# IDS project


| | |
|---|---|
| **Naman Mantri** | **18ucs151** |
| **Suryansh Bhandari** | **18ucs152** |
| **Prakhar Sharma** | **18ucs154** |
| **Anurag Gupta** | **18ucs210** |


## Problem Statement

Data Preprocessing and Preliminary Analysis and get inferences from the data.

## Data Sources

https://archive.ics.uci.edu/ml/datasets/Internet+Firewall+Data

## Specification of the Dataset

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 65532 | **Area:** | computer |
| **Attribute Characteristics:** | N/A | **Number of Attributes:** | 12 | **Date Donated** | 2019-02-04 |
| **Associated Tasks:** | Classification | **Missing Values?** | N/A | **Number of Web Hits:** | 4243 |

## 1. Goal

The goal of the project is to find insight about Internet Firewall's action based on different attributes in the dataset and also to perform preprocessing and statistical and descriptive analysis of the data using different visualization techniques and descriptive tables.

## 2. Importing Libraries

```
# importing all required libraries
import pandas as pd # library to manage dataframes
import numpy as np  # library for array calculations
import matplotlib.pyplot as plt # library for creating plots
import seaborn as sns # library for statistical analysis and visualizations
from collections import Counter
from sklearn.preprocessing import StandardScaler # library for normalizing
from sklearn import preprocessing
```

## 3. Importing Dataset

```
# url to extract dataset
url='https://archive.ics.uci.edu/ml/machine-learning-databases/00542/log2.csv'
df=pd.read_csv(url) # reading dataset from the url
print(df.head())  # printing first 5 values from dataset
```

```
   Source Port  Destination Port  ...  pkts_sent  pkts_received
0        57222                53  ...          1              1
1        56258              3389  ...         10              9
2         6881             50321  ...          1              1
3        50553              3389  ...          8              7
4        50002               443  ...         13             18
```

# 4. Exploring Dataset

## 4.1 Count of Null values

```
#Calculating count of null Values
print(df.isnull().sum())
```

```
Source Port              0
Destination Port         0
NAT Source Port          0
NAT Destination Port     0
Action                   0
Bytes                    0
Bytes Sent               0
Bytes Received           0
Packets                  0
Elapsed Time (sec)       0
pkts_sent                0
pkts_received            0
dtype: int64
```

## Insight:

- This dataset does not have any missing values.

## 4.2 Shape of dataset

```
# printing shape of dataframe (rows x columns)
print(df.shape)
```

```
(65532, 12)
```

## 4.3 Describe

```
# describing dataframe
print(df.describe())
```

| Index | Source Port | Destination Port | NAT Source Port | NAT Destination Port | Bytes | Bytes Sent | Bytes Received | Packets | :lapsed Time (sec | pkts_sent | pkts_received |
|---|---|---|---|---|---|---|---|---|---|---|---|
| min | 0 | 0 | 0 | 0 | 60 | 60 | 0 | 1 | 0 | 1 | 0 |
| 25% | 49364 | 53 | 0 | 0 | 66 | 66 | 0 | 1 | 0 | 1 | 0 |
| 50% | 54542 | 445 | 0 | 0 | 70 | 70 | 0 | 1 | 0 | 1 | 0 |
| 75% | 58715 | 25174 | 31549.8 | 53 | 205 | 102 | 98 | 2 | 30 | 1 | 1 |
| mean | 49418.6 | 12743.9 | 15696.7 | 2986.78 | 240.732 | 127.971 | 112.761 | 2.07544 | 35.4841 | 1.4099 | 0.665542 |
| std | 15978.6 | 19716.4 | 21070.4 | 10433.3 | 434.602 | 224.931 | 289.473 | 2.27927 | 181.477 | 1.22599 | 1.18357 |
| count | 53050 | 53050 | 53050 | 53050 | 53050 | 53050 | 53050 | 53050 | 53050 | 53050 | 53050 |
| max | 65534 | 65535 | 65535 | 65535 | 12807 | 9440 | 12678 | 24 | 3632 | 15 | 14 |

## Insight:

The above statistics show that data across all attributes are not in the same range, so we will have to normalize the data.

The features are not on the same scale. i.e. Source Port’s mean value is 49418.554741 while Destination Port’s mean value is 12743.891725. For most of the machine learning algorithms to be applied feature must be on the same scale. Let’s get an insight into the ‘action’ class label that is describing the types of classes in our Internet Firewall data set with 65532 instances and 12 attributes.

## 4.4 Unique class labels

```
#Types of classes
print(df['Action'].unique()) # printing all unique class labels
```

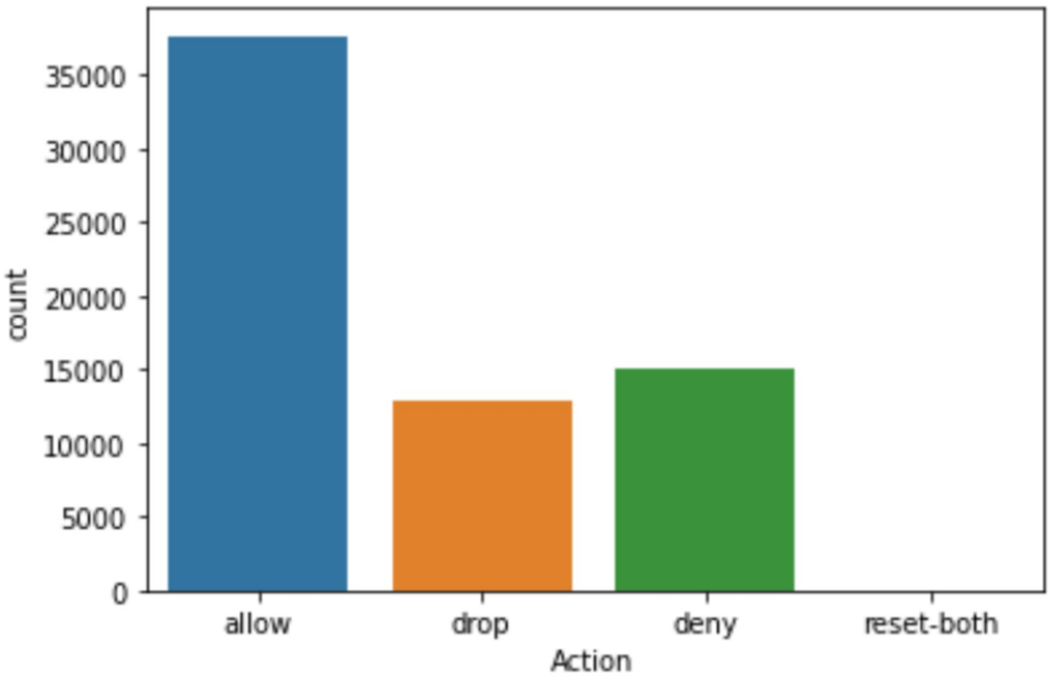
```
['allow' 'drop' 'deny' 'reset-both']
```

