ‘proto’ in testing set

[ ]:
```python
# Apply the custom function to create the "category" column
test['category'] = test['proto'].apply(categorize_protocol)

# Perform one-hot encoding
one_hot_encoded = pd.get_dummies(test['category'], prefix='proto',␣
 ↪prefix_sep='_')
```

```python
# Concatenate the one-hot encoded columns with the original DataFrame
test = pd.concat([test, one_hot_encoded], axis=1)

# Drop the original "proto" and "category
test.drop(['proto', 'category'], axis=1, inplace=True)
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 53952 entries, 0 to 82328
Data columns (total 45 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   dur                53952 non-null  float64
 1   service            53952 non-null  object
 2   state              53952 non-null  object
 3   spkts              53952 non-null  int64
 4   dpkts              53952 non-null  int64
 5   sbytes             53952 non-null  int64
 6   dbytes             53952 non-null  int64
 7   rate               53952 non-null  float64
 8   sttl               53952 non-null  int64
 9   dttl               53952 non-null  int64
 10  sload              53952 non-null  float64
 11  dload              53952 non-null  float64
 12  sloss              53952 non-null  int64
 13  dloss              53952 non-null  int64
 14  sinpkt             53952 non-null  float64
 15  dinpkt             53952 non-null  float64
 16  sjit               53952 non-null  float64
 17  djit               53952 non-null  float64
 18  swin               53952 non-null  int64
 19  stcpb              53952 non-null  int64
 20  dtcpb              53952 non-null  int64
 21  dwin               53952 non-null  int64
 22  tcprtt             53952 non-null  float64
 23  synack             53952 non-null  float64
 24  ackdat             53952 non-null  float64
 25  smean              53952 non-null  int64
 26  dmean              53952 non-null  int64
 27  trans_depth        53952 non-null  int64
 28  response_body_len  53952 non-null  int64
 29  ct_srv_src         53952 non-null  int64
 30  ct_state_ttl       53952 non-null  int64
 31  ct_dst_ltm         53952 non-null  int64
 32  ct_src_dport_ltm   53952 non-null  int64
 33  ct_dst_sport_ltm   53952 non-null  int64
 34  ct_dst_src_ltm     53952 non-null  int64
```

```
 35   is_ftp_login       53952 non-null   int64
 36   ct_ftp_cmd         53952 non-null   int64
 37   ct_flw_http_mthd   53952 non-null   int64
 38   ct_src_ltm         53952 non-null   int64
 39   ct_srv_dst         53952 non-null   int64
 40   is_sm_ips_ports    53952 non-null   int64
 41   label              53952 non-null   int64
 42   proto_other        53952 non-null   uint8
 43   proto_tcp          53952 non-null   uint8
 44   proto_udp          53952 non-null   uint8
dtypes: float64(11), int64(29), object(2), uint8(3)
memory usage: 17.9+ MB
```

[ ]:
```
train.shape
```

[ ]:
```
(101269, 45)
```

[ ]:
```
test.shape
```

[ ]:
```
(53952, 45)
```

3.2.2 'service' attributes

[ ]:
```
train['service'].value_counts()
```

[ ]:
```
-           58668
http        17977
dns         11044
smtp         5020
ftp-data     3282
ftp          2689
ssh          1295
pop3         1104
snmp           59
ssl            56
dhcp           38
irc            25
radius         12
Name: service, dtype: int64
```

[ ]:
```
test['service'].value_counts()
```

[ ]:
```
-           35068
http         7900
dns          6091
smtp         1759
ftp          1291
```

```
ftp-data     1177
pop3          381
ssh           204
ssl            30
snmp           24
dhcp           16
radius          6
irc             5
Name: service, dtype: int64
```

Both sets have 13 unique values.

The strategy would be to encode the top values: '-', 'dns', 'http', 'smtp', 'ftp', 'ftp-data' and the rest as others.

Encode 'service' in training set

[ ]:
```
# Define the list of top values and the "other" category
top_values = ['-', 'dns', 'http', 'smtp', 'ftp', 'ftp-data']
other_category = 'other'

# Create a new column with values either in the top values list or as "other"
train['service_encoded'] = train['service'].apply(lambda x: x if x in␣
↪top_values else other_category)

# Perform one-hot encoding
one_hot_encoded = pd.get_dummies(train['service_encoded'], prefix='service',␣
↪prefix_sep='_')

# Concatenate the one-hot encoded columns with the original DataFrame
train = pd.concat([train, one_hot_encoded], axis=1)

# Drop the original "service" and "service_encoded" columns
train.drop(['service', 'service_encoded'], axis=1, inplace=True)
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101269 entries, 0 to 175337
Data columns (total 51 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   dur                101269 non-null  float64
 1   state              101269 non-null  object
 2   spkts              101269 non-null  int64
 3   dpkts              101269 non-null  int64
 4   sbytes             101269 non-null  int64
 5   dbytes             101269 non-null  int64
 6   rate               101269 non-null  float64
 7   sttl               101269 non-null  int64
```

```
 8   dttl              101269 non-null  int64
 9   sload             101269 non-null  float64
 10  dload             101269 non-null  float64
 11  sloss             101269 non-null  int64
 12  dloss             101269 non-null  int64
 13  sinpkt            101269 non-null  float64
 14  dinpkt            101269 non-null  float64
 15  sjit              101269 non-null  float64
 16  djit              101269 non-null  float64
 17  swin              101269 non-null  int64
 18  stcpb             101269 non-null  int64
 19  dtcpb             101269 non-null  int64
 20  dwin              101269 non-null  int64
 21  tcprtt            101269 non-null  float64
 22  synack            101269 non-null  float64
 23  ackdat            101269 non-null  float64
 24  smean             101269 non-null  int64
 25  dmean             101269 non-null  int64
 26  trans_depth       101269 non-null  int64
 27  response_body_len 101269 non-null  int64
 28  ct_srv_src        101269 non-null  int64
 29  ct_state_ttl      101269 non-null  int64
 30  ct_dst_ltm        101269 non-null  int64
 31  ct_src_dport_ltm  101269 non-null  int64
 32  ct_dst_sport_ltm  101269 non-null  int64
 33  ct_dst_src_ltm    101269 non-null  int64
 34  is_ftp_login      101269 non-null  int64
 35  ct_ftp_cmd        101269 non-null  int64
 36  ct_flw_http_mthd  101269 non-null  int64
 37  ct_src_ltm        101269 non-null  int64
 38  ct_srv_dst        101269 non-null  int64
 39  is_sm_ips_ports   101269 non-null  int64
 40  label             101269 non-null  int64
 41  proto_other       101269 non-null  uint8
 42  proto_tcp         101269 non-null  uint8
 43  proto_udp         101269 non-null  uint8
 44  service_-         101269 non-null  uint8
 45  service_dns       101269 non-null  uint8
 46  service_ftp       101269 non-null  uint8
 47  service_ftp-data  101269 non-null  uint8
 48  service_http      101269 non-null  uint8
 49  service_other     101269 non-null  uint8
 50  service_smtp      101269 non-null  uint8
dtypes: float64(11), int64(29), object(1), uint8(10)
memory usage: 33.4+ MB
```

Encode ‘service’ in testing set

[ ]:

```python
# For testing data

# Create a new column with values either in the top values list or as "other"
test['service_encoded'] = test['service'].apply(lambda x: x if x in top_values␣
 ↪else other_category)

# Perform one-hot encoding
one_hot_encoded = pd.get_dummies(test['service_encoded'], prefix='service',␣
 ↪prefix_sep='_')

# Concatenate the one-hot encoded columns with the original DataFrame
test = pd.concat([test, one_hot_encoded], axis=1)

# Drop the original "service" and "service_encoded" columns
test.drop(['service', 'service_encoded'], axis=1, inplace=True)
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 53952 entries, 0 to 82328
Data columns (total 51 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   dur                53952 non-null  float64
 1   state              53952 non-null  object
 2   spkts              53952 non-null  int64
 3   dpkts              53952 non-null  int64
 4   sbytes             53952 non-null  int64
 5   dbytes             53952 non-null  int64
 6   rate               53952 non-null  float64
 7   sttl               53952 non-null  int64
 8   dttl               53952 non-null  int64
 9   sload              53952 non-null  float64
 10  dload              53952 non-null  float64
 11  sloss              53952 non-null  int64
 12  dloss              53952 non-null  int64
 13  sinpkt             53952 non-null  float64
 14  dinpkt             53952 non-null  float64
 15  sjit               53952 non-null  float64
 16  djit               53952 non-null  float64
 17  swin               53952 non-null  int64
 18  stcpb              53952 non-null  int64
 19  dtcpb              53952 non-null  int64
 20  dwin               53952 non-null  int64
 21  tcprtt             53952 non-null  float64
 22  synack             53952 non-null  float64
 23  ackdat             53952 non-null  float64
 24  smean              53952 non-null  int64
```

```
 25  dmean              53952 non-null  int64
 26  trans_depth        53952 non-null  int64
 27  response_body_len  53952 non-null  int64
 28  ct_srv_src         53952 non-null  int64
 29  ct_state_ttl       53952 non-null  int64
 30  ct_dst_ltm         53952 non-null  int64
 31  ct_src_dport_ltm   53952 non-null  int64
 32  ct_dst_sport_ltm   53952 non-null  int64
 33  ct_dst_src_ltm     53952 non-null  int64
 34  is_ftp_login       53952 non-null  int64
 35  ct_ftp_cmd         53952 non-null  int64
 36  ct_flw_http_mthd   53952 non-null  int64
 37  ct_src_ltm         53952 non-null  int64
 38  ct_srv_dst         53952 non-null  int64
 39  is_sm_ips_ports    53952 non-null  int64
 40  label              53952 non-null  int64
 41  proto_other        53952 non-null  uint8
 42  proto_tcp          53952 non-null  uint8
 43  proto_udp          53952 non-null  uint8
 44  service_-          53952 non-null  uint8
 45  service_dns        53952 non-null  uint8
 46  service_ftp        53952 non-null  uint8
 47  service_ftp-data   53952 non-null  uint8
 48  service_http       53952 non-null  uint8
 49  service_other      53952 non-null  uint8
 50  service_smtp       53952 non-null  uint8
dtypes: float64(11), int64(29), object(1), uint8(10)
memory usage: 17.8+ MB
```

