ELEC-E7130 Internet Traffic Measurements and Analysis

Assignment 6. Distributions and Sampling

Name: Xingji Chen

Student ID: 101659554

E-mail: xingji.chen@aalto.fi

Aalto University
School of Electrical
Engineering

Task 1: Introduction to distribution and sampling

1. What is sampling in statistics, and how does it help us understand data distributions?

   - In statistics, sampling refers to the process of selecting a subset of elements from a larger population for statistical inference. The main goal is to obtain a sample that is representative of the population so that the results can be generalized to the entire population to some extent.
   - Sampling allows estimation of population parameters such as mean, median, variance, etc. based on sample statistics. Sampling allows statisticians to make inferences about a population and calculate error margins and confidence intervals, which provide the range within which a population parameter is likely to exist. These all help to better understand the data distribution.

2. Choose at least three distributions from the following options and explain their respective parameters and typical applications.

   - Normal Distribution
     - ✓ Parameters:
       Mean (μ): The average value of the distribution.
       Standard Deviation (σ): A measure of the spread of the distribution.
     - ✓ Typical Applications:
       Measurement Errors: Errors in scientific experiments are often normally distributed.
       Standardized Testing: Scores are often normally distributed.

   - Beta Distribution
     - ✓ Parameters:
       Shape parameter α (alpha): Determines the shape of the distribution.
       Shape parameter β (beta): Determines the shape of the distribution.
     - ✓ Typical Applications:
       Probabilistic Modeling: Because the beta distribution is defined on the interval [0,1], it is very useful for modeling variables that represent probabilities.
       Bayesian Inference: The Beta distribution is the conjugate prior of the Bernoulli, binomial, negative binomial, and geometric distributions, making it very useful in

Bayesian statistics.

- Exponential Distribution
  - ✓ Parameters:
    Rate (λ): The rate parameter is the inverse of the mean.
  - ✓ Typical Applications:
    Survival Analysis: Models the timing of events, such as system failure or the lifespan of a biological organism.
    Queuing Theory: Modeling the time between customers arriving at a service point.

- Weibull Distribution
  - ✓ Parameters:
    Scale parameter (λ): Determines the scale of the distribution.
    Shape parameter (k): Determines the shape of the distribution.
  - ✓ Typical Applications:
    Reliability Analysis: used to model the life of products and systems, especially suitable for modeling product failures.
    Wind Speed Modeling: Used in the renewable energy sector to model wind speed.

- Log-gamma Distribution
  - ✓ Parameters:
    Shape parameter (α): Determines the shape of the distribution.
    Scale parameter (β): Determines the scale of the distribution.
    Location parameter (μ): Shifts the distribution along the x-axis.
  - ✓ Typical Applications:
    Skewed Data Modeling: Used when the data is skewed and can be modeled better by the log-gamma distribution than other distributions.
    Bayesian Inference: Like the Beta distribution, the log-gamma distribution is used in Bayesian statistics.

- Pareto Distribution
  - ✓ Parameters:
    Scale parameter (x_m): The minimum possible value that a random variable following a Pareto distribution can take.
    Shape parameter (α): Determines the tail heaviness of the distribution.

- ✓ Typical Applications:
  Wealth Distribution: Model the distribution of wealth among individuals.
  Network Traffic: Modeling the distribution of file sizes transmitted over the Internet.

3. What are the components of the following goodness-of-fit plots used to validate a model?

- Probability-Probability (P-P) plot
  X-axis: Theoretical cumulative probabilities of the chosen distribution.
  Y-axis: Empirical cumulative probabilities of the data.
  Diagonal Line (45-degree line): Represents a perfect fit between the theoretical and empirical distributions.

- Quantile-Quantile (Q-Q) plot
  X-axis: Theoretical quantiles of the chosen distribution.
  Y-axis: Sample quantiles of the data.
  Diagonal Line: If the distribution of the sample matches the theoretical distribution, the points should roughly follow this line.

- Comparison of probability density (empirical and theoretical)
  X-axis: Data values.
  Y-axis: Probability density.
  Empirical Density Curve: Estimated from the sample data.
  Theoretical Density Curve: Based on the assumed theoretical distribution.

- Comparison of cumulative distribution function (empirical and theoretical)
  X-axis: Data values.
  Y-axis: Cumulative probability.
  Empirical CDF: A step function that increases by 1/n at each data point, where n is the sample size.
  Theoretical CDF: The cumulative distribution function of the assumed theoretical distribution.