## 4.5 Class counts

```
#Count Number of Values Belonging to each class
print(df['Action'].value_counts())
```

```
allow          37640
deny           14987
drop           12851
reset-both        54
Name: Action, dtype: int64
```

```
# creating plot of  Number of Values Belonging to each class
sns.countplot(x=df['Action'])
```




As we can see The dataset is very very unbalanced.

The occurrences of the ‘allow’ class label constitute more than 50 % of the class types.

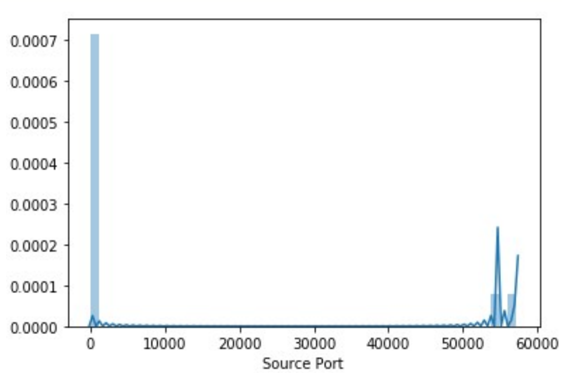
## 4.6 Attribute Information:

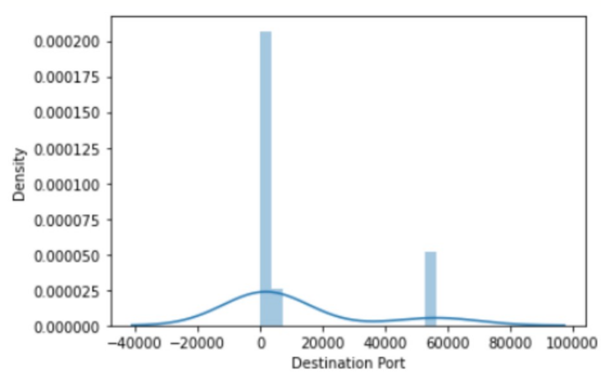
1. Source Port
2. Destination Port
3. NAT Source Port
4. NAT Destination Port
5. Bytes
6. Bytes Sent
7. Bytes Received
8. Packets
9. Elapsed Time (sec)
10. Pkts_sent
11. pkts_received
12. Type of Action:
    - Allow
    - Drop
    - Deny
    - Reset-both

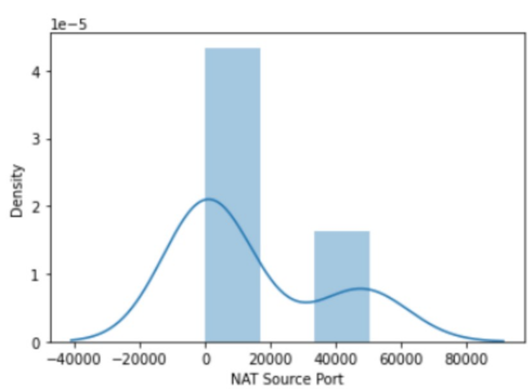
# 5. Data Visualization

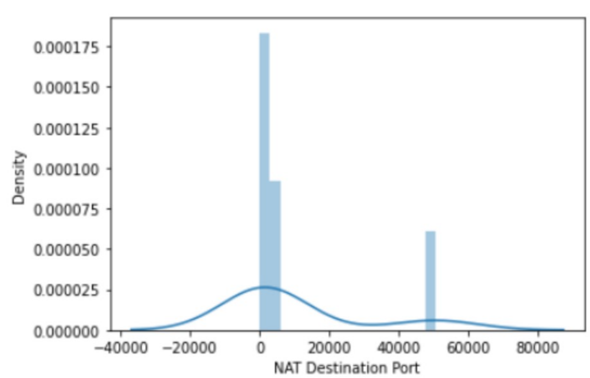
## 5.1 Using Univariate Plots

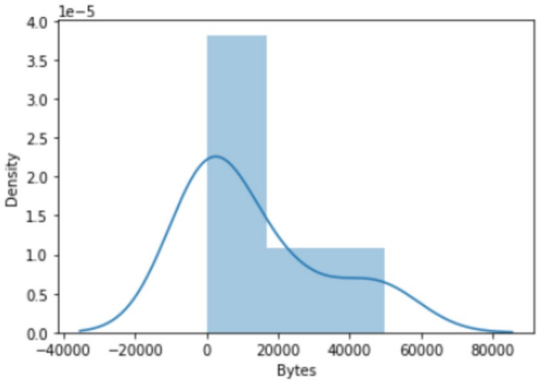
```python
# list of all features present in dataset
features=['Source Port','Destination Port','NAT Source Port',
          'NAT Destination Port','Bytes','Bytes Sent','Bytes Received',
          'Packets','Elapsed Time (sec)','pkts_sent','pkts_received']
label=['Action'] # label stores class label values

y = df[label] # storing value of class label
x=df[features] # storing all the values of features

x_val = x.values

# for loop creates distplot of each feature using sns library
for i in range(11):
 sns.distplot(x_val[i])
 plt.xlabel(features[i])
 plt.show()
```