[ ]:
```
train.shape
```

[ ]: (101269, 51)

[ ]:
```
test.shape
```

[ ]: (53952, 51)

3.2.3 'state' attributes

[ ]:
```
train['state'].value_counts()
```

[ ]:
```
FIN    74306
INT    13721
CON    12363
REQ      783
RST       83
ECO       10
```

```
PAR         1
URN         1
no          1
Name: state, dtype: int64
```

[ ]:
```
test['state'].value_counts()
```

[ ]:
```
FIN    37323
INT     8718
CON     6697
REQ     1208
ACC        4
RST        1
CLO        1
Name: state, dtype: int64
```

Train set has 7 values. Test set has 9 values.

We will encode the top values: 'INT', 'FIN', 'CON', 'REQ' and the rest as 'others'.

Encode 'state' in training set

[ ]:
```
# Define the list of top values and the "other" category
top_values = ['INT', 'FIN', 'CON', 'REQ']
other_category = 'other'

# Create a new column with values either in the top values list or as "other"
train['state_encoded'] = train['state'].apply(lambda x: x if x in top_values␣
↪else other_category)

# Perform one-hot encoding
one_hot_encoded = pd.get_dummies(train['state_encoded'], prefix='state',␣
↪prefix_sep='_')

# Concatenate the one-hot encoded columns with the original DataFrame
train = pd.concat([train, one_hot_encoded], axis=1)

# Drop the original "state" and "state_encoded" columns
train.drop(['state', 'state_encoded'], axis=1, inplace=True)
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101269 entries, 0 to 175337
Data columns (total 55 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   dur                 101269 non-null  float64
 1   spkts               101269 non-null  int64
 2   dpkts               101269 non-null  int64
```

```
3   sbytes             101269 non-null  int64
4   dbytes             101269 non-null  int64
5   rate               101269 non-null  float64
6   sttl               101269 non-null  int64
7   dttl               101269 non-null  int64
8   sload              101269 non-null  float64
9   dload              101269 non-null  float64
10  sloss              101269 non-null  int64
11  dloss              101269 non-null  int64
12  sinpkt             101269 non-null  float64
13  dinpkt             101269 non-null  float64
14  sjit               101269 non-null  float64
15  djit               101269 non-null  float64
16  swin               101269 non-null  int64
17  stcpb              101269 non-null  int64
18  dtcpb              101269 non-null  int64
19  dwin               101269 non-null  int64
20  tcprtt             101269 non-null  float64
21  synack             101269 non-null  float64
22  ackdat             101269 non-null  float64
23  smean              101269 non-null  int64
24  dmean              101269 non-null  int64
25  trans_depth        101269 non-null  int64
26  response_body_len  101269 non-null  int64
27  ct_srv_src         101269 non-null  int64
28  ct_state_ttl       101269 non-null  int64
29  ct_dst_ltm         101269 non-null  int64
30  ct_src_dport_ltm   101269 non-null  int64
31  ct_dst_sport_ltm   101269 non-null  int64
32  ct_dst_src_ltm     101269 non-null  int64
33  is_ftp_login       101269 non-null  int64
34  ct_ftp_cmd         101269 non-null  int64
35  ct_flw_http_mthd   101269 non-null  int64
36  ct_src_ltm         101269 non-null  int64
37  ct_srv_dst         101269 non-null  int64
38  is_sm_ips_ports    101269 non-null  int64
39  label              101269 non-null  int64
40  proto_other        101269 non-null  uint8
41  proto_tcp          101269 non-null  uint8
42  proto_udp          101269 non-null  uint8
43  service_-          101269 non-null  uint8
44  service_dns        101269 non-null  uint8
45  service_ftp        101269 non-null  uint8
46  service_ftp-data   101269 non-null  uint8
47  service_http       101269 non-null  uint8
48  service_other      101269 non-null  uint8
49  service_smtp       101269 non-null  uint8
50  state_CON          101269 non-null  uint8
```

```
 51  state_FIN          101269 non-null  uint8
 52  state_INT          101269 non-null  uint8
 53  state_REQ          101269 non-null  uint8
 54  state_other        101269 non-null  uint8
dtypes: float64(11), int64(29), uint8(15)
memory usage: 33.1 MB
```

Encode ‘state’ in testing set

```
[ ]: # Create a new column with values either in the top values list or as "other"
test['state_encoded'] = test['state'].apply(lambda x: x if x in top_values else␣
 ↪other_category)

# Perform one-hot encoding
one_hot_encoded = pd.get_dummies(test['state_encoded'], prefix='state',␣
 ↪prefix_sep='_')

# Concatenate the one-hot encoded columns with the original DataFrame
test = pd.concat([test, one_hot_encoded], axis=1)

# Drop the original "state" and "state_encoded" columns
test.drop(['state', 'state_encoded'], axis=1, inplace=True)
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 53952 entries, 0 to 82328
Data columns (total 55 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   dur                53952 non-null  float64
 1   spkts              53952 non-null  int64
 2   dpkts              53952 non-null  int64
 3   sbytes             53952 non-null  int64
 4   dbytes             53952 non-null  int64
 5   rate               53952 non-null  float64
 6   sttl               53952 non-null  int64
 7   dttl               53952 non-null  int64
 8   sload              53952 non-null  float64
 9   dload              53952 non-null  float64
 10  sloss              53952 non-null  int64
 11  dloss              53952 non-null  int64
 12  sinpkt             53952 non-null  float64
 13  dinpkt             53952 non-null  float64
 14  sjit               53952 non-null  float64
 15  djit               53952 non-null  float64
 16  swin               53952 non-null  int64
 17  stcpb              53952 non-null  int64
 18  dtcpb              53952 non-null  int64
```

```
 19  dwin               53952 non-null  int64
 20  tcprtt             53952 non-null  float64
 21  synack             53952 non-null  float64
 22  ackdat             53952 non-null  float64
 23  smean              53952 non-null  int64
 24  dmean              53952 non-null  int64
 25  trans_depth        53952 non-null  int64
 26  response_body_len  53952 non-null  int64
 27  ct_srv_src         53952 non-null  int64
 28  ct_state_ttl       53952 non-null  int64
 29  ct_dst_ltm         53952 non-null  int64
 30  ct_src_dport_ltm   53952 non-null  int64
 31  ct_dst_sport_ltm   53952 non-null  int64
 32  ct_dst_src_ltm     53952 non-null  int64
 33  is_ftp_login       53952 non-null  int64
 34  ct_ftp_cmd         53952 non-null  int64
 35  ct_flw_http_mthd   53952 non-null  int64
 36  ct_src_ltm         53952 non-null  int64
 37  ct_srv_dst         53952 non-null  int64
 38  is_sm_ips_ports    53952 non-null  int64
 39  label              53952 non-null  int64
 40  proto_other        53952 non-null  uint8
 41  proto_tcp          53952 non-null  uint8
 42  proto_udp          53952 non-null  uint8
 43  service_-          53952 non-null  uint8
 44  service_dns        53952 non-null  uint8
 45  service_ftp        53952 non-null  uint8
 46  service_ftp-data   53952 non-null  uint8
 47  service_http       53952 non-null  uint8
 48  service_other      53952 non-null  uint8
 49  service_smtp       53952 non-null  uint8
 50  state_CON          53952 non-null  uint8
 51  state_FIN          53952 non-null  uint8
 52  state_INT          53952 non-null  uint8
 53  state_REQ          53952 non-null  uint8
 54  state_other        53952 non-null  uint8
dtypes: float64(11), int64(29), uint8(15)
memory usage: 17.6 MB
```

Shifting target label to the last column

[ ]:
```
train = train[[col for col in train.columns if col != 'label'] + ['label']]

test = test[[col for col in test.columns if col != 'label'] + ['label']]
```

