Assignment 5

April 21, 2024

0.1 Assingment 5.

Name: Sifael Sebastian Ndandala Date: Mittwoch, 17. April. 2024

1. Find and Download a Data Set [5 Points]

See the project guidelines on TEAMS for hints how to find and choose interesting data sets.The data set should have at least 2 features. It does not have to be huge, but should contain atleast several hundred data points.
1. Describe the raw dataset you have chosen for the project in mathematical correct formalism, define the universe in which the data has been taken, the RV functions, and the data value spaces S. 2. Describe whether you had to clean the data, whether there are any missing values. 3. How did you deal with missing values? 4. Show excerpts of the raw data.

2. Distribution, Moments [5 Points]

Choose from this data set 1 numerical features where you have at least 100 measurements withresults in a discrete DVS S (if it is a continous S, choose intervals with some bin width d, tomap it on a discrete S). a) Plot the corresponding histograms and the pmf of this discrete distribution. b) Calculate the mean value, the standard deviation, the skewness and the kurtosis of this dis-tribution. c) Check all kinds of distributions which we reviewed in the lecture. To which kind of dis-tribution function does the pmf of the feature you choose resemble most closely? Find thehyperparameters of that distribution function which minimise the MSE.

0.2 Question 1: Summarizing the Dataset in Mathematical Formulation

1.1. Dataset Summary: Financial Phrase Bank The selected dataset for my project is the Financial Phrase Bank dataset. The dataset is a Polar Sentiment dataset of sentences from financial news. The dataset consists of 4840 sentences from English language financial news categorised by sentiment. The selected collection of phrases was annotated by 16 people with adequate background knowledge on financial markets. The dataset is divided by agreement rate of 5-8 annotators, all of which determine whether the financial news phrase is `positive`, `neutral` or `negative`. In essence, that data is a pair of `financial news phrases` and their respective sentiment *positive* ,*neutral* ,*negative*

1.2. Mathematical Formalism Below is the Mathematical Formalism of the Dataset

1.3. Universe of the Dataset. The universe Ω of the dataset "financial_phrasebank" is the collection of all individual phrases (or sentences) from global financial news articles. Each phrase is collected under certain constraints, likely within a specific time frame and from specific sources that publish financial news.

$$\Omega = \text{All Financial News Articles Published and their respective Sentiment}$$

1.4. Random Variable Function of the Dataset The Random Variable Function of the Dataset for Financial News and respective sentiment can be given as follows:

$ X_{F}: financial news $ $ X_{S}: sentiment associated with the financial news $

Therefore:

$ X_F : S_{f} $ where $S_{f} = {financial news phrases selected } < *br* >X_S : S_{s}$ where $S_{s} = {positive, negative, neutral} $

1.4. Product RV and DVS Based on the above definition, the Product RV and Data Value space is given by the following formalism:

$$X = X_F \otimes X_S$$

Therefore, the product DVS is:

$$S_I = S_F \times S_S$$

The index I contains a paired set such that $i = \{$ financial news, sentiment$\}$

[1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

[2]:
```python
pd.set_option('max_colwidth', 150)
```

[3]:
```python
financial_news = pd.read_csv('financial_news.txt')
financial_news.head()
```

[3]:
```
                                                                    sentence  \
0                          According to Gran , the company has no plans to move
all production to Russia , although that is where the company is growing .
1  Technopolis plans to develop in stages an area of no less than 100,000 square
meters in order to host companies working in computer technologies a…
2  The international electronic industry company Elcoteq has laid off tens of
employees from its Tallinn facility ; contrary to earlier layoffs the c…
3  With the new production plant the company would increase its capacity to meet
the expected increase in demand and would improve the use of raw mat…
4  According to the company 's updated strategy for the years 2009-2012 ,
Basware targets a long-term net sales growth in the range of 20 % -40 % wit…
```

```
   label
0      1
1      1
2      0
3      2
4      2
```

0.2.1 Data Acquisition and Basic Preprocessing

The dataset is available through HuggingFace, an online collective of datasets for specific analysis and machine learning processing. The data can be downloaded using the following code.

```
from datasets import load_dataset

dataset = load_dataset("financial_phrasebank", "sentences_50agree")
```

0.2.2 Cleaning and Missing Values

The data is well curated and does not contain missing values. However, feature extraction is need to understand the dataset. For example computing the length of characters or tokens from the text to understand the distribution of labels to financial text.