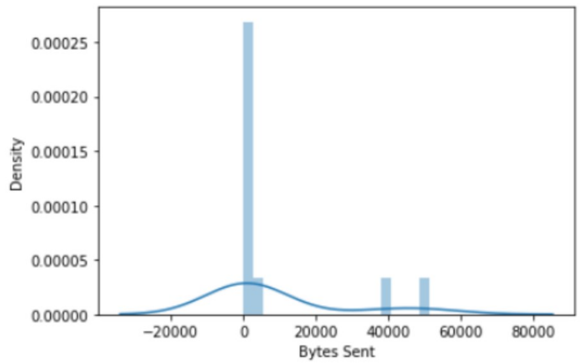


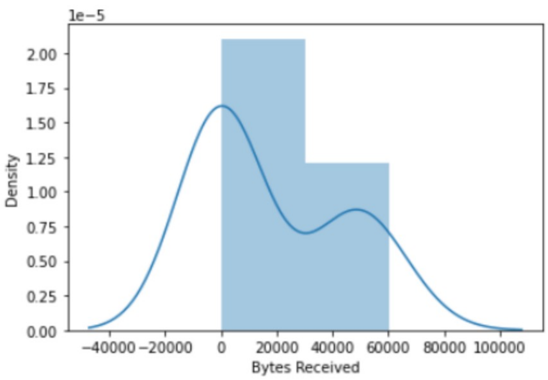



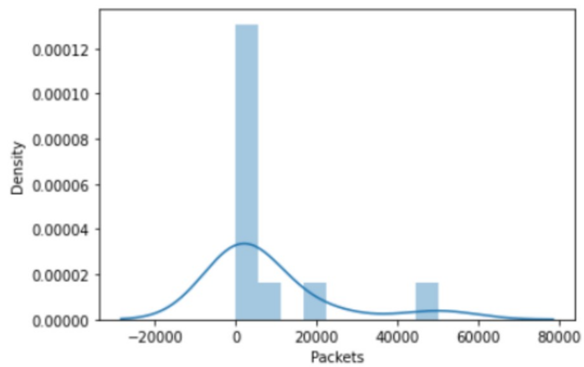



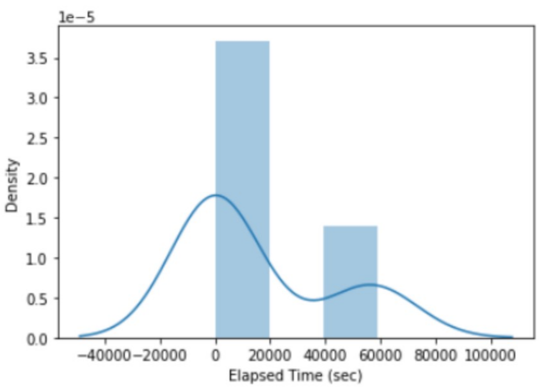



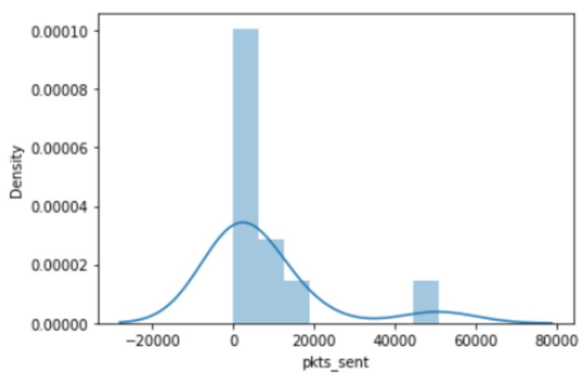

1e−5
Density
Bytes

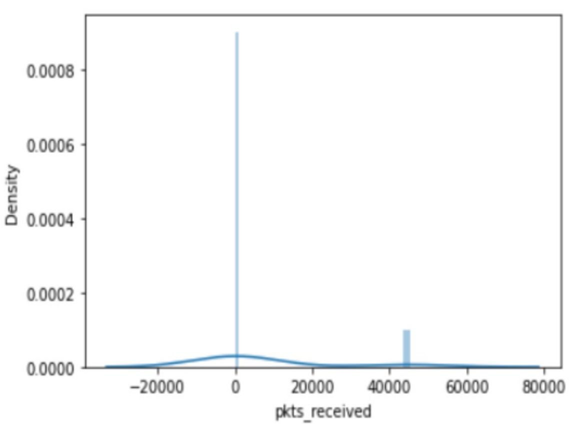

Density
Bytes Sent


1e−5
Density
Bytes Received


Density
Packets


1e−5
Density
Elapsed Time (sec)