[ ]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 101269 entries, 0 to 175337
Data columns (total 55 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   dur                101269 non-null  float64
 1   spkts              101269 non-null  int64
 2   dpkts              101269 non-null  int64
 3   sbytes             101269 non-null  int64
 4   dbytes             101269 non-null  int64
 5   rate               101269 non-null  float64
 6   sttl               101269 non-null  int64
 7   dttl               101269 non-null  int64
 8   sload              101269 non-null  float64
 9   dload              101269 non-null  float64
 10  sloss              101269 non-null  int64
 11  dloss              101269 non-null  int64
 12  sinpkt             101269 non-null  float64
 13  dinpkt             101269 non-null  float64
 14  sjit               101269 non-null  float64
 15  djit               101269 non-null  float64
 16  swin               101269 non-null  int64
 17  stcpb              101269 non-null  int64
 18  dtcpb              101269 non-null  int64
 19  dwin               101269 non-null  int64
 20  tcprtt             101269 non-null  float64
 21  synack             101269 non-null  float64
 22  ackdat             101269 non-null  float64
 23  smean              101269 non-null  int64
 24  dmean              101269 non-null  int64
 25  trans_depth        101269 non-null  int64
 26  response_body_len  101269 non-null  int64
 27  ct_srv_src         101269 non-null  int64
 28  ct_state_ttl       101269 non-null  int64
 29  ct_dst_ltm         101269 non-null  int64
 30  ct_src_dport_ltm   101269 non-null  int64
 31  ct_dst_sport_ltm   101269 non-null  int64
 32  ct_dst_src_ltm     101269 non-null  int64
 33  is_ftp_login       101269 non-null  int64
 34  ct_ftp_cmd         101269 non-null  int64
 35  ct_flw_http_mthd   101269 non-null  int64
 36  ct_src_ltm         101269 non-null  int64
 37  ct_srv_dst         101269 non-null  int64
 38  is_sm_ips_ports    101269 non-null  int64
 39  proto_other        101269 non-null  uint8
 40  proto_tcp          101269 non-null  uint8
 41  proto_udp          101269 non-null  uint8
 42  service_-          101269 non-null  uint8
 43  service_dns        101269 non-null  uint8
```

```
 44  service_ftp        101269 non-null  uint8
 45  service_ftp-data   101269 non-null  uint8
 46  service_http       101269 non-null  uint8
 47  service_other      101269 non-null  uint8
 48  service_smtp       101269 non-null  uint8
 49  state_CON          101269 non-null  uint8
 50  state_FIN          101269 non-null  uint8
 51  state_INT          101269 non-null  uint8
 52  state_REQ          101269 non-null  uint8
 53  state_other        101269 non-null  uint8
 54  label              101269 non-null  int64
dtypes: float64(11), int64(29), uint8(15)
memory usage: 33.1 MB
```

[ ]:
```python
train.shape
```

[ ]:
```
(101269, 55)
```

[ ]:
```python
test.shape
```

[ ]:
```
(53952, 55)
```

3.3 Filter by correlation to determine the attributes used to train models

[ ]:
```python
correlation = train.corr()['label'].abs().sort_values(ascending=False)

correlation
```

[ ]:
```
label                1.000000
sttl                 0.587313
dttl                 0.553588
ct_state_ttl         0.500807
ackdat               0.387279
tcprtt               0.379212
dload                0.351894
state_CON            0.334804
synack               0.324846
state_INT            0.246223
service_http         0.208635
dmean                0.207194
ct_dst_sport_ltm     0.173568
rate                 0.170367
ct_flw_http_mthd     0.140827
smean                0.130069
sload                0.110136
service_-            0.108634
service_dns          0.104578
```

```
ct_src_dport_ltm     0.102564
service_ftp-data     0.097092
ct_srv_dst           0.096549
service_smtp         0.090946
trans_depth          0.079419
ct_srv_src           0.078478
is_sm_ips_ports      0.071756
state_FIN            0.070004
dur                  0.068186
ct_dst_ltm           0.066179
dpkts                0.066086
sinpkt               0.065341
proto_udp            0.057684
state_REQ            0.056114
dloss                0.055893
proto_other          0.052552
dwin                 0.051157
ct_dst_src_ltm       0.051140
dbytes               0.046332
sbytes               0.045617
sjit                 0.040358
swin                 0.038461
proto_tcp            0.038326
dtcpb                0.037000
sloss                0.036260
stcpb                0.030517
ct_src_ltm           0.026573
state_other          0.022346
service_ftp          0.020255
djit                 0.019208
dinpkt               0.008674
response_body_len    0.008242
spkts                0.006206
service_other        0.004456
is_ftp_login         0.000882
ct_ftp_cmd           0.000882
Name: label, dtype: float64
```

From the result shown above, most of the attributes have the correlation higher than 0.01. Thus, the attributes with absolute correlation higher than 0.01 will be used to train models.

[ ]:
```
# Filter attributes with correlation lower than 0.01
attributes_filtered = correlation[correlation < 0.01].index
attributes_filtered
```

[ ]:
```
Index(['dinpkt', 'response_body_len', 'spkts', 'service_other', 'is_ftp_login',
       'ct_ftp_cmd'],
```

```
dtype='object')
```

[ ]:
```
# Remove the filtered attributes from training set
train = train.drop(attributes_filtered, axis=1)
train.shape
```

[ ]:
```
(101269, 49)
```

[ ]:
```
# Remove the filtered attributes from testing set
test = test.drop(attributes_filtered, axis=1)
test.shape
```

[ ]:
```
(53952, 49)
```

3.4 Check for attributes outliers

[ ]:
```
outliers = []
def outlierFinder(dataF, col):
  # Use Quatile 3 value to subtract Quatile 1 value to get interquatile range
  # The lower quartile, or first quartile (Q1), is the value under which 25% of␣
↪data
  # The upper quartile, or third quartile (Q3), is the value under which 75% of␣
↪data
  Q1 = np.percentile(np.array(dataF[col].tolist()), 25)
  Q3 = np.percentile(np.array(dataF[col].tolist()), 75)
  interquatileRange = Q3-Q1

  upperBound = Q3 + (3 * interquatileRange)
  lowerBound = Q1 - (3 * interquatileRange)

  count = 0

  for value in dataF[col].tolist():
    if((value <lowerBound ) |(value>upperBound)):
      # Increment the outliers count when the values fall outside of the interqu
      count+=1
  outliers.append(count)
  return lowerBound, upperBound, count
```

Finding outliers in columns of data type Sparse[uint8, 0] may not be meaningful because these columns typically represent categorical variables encoded using one-hot encoding. The Sparse[uint8, 0] data type is used to efficiently store binary data where most of the entries are zero. Thus, only the outliers of the integer and float attributes will be evaluated.

[ ]:
```
import numpy as np
# Select the numerical attributes
numerical_att = train.select_dtypes(include=['float64', 'int64'])
```

```python
# Display the number of outliers of each attribute
for attribute in numerical_att:
  if(outlierFinder(train, attribute)[2] > 0):
    print(f'There is {outlierFinder(train, attribute)[2]} outliers in␣
  ↪{attribute}')
```

```
There is 4056 outliers in dur
There is 6903 outliers in dpkts
There is 9025 outliers in sbytes
There is 15491 outliers in dbytes
There is 13238 outliers in rate
There is 15446 outliers in sload
There is 12423 outliers in dload
There is 4906 outliers in sloss
There is 6224 outliers in dloss
There is 2736 outliers in sinpkt
There is 1938 outliers in sjit
There is 12310 outliers in djit
There is 25161 outliers in swin
There is 222 outliers in tcprtt
There is 271 outliers in synack
There is 288 outliers in ackdat
There is 12171 outliers in smean
There is 18375 outliers in dmean
There is 17095 outliers in trans_depth
There is 3487 outliers in ct_srv_src
There is 724 outliers in ct_state_ttl
There is 5323 outliers in ct_dst_ltm
There is 23099 outliers in ct_src_dport_ltm
There is 6881 outliers in ct_dst_sport_ltm
There is 5120 outliers in ct_dst_src_ltm
There is 17095 outliers in ct_flw_http_mthd
There is 5384 outliers in ct_src_ltm
There is 3430 outliers in ct_srv_dst
There is 545 outliers in is_sm_ips_ports
```

3.4.1 Find the top 3 attributes with the most outliers

[ ]:
```python
# To remove outliers in the list
unique_outliers = frozenset(outliers)
total_outliers = 0
sorted_outliers = sorted(unique_outliers, reverse = True)

for outlier in unique_outliers:
 total_outliers += outlier
```

```
print("Total number of outliers: ", total_outliers)
# We arrange the outliers number in descending order
print("Outliers in descending orders: ", sorted_outliers)
```

```
Total number of outliers:  232272
Outliers in descending orders:  [25161, 23099, 18375, 17095, 15491, 15446,
13238, 12423, 12310, 12171, 9025, 6903, 6881, 6224, 5384, 5323, 5120, 4906,
4056, 3487, 3430, 2736, 1938, 724, 545, 288, 271, 222, 0]
```

From the result shown above, the top 3 attributes with the most outliers are sload, rate, and dload. However, these attributes have relatively high correlations with the target attribute, so they will not be removed.

The presence of outliers will be kept in mind as we move to select and train models for classification.

3.5 Validating Training Set

Checking the order and sequence of both sets

[ ]:
```
if list(train.columns) == list(test.columns):
    print("Both DataFrames have the same columns in the same sequence.")
else:
    print("The DataFrames have different columns or different sequence.")
```

```
Both DataFrames have the same columns in the same sequence.
```

[ ]:
```
# Check for missing values for Train
missing_values = train.isnull().any()

# Get the columns with missing data for Train
columns_with_missing_data = missing_values[missing_values].index

# Print the columns with missing data
if len(columns_with_missing_data) > 0:
    for column in columns_with_missing_data:
        print("Columns With Missing Data In Train Set: "+column)
else:
    print("No Missing Data For Train.")