Aalto University
School of Electrical
Engineering

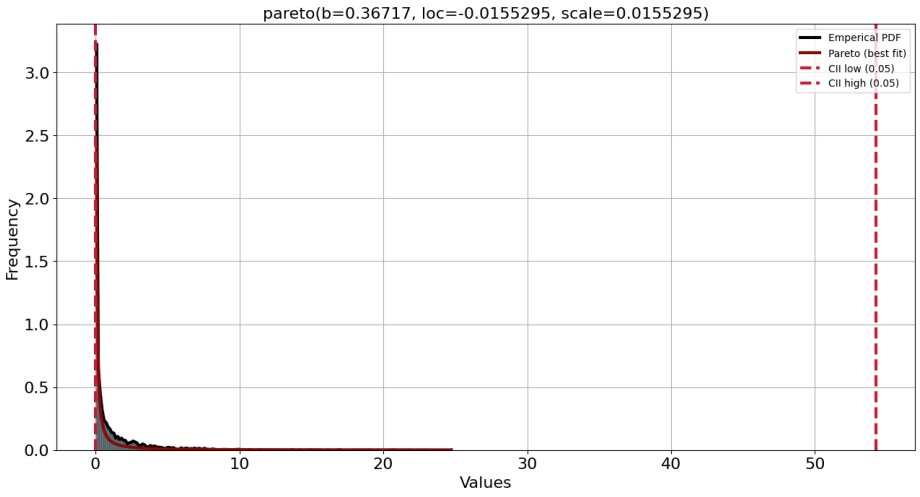
Task 2: Distributions

```
dist = distfit()

dist.fit_transform(data['value'].to_numpy())

dist.plot()
plt.show()
```

First, create an object of the distfit class named dist, used for fitting distributions. Next, fit the distribution and transform the data. Take a column from a pandas DataFrame, convert it to a NumPy array, and then use the fit_transform method of the dist object to perform the distribution fitting on this data. Finally, plot the results of the distribution fitting to display the fitted distribution alongside the original data.

The distfit is a library in Python that can automatically fit data for analysis. It performs probability density fitting on 89 univariate distributions through the Residual Sum of Squares (RSS) and Goodness of Fit (GOF) tests, returning the best distribution.

1. distr_a.txt



[norm] [0.00 sec] [RSS: 9.84458] [loc=1.147 scale=2.220]

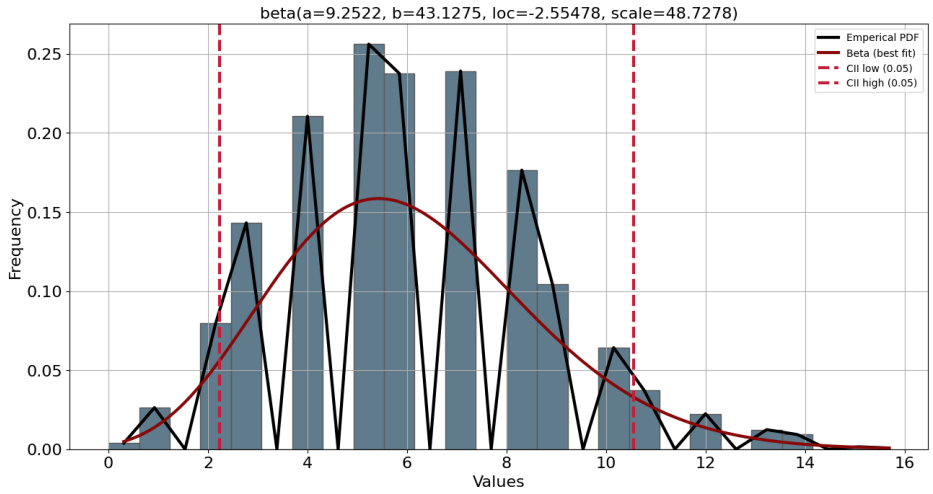
[expon] [0.00 sec] [RSS: 6.2506] [loc=0.000 scale=1.147]

