toydat

October 26, 2024

[15]:

```
! pip install openpyxl
import pandas as pd
import numpy as np
df = pd.read_excel("OPERATIONAL AVAIL.xlsx")
```

```
Collecting openpyxl
  Downloading openpyxl-3.1.5-py2.py3-none-any.whl.metadata (2.5 kB)
Collecting et-xmlfile (from openpyxl)
  Downloading et_xmlfile-1.1.0-py3-none-any.whl.metadata (1.8 kB)
Downloading openpyxl-3.1.5-py2.py3-none-any.whl (250 kB)
Downloading et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-1.1.0 openpyxl-3.1.5
```

[16]:

```
df
```

[16]:

```
     RowNo Application ID_x   Possession Work Type DS               BidKPI  \
0      422             ID62         Sigs Points Mntce     12 Weeks or More
1      475             ID62         Sigs Points Mntce     12 Weeks or More
2     2800             ID62         Sigs Points Mntce     12 Weeks or More
3     2927             ID62         Sigs Points Mntce     12 Weeks or More
4     3416           ID1483         Sigs Points Mntce   Inside 4 Weeks Out
..     …              …                       …                  …
109   1231            ID499  OverHead Work by Yandina   Inside 8 Weeks Out
110   3015           ID1224   Replacement of Sleepers  Inside 10 Weeks Out
111   3468           ID1488       Turnout Replacement       Inside 2 Weeks
112   3826           ID1573                Contractor       Inside 2 Weeks
113   4224           ID1573                Contractor       Inside 2 Weeks

     Planned? CalenderDate_x  Possession_Start_x    Possession_End_x  \
0     Planned     2021-01-02 2021-01-02 12:04:00 2021-01-02 16:50:00
1     Planned     2021-01-02 2021-01-02 12:04:00 2021-01-02 16:50:00
2     Planned     2021-01-02 2021-01-02 12:04:00 2021-01-02 16:50:00
3     Planned     2021-01-02 2021-01-02 12:04:00 2021-01-02 16:50:00
4    Reactive     2021-01-02 2021-01-02 08:30:00 2021-01-02 12:18:00
..        …            …                 …                 …
109  Reactive     2021-01-31 2021-01-31 07:48:00 2021-01-31 11:30:00
110  Reactive     2021-01-31 2021-01-30 06:35:00 2021-02-01 17:11:00
```

```
111  Reactive     2021-01-31 2021-01-31 08:40:00 2021-01-31 18:30:00
112  Reactive     2021-01-31 2021-01-28 12:22:00 2021-02-01 00:00:00
113  Reactive     2021-01-31 2021-01-28 12:22:00 2021-02-01 00:00:00

    Corridor NM_x System NM_x  …  MinutesPossessed  SystemLength_x  \
0             CR4        SYS2  …               286             300
1             CR2        SYS2  …               286             300
2            CR26        SYS6  …               286             800
3            CR29        SYS6  …               286             800
4            CR35        SYS7  …               228            1000
..            …       …  …               …             …
109           CR8        SYS3  …               222             300
110          CR24        SYS6  …              1440             800
111          CR43        SYS8  …               590            1000
112          CR50        SYS8  …              1440            1000
113          CR39        SYS8  …              1440            1000

     MinutesKmPossessed  Downtime_corr  Uptime_corr  avail_daily  Unnamed: 17  \
0              1473.186       1473.186      7417.44     0.801389          NaN
1              3431.142       3431.142     17275.68     0.801389          NaN
2              4864.860       4864.860     24494.40     0.801389          NaN
3              7263.828       7263.828     36573.12     0.801389          NaN
4             14791.044      14791.044     93417.12     0.841667          NaN
..                  …            …          …          …          …
109            4662.666       4662.666     30244.32     0.845833          NaN
110           71971.200      71971.200     71971.20     0.000000          NaN
111            1538.130       1538.130      3754.08     0.590278          NaN
112           14873.760      14873.760     14873.76     0.000000          NaN
113           16441.920      16441.920     16441.92     0.000000          NaN

       MONTH  YEAR  days
0    January  2021    31
1    January  2021    31
2    January  2021    31
3    January  2021    31
4    January  2021    31
..       …   …   …
109  January  2021    31
110  January  2021    31
111  January  2021    31
112  January  2021    31
113  January  2021    31

[114 rows x 21 columns]
```