[4]:
```
len(financial_news)
```

[4]:
```
4846
```

[5]:
```
financial_news.label.value_counts(normalize=True)
```

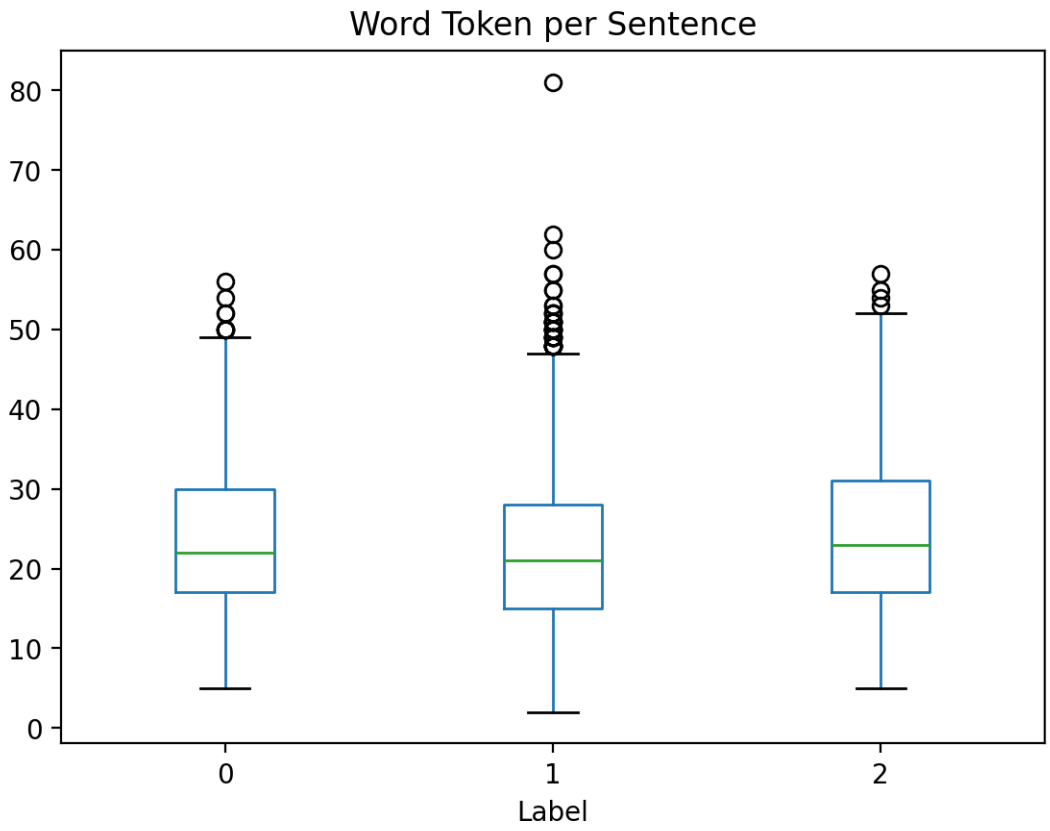
[5]:
```
label
1    0.594098
2    0.281263
0    0.124639
Name: proportion, dtype: float64
```

0.3 Question 2: Distribution, Moments

Choose from this data set 1 numerical features where you have at least 100 measurements withresults in a discrete DVS S (if it is a continous S, choose intervals with some bin width d, tomap it on a discrete S).