Density
pkts_sent

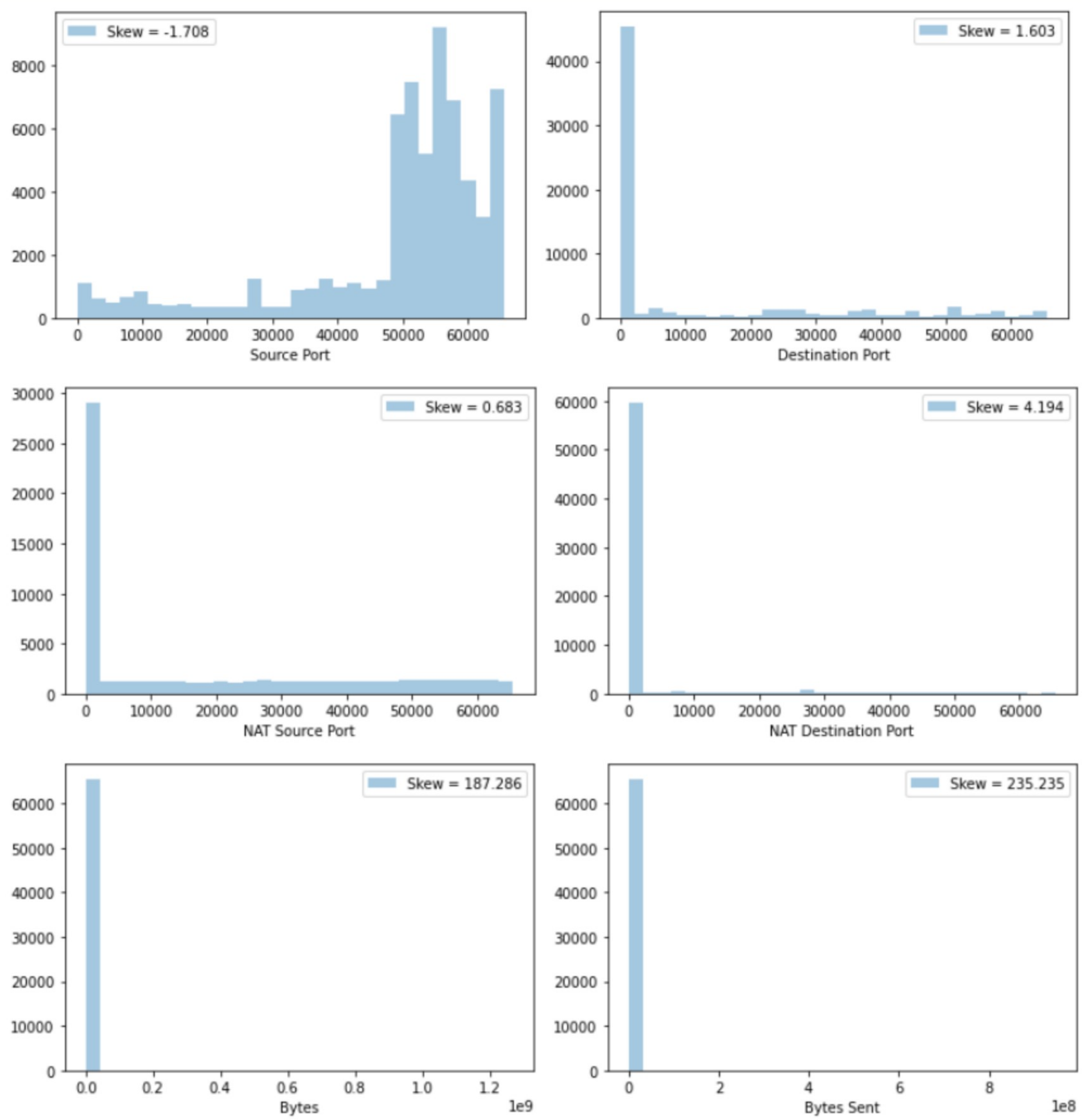

Density
pkts_received

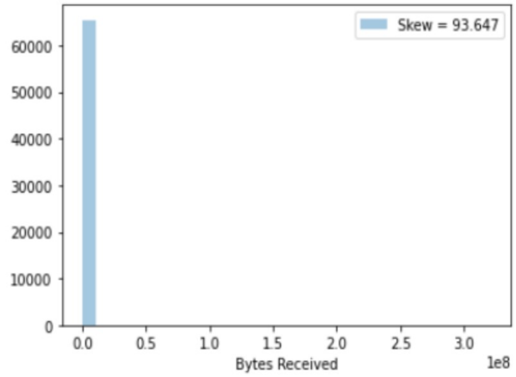
## Insight:

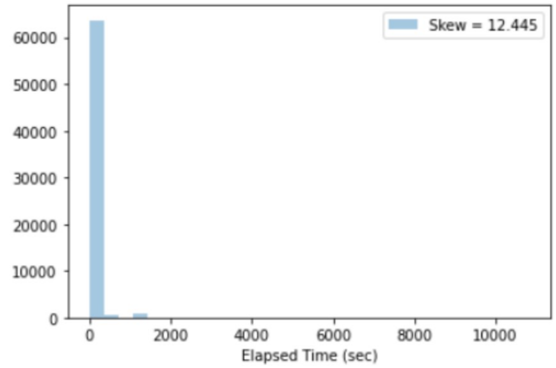
These univariate plots tell us that our data needs to be normalized as it is skewed either towards the left or right.

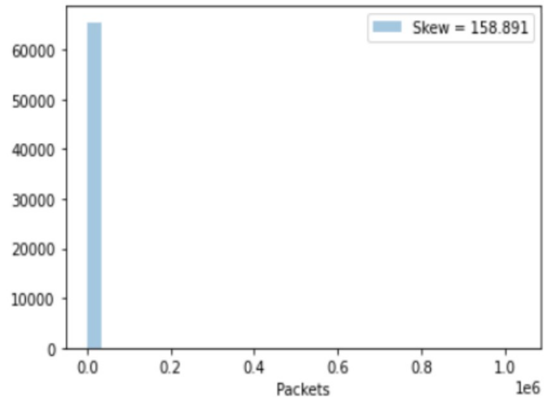
## 5.1.1 Skewness Plot

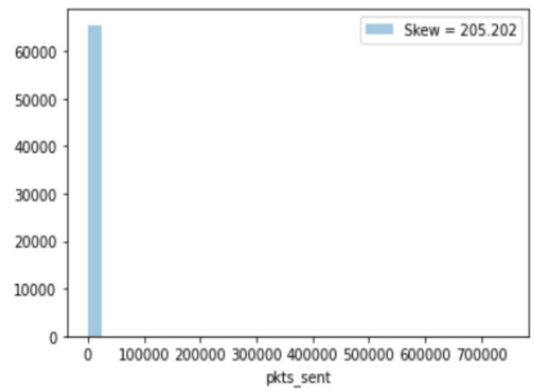
```python
# checking which features are not normalized using skewness(positive/negative/zero)
for j in features:
    skew = df[j].skew()
    sns.distplot(df[j], kde= False, label='Skew = %.3f'%(skew), bins=30)
    plt.legend(loc='best')
    plt.show()
```