[17]:
```
df.columns
```

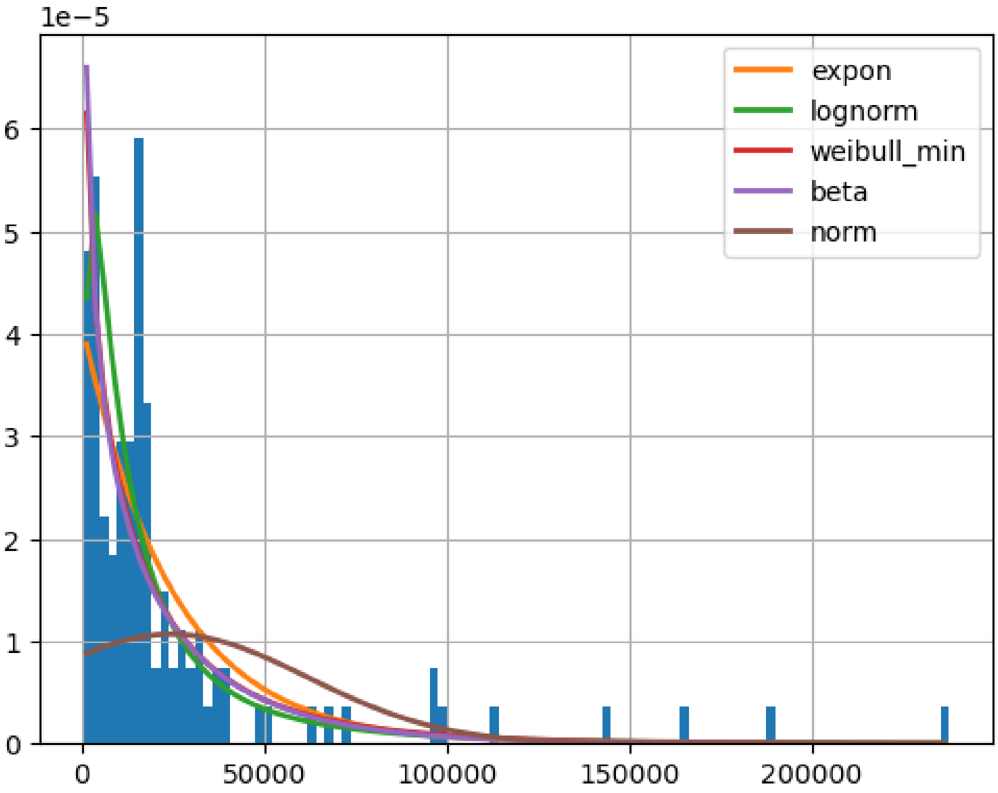
[17]:
```
Index(['RowNo', 'Application ID_x', 'Possession Work Type DS', 'BidKPI',
       'Planned?', 'CalenderDate_x', 'Possession_Start_x', 'Possession_End_x',
       'Corridor NM_x', 'System NM_x', 'CorridorLengthPossessed(km)',
       'MinutesPossessed', 'SystemLength_x', 'MinutesKmPossessed',
       'Downtime_corr', 'Uptime_corr', 'avail_daily', 'Unnamed: 17', 'MONTH',
       'YEAR', 'days'],
      dtype='object')
```