# Check for missing values for Test
missing_values = test.isnull().any()

# Get the columns with missing data for Train
columns_with_missing_data = missing_values[missing_values].index

# Print the columns with missing data
if len(columns_with_missing_data) > 0:
    for column in columns_with_missing_data:
        print("Columns With Missing Data In Test Set: "+column)
```

```
else:
    print("No Missing Data For Test.")
```

```
No Missing Data For Train.
No Missing Data For Test.
```

[ ]:
```
train.shape
```

[ ]: (101269, 49)

[ ]:
```
test.shape
```

[ ]: (53952, 49)

4 Select and train models

4.1 Scaling the data

Using pyspark

[ ]:
```
!pip install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.4.1.tar.gz (310.8 MB)
                           310.8/310.8

MB 4.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-
packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) … done
  Created wheel for pyspark: filename=pyspark-3.4.1-py2.py3-none-any.whl
size=311285388
sha256=05c4dfef5d9716ff348ab8309fb66e4affe1308e07d6beaf4fd7ad9b8d9886a0
  Stored in directory: /root/.cache/pip/wheels/0d/77/a3/ff2f74cc9ab41f8f594dabf0
579c2a7c6de920d584206e0834
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.4.1
```

[ ]:
```
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import MinMaxScaler

# Create a Spark session
spark = SparkSession.builder.appName("ScalingExample").getOrCreate()
```

```python
train = spark.createDataFrame(train)

test = spark.createDataFrame(test)

# Assemble features into a single vector column
feature_columns = train.columns[:-1]
vector_assembler = VectorAssembler(inputCols=feature_columns,␣
↪outputCol="features")
train = vector_assembler.transform(train)

# Assemble features into a single vector column
feature_columns = test.columns[:-1]
vector_assembler = VectorAssembler(inputCols=feature_columns,␣
↪outputCol="features")
test = vector_assembler.transform(test)

# Initialize MinMaxScaler
min_max_scaler = MinMaxScaler(inputCol="features", outputCol="scaled_features")

# Fit and transform the scaler
min_max_scaler_model = min_max_scaler.fit(train)
train_scaled = min_max_scaler_model.transform(train)

# Fit and transform the scaler
test_scaled = min_max_scaler_model.transform(test)

# Show the scaled features
train_scaled.select("scaled_features").show(truncate=False)
```

```
+-------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------+
|scaled_features
|
+-------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------+
```

|(48,[0,1,2,3,4,5,6,7,8,11,12,13,14,15,16,17,21,22,35,37,44],[0.0020246337045161
794,3.6449790413705123E-4,1.773978891965071E-5,1.1736168209313195E-
5,7.408748977773754E-5,0.9882352941176471,1.0,2.36455273458158E-
6,3.7887292198585993E-4,2.879598105028267E-4,2.0662759277358024E-
5,4.088142208481507E-
5,1.0,0.14476801864577,0.512827516559219,1.0,0.01016260162601626,0.0294924554183
8134,1.0,1.0,1.0])
|
|(48,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,21,22,24,25,29,32,35,37,44],[0
.010831701985812032,0.0034627300893019866,5.445343903162349E-
5,0.00286676378573305,7.847337176457989E-
5,0.24313725490196078,0.9921259842519685,1.4019893029543652E-
6,0.022458073236398958,4.1640641265875496E-
4,0.003099927060539752,5.91609754080928E-4,4.205941425219748E-
5,0.00479555833910845,1.0,0.3301275226890628,0.7165244645733828,1.0,0.0162601626
01626015,0.7585733882030178,0.6774193548387096,0.16666666666666666,0.015625,0.08
19672131147541,1.0,1.0,1.0])
|
|[0.027052154959561744,0.001457991616548205,2.591551772609843E-
5,8.997274070232778E-4,1.4170160957489517E-
5,0.24313725490196078,0.9921259842519685,2.625704381733413E-
7,0.0027172976024774863,2.0820320632937748E-
4,0.0010940919037199124,0.0027482690481154914,0.011762972907395124,0.03946575388
781213,1.0,0.49270569285187477,0.6899176800279365,1.0,0.044423085855572274,0.029
26080961667377,0.03316426499325393,0.012195121951219513,0.5651577503429355,0.0,
0.0967741935483871,0.16666666666666666,0.02,0.0,0.0,0.03125,0.0,0.01694915254237
288,0.0819672131147541,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0
.0] |
|(48,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,21,22,25,26,29,31,35,39,44],[0
.02802737180501817,0.0010934937124111536,4.627771022517577E-
5,5.253982279750674E-5,1.3677107958968677E-
5,0.24313725490196078,0.9921259842519685,4.576116963679702E-
7,1.497864920997577E-4,2.0820320632937748E-4,5.470459518599562E-
4,0.0018119454347472988,1.7739384939314363E-
4,0.01724943680051313,1.0,0.25777177370637167,0.24388164995802109,1.0,0.01626016
2601626015,0.0438957475994513,0.16666666666666666,0.02,0.03125,0.016949152542372
88,1.0,1.0,1.0])
|
|[0.0074909013733319195,5.467468562055768E-4,3.902753562323156E-
5,1.8286587674976374E-5,3.3373825899878524E-
5,0.996078431372549,0.9921259842519685,1.429775994819196E-6,1.7781330881654466E-
4,4.1640641265875496E-4,1.8234865061998541E-4,5.659533760074611E-
4,0.001654139466157543,4.001786254849016E-
4,1.0,0.5672085806516771,0.46035123316232995,1.0,0.05096723044607294,0.033873845
90773356,0.037632061353791615,0.016937669376693765,0.030864197530864196,0.0,0.67
74193548387096,0.16666666666666666,0.02,0.02,0.0,0.609375,0.0,0.0169491525423728
8,0.6229508196721312,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0
] |

|[0.006342284496085491,5.467468562055768E-4,3.902753562323156E-
5,1.8286587674976374E-5,3.941797988174606E-
5,0.996078431372549,0.9921259842519685,1.688714922793751E-6,2.1001612051699327E-
4,4.1640641265875496E-4,1.8234865061998541E-4,4.7324961501216004E-
4,0.0015226023756835846,2.85258625325683E-
4,1.0,0.9276353007570669,0.4181815525506401,1.0,0.06865476223086889,0.0568147624
7790847,0.035244634041780964,0.016937669376693765,0.030864197530864196,0.0,0.677
4193548387096,0.16666666666666666,0.02,0.02,0.0,0.609375,0.0,0.01694915254237288
,0.6229508196721312,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0]
|
|(48,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,25,29,32,35,
37,44],[0.010618485280055635,7.289958082741025E-4,3.902753562323156E-
5,2.4154671779632972E-5,2.66830329199509E-
5,0.996078431372549,0.9921259842519685,1.0086477863704353E-6,1.735999024204457E-
4,4.1640641265875496E-4,1.8234865061998541E-4,8.091331935136009E-
4,0.002935218916408644,4.126729778086786E-
4,1.0,0.41614116973044024,0.41146195035651933,1.0,0.05690475935262038,0.03291638
7348406365,0.04878807325213495,0.016937669376693765,0.03017832647462277,0.677419
3548387096,0.16666666666666666,0.609375,0.6229508196721312,1.0,1.0,1.0])
|
|[0.008693068260395849,7.289958082741025E-4,3.902753562323156E-
5,2.4154671779632972E-5,3.2593025902220924E-
5,0.996078431372549,0.9921259842519685,1.2320519419830832E-6,2.120503199209017E-
4,4.1640641265875496E-4,1.8234865061998541E-4,6.612896848480676E-
4,0.0025817407185939885,4.1108312069222756E-
4,1.0,0.04795987718916742,0.07357741274192944,1.0,0.04629613087971581,0.02818337
116826132,0.03775435864931185,0.016937669376693765,0.03017832647462277,0.0,0.677
4193548387096,0.16666666666666666,0.04,0.04,0.0,0.609375,0.0,0.03389830508474576
,0.6229508196721312,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0]
|
|[0.00904841832554336,7.289958082741025E-4,3.902753562323156E-
5,2.4154671779632972E-5,3.131303090606091E-
5,0.996078431372549,0.9921259842519685,1.183666680691605E-6,2.0372267582939276E-
4,4.1640641265875496E-4,1.8234865061998541E-4,7.136401729797466E-
4,0.0027803362952691026,3.684031391822514E-
4,1.0,0.2058447803363189,0.7940421327866988,1.0,0.04707782347245397,0.0314866270
03473704,0.034487179824365305,0.016937669376693765,0.03017832647462277,0.0,0.677
4193548387096,0.16666666666666666,0.04,0.04,0.0,0.609375,0.0,0.03389830508474576
,0.6229508196721312,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0]
|
|[0.004311450790432645,5.467468562055768E-4,3.902753562323156E-
5,1.8286587674976374E-5,5.79851348260446E-
5,0.996078431372549,0.9921259842519685,2.484154890456972E-6,3.089405712863688E-
4,4.1640641265875496E-4,1.8234865061998541E-4,3.260000391601448E-
4,9.679601223656865E-4,1.9765968765526643E-
4,1.0,0.784279445795015,0.13617585427244916,1.0,0.03490977981200472,0.0300502011
09147422,0.016318141291512045,0.016937669376693765,0.030864197530864196,0.0,0.67
74193548387096,0.16666666666666666,0.04,0.04,0.0,0.609375,0.0,0.0338983050847457

6,0.6229508196721312,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0
] |
|(48,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,25,29,32,35,
37,44],[0.005080884264828782,5.467468562055768E-4,3.1731083311062184E-
4,1.8286587674976374E-5,5.5764582832706254E-
5,0.996078431372549,0.9921259842519685,1.6640191130613072E-
5,2.6215555215622723E-4,6.246096189881324E-4,1.8234865061998541E-
4,3.0755431905581E-4,0.001007647604128532,2.778441713734743E-
4,1.0,0.03193285309083468,0.6063246745965722,1.0,0.03881109677941858,0.017381848
375891278,0.04027460345430684,0.21476964769647697,0.030864197530864196,0.1612903
2258064516,0.16666666666666666,0.03125,0.0819672131147541,1.0,1.0,1.0])
|
|(48,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,25,29,35,42,
44],[0.034884756395538674,0.0025514853289593585,0.004342468938979369,1.509325818
5465574E-4,4.25209668724371E-
5,0.24313725490196078,0.9921259842519685,3.537493586239419E-
5,3.635845894768389E-
4,0.005829689777222569,0.0014587892049598833,4.066879174454842E-
4,0.0022275408066932422,3.666819429481526E-
4,1.0,0.4248521811179431,0.20040521118642984,1.0,0.05206255287540996,0.025163401
18227802,0.051475326191872624,0.5968834688346883,0.0541838134430727,0.0161290322
58064516,0.16666666666666666,0.015625,1.0,1.0,1.0])
|
|(48,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,25,29,32,35,
37,44],[0.006949201274020234,5.467468562055768E-4,3.902753562323156E-
5,1.8286587674976374E-5,3.5975362892073915E-
5,0.996078431372549,0.9921259842519685,1.541229039486534E-6,1.9167409343108535E-
4,4.1640641265875496E-4,1.8234865061998541E-4,5.344077103954634E-
4,0.0017877056271888077,3.450757527370502E-
4,1.0,0.02058413661010964,0.8642806935523102,1.0,0.08772742629401091,0.045010074
501797795,0.08313520294775933,0.016937669376693765,0.030864197530864196,0.677419
3548387096,0.16666666666666666,0.609375,0.6229508196721312,1.0,1.0,1.0])
|
|[0.016603686377342504,7.289958082741025E-4,4.1341421134490354E-
5,2.4154671779632972E-5,1.706448694880654E-
5,0.996078431372549,0.9921259842519685,6.812651706072339E-7,1.110216049071634E-
4,4.1640641265875496E-4,1.8234865061998541E-
4,0.001311950448288839,0.004479907376562145,6.995205687150214E-
4,1.0,0.5405827103426821,0.6927158843121668,1.0,0.06718268699782007,0.0357844780
30349204,0.061849555916164554,0.018970189701897018,0.03017832647462277,0.0,0.161
29032258064516,0.16666666666666666,0.0,0.0,0.0,0.03125,0.0,0.01694915254237288,0
.03278688524590164,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0]
|
|(48,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,25,29,32,35,
37,44],[0.009612585095640601,7.289958082741025E-4,3.902753562323156E-
5,2.4154671779632972E-5,2.947525391157424E-
5,0.996078431372549,0.9921259842519685,1.1141967705019418E-6,1.917660977945148E-
4,4.1640641265875496E-4,1.8234865061998541E-4,7.595443015494237E-