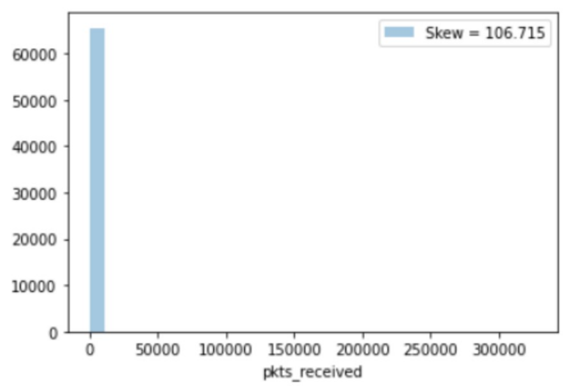




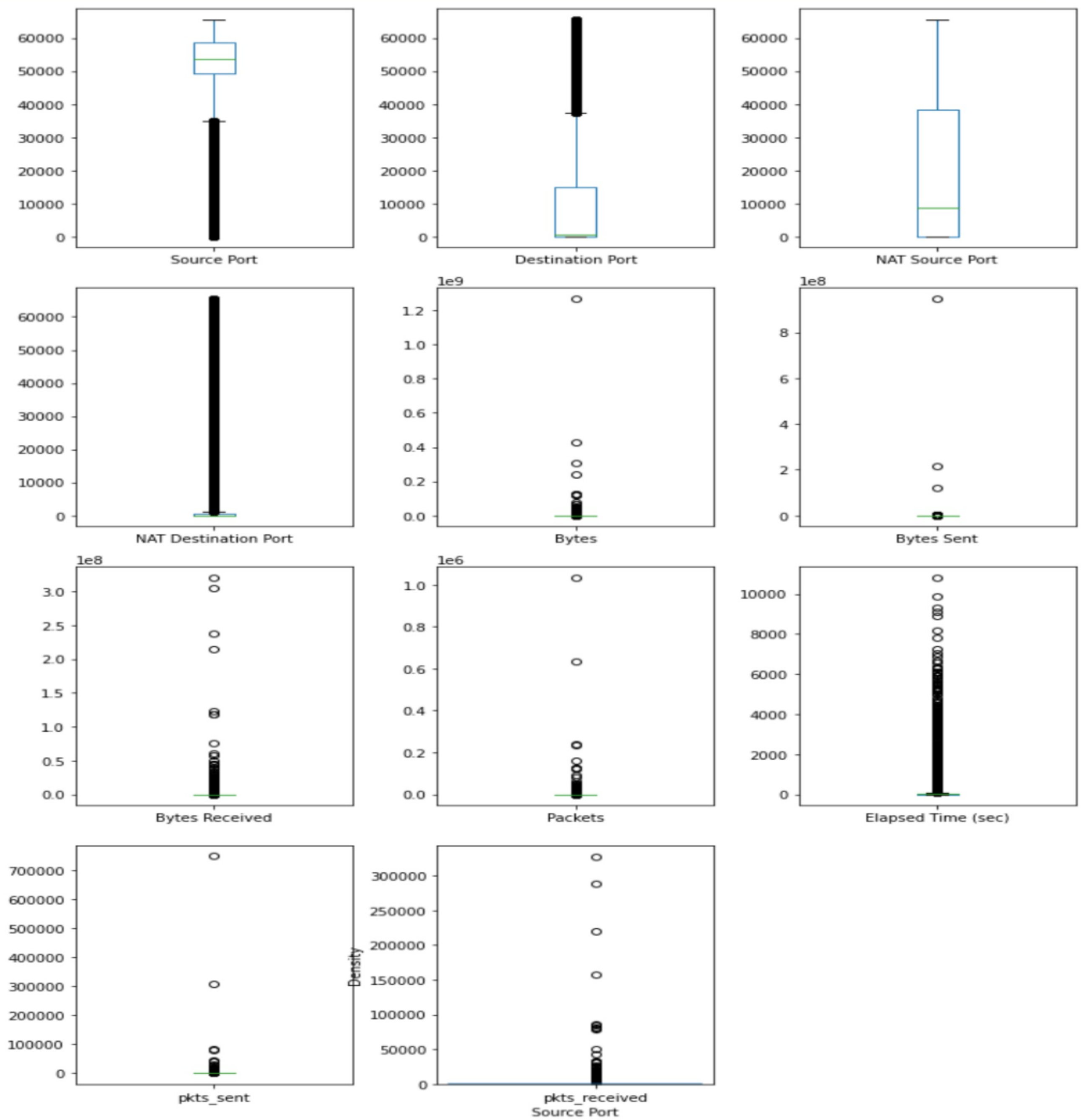




## Insight:

From the above graphs of multiple attributes, we can see that most of the attributes of our dataset are right-skewed and the ‘source port’ feature is left-skewed and thus data is not normalized.

## 5.1.2 Box Plot

```python
# creating box plot to show outliers in all features
plt.figure(figsize=(10,15))
for i,col in enumerate(list(x.columns.values)):
    plt.subplot(4,3,i+1)
    df.boxplot(col)
    plt.grid()
    plt.tight_layout()
```



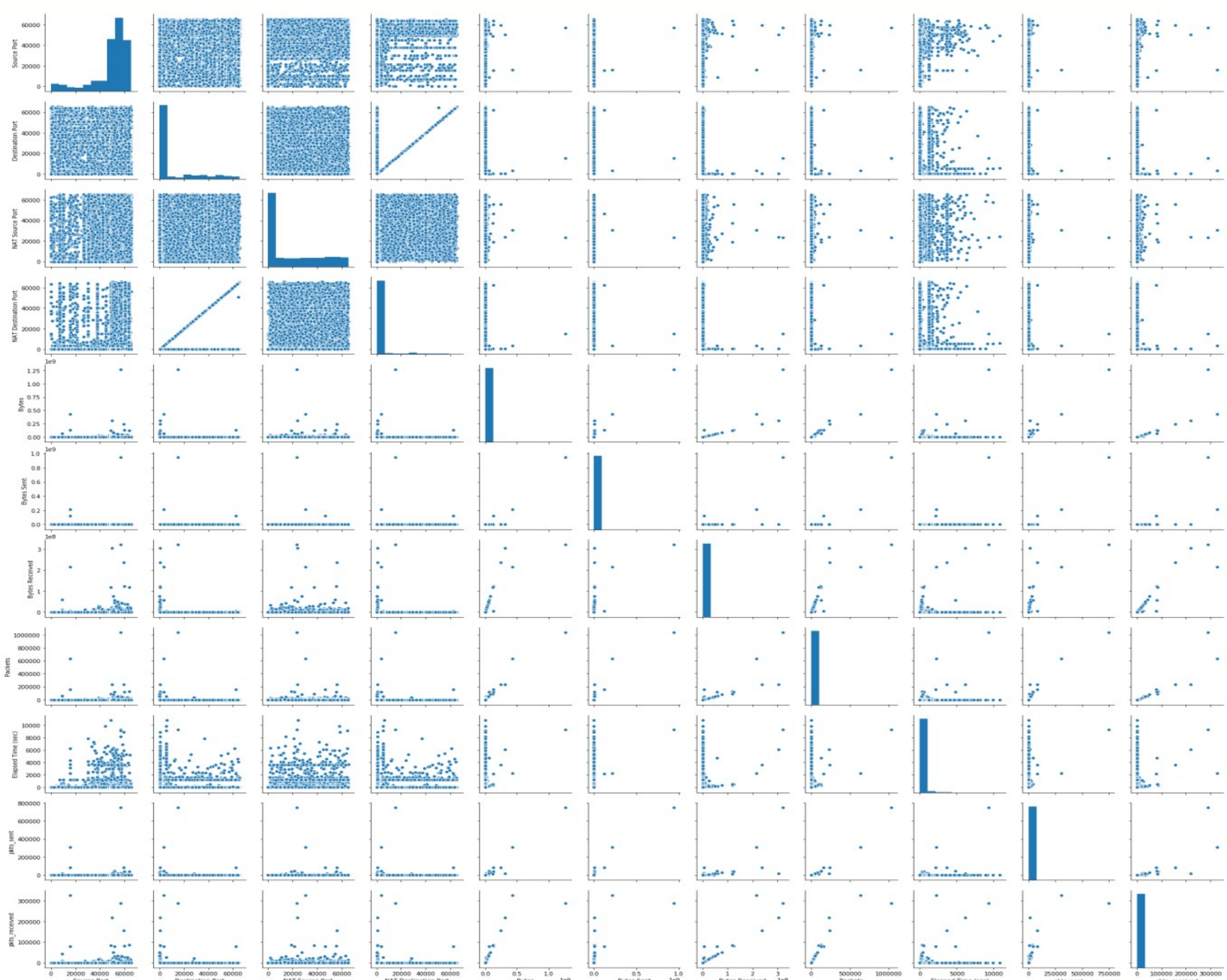
## Insight:

Above box plots of different attributes show outliers present in the dataset that might give problem while training the model on our data and thus needs to be removed

# 5.2 Using Multivariate Plots

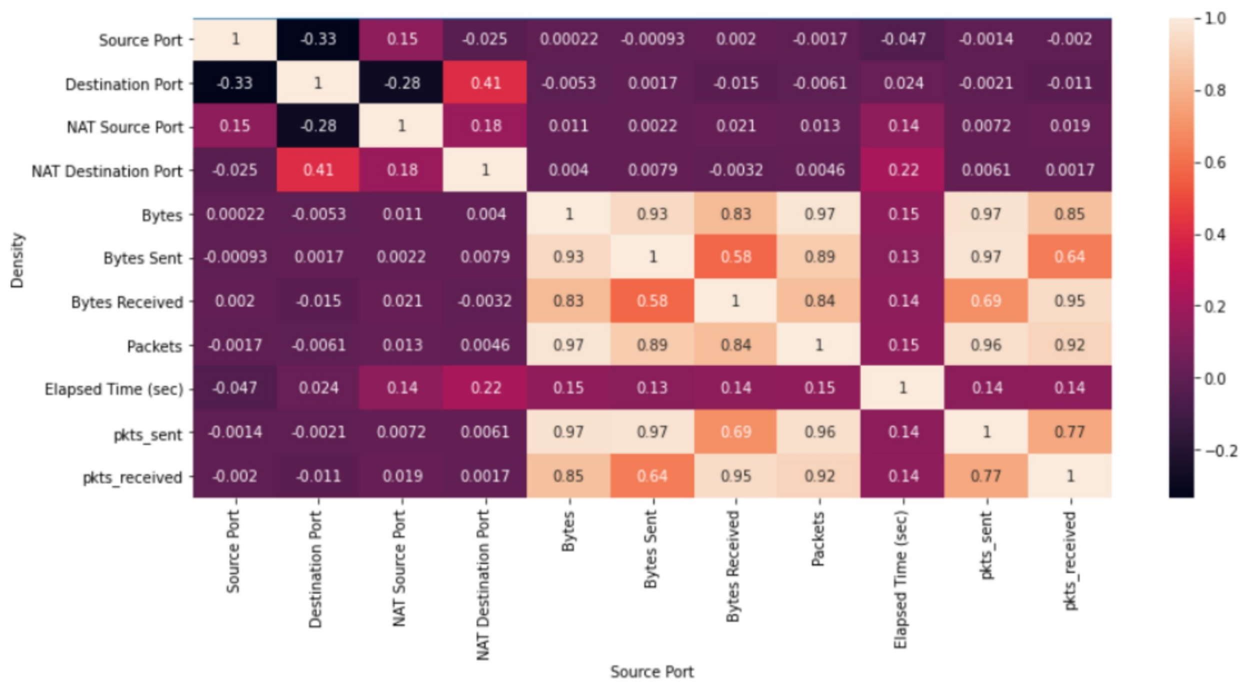
## 5.2.1 Pair plot of all the features

```
# creating pairplot of all the features
sns.pairplot(df)
```



## 5.2.2 Using Correlation Matrix to make heatmap

```
# creating a Heatmap using correlation matrix
corr=df.corr()
plt.figure(figsize=(14,6))
sns.heatmap(corr,color="k",annot=True)
```




## Insight:

1. From the correlation matrix, we can see that there are some attributes with a strong correlation between them ex: Bytes Sent and pkts_sent have a strong correlation(+0.97) between them.
2. We can observe that there are many attributes with less correlation between them ex: NAT Source Port and Elapsed Time have a very weak correlation(+0.14) between them.

# 6. Outlier Detection

```
# Detecting all the observations with more than four outlier using inter quartile range
def Iqr(df):
    out_index = []
    for col in df.columns.tolist():
        quartile1 = np.percentile(df[col], 25)
        quartile3 = np.percentile(df[col],75)
        IQR = quartile3 - quartile1
        out_list_col = df[(df[col] > quartile3 + 1.5 * IQR )|(df[col] < quartile1 - 1.5 * IQR) ].index
        out_index.extend(out_list_col)
    out_index = Counter(out_index)
    result = list( k for k, v in out_index.items() if v >4 )
    # taking feature with more than 4 outliers
    return result

print('Number of observations with more than 4 outliers in this dataset are %d'%(len(Iqr(df[features]))))

print(df.info())
```

# Insight:

In our data, There exists approximately 12482 observations consisting greater than 4 outliers, these might degrade the accuracy of any machine learning algorithm that is to be applied to the dataset.

# 7. Data treatment

# 7.1 Removing outliers

```
# removing outliers from dataset
out_index = Iqr(df[features])
df = df.drop(out_index).reset_index(drop=True)

print(df.shape) # printing shape of dataset after removing outliers


print(df.info())
```

# Insight:

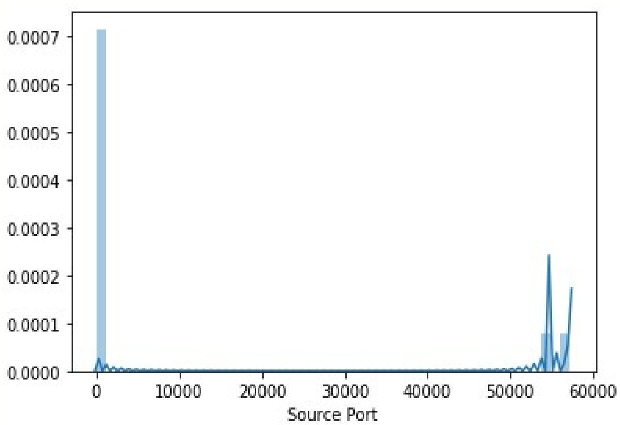
1. Removing observations with multiple outliers (more than 4) left us with 53050 observations to train from.