[pareto] [0.00 sec] [RSS: 0.757524] [loc=-0.016 scale=0.016]
[dweibull] [0.22 sec] [RSS: 6.05357] [loc=0.000 scale=0.770]
[t] [0.20 sec] [RSS: 3.1662] [loc=0.011 scale=0.030]
[genextreme] [0.34 sec] [RSS: 2.82731] [loc=0.023 scale=0.076]
[gamma] [0.16 sec] [RSS: 1.89699] [loc=0.000 scale=1.541]
[lognorm] [0.00 sec] [RSS: 2.52195] [loc=-0.000 scale=0.094]
[beta] [0.26 sec] [RSS: 0.963583] [loc=0.000 scale=125.780]
[uniform] [0.00 sec] [RSS: 11.0723] [loc=0.000 scale=24.788]
[loggamma] [0.07 sec] [RSS: 9.89923] [loc=-994.810 scale=125.422]

Distribution: pareto
Parameters: (0.3671697261087662, -0.015529498528188095, 0.015529498529021512)

The Residual Sum of Squares (RSS) is the smallest under the pareto distribution, being 0.757524, hence this distribution is chosen.

2. distr_b.txt



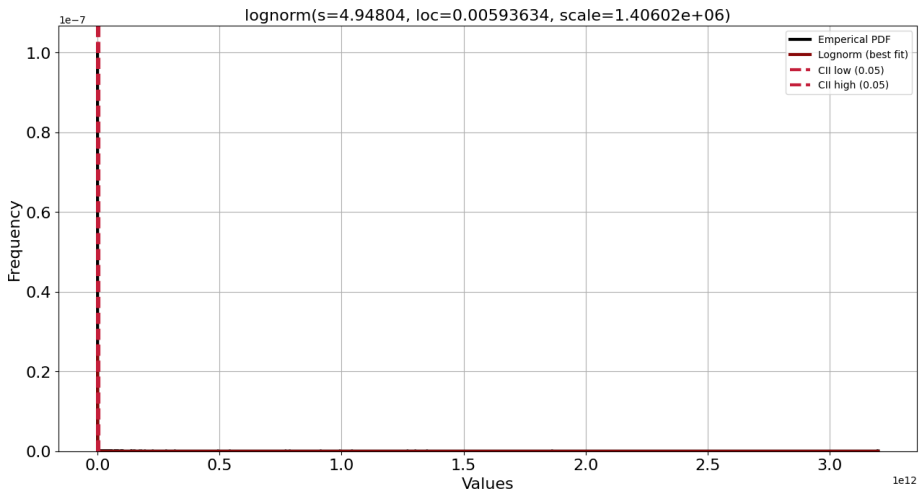
[norm] [0.00 sec] [RSS: 0.120892] [loc=6.052 scale=2.543]
[expon] [0.00 sec] [RSS: 0.219055] [loc=0.000 scale=6.052]
[pareto] [0.00 sec] [RSS: 0.219055] [loc=-1073741824.000 scale=1073741824.000]
[dweibull] [0.02 sec] [RSS: 0.135504] [loc=5.630 scale=2.234]
[t] [0.29 sec] [RSS: 0.120931] [loc=6.014 scale=2.445]

[genextreme] [0.08 sec] [RSS: 0.120738] [loc=5.010 scale=2.317]
[gamma] [0.02 sec] [RSS: 0.120589] [loc=-4.353 scale=0.623]
[lognorm] [0.03 sec] [RSS: 0.120624] [loc=-9.400 scale=15.246]
[beta] [0.07 sec] [RSS: 0.120484] [loc=-2.555 scale=48.728]
[uniform] [0.00 sec] [RSS: 0.197791] [loc=0.000 scale=16.000]
[loggamma] [0.05 sec] [RSS: 0.121026] [loc=-728.228 scale=100.080]

Distribution: beta
Parameters: (9.252203593352178, 43.12746028487105, -2.5547819163354797, 48.727754588960416)

The Residual Sum of Squares (RSS) is the smallest under the beta distribution, being 0.120484, hence this distribution is chosen.

3. distr_c.txt



[norm] [0.05 sec] [RSS: 1.0466e-14] [loc=6785925289.021 scale=88649550719.865]
[expon] [0.04 sec] [RSS: 1.04335e-14] [loc=0.006 scale=6785925289.015]
[pareto] [0.06 sec] [RSS: 6.86481e-15] [loc=-4788.347 scale=4788.353]
[dweibull] [0.23 sec] [RSS: 7.65242e-15] [loc=60.392 scale=44671299.022]
[t] [0.62 sec] [RSS: 8.67949e-15] [loc=4724.006 scale=14439.276]
[genextreme] [0.46 sec] [RSS: 9.11684e-15] [loc=3.560 scale=34.646]