```
4,0.0028720317402822987,4.0254995173358266E-
4,1.0,0.8782975689434727,0.9969382294478185,1.0,0.0449844435631049,0.02420975150
8318604,0.04106953587518838,0.016937669376693765,0.03017832647462277,0.677419354
8387096,0.16666666666666666,0.609375,0.6229508196721312,1.0,1.0,1.0])
|
|(48,[0,2,4,5,7,11,21,25,29,31,36,45],[3.333333944444556E-8,8.484246874615557E-
6,0.49999999980000004,0.996078431372549,0.04609218239819661,2.3704688133063327E-
8,0.027777777777777776,0.3333333333333333,0.046875,0.01694915254237288,1.0,1.0])
|
|[0.012137535558548187,5.467468562055768E-4,3.902753562323156E-
5,1.8286587674976374E-5,2.0597265938208205E-
5,0.996078431372549,0.9921259842519685,8.824123293424254E-7,1.0974073041061459E-
4,4.1640641265875496E-4,1.8234865061998541E-4,9.590548092213512E-
4,0.003652088456238075,8.010061957255622E-
4,1.0,0.4041805782461975,0.9877759654563207,1.0,0.05342465916575258,0.0380802836
85782194,0.035892941210506527,0.016937669376693765,0.030864197530864196,0.0,0.67
74193548387096,0.16666666666666666,0.02,0.02,0.0,0.609375,0.0,0.0169491525423728
8,0.6229508196721312,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0
]             |
|[0.00655926786919911,7.289958082741025E-4,6.41717581789104E-
5,7.478395556632129E-5,4.319588587041235E-
5,0.24313725490196078,0.9921259842519685,2.627499189605913E-
6,8.693878412664292E-4,4.1640641265875496E-4,3.6469730123997083E-
4,5.182845637820622E-4,0.001454890070197304,2.4812984670884885E-
4,1.0,0.9040764958768303,0.7180805324975449,1.0,0.04776423611483299,0.0281862278
32287157,0.040181894214154405,0.03929539295392954,0.09396433470507544,0.00581395
3488372093,0.03225806451612903,0.16666666666666666,0.0,0.0,0.0,0.015625,0.033333
33333333333,0.0,0.03278688524590164,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,
1.0,0.0,0.0,0.0]|
|[0.006464201185103551,5.467468562055768E-4,3.902753562323156E-
5,1.8286587674976374E-5,3.867454488397637E-
5,0.996078431372549,0.9921259842519685,1.6568652548167159E-
6,2.0605514350839526E-4,4.1640641265875496E-4,1.8234865061998541E-
4,4.518074089856129E-4,0.0015242815633295182,3.64357750607194E-
4,1.0,0.12278364908307099,0.605971794813887,1.0,0.04604363901126408,0.0221129601
1335243,0.04571946315432341,0.016937669376693765,0.030864197530864196,0.0,0.6774
193548387096,0.16666666666666666,0.02,0.02,0.0,0.609375,0.0,0.01694915254237288,
0.6229508196721312,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0]
|
|[0.008964001643400301,7.289958082741025E-4,3.902753562323156E-
5,2.4154671779632972E-5,3.160791290517626E-
5,0.996078431372549,0.9921259842519685,1.1948135669885985E-
6,2.0564117843812954E-4,4.1640641265875496E-4,1.8234865061998541E-
4,7.072227805466434E-4,0.002665833062655203,3.6226648754173743E-
4,1.0,0.2519796796482183,0.11429572586312621,1.0,0.0454838693029041,0.0282657383
14339692,0.03629533876350859,0.016937669376693765,0.03017832647462277,0.0,0.6774
193548387096,0.16666666666666666,0.02,0.02,0.0,0.609375,0.0,0.01694915254237288,
0.6229508196721312,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0]
```

```
|
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------------------------------------------+
only showing top 20 rows
```

[ ]:
```python
# Show the scaled features
test_scaled.select("scaled_features").show(truncate=False)
```