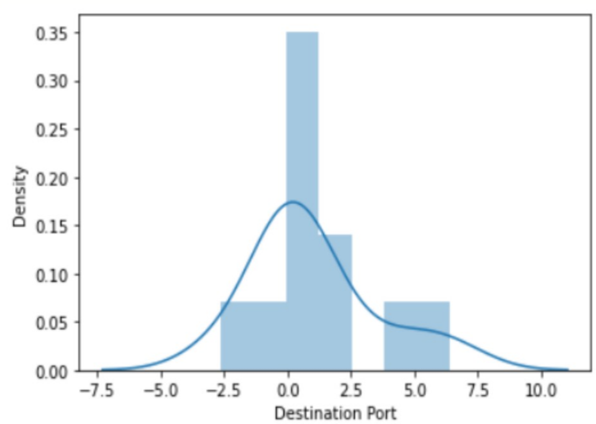
## 7.2 Normalizing the data

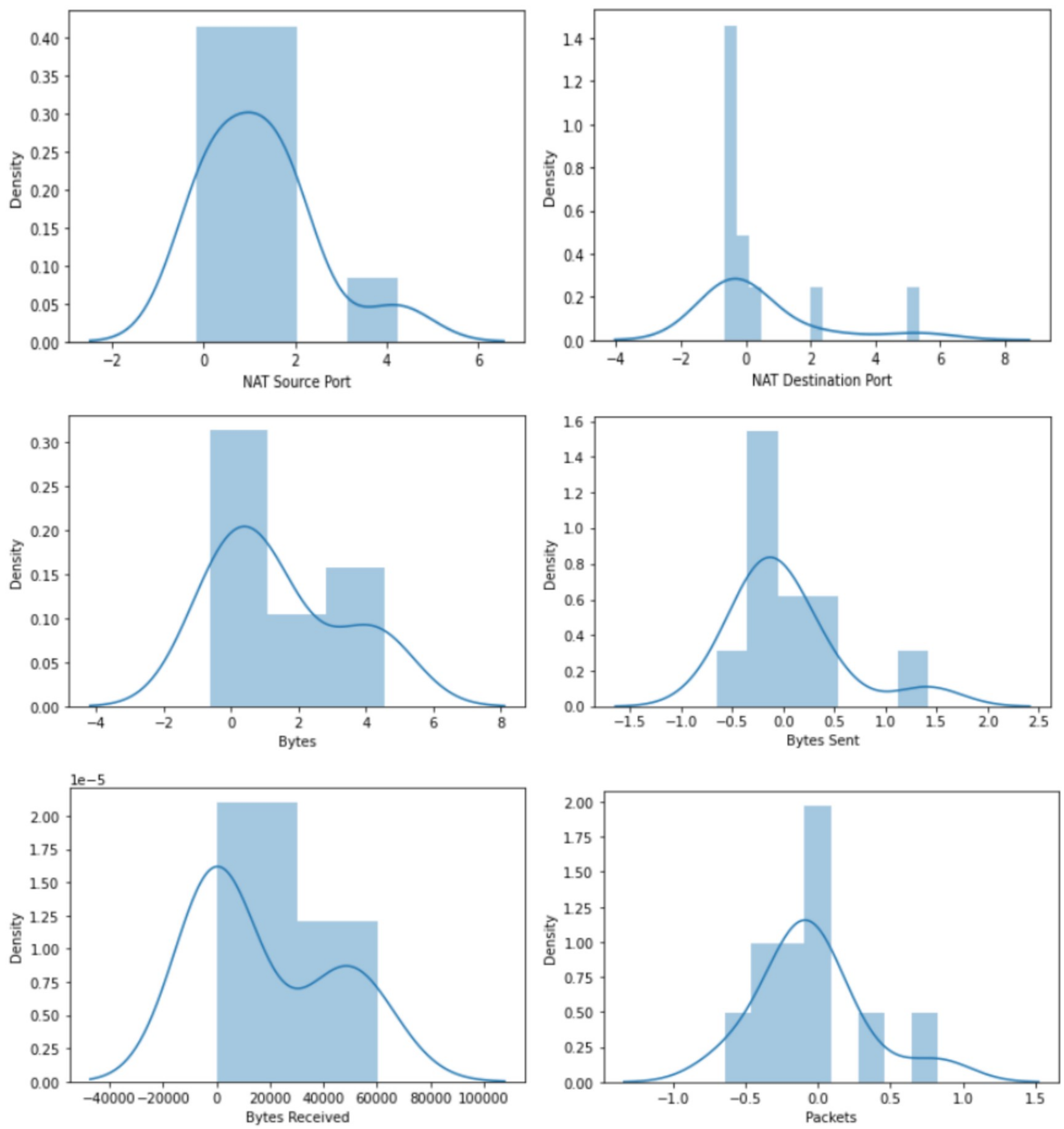
```
y = df[label]  # stores class label values
x=df[features] # x stores all features values after removing outliers


# Normalizing the data using Standard Scaler method
scaler=StandardScaler()
x=scaler.fit_transform(x) # normalizing on data (without outliers)
```