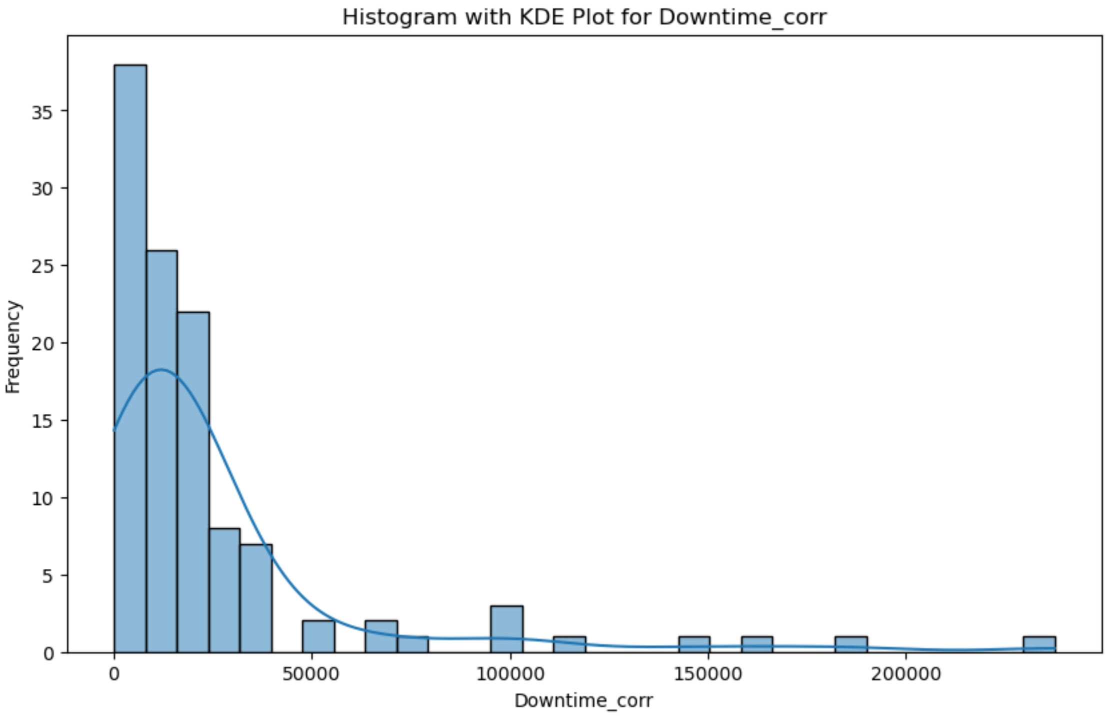
1 FINDING THE BEST FIT FOR THE COLUMNS IN QR DATASET

[19]:
```
!pip install fitter
from fitter import Fitter
import matplotlib.pyplot as plt
import seaborn as sns

# Define each dataset from the DataFrame
downtime_data = df['Downtime_corr']
uptime_data = df['Uptime_corr']
avail_daily_data = df['avail_daily']

# List of datasets for fitting
datasets = {'Downtime_corr': downtime_data, 'Uptime_corr': uptime_data,␣
 ↪'avail_daily': avail_daily_data}
results = {}

for name, data in datasets.items():
    # Fit common distributions to each dataset individually
    f = Fitter(data.dropna(), distributions=['norm', 'lognorm', 'expon',␣
 ↪'gamma', 'beta','uniform', 'weibull_min'])  # Drop NaN values if any
    f.fit()

    # Store and print the best-fitting distribution for each dataset
    best_distribution = f.get_best()
    results[name] = best_distribution
    print(f"Best fitting distribution for {name}:", best_distribution)

    # Summary of distribution fitting
    f.summary()

    # Plot histogram with KDE
    plt.figure(figsize=(10, 6))
    sns.histplot(data, bins=30, kde=True)
    plt.title(f'Histogram with KDE Plot for {name}')
    plt.xlabel(name)
    plt.ylabel("Frequency")
    plt.show()

# Optional: View all best-fitting distributions at once
print("Best-fitting distributions summary:")
print(results)
```