[gamma] [0.18 sec] [RSS: 9.77155e-15] [loc=0.006 scale=1271224920487.997]
[lognorm] [0.08 sec] [RSS: 5.64665e-15] [loc=0.006 scale=1406024.683]
[beta] [0.31 sec] [RSS: 7.15044e-15] [loc=0.006 scale=8399542389770.129]
[uniform] [0.03 sec] [RSS: 1.04671e-14] [loc=0.006 scale=3199111619846.530]
[loggamma] [0.12 sec] [RSS: 1.04662e-14] [loc=-40401549125895.375 scale=5306499338792.770]

Distribution: lognorm
Parameters: (4.948038020458007, 0.005936338888217407, 1406024.683322883)

The Residual Sum of Squares (RSS) is the smallest under the lognorm distribution, being 5.64665e-15, hence this distribution is chosen.

Aalto University
School of Electrical
Engineering

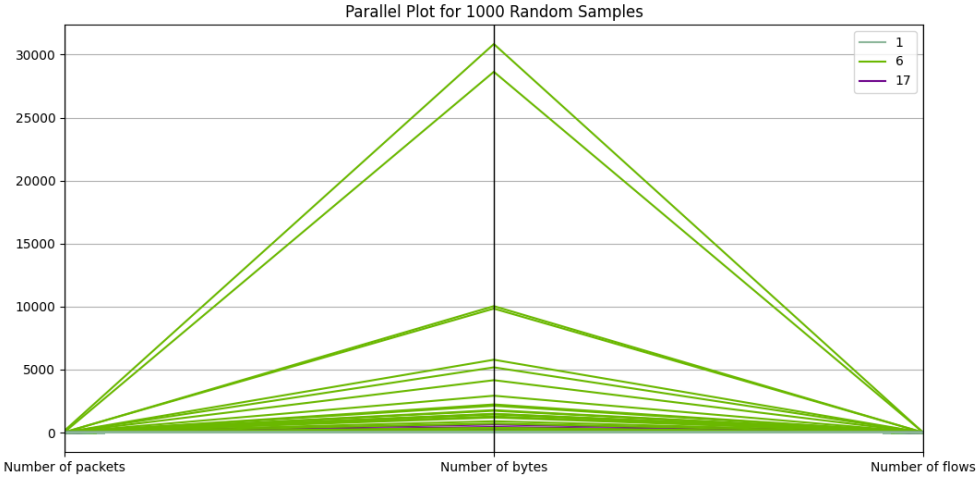
Task 3: Sampling

1. Overview of the data set

```
# Load dataset and specify column names
columns = [
    'Source IP (Anonymized)', 'Destination IP (Anonymized)',
'Protocol',
    'Is the port number valid', 'Source port', 'Destination port',
    'Number of packets', 'Number of bytes', 'Number of flows',
    'First packet arrival time', 'Last packet arrival time'
]
df = pd.read_csv('file/flowdata.txt', names=columns, sep='\t')

# Select 1000 random samples
sample_df = df.sample(1000)

# Create a parallel coordinate plot
numerical_columns = ['Number of packets', 'Number of bytes', 'Number
of flows']
plt.figure(figsize=(12, 6))
parallel_coordinates(sample_df[numerical_columns + ['Protocol']],
'Protocol')
plt.title('Parallel Plot for 1000 Random Samples')
plt.show()
```

Firstly, load the dataset named flowdata.txt. Provide a list of column names to correctly label the data when loading the dataset. Then, from the loaded dataset, select 1000 random samples using the sample() function. This helps to reduce the size of the data for quicker visualization and analysis. Finally, create a parallel coordinate plot focusing on three numerical columns: 'Number of packets', 'Number of bytes', and 'Number of flows'. In this plot, each line represents a sample, and each vertical line represents a dimension of the data.