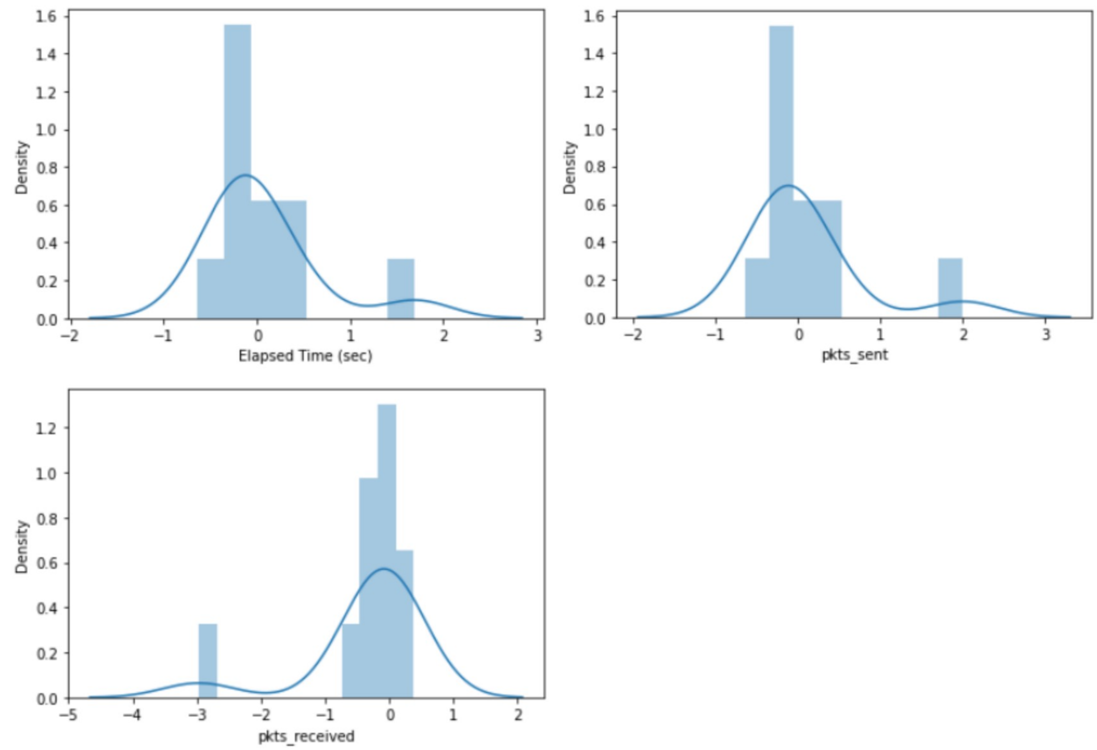
## 7.3 Visualization of Data after Being Preprocessed

```
# creating distplot for each feature after removing outliers from each instance
x2 = x
for i in range(11):
 sns.distplot(x2[i])
 plt.xlabel(features[i])
 plt.show()
```




Density
NAT Source Port
Density
NAT Destination Port
Density
Bytes
Density
Bytes Sent
1e−5
Density
Bytes Received
Density
Packets




## Insight:

According to the Diagrams above after preprocessing: Skewness is reduced and each feature is more normalized.

## 8. Code:

**Link to the github repository:**
[https://github.com/SuryanshBhandari/Ids_Project](https://github.com/SuryanshBhandari/Ids_Project)

```
# importing all required libraries
import pandas as pd # library to manage dataframes
import numpy as np  # library for array calculations
import matplotlib.pyplot as plt # library for creating
plots
import seaborn as sns # library for statistical
analysis and visualizations
from collections import Counter
```

```
from sklearn.preprocessing import StandardScaler
# library for normalizing
from sklearn import preprocessing

# url to extract dataset
url='https://archive.ics.uci.edu/ml/machine-learning-databases/00542/log2.csv'
df=pd.read_csv(url) # reading dataset from the url
print(df.head())  # printing first 5 values from dataset


#Calculating count of null Values
print(df.isnull().sum())

# printing shape of dataframe (rows x columns)
print(df.shape) # printing shape of dataframe (rows x columns)

#Types of classes
print(df['Action'].unique()) # printing all unique class labels

# describing dataframe
print(df.describe())

#Count Number of Values Belonging to each class
print(df['Action'].value_counts())

# creating plot of  Number of Values Belonging to each class
sns.countplot(x=df['Action'])

#sns.pairplot(df)
# creating pairplot of all the features
# creating a Heatmap using correlation matrix
corr=df.corr()
plt.figure(figsize=(14,6))
sns.heatmap(corr,color="k",annot=True)
```

```
# list of all features present in dataset
features=['Source Port','Destination Port','NAT Source
Port','NAT Destination Port','Bytes','Bytes Sent',
'Bytes Received','Packets','Elapsed Time (sec)',
'pkts_sent','pkts_received']
label=['Action'] # label stores class label values

y = df[label] # storing value of class label
x=df[features] # storing all the values of features

x_val = x.values

# for loop creates distplot of each feature using sns
library
for i in range(11):
 sns.distplot(x_val[i])
 plt.xlabel(features[i])
 plt.show()

# checking which features are not normalized using skew
ness(positive/negative/zero)
for j in features:
    skew = df[j].skew()
    sns.distplot(df[j], kde= False, label='Skew = %.3f'
%(skew), bins=30)
    plt.legend(loc='best')
    plt.show()

# creating box plot to show outliers in all features
plt.figure(figsize=(10,15))
for i,col in enumerate(list(x.columns.values)):
    plt.subplot(4,3,i+1)
    df.boxplot(col)
    plt.grid()
    plt.tight_layout()
```

```
# Detecting all the observations with more than four
outlier using inter quartile range
def Iqr(df):
    out_index = []
    for col in df.columns.tolist():
        quartile1 = np.percentile(df[col], 25)
        quartile3 = np.percentile(df[col],75)
        IQR = quartile3 - quartile1
        out_list_col = df[(df[col] > quartile3 +1.5
*IQR )|(df[col] < quartile1 - 1.5 * IQR) ].index
        out_index.extend(out_list_col)
    out_index = Counter(out_index)
    result = list( k for k, v in out_index.items()
if v >4 )
    # taking feature with more than 4 outliers
    return result

print('Number of observations with more than 4 outliers
 in this dataset are %d'%(len(Iqr(df[features]))))


print(df.info())

# removing outliers from dataset
out_index = Iqr(df[features])
df = df.drop(out_index).reset_index(drop=True)


print(df.shape) # printing shape of dataset after
removing outliers


print(df.info())

y = df[label]  # stores class label values
x=df[features] # x stores all features values after
removing outliers
```

```
# Normalizing the data using Standard Scaler method
scaler=StandardScaler()
x=scaler.fit_transform(x) # normalizing on data(without
 outliers)


# creating distplot for each feature after removing
outliers from each instance
x2 = x
for i in range(11):
 sns.distplot(x2[i])
 plt.xlabel(features[i])
 plt.show()
```