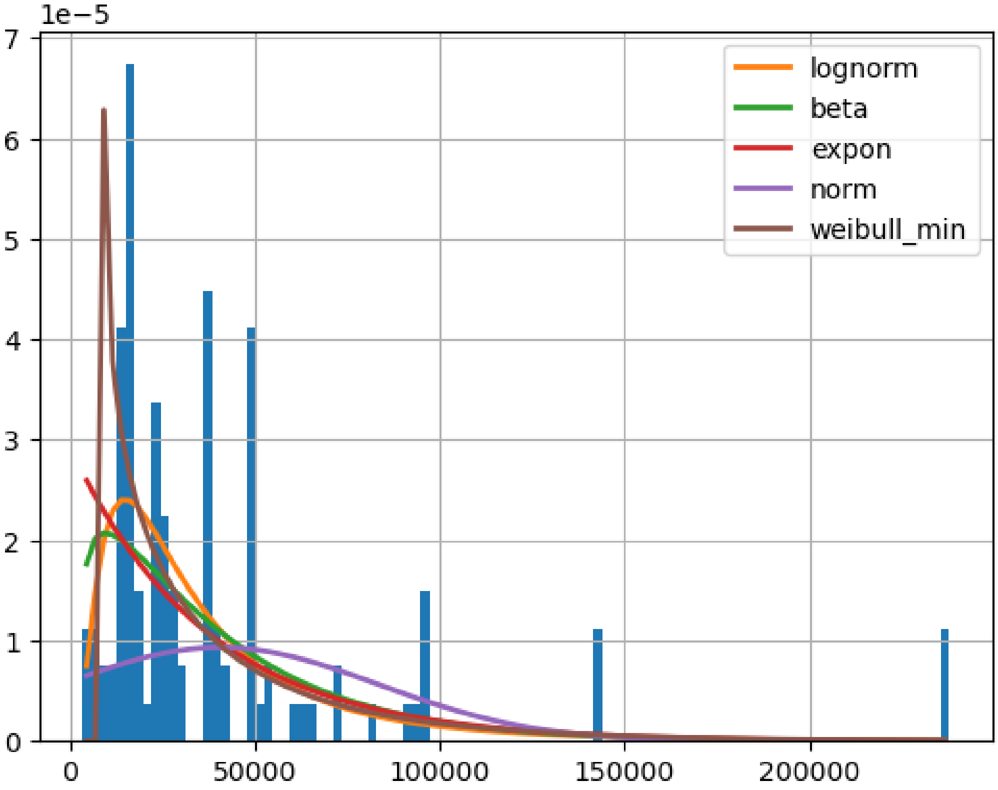
```
2024-10-25 14:59:56.557 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted lognorm distribution with error=0.0)
2024-10-25 14:59:56.573 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted norm distribution with error=0.0)
2024-10-25 14:59:56.624 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted expon distribution with error=0.0)
2024-10-25 14:59:56.625 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted uniform distribution with error=0.0)
2024-10-25 14:59:56.630 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted beta distribution with error=0.0)
2024-10-25 14:59:56.649 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted weibull_min distribution with error=0.0)
2024-10-25 14:59:56.663 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted gamma distribution with error=0.0)
```

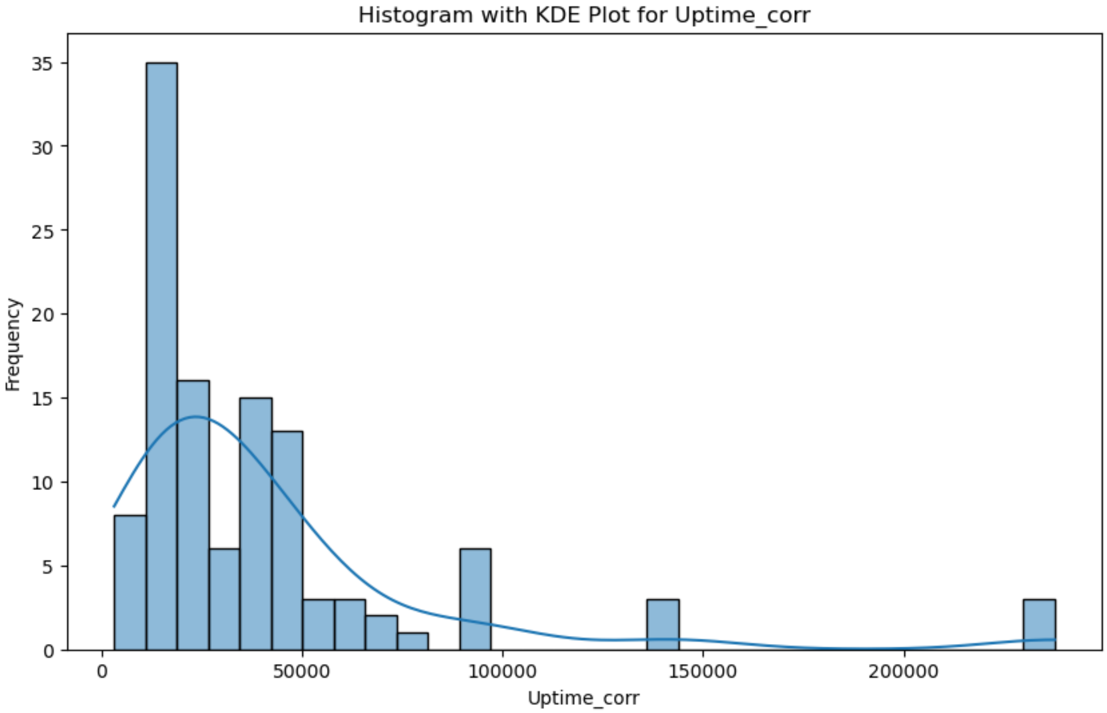
```
Best fitting distribution for Downtime_corr: {'expon': {'loc': 75.54, 'scale':
24443.78942105263}}
```