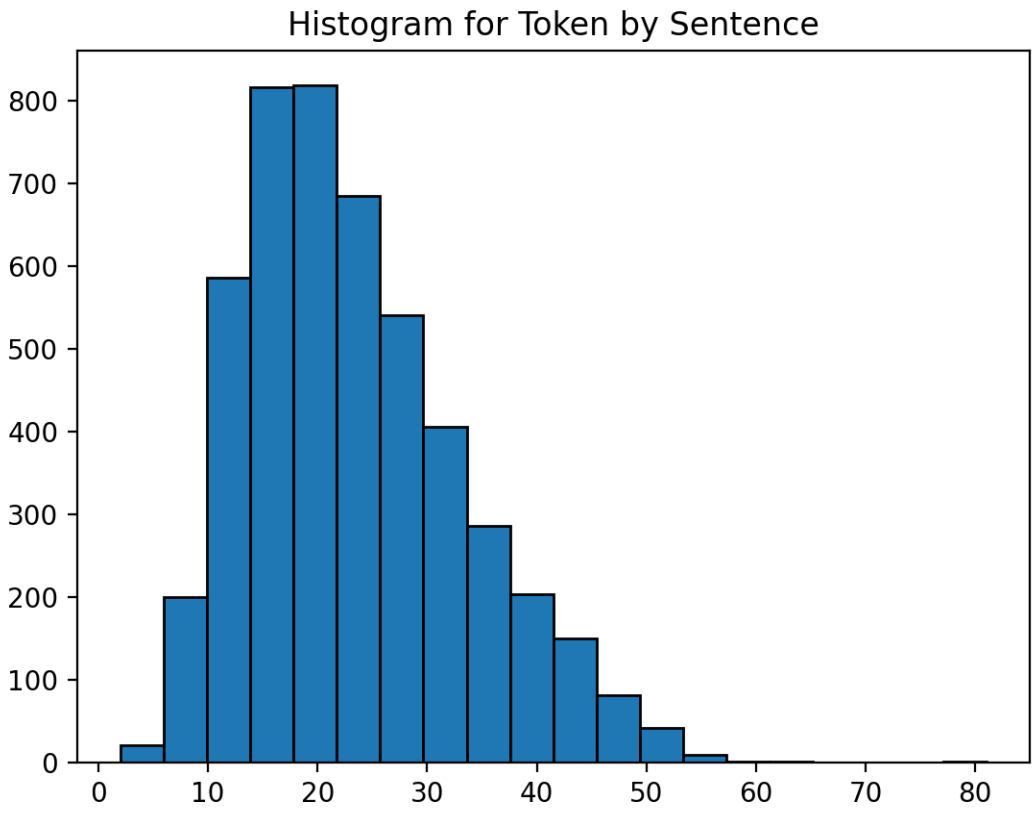
[6]:
```
financial_news['tokens'] = financial_news['sentence'].str.split().apply(len)
```

[7]:
```
financial_news.boxplot("tokens", by="label", grid=False, showfliers=True)
plt.title('Word Token per Sentence')
plt.suptitle('')
plt.xlabel('Label')
plt.show();
```



Plotting the Histogram and the PMF Function The code below implements the visualization of the Histogram and the PMF Function.

[8]:
```
plt.hist(financial_news.tokens, ec='black', bins=20 )
plt.title('Histogram for Token by Sentence')
plt.show()
#plt.savefig('token-distribution.png')
```



Compute Mean, Standard Deviation, Skewness and Kurtosis The code below implements the computation of Mean, Standard Deviation, Skewness and Kurtosis

[9]:
```
from scipy import stats

mean = np.mean(financial_news.tokens)
standard_deviation = np.std(financial_news.tokens)
skewness = stats.skew(financial_news.tokens)
kurtosis = stats.kurtosis(financial_news.tokens)

print(f"The Mean: {mean}")
print(f"The Standard Deviation: { standard_deviation }")
print(f"The Skewness: { skewness }")
print(f"The Kurtosis: {kurtosis}")
```

```
The Mean: 23.101114321089558
The Standard Deviation: 9.957446127636572
The Skewness: 0.7076098948337926
The Kurtosis: 0.20661787520453512
```

0.3.1 Fitting Probability Distributions

To fit the probability distributions, I first compute the pm by the bin. I will then use the pmf and bins with the `curve_fit` method to fit the distribution.

[10]:
```
pmf, bins = np.histogram(financial_news.tokens, bins=20, density=True)
```