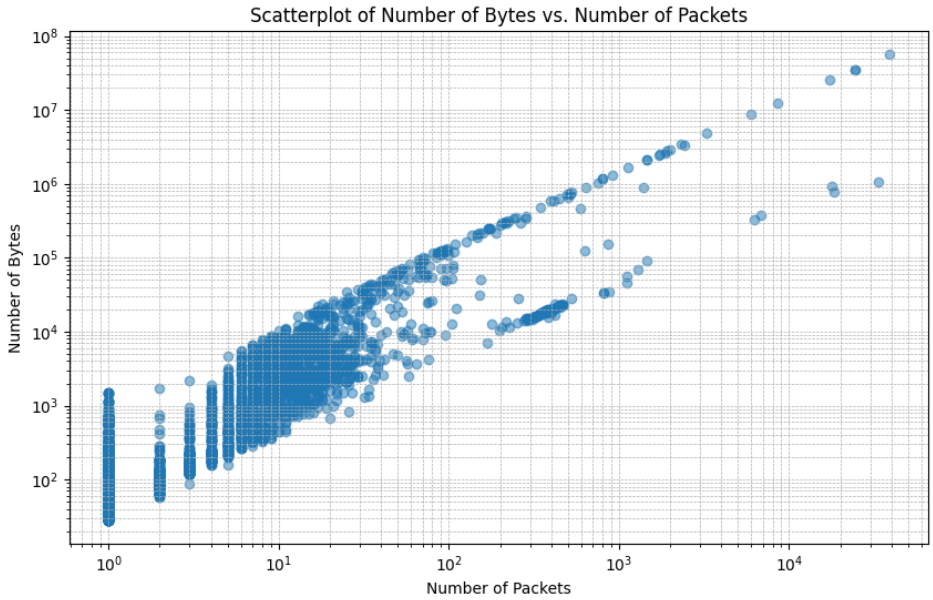
Aalto University
School of Electrical
Engineering

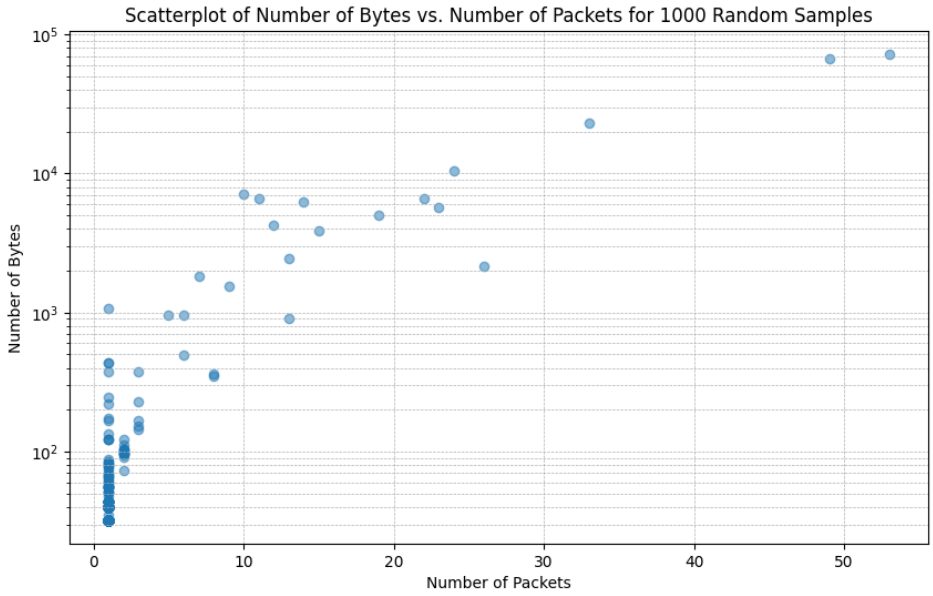


2. Number of bytes against packets

```
# Calculate the average packet size
df['Average packet size'] = df['Number of bytes'] / df['Number of
packets']
# Calculate the max average packet size for the whole dataset
max_avg_size_original = df['Average packet size'].max()
print(f"Maximum Average Packet Size for Original Dataset:
{max_avg_size_original:.2f} bytes/packet")
# Scatter plot
plt.figure(figsize=(10, 6))
# Apply log scale if needed. To determine the necessity, we'll check
the data range.
if df['Number of packets'].max() / df['Number of packets'].min() >
100:
    plt.xscale('log')
if df['Number of bytes'].max() / df['Number of bytes'].min() > 100:
    plt.yscale('log')
plt.scatter(df['Number of packets'], df['Number of bytes'],
alpha=0.5)
plt.xlabel('Number of Packets')
plt.ylabel('Number of Bytes')
plt.title('Scatterplot of Number of Bytes vs. Number of Packets')
plt.grid(True, which="both", ls="--", linewidth=0.5)
plt.show()
```

First, the average packet size and the maximum average packet size for the entire dataset are calculated. Next, it checks whether there's a need to apply a logarithmic scale to the x and y axes. If the data range on the x or y axis varies greatly (for instance, if the maximum value is more than 100 times the minimum), then a logarithmic scale is used to make the graph clearer. Lastly, a scatter plot is drawn.