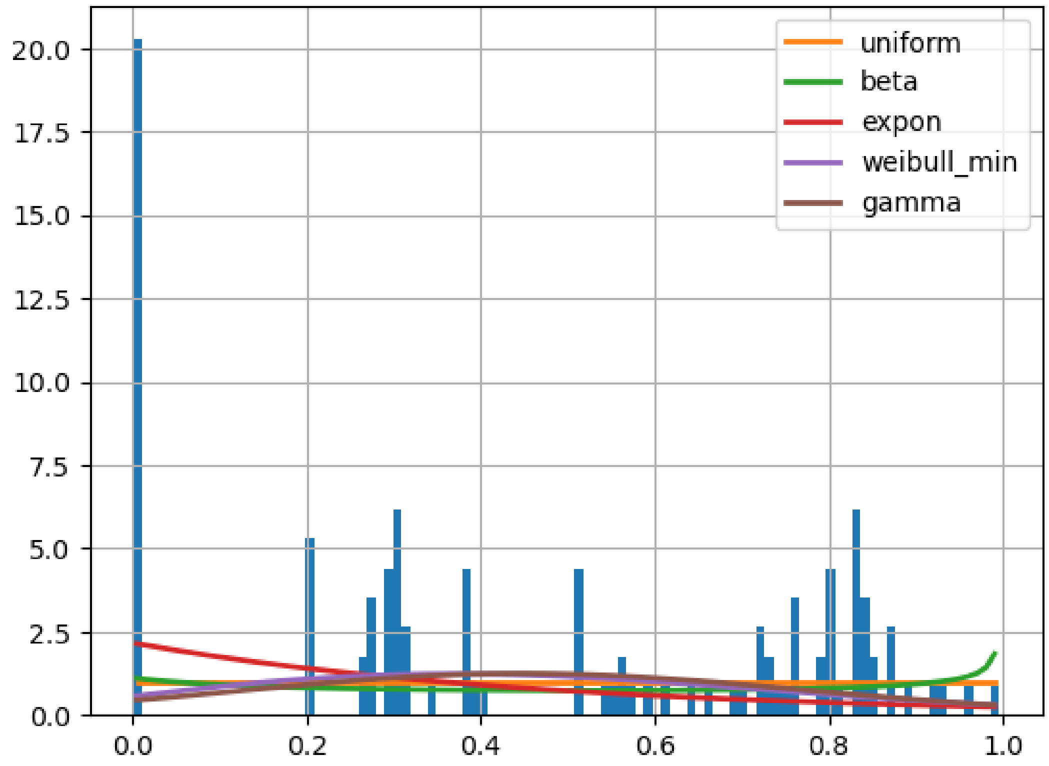
```
Best fitting distribution for Uptime_corr: {'lognorm': {'s': 0.8138399890526843,
'loc': 183.16594517411, 'scale': 28321.002881801716}}

2024-10-25 14:59:56.948 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted norm distribution with error=0.0)
2024-10-25 14:59:56.950 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted expon distribution with error=0.0)
2024-10-25 14:59:56.950 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted lognorm distribution with error=0.0)
2024-10-25 14:59:56.953 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted uniform distribution with error=0.0)
2024-10-25 14:59:56.967 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted gamma distribution with error=0.0)
2024-10-25 14:59:56.975 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted weibull_min distribution with error=0.0)
2024-10-25 14:59:56.980 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted beta distribution with error=0.0)
```