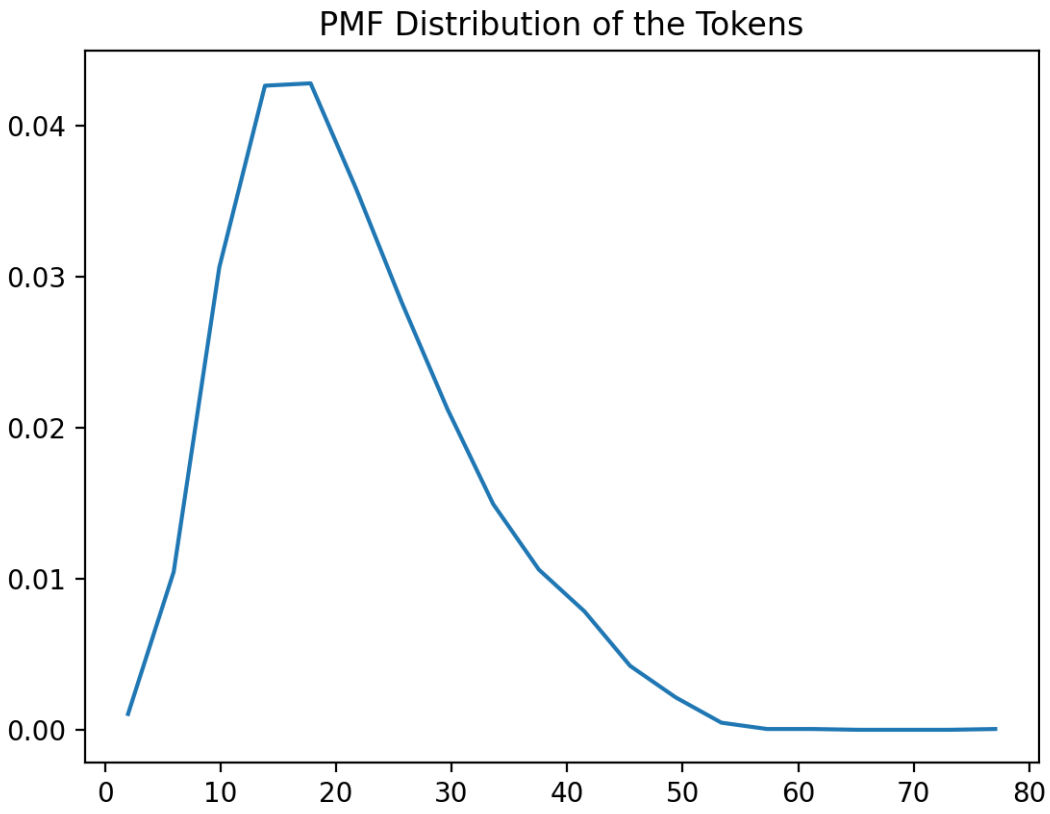
[11]:
```
pmf, bins
```

[11]:
```
(array([1.04483928e-03, 1.04483928e-02, 3.06137908e-02, 4.26294425e-02,
        4.27861684e-02, 3.57857453e-02, 2.82629025e-02, 2.12102373e-02,
        1.49412017e-02, 1.06051187e-02, 7.83629458e-03, 4.23159907e-03,
        2.14192052e-03, 4.70177675e-04, 5.22419639e-05, 5.22419639e-05,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 5.22419639e-05]),
 array([ 2.  ,  5.95,  9.9 , 13.85, 17.8 , 21.75, 25.7 , 29.65, 33.6 ,
        37.55, 41.5 , 45.45, 49.4 , 53.35, 57.3 , 61.25, 65.2 , 69.15,
        73.1 , 77.05, 81.  ]))
```

[12]:
```
bins.size, pmf.size
```

[12]:
```
(21, 20)
```

[13]:
```
plt.plot(bins[:20], pmf)
plt.title('PMF Distribution of the Tokens')
plt.show();
```



[14]:
```
len(pmf), len(bins[:20])
```

[14]: (20, 20)

0.3.2 Modelling the Distribution into Functions

1. Normal Distribution The probability density function (PDF) of the normal distribution is:

$$f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where: - $ $ is the mean of the distribution, - $ $ is the standard deviation, - $ x $ represents the variable.

[15]:
```
from scipy.optimize import curve_fit
from scipy.stats import poisson, norm, lognorm, weibull_min, levy

# Convert the dataset to a NumPy array for better handling
data = np.array(financial_news.tokens)

# We need to create histogram data to fit
```