```
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------+
|scaled_features
|
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------+
|(48,[0,2,4,5,7,11,21,24,25,29,32,36,37,45],[1.8333336694445063E-
7,3.60966139756371E-
5,0.09090908992727274,0.996078431372549,0.030120845739656562,1.3037578473184828E
-7,0.14905149051490515,0.016129032258064516,0.3333333333333333,0.015625,0.016393
44262295082,1.0,1.0,1.0])                                      |
|(48,[0,2,4,5,7,11,21,24,25,29,32,36,37,45],[1.3333335777778225E-
7,1.3374258255075797E-
4,0.12499999992500001,0.996078431372549,0.14712758222033048,9.48187525322533E-
8,0.5779132791327913,0.016129032258064516,0.3333333333333333,0.015625,0.01639344
262295082,1.0,1.0,1.0])                                      |
|(48,[0,2,4,5,7,11,21,24,25,29,32,36,37,45],[8.333334861111393E-
8,8.0214697723638E-
5,0.2000000045,0.996078431372549,0.14268536464137385,5.926172033265832E-
8,0.3428184281842818,0.03225806451612903,0.3333333333333333,0.03125,0.0327868852
4590164,1.0,1.0,1.0])                                          |
|(48,[0,2,4,5,7,11,21,24,25,26,27,29,31,32,36,37,45],[1.0000001833333671E-
7,6.725693886058878E-
5,0.16666666030000005,0.996078431372549,0.10020039651781872,7.111406439918998E-
8,0.2859078590785908,0.03225806451612903,0.3333333333333333,0.02,0.02,0.03125,0.
01694915254237288,0.03278688524590164,1.0,1.0,1.0])   |
|(48,[0,2,4,5,7,11,21,24,25,26,27,29,31,32,36,37,45],[1.6666669722222786E-
7,1.6181772675403128E-
4,0.1000000022,0.996078431372549,0.14201736199792173,1.1852344066531664E-
```

7,0.7012195121951219,0.03225806451612903,0.3333333333333333,0.02,0.02,0.03125,0.
01694915254237288,0.03278688524590164,1.0,1.0,1.0]) |
|(48,[0,2,4,5,7,11,21,24,25,26,27,29,31,32,36,37,45],[5.0000009166668354E-
8,5.830991488372147E-
5,0.3333333205000001,0.996078431372549,0.17457135392614118,3.555703219959499E-
8,0.24661246612466126,0.016129032258064516,0.3333333333333333,0.02,0.02,0.015625
,0.01694915254237288,0.01639344262295082,1.0,1.0,1.0])|
|(48,[0,2,4,5,7,11,21,24,25,26,27,29,31,32,36,37,45],[1.0000001833333671E-
7,1.4901422692506597E-
4,0.16666666030000005,0.996078431372549,0.2182141897356659,7.111406439918998E-
8,0.6449864498644986,0.016129032258064516,0.3333333333333333,0.02,0.02,0.015625,
0.01694915254237288,0.01639344262295082,1.0,1.0,1.0])|
|(48,[0,2,4,5,7,11,21,24,25,29,32,36,37,45],[4.6666675222223795E-
7,1.0458762510889724E-
4,0.035714285112857146,0.996078431372549,0.033018416758063676,3.318656338628866E
-7,0.44986449864498645,0.03225806451612903,0.3333333333333333,0.03125,0.03278688
524590164,1.0,1.0,1.0]) |
|(48,[2,11,21,24,25,26,27,28,29,31,32,33,34,37,45],[1.388331306755273E-
6,0.7111487984046176,0.012195121951219513,0.016129032258064516,0.333333333333333
3,0.02,0.02,0.022222222222222223,0.015625,0.01694915254237288,0.0163934426229508
2,1.0,1.0,1.0,1.0]) |
|(48,[2,11,21,24,25,26,27,28,29,31,32,33,34,37,45],[1.388331306755273E-
6,0.7111490828608752,0.012195121951219513,0.016129032258064516,0.333333333333333
3,0.02,0.02,0.022222222222222223,0.015625,0.01694915254237288,0.016393442622950 8
2,1.0,1.0,1.0,1.0]) |
|(48,[0,2,4,5,7,11,21,24,25,29,32,36,37,45],[6.666667888889113E-
8,1.0998669130183441E-
4,0.24999999985000002,0.996078431372549,0.24281896089484736,4.740937626612665E-
8,0.4735772357723577,0.03225806451612903,0.3333333333333333,0.03125,0.0327868852
4590164,1.0,1.0,1.0]) |
|(48,[0,2,4,5,7,11,21,24,25,29,32,36,37,45],[1.1666668805555949E-
7,1.5688143766334585E-
4,0.14285714047142858,0.996078431372549,0.1967744899174567,8.296640846572165E-
8,0.6795392953929539,0.03225806451612903,0.3333333333333333,0.03125,0.0327868852
4590164,1.0,1.0,1.0]) |
|(48,[0,2,4,5,7,11,21,25,36,37,45],[1.8333336694445063E-7,1.5518458828842275E-
4,0.09090908992727274,0.996078431372549,0.1238841256322084,1.3037578473184828E-
7,0.6720867208672087,0.3333333333333333,1.0,1.0,1.0])
|
|(48,[0,2,4,5,7,11,21,24,25,29,31,32,36,37,45],[6.666667888889113E-
8,7.898062545096665E-
5,0.24999999985000002,0.996078431372549,0.17568469522790883,4.740937626612665E-
8,0.33739837398373984,0.03225806451612903,0.3333333333333333,0.03125,0.016949152
54237288,0.03278688524590164,1.0,1.0,1.0]) |
|(48,[0,2,4,5,7,11,21,24,25,29,32,36,37,45],[5.0000009166668354E-
8,2.205904187400045E-
5,0.3333333205000001,0.996078431372549,0.06991760823331535,3.555703219959499E-
8,0.08739837398373984,0.016129032258064516,0.3333333333333333,0.015625,0.0163934

```
4262295082,1.0,1.0,1.0])                                       |
|(48,[0,2,4,5,7,11,21,24,25,29,32,36,37,45],[1.6666669722222786E-
7,1.346681367552615E-
4,0.1000000022,0.996078431372549,0.11850366894840694,1.1852344066531664E-
7,0.5819783197831978,0.016129032258064516,0.3333333333333333,0.015625,0.01639344
262295082,1.0,1.0,1.0])                                             |
|(48,[0,2,4,5,7,11,21,24,25,29,32,36,37,45],[3.333333944444556E-
8,1.187794562446178E-
4,0.49999999980000004,0.996078431372549,0.5237140724664658,2.3704688133063327E-
8,0.5121951219512195,0.03225806451612903,0.3333333333333333,0.015625,0.016393442
62295082,1.0,1.0,1.0])                                              |
|(48,[0,2,4,5,7,11,21,24,25,29,31,32,36,37,45],[6.666667888889113E-
8,1.562644015270102E-
4,0.24999999985000002,0.996078431372549,0.3430193574126661,4.740937626612665E-
8,0.676829268292683,0.03225806451612903,0.3333333333333333,0.015625,0.0169491525
4237288,0.01639344262295082,1.0,1.0,1.0])                           |
|(48,[0,2,4,5,7,11,21,24,25,29,32,36,37,45],[1.6666669722222786E-
7,1.6521142550387748E-
4,0.1000000022,0.996078431372549,0.14495657362911107,1.1852344066531664E-
7,0.7161246612466124,0.016129032258064516,0.3333333333333333,0.015625,0.01639344
262295082,1.0,1.0,1.0])                                             |
|(48,[0,2,4,5,7,11,21,24,25,29,32,36,37,45],[1.5000002750000507E-
7,1.3420535965300973E-
5,0.11111110686666668,0.996078431372549,0.014992947922813181,1.0667109659878496E
-7,0.0494579945799458,0.016129032258064516,0.3333333333333333,0.015625,0.0163934
4262295082,1.0,1.0,1.0])                                            |
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------------------------------------------------+
only showing top 20 rows
```

4.2 Decision Tree

[ ]:
```python
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler, StandardScaler
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Create a Spark session
spark = SparkSession.builder.appName("DecisionTreeClassification").getOrCreate()

# Assuming train_scaled is your scaled DataFrame
# Initialize a Decision Tree classifier
dt = DecisionTreeClassifier(labelCol="label", featuresCol="scaled_features")
```

```
# Train the Decision Tree model
dt_model = dt.fit(train_scaled)

# Make predictions on the testing set
dtpredictions = dt_model.transform(test_scaled)

# Initialize evaluator for precision, recall, F1-score, and accuracy
evaluator = MulticlassClassificationEvaluator(labelCol="label",␣
↪predictionCol="prediction")

# Calculate metrics
precision = evaluator.evaluate(dtpredictions, {evaluator.metricName:␣
↪"weightedPrecision"})
recall = evaluator.evaluate(dtpredictions, {evaluator.metricName:␣
↪"weightedRecall"})
f1_score = evaluator.evaluate(dtpredictions, {evaluator.metricName: "f1"})
accuracy = evaluator.evaluate(dtpredictions, {evaluator.metricName: "accuracy"})

# Print metrics
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
print("Accuracy:", accuracy)
```

```
Precision: 0.8461317278435517
Recall: 0.7349681198102016
F1 Score: 0.7351905969178332
Accuracy: 0.7349681198102017
```

[ ]:
```
evaluation = evaluator.evaluate(dtpredictions)

print("evaluation (area under ROC): %f" % evaluation)
```

```
evaluation (area under ROC): 0.735191
```

4.2.1 Random Forest

[ ]:
```
from pyspark.ml.classification import RandomForestClassifier