The scatter plot of the original dataset in logarithmic coordinates uses a logarithmic scale to illustrate the relationship between the number of "bytes" and the number of "packets". The data mainly follows a diagonal line, indicating a certain positive correlation between bytes and packets. For smaller numbers of packets, the quantity of bytes is also relatively low. However, as the number of packets increases, the quantity of bytes also significantly rises. Using logarithmic coordinates allows for a clearer visualization of this trend, especially when there's a large disparity in the data quantities.

The scatter plot of 1000 random samples displays the relationship between bytes and packets among 1000 samples randomly selected from the original dataset. Compared to the first graph, there are fewer data points, and the distribution is relatively sparse. From the chart, it can be seen that the number of packets roughly ranges from 0 to 50, while the number of bytes is primarily concentrated at lower values.

Both of these charts aim to depict the relationship between bytes and packets in the dataset, but they utilize different visualization methods and subsets of data. The first graph offers a comprehensive view, showcasing all data points in the entire dataset. Given that the volume of data can be quite substantial, a logarithmic scale is used to better visualize the data. The second chart provides a view of a random sample, displaying only a small fraction of the original dataset, yet some trends and patterns are still observable. In essence, the second chart is a subset randomly selected from the first one. Therefore, although the second graph has fewer data points, they still reflect some of the primary characteristics and patterns found in the original dataset.

Maximum average packet size for original dataset: 1500.00 bytes/packet.
Maximum average packet size for 1000 random samples: 1358.21 bytes/packet。

3. Average throughput

```
df['Transfer Time'] = df['Last packet arrival time'] - df['First
packet arrival time']

# Calculate throughput, assigning NaN for flows with zero transfer
time
df['Throughput'] = df.apply(lambda x: x['Number of bytes'] /
x['Transfer Time'] if x['Transfer Time'] != 0 else None, axis=1)
```

```
# Handling flows transferred in zero time
flows_zero_time = df[df['Transfer Time'] == 0].shape[0]

print(f"Number of flows transferred in zero time: {flows_zero_time}")

# Calculate the average throughput, excluding NaN values
average_throughput = df['Throughput'].mean(skipna=True)

print(f"Average Throughput (excluding zero-time flows):
{average_throughput:.2f} bytes/second")
```

First, the transfer time is calculated by subtracting 'First packet arrival time' from 'Last packet arrival time'. Throughput is computed by dividing the number of bytes transferred by the transfer time. However, for flows with a transfer time of zero, the value is set to NaN (Not a Number) indicating that the throughput is indeterminable. Next, using the shape attribute (which provides the number of rows and columns in a dataframe), all rows with a 'Transfer Time' of zero are counted. In this context, the row count represents the number of flows transferred in zero time. Finally, the mean method is utilized to compute the average throughput of the dataset. This computation explicitly excludes NaN values, thereby omitting flows with a transfer time of zero.

Average throughput for original dataset: 1128591.57 bytes/second.
Average throughput for the 1000 sample: 1820777.64 bytes/second.

For flows transferred in zero time, mathematically, the throughput becomes infinite because the elapsed time is zero. This zero-time scenario might be due to the limitations of clock resolution. It doesn't necessarily mean that the flow is transmitted instantaneously, but rather that the transfer speed is faster than the clock's measuring capability. In practical applications, no flow is transferred in an absolute zero time.

To address this issue, one could set a minimum time threshold (for example, the smallest non-zero time in the dataset) and use that for flows with zero time. Alternatively, one can exclude flows with zero time from the throughput calculation, though if there are many such flows, this might introduce a bias in the results.

The average throughput for the original dataset is 1,128,591.57 bytes/second, while the average throughput for the 1000 random sample data is 1,820,777.64 bytes/second. Comparing these two values, the average throughput of the 1000 random samples is significantly higher than that of the original dataset. This suggests that the 1000 samples chosen might contain a certain number of high-throughput connections, leading to an increased average throughput. These 1000 samples may not fully represent the overall trend of the original dataset.

When selecting samples randomly from a large dataset, caution should be exercised as the selection might introduce bias, resulting in an analysis that differs from the entire dataset. To obtain more accurate results, it might be necessary to take a larger sample or use stratified sampling techniques.