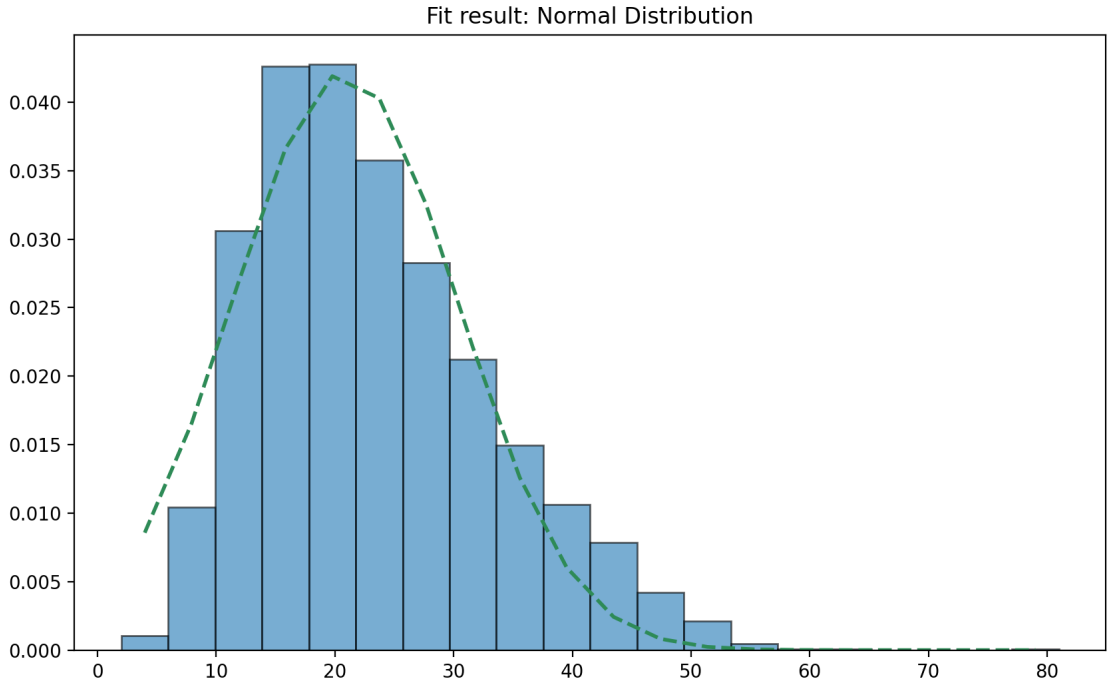
```
values, base = np.histogram(data, bins=20, density=True)

# Calculate mid points for bins
x_data = base[:-1] + np.diff(base) / 2
```

[16]:
```
def fit_normal(x, mean, std):
    return norm.pdf(x, mean, std)

params_norm, _ = curve_fit(fit_normal, x_data, values, p0=[np.mean(data), np.
 ↪std(data)])

# Plotting the result
plt.figure(figsize=(10, 6))
plt.hist(data, bins=20, density=True, alpha=0.6, ec='black')
plt.plot(x_data, fit_normal(x_data, *params_norm), color='seagreen',␣
 ↪linestyle='--', linewidth=2)
plt.title("Fit result: Normal Distribution")
plt.show()
```



0.3.3 2. Poisson Distribution

The probability mass function (PMF) of the Poisson distribution is:

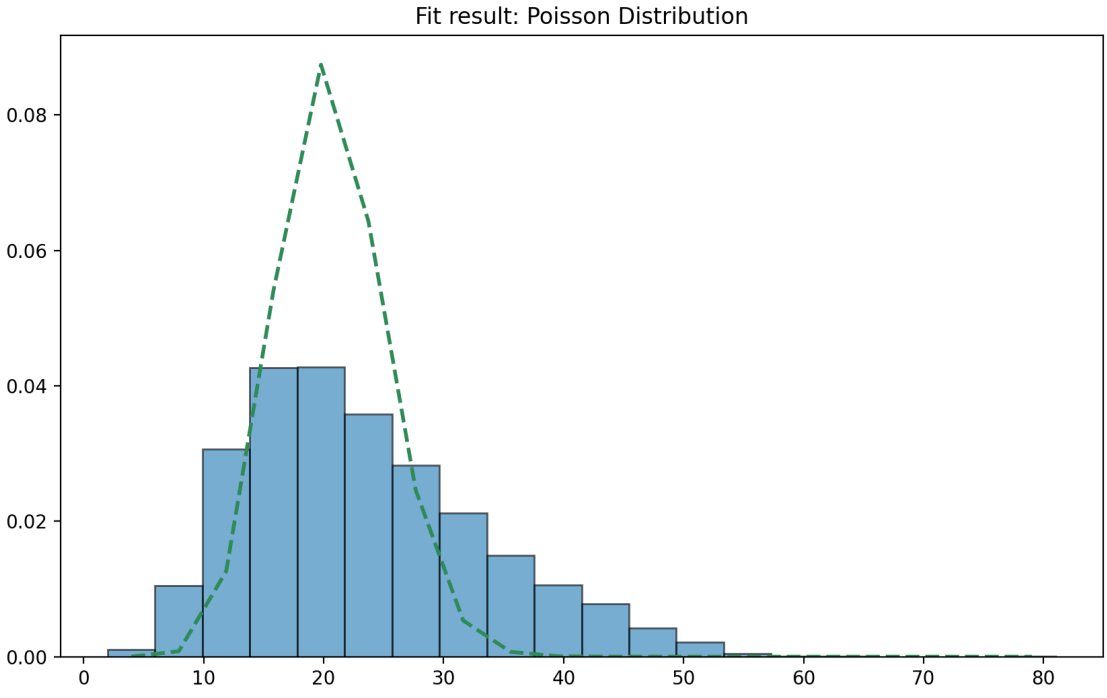
$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

where: - $ $ is the average number of events in an interval, - $ k $ is the number of occurrences (an integer), - $ e $ is the base of the natural logarithm.