# Initialize a Spark session
spark = SparkSession.builder.appName("RandomForest").getOrCreate()

# Train a Random Forest Classifier
rf_classifier = RandomForestClassifier(labelCol="label",␣
↪featuresCol="scaled_features")
rf_model = rf_classifier.fit(train_scaled)
```

```
# Make predictions on the test data
rfpredictions = rf_model.transform(test_scaled)

# Calculate metrics
precision = evaluator.evaluate(rfpredictions, {evaluator.metricName:␣
 ↪"weightedPrecision"})
recall = evaluator.evaluate(rfpredictions, {evaluator.metricName:␣
 ↪"weightedRecall"})
f1_score = evaluator.evaluate(rfpredictions, {evaluator.metricName: "f1"})
accuracy = evaluator.evaluate(rfpredictions, {evaluator.metricName: "accuracy"})

# Print metrics
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
print("Accuracy:", accuracy)
```

```
Precision: 0.8463259427029326
Recall: 0.7350978647686832
F1 Score: 0.7353163775585887
Accuracy: 0.7350978647686833
```

[ ]:
```
evaluation = evaluator.evaluate(rfpredictions)

print("evaluation (area under ROC): %f" % evaluation)
```

```
evaluation (area under ROC): 0.735316
```

4.2.2 XGBoost

[ ]:
```
!pip install sparkxgb
```

```
Collecting sparkxgb
  Downloading sparkxgb-0.1.tar.gz (3.6 kB)
  Preparing metadata (setup.py) … done
Collecting pyspark==3.1.1 (from sparkxgb)
  Downloading pyspark-3.1.1.tar.gz (212.3 MB)
                          212.3/212.3

MB 6.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Collecting py4j==0.10.9 (from pyspark==3.1.1->sparkxgb)
  Downloading py4j-0.10.9-py2.py3-none-any.whl (198 kB)
                          198.6/198.6 kB
22.8 MB/s eta 0:00:00
Building wheels for collected packages: sparkxgb, pyspark
  Building wheel for sparkxgb (setup.py) … done
  Created wheel for sparkxgb: filename=sparkxgb-0.1-py3-none-any.whl size=5627
```

```
sha256=673364ed3e8dd4446be8fc3db26988a33747a232ba0114510df34effef7c2073
  Stored in directory: /root/.cache/pip/wheels/b7/0c/a1/786408e13056fabeb8a72134
e101b1e142fc95905c7b0e2a71
  Building wheel for pyspark (setup.py) … done
  Created wheel for pyspark: filename=pyspark-3.1.1-py2.py3-none-any.whl
size=212767581
sha256=e649aca040b87d5d8b5f6a0a578224ef53c7260f44f3038cd0027624d524a148
  Stored in directory: /root/.cache/pip/wheels/a0/3f/72/8efd988f9ae041f051c75e68
34cd92dd6d13a726e206e8b6f3
Successfully built sparkxgb pyspark
Installing collected packages: py4j, pyspark, sparkxgb
  Attempting uninstall: py4j
    Found existing installation: py4j 0.10.9.7
    Uninstalling py4j-0.10.9.7:
      Successfully uninstalled py4j-0.10.9.7
  Attempting uninstall: pyspark
    Found existing installation: pyspark 3.4.1
    Uninstalling pyspark-3.4.1:
      Successfully uninstalled pyspark-3.4.1
Successfully installed py4j-0.10.9 pyspark-3.1.1 sparkxgb-0.1
```

[ ]:

```
from pyspark.ml.classification import GBTClassifier  # Gradient-Boosted Trees␣
 ↪(GBTs)

# Create a Spark session
spark = SparkSession.builder.appName("XGBoostClassification").getOrCreate()

# Assuming train_scaled is your scaled DataFrame
# Create an XGBoost classifier (here we use GBTs as a similar gradient boosting␣
 ↪method in Spark)
gbt = GBTClassifier(labelCol="label", featuresCol="features", maxIter=100,␣
 ↪seed=42)

# Train the XGBoost model
xgb_model = gbt.fit(train_scaled)

# Make predictions on the testing set
xgPredictions = xgb_model.transform(test_scaled)

# Initialize evaluator for precision, recall, F1-score, and accuracy
evaluator = MulticlassClassificationEvaluator(labelCol="label",␣
 ↪predictionCol="prediction")

# Calculate metrics
precision = evaluator.evaluate(xgPredictions, {evaluator.metricName:␣
 ↪"weightedPrecision"})
```

```
recall = evaluator.evaluate(xgPredictions, {evaluator.metricName:␣
 ↪"weightedRecall"})
f1_score = evaluator.evaluate(xgPredictions, {evaluator.metricName: "f1"})
accuracy = evaluator.evaluate(xgPredictions, {evaluator.metricName: "accuracy"})

# Print metrics
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
print("Accuracy:", accuracy)
```

```
Precision: 0.8670100343354672
Recall: 0.8168371886120996
F1 Score: 0.820000235945711
Accuracy: 0.8168371886120996
```

[ ]:
```
evaluation = evaluator.evaluate(xgPredictions)

print("evaluation (area under ROC): %f" % evaluation)
```

```
evaluation (area under ROC): 0.820000
```

4.3 Linear Support Vector Classifier

[ ]:
```
from pyspark.ml.classification import LinearSVC

# Create a Spark session
spark = SparkSession.builder.appName("LinearSVMClassification").getOrCreate()

# Assuming train_scaled is your scaled DataFrame
# Initialize a Linear SVM classifier
svm = LinearSVC(labelCol="label", featuresCol="scaled_features")

# Train the Linear SVM model
svm_model = svm.fit(train_scaled)

# Make predictions on the testing set
svPredictions = svm_model.transform(test_scaled)

# Initialize evaluator for precision, recall, F1-score, and accuracy
evaluator = MulticlassClassificationEvaluator(labelCol="label",␣
 ↪predictionCol="prediction")

# Calculate metrics
precision = evaluator.evaluate(svPredictions, {evaluator.metricName:␣
 ↪"weightedPrecision"})
```

```
recall = evaluator.evaluate(svPredictions, {evaluator.metricName:␣
 ↪"weightedRecall"})
f1_score = evaluator.evaluate(svPredictions, {evaluator.metricName: "f1"})
accuracy = evaluator.evaluate(svPredictions, {evaluator.metricName: "accuracy"})

# Print metrics
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
print("Accuracy:", accuracy)
```

```
Precision: 0.8419885752034924
Recall: 0.7341525800711743
F1 Score: 0.7346500162303332
Accuracy: 0.7341525800711743
```

[ ]:
```
evaluation = evaluator.evaluate(svPredictions)

print("evaluation (area under ROC): %f" % evaluation)
```

```
evaluation (area under ROC): 0.734650
```

4.4 Logistic Regression

[ ]:
```
from pyspark.ml.classification import LogisticRegression

# Create a Spark session
spark = SparkSession.builder.appName("LogReg").getOrCreate()

# Create a Logistic Regression model instance
lr = LogisticRegression(labelCol="label", featuresCol="features")

# Fit the model
lr_model = lr.fit(train_scaled)

# Predictions
lrPredictions = lr_model.transform(test_scaled)

# Calculate metrics
precision = evaluator.evaluate(lrPredictions, {evaluator.metricName:␣
 ↪"weightedPrecision"})
recall = evaluator.evaluate(lrPredictions, {evaluator.metricName:␣
 ↪"weightedRecall"})
f1_score = evaluator.evaluate(lrPredictions, {evaluator.metricName: "f1"})
accuracy = evaluator.evaluate(lrPredictions, {evaluator.metricName: "accuracy"})

# Print metrics
```

```
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
print("Accuracy:", accuracy)
```

```
Precision: 0.8323662545602685
Recall: 0.7408807829181494
F1 Score: 0.7427421365665219
Accuracy: 0.7408807829181495
```

[ ]:
```
evaluation = evaluator.evaluate(lrPredictions)

print("evaluation (area under ROC): %f" % evaluation)
```

```
evaluation (area under ROC): 0.742742
```

5 Fine-Tuning Models

Fine-tuning Decision Tree Model

[ ]:
```
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator

# Initialize a Spark session
spark = SparkSession.builder.appName("DecisionTreeTuning").getOrCreate()

# Initialize a Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(labelCol="label", featuresCol="features")

# Define a grid of hyperparameters to search
param_grid = ParamGridBuilder() \
    .addGrid(dt_classifier.maxDepth, [5, 10, 15]) \
    .addGrid(dt_classifier.maxBins, [16, 32, 64]) \
    .build()

# Initialize a CrossValidator
cross_validator = CrossValidator(estimator=dt_classifier,
                                 estimatorParamMaps=param_grid,
                                 ␣
 ↪evaluator=MulticlassClassificationEvaluator(labelCol="label",
                                                                                        ␣
 ↪predictionCol="prediction",
                                                                                        ␣
 ↪metricName="accuracy"),
                                 numFolds=3)