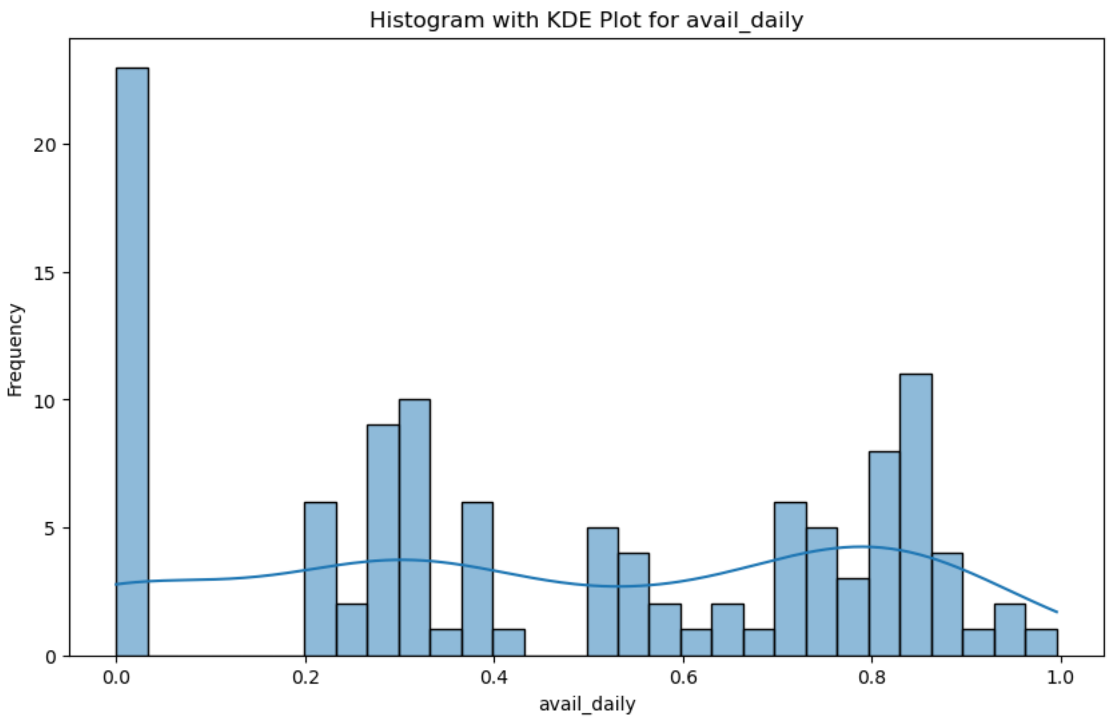

1e−5
7
6
5
4
3
2
1
0
0
50000
100000
150000
200000
lognorm
beta
expon
norm
weibull_min


Histogram with KDE Plot for Uptime_corr
Frequency
35
30
25
20
15
10
5
0
0
50000
100000
150000
200000
Uptime_corr

```
Best fitting distribution for avail_daily: {'uniform': {'loc':
-1.8479805219745941e-16, 'scale': 0.9958333333333335}}
```

```
2024-10-25 14:59:57.191 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted expon distribution with error=587.5393)
2024-10-25 14:59:57.192 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted norm distribution with error=602.962196)
2024-10-25 14:59:57.195 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted uniform distribution with error=580.41924)
2024-10-25 14:59:57.209 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted gamma distribution with error=602.885718)
2024-10-25 14:59:57.211 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted beta distribution with error=580.582991)
2024-10-25 14:59:57.211 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted weibull_min distribution with error=601.207902)
2024-10-25 14:59:57.231 | INFO     | fitter.fitter:_fit_single_distribution:333
- Fitted lognorm distribution with error=603.165602)
```





```
Best-fitting distributions summary:
{'Downtime_corr': {'expon': {'loc': 75.54, 'scale': 24443.78942105263}},
'Uptime_corr': {'lognorm': {'s': 0.8138399890526843, 'loc': 183.16594517411,
'scale': 28321.002881801716}}, 'avail_daily': {'uniform': {'loc':
-1.8479805219745941e-16, 'scale': 0.9958333333333335}}}
```

[20]:
```
x=df['Uptime_corr'].unique()
y=df['Downtime_corr'].unique()
```

[21]:
```
print ( max(x), min(x))

print ( max(y), min(y))
```

```
237650.4 3221.28
237650.4 75.54
```

[22]:
```
# df['DT']= np.random.randint(1,101, size=len(df))
# df['UT'] = df['DT'] + np.random.randint(1, 101, size=len(df))

# df['V_red']=np.random.rand( len(df))

# MDT= sum(df['DT'])/len(df['DT'])
# MDT

# OH=15
# V=70
# D=10

# N_pd= MDT/OH
# int(N_pd)

# TSR= D*( (1/V)-(1/df['V_red']))

# DT = (N_pd*TSR) + df['DT']

# Ao= (df['UT'] - DT)/df['UT']
```