[17]:
```
def fit_poisson(x, mu):
    # Because Poisson is a PMF and expects integers, we use the round function
    return poisson.pmf(np.round(x), mu)

params_poisson, _ = curve_fit(fit_poisson, x_data, values, p0=[np.mean(data)])

plt.figure(figsize=(10, 6))
plt.hist(data, bins=20, density=True, alpha=0.6, ec='black')
plt.plot(x_data, fit_poisson(x_data, *params_poisson), color='seagreen',␣
 ↪linestyle='--', linewidth=2)
plt.title("Fit result: Poisson Distribution")
plt.show()
```



0.3.4 3. Lognormal Distribution

The PDF of the lognormal distribution is:

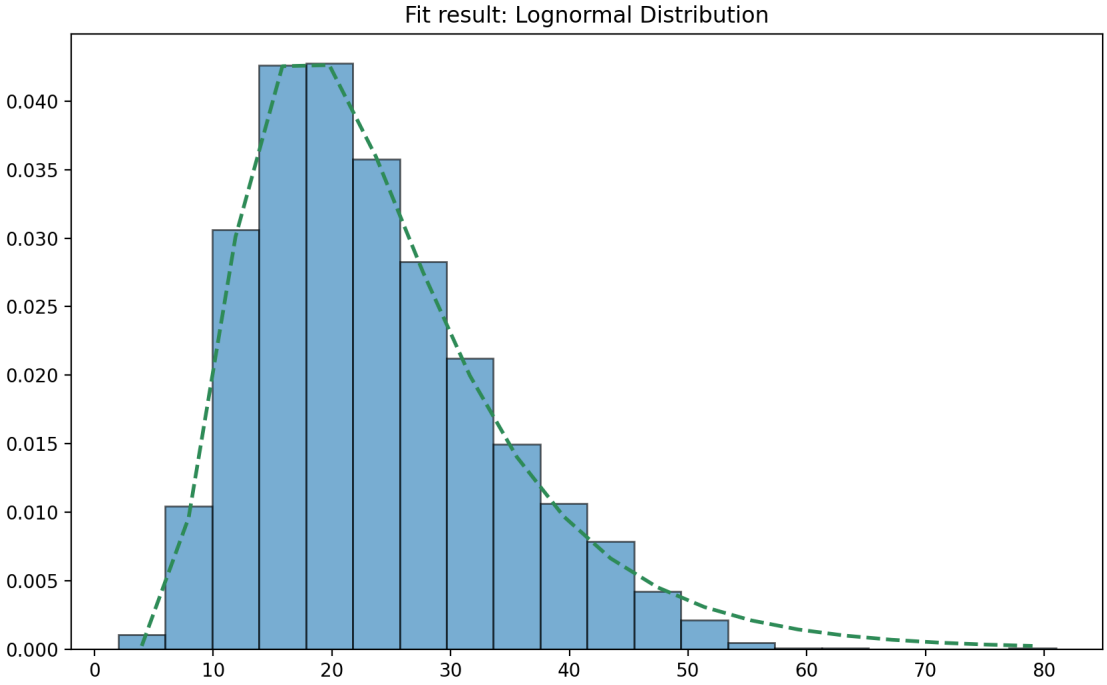
$$f(x|\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right)$$

where: - $ $ and $ $ are the mean and standard deviation of the variable's natural logarithm, - $ x $ must be greater than 0.