# Fit the CrossValidator to the training data
cv_model = cross_validator.fit(train_scaled)
```

```
# Make predictions on the test data using the best model
predictions = cv_model.transform(test_scaled)

# Calculate metrics
precision = evaluator.evaluate(predictions, {evaluator.metricName:␣
↪"weightedPrecision"})
recall = evaluator.evaluate(predictions, {evaluator.metricName:␣
↪"weightedRecall"})
f1_score = evaluator.evaluate(predictions, {evaluator.metricName: "f1"})
accuracy = evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"})

# Print metrics
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
print("Accuracy:", accuracy)
```

```
Precision: 0.8494685076116654
Recall: 0.8084223013048636
F1 Score: 0.8118898187035799
Accuracy: 0.8084223013048636
```

Fine-tuning Linear SVC

[ ]:
```
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder

# Create a Spark session
spark = SparkSession.builder.appName("LinearSVCTuning").getOrCreate()

# Assuming train_scaled is your scaled DataFrame
# Initialize a Linear SVM classifier
svm = LinearSVC(labelCol="label", featuresCol="scaled_features")

# Define a parameter grid for hyperparameter tuning
param_grid = ParamGridBuilder() \
    .addGrid(svm.maxIter, [50, 100, 200]) \
    .addGrid(svm.regParam, [0.1, 0.01, 0.001]) \
    .build()

# Initialize CrossValidator
crossval = CrossValidator(estimator=svm,
                          estimatorParamMaps=param_grid,
                          evaluator=evaluator,
                          numFolds=5)

# Fit the CrossValidator to the training data
cv_model = crossval.fit(train_scaled)
```

```
# Make predictions on the testing set using the best model
predictions = cv_model.bestModel.transform(test_scaled)

# Calculate metrics
precision = evaluator.evaluate(predictions, {evaluator.metricName:␣
↪"weightedPrecision"})
recall = evaluator.evaluate(predictions, {evaluator.metricName:␣
↪"weightedRecall"})
f1_score = evaluator.evaluate(predictions, {evaluator.metricName: "f1"})
accuracy = evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"})

# Print metrics
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
print("Accuracy:", accuracy)
```

```
Precision: 0.8417442899624353
Recall: 0.7342081850533808
F1 Score: 0.7347317247860206
Accuracy: 0.7342081850533808
```

Fine-tuning Random Forest Model

[ ]:
```
# Initialize a Spark session
spark = SparkSession.builder.appName("RandomForestTuning").getOrCreate()

# Initialize a Random Forest Classifier
rf_classifier = RandomForestClassifier(labelCol="label", featuresCol="features")

# Define a grid of hyperparameters to search
param_grid = ParamGridBuilder() \
    .addGrid(rf_classifier.numTrees, [10, 20, 30]) \
    .addGrid(rf_classifier.maxDepth, [5, 10, 15]) \
    .addGrid(rf_classifier.maxBins,  [10, 20, 30]) \
    .build()

# Initialize a CrossValidator
cross_validator = CrossValidator(estimator=rf_classifier,
                                 estimatorParamMaps=param_grid,
                                ␣
↪evaluator=MulticlassClassificationEvaluator(labelCol="label",
                                                                            ␣
↪predictionCol="prediction",
                                                                            ␣
↪metricName="accuracy"),
```

```
                                    numFolds=3)

# Fit the CrossValidator to the training data
cv_model = cross_validator.fit(train_scaled)

# Make predictions on the test data using the best model
predictions = cv_model.transform(test_scaled)

# Calculate metrics
precision = evaluator.evaluate(predictions, {evaluator.metricName:␣
 ↪"weightedPrecision"})
recall = evaluator.evaluate(predictions, {evaluator.metricName:␣
 ↪"weightedRecall"})
f1_score = evaluator.evaluate(predictions, {evaluator.metricName: "f1"})
accuracy = evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"})

# Print metrics
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
print("Accuracy:", accuracy)
```

```
Precision: 0.8683491900711874
Recall: 0.8130931198102016
F1 Score: 0.8161905127401241
Accuracy: 0.8130931198102017
```

Fine-tuning Logistic Regression

[ ]:
```
# Create a Spark session
spark = SparkSession.builder.appName("LogisticRegressionTuning").getOrCreate()

# Assuming train_scaled is your scaled DataFrame
# Initialize a Logistic Regression classifier
lr = LogisticRegression(labelCol="label", featuresCol="scaled_features")

# Define a parameter grid for hyperparameter tuning
param_grid = ParamGridBuilder() \
    .addGrid(lr.regParam, [0.01, 0.1, 1.0]) \
    .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0]) \
    .build()

# Initialize CrossValidator
crossval = CrossValidator(estimator=lr,
                          estimatorParamMaps=param_grid,
                          evaluator=evaluator,
                          numFolds=5)
```

```python
# Fit the CrossValidator to the training data
cv_model = crossval.fit(train_scaled)

# Make predictions on the testing set using the best model
predictions = cv_model.bestModel.transform(test_scaled)

# Calculate metrics
precision = evaluator.evaluate(predictions, {evaluator.metricName:␣
 ↪"weightedPrecision"})
recall = evaluator.evaluate(predictions, {evaluator.metricName:␣
 ↪"weightedRecall"})
f1_score = evaluator.evaluate(predictions, {evaluator.metricName: "f1"})
accuracy = evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"})

# Print metrics
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
print("Accuracy:", accuracy)
```

```
Precision: 0.8355081692262887
Recall: 0.7366362692763938
F1 Score: 0.7379006371907408
Accuracy: 0.7366362692763938
```

[ ]:
```python
# Stop the Spark session
spark.stop()
```

6 Evaluate outcomes

[ ]:
```python
results ={
    "Decision Tree": {
        "Precision": 0.849,
        "Recall": 0.808,
        "F1 Score": 0.812,
        "Accuracy": 0.808
    },
    "Random Forest": {
        "Precision": 0.868,
        "Recall": 0.813,
        "F1 Score": 0.816,
        "Accuracy": 0.813
    },
    "XGBoost": {
        "Precision": 0.867,
        "Recall": 0.817,
```

```python
        "F1 Score": 0.820,
        "Accuracy": 0.817
    },
    "Linear SVC": {
        "Precision": 0.842,
        "Recall": 0.734,
        "F1 Score": 0.735,
        "Accuracy": 0.734
    },

    "Log Regression": {
        "Precision": 0.836,
        "Recall": 0.737,
        "F1 Score": 0.738,
        "Accuracy": 0.736
    },
}

for model, metrics in results.items():
    print(model)
    for metric, value in metrics.items():
        print(f"{metric}: {value:.3f}")
    print()
```

```
Decision Tree
Precision: 0.852
Recall: 0.809
F1 Score: 0.812
Accuracy: 0.809

Random Forest
Precision: 0.866
Recall: 0.809
F1 Score: 0.813
Accuracy: 0.809

NB
Precision: 0.794
Recall: 0.679
F1 Score: 0.678
Accuracy: 0.679

XGBoost
Precision: 0.867
Recall: 0.817
F1 Score: 0.820
Accuracy: 0.817
```

```
Linear SVC
Precision: 0.842
Recall: 0.735
F1 Score: 0.736
Accuracy: 0.735

Log Regression
Precision: 0.838
Recall: 0.738
F1 Score: 0.739
Accuracy: 0.738
```

6.1 Comparison of Spark vs SKLearn

Based on your experience in the assignments, write a brief report that compares Spark MLlib and Scikit-Learn (e.g., their pros/cons or similarity/difference).

1. **SKLearn's ease of use and syntax, as compared to Spark.**

SKLearn is very intuitive and easy to implement, in terms of syntax, readability and understandability.

The output in SKLearn's functions like describe, info and shape gives a simple result on the current progress of a data set, and grants a clear, concise, and instant overview of progress.

As opposed to Spark, which requires some intepretation and understanding.

2. **Algorithms:**

SKLearn: Offers a more comprehensive collection of machine learning algorithms, such as regression, classification, clustering, dimensionality reduction.

Spark: Provides a subset of machine learning algorithms, primarily focusing on scalable algorithms that can be distributed across a cluster. It might not have the same breadth of algorithms as scikit-learn, for example, KNN.

3. **Data Processing:**

SKLearn: It primarily focuses on machine learning algorithms. For data preprocessing and transformation, you might need to use other libraries or tools, like importing SimpleImputer and the like.

Spark: It provides tools for data preprocessing, feature extraction, and transformation, making it more suitable for end-to-end machine learning pipelines.

4. **Training Times:**

Spark is designed for distributed and parallel processing, suitable for larger datasets and training complex models. This is apparent in the faster training times of the ML models.

5. **Similarities:**

From an end-to-end data mining project standpoint, conceptually, Spark and SKLearn utilises the same processes, differing only in implementation.

The steps of visualising, pre-processing, selecting and training models, and evaluating outcomes remain the same.

The main draw back of Spark is the absence of a built in classification report function, only present in SKLearn, that presents a clear summary of a model's peformance.