[18]:
```
def fit_lognormal(x, mu, sigma):
    return lognorm.pdf(x, sigma, scale=np.exp(mu))

params_lognorm, _ = curve_fit(fit_lognormal, x_data, values, p0=[np.mean(np.
↪log(data)), np.std(np.log(data))])

plt.figure(figsize=(10, 6))
plt.hist(data, bins=20, density=True, alpha=0.6, ec='black')
plt.plot(x_data, fit_lognormal(x_data, *params_lognorm), color='seagreen',␣
↪linestyle='--', linewidth=2)
plt.title("Fit result: Lognormal Distribution")
plt.show()
```



0.3.5 4. Weibull Distribution

The PDF of the Weibull distribution is:

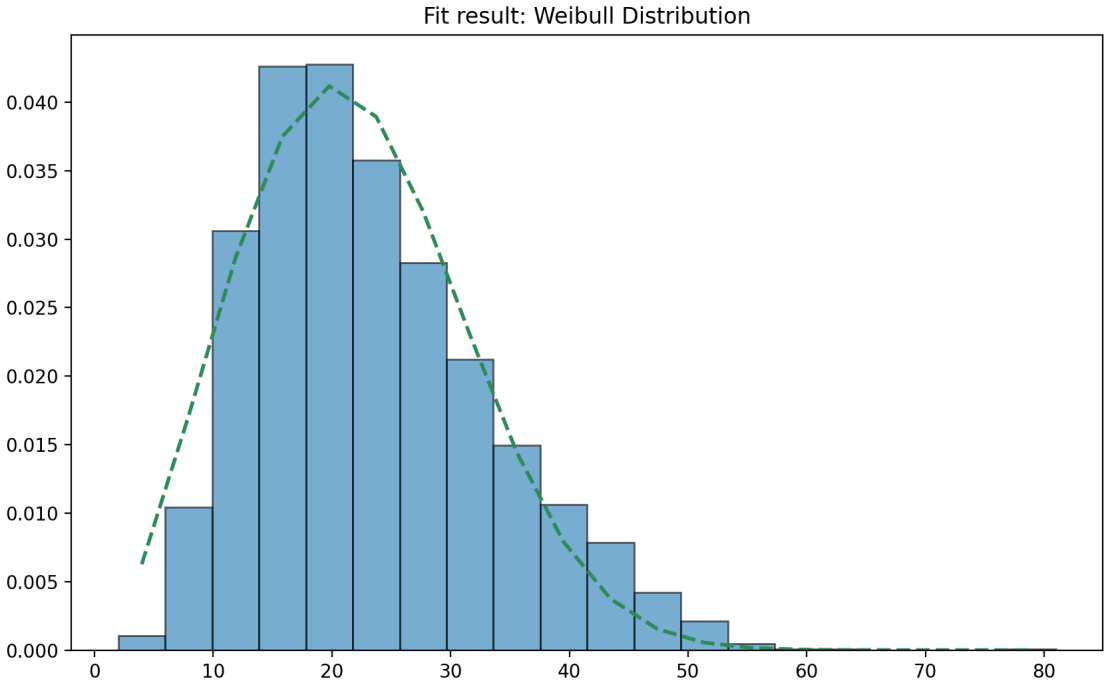
$$f(x|k,\lambda) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k}$$

where: - $ k $ is the shape parameter, - $ $ is the scale parameter, - $ x $ is the variable, typically $ x 0 $.

[19]:
```
def fit_weibull(x, k, lam):
    return weibull_min.pdf(x, k, scale=lam)
```

```
params_weibull, _ = curve_fit(fit_weibull, x_data, values, p0=[1.5, np.
↪mean(data)])

plt.figure(figsize=(10, 6))
plt.hist(data, bins=20, density=True, alpha=0.6, ec='black')
plt.plot(x_data, fit_weibull(x_data, *params_weibull), color='seagreen',␣
↪linestyle='--', linewidth=2)
plt.title("Fit result: Weibull Distribution")
plt.show()
```



0.3.6 5. Lévy Distribution

The PDF of the Lévy distribution is:

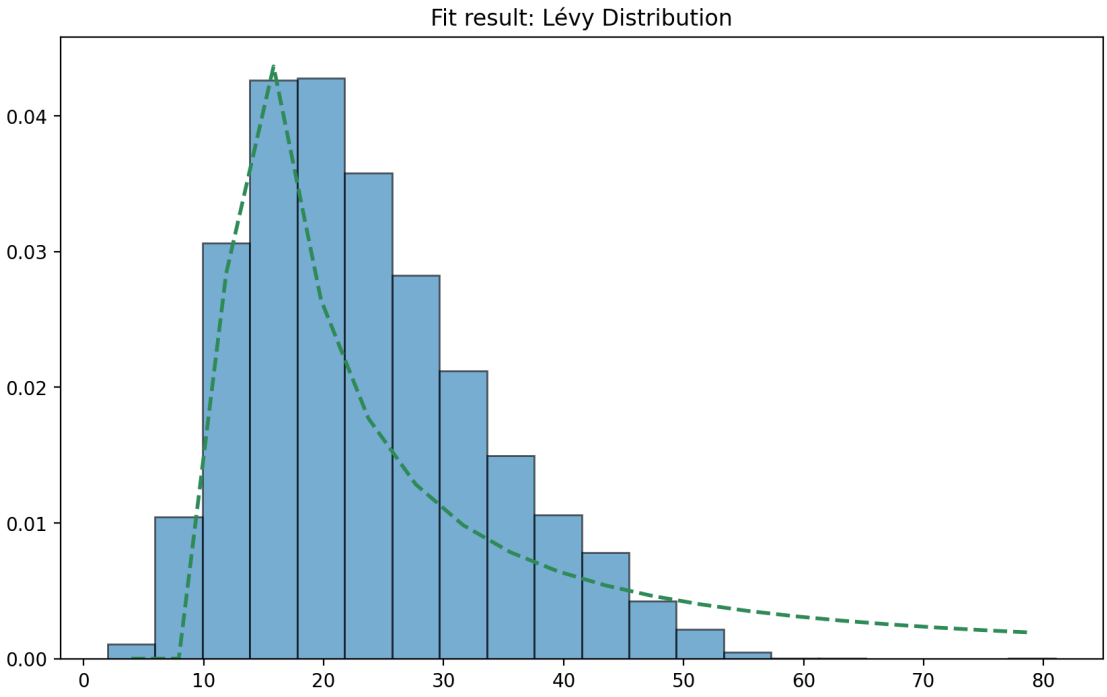
$$f(x|\mu, c) = \sqrt{\frac{c}{2\pi}} \frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{3/2}}$$

where: - $ $ is the location parameter, - $ c $ is the scale parameter, - $ x > $.

[20]:
```
def fit_levy(x, mu, c):
    return levy.pdf(x, loc=mu, scale=c)

params_levy, _ = curve_fit(fit_levy, x_data, values, p0=[np.min(data), 3])
```

```
plt.figure(figsize=(10, 6))
plt.hist(data, bins=20, density=True, alpha=0.6, ec='black')
plt.plot(x_data, fit_levy(x_data, *params_levy), color='seagreen',␣
 ↪linestyle='--', linewidth=2)
plt.title("Fit result: Lévy Distribution")
plt.show()
```



0.3.7 MSE Calculations

Below is the calculation of the MSE for each respective distribution fit.

[21]:
```
# Import necessary library for calculating MSE
from sklearn.metrics import mean_squared_error

# Evaluate each PDF at the bin centers and compute MSE
normal_pdf_values = fit_normal(x_data, *params_norm)
poisson_pdf_values = fit_poisson(x_data, *params_poisson)
lognormal_pdf_values = fit_lognormal(x_data, *params_lognorm)
weibull_pdf_values = fit_weibull(x_data, *params_weibull)
levy_pdf_values = fit_levy(x_data, *params_levy)
```

[22]:
```
# Calculate MSE for each fitted distribution
mse_normal = mean_squared_error(values, normal_pdf_values)
```

```
mse_poisson = mean_squared_error(values, poisson_pdf_values)
mse_lognormal = mean_squared_error(values, lognormal_pdf_values)
mse_weibull = mean_squared_error(values, weibull_pdf_values)
mse_levy = mean_squared_error(values, levy_pdf_values)

# Printing MSE for each distribution
print(f"MSE - Normal Distribution: {mse_normal}")
print(f"MSE - Poisson Distribution: {mse_poisson}")
print(f"MSE - Lognormal Distribution: {mse_lognormal}")
print(f"MSE - Weibull Distribution: {mse_weibull}")
print(f"MSE - Lévy Distribution: {mse_levy}")
```

```
MSE - Normal Distribution: 1.2791483349429298e-05
MSE - Poisson Distribution: 0.00020105453762331196
MSE - Lognormal Distribution: 7.09811725025554e-07
MSE - Weibull Distribution: 8.317194676894815e-06
MSE - Lévy Distribution: 6.002745901618657e-05
```

[23]:
```
params_lognorm
```

[23]:
```
array([3.0878366 , 0.46146662])
```

The LogNormal has the best fit with the lowest MSE at 7.09e-07 with parameters μ = 3.0878366 and σ = $0.46146662 $