2 TOY DATA FOR UT-DT/UT

[24]:

```
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Initialize lists to store test case data
test_cases = []

for _ in range(100):
    UT = random.lognormvariate(6.2, 0.2)   # Total Uptime in minutes (mean ~500)
    MDT = random.uniform(5, 60)            # Mean Down Time in minutes
    OH = random.uniform(3, 15)             # Headway in minutes
    TSR = random.uniform(1, 10)            # Speed Restriction Delay in minutes
    dt = random.expovariate(1 / 30)


    # # Generate random values within specified ranges
    # UT = random.weibullvariate( 1000)   # Total Uptime in minutes
    # MDT = random.weibullvariate( 60)      # Mean Down Time in minutes
    # OH = random.weibullvariate( 15)       # Headway in minutes
    # TSR = random.weibullvariate( 10)      # Speed Restriction Delay in minutes
    # dt = random.weibullvariate(30)       # Other Downtime in minutes

    # Calculate N_pd
    N_pd = int(MDT / OH)

    # Calculate DT
    DT = ((N_pd) * TSR) + dt

    # Calculate Availability
    Availability = (UT - DT) / UT
    # Store the test case data
    test_cases.append({
        'UT (min)': UT,
        'MDT (min)': MDT,
        'OH (min)': OH,
        'TSR (min)': TSR,
        'dt (min)': dt,
        'N_pd': N_pd,
        'DT (min)': DT,
        'Availability': Availability
    })

# Convert the list of test cases into a DataFrame
df = pd.DataFrame(test_cases)

# Perform checks
if df['Availability'].between(0, 1).all():
    print("All availability values are within the expected range (0 to 1).")
else:
    print("Some availability values are outside the expected range!")

# Display first few test cases
print(df.head())
df.to_csv("toy_avail")
```

```
All availability values are within the expected range (0 to 1).
     UT (min)  MDT (min)   OH (min)  TSR (min)   dt (min)  N_pd   DT (min)  \
0  437.174602  54.846200  14.406013   7.477627  54.533424     3  76.966306
1  631.174891  14.820423  14.710671   6.741566  63.730014     1  70.471580
2  412.047373  56.440115  12.773297   7.978458  40.800375     4  72.714209
3  299.974647  39.551828  14.394654   1.192152  53.745396     2  56.129700
4  634.971523  46.614183  10.817858   6.358134   3.290550     4  28.723086

   Availability
0      0.823946
1      0.888349
2      0.823529
3      0.812885
4      0.954765
```

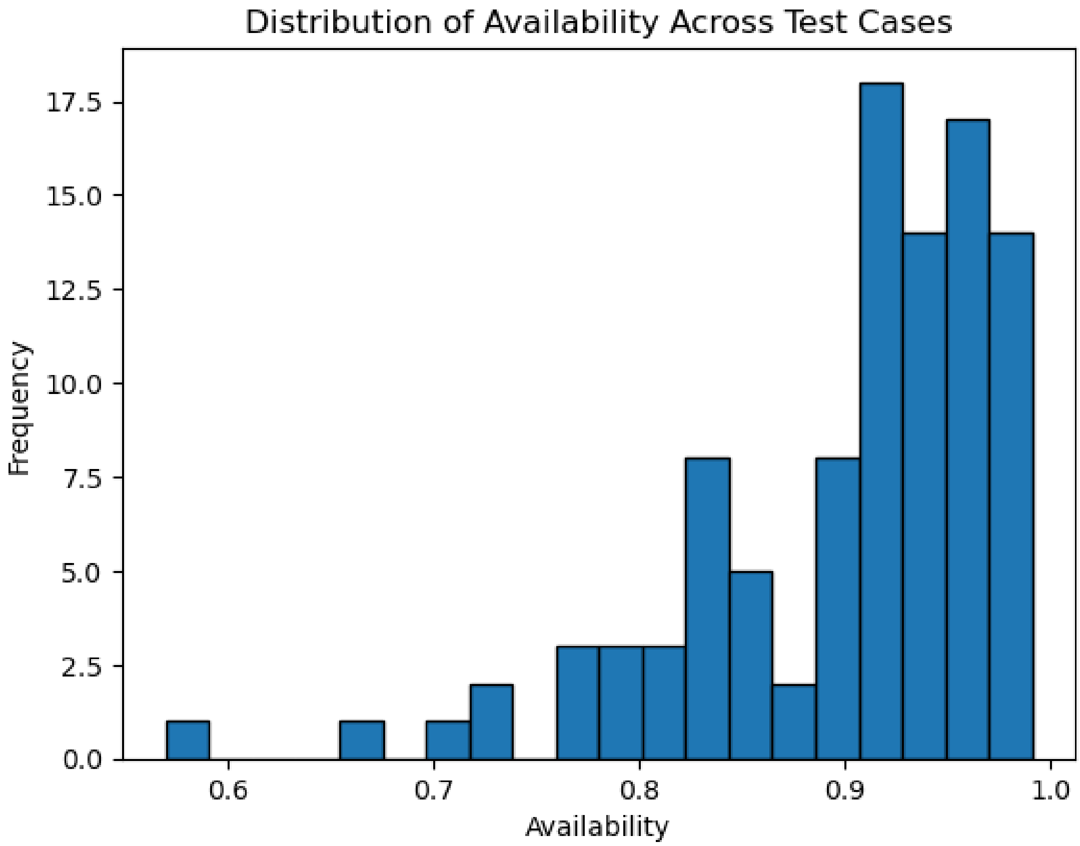
[ ]:

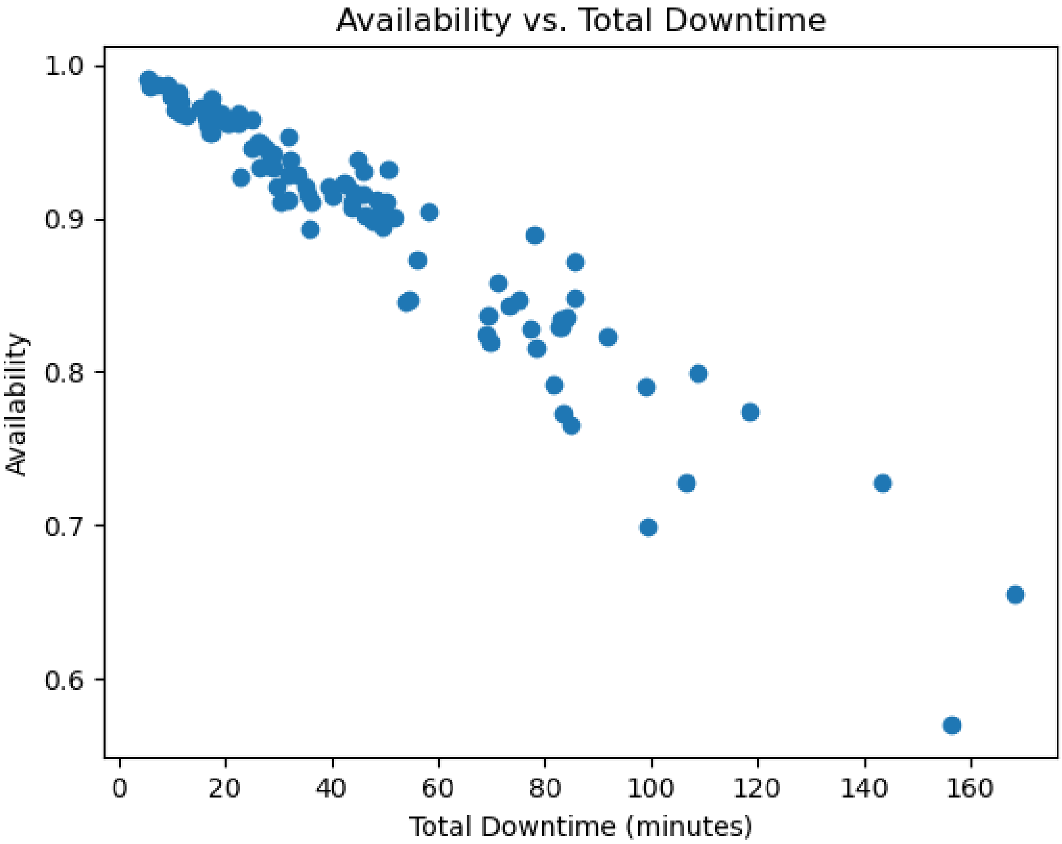
[3]:

```
# Analyze correlation
correlation = df['DT (min)'].corr(df['Availability'])
print(f"Correlation between DT and Availability: {correlation}")
```

```python
# Plot Availability Distribution
plt.hist(df['Availability'], bins=20, edgecolor='black')
plt.xlabel('Availability')
plt.ylabel('Frequency')
plt.title('Distribution of Availability Across Test Cases')
plt.show()

# Plot Availability vs. Total Downtime
plt.scatter(df['DT (min)'], df['Availability'])
plt.xlabel('Total Downtime (minutes)')
plt.ylabel('Availability')
plt.title('Availability vs. Total Downtime')
plt.show()
```

```
Correlation between DT and Availability: -0.9550949075710531
```





3 TOY DATA - UT/UT+DT & UT-DT/UT

[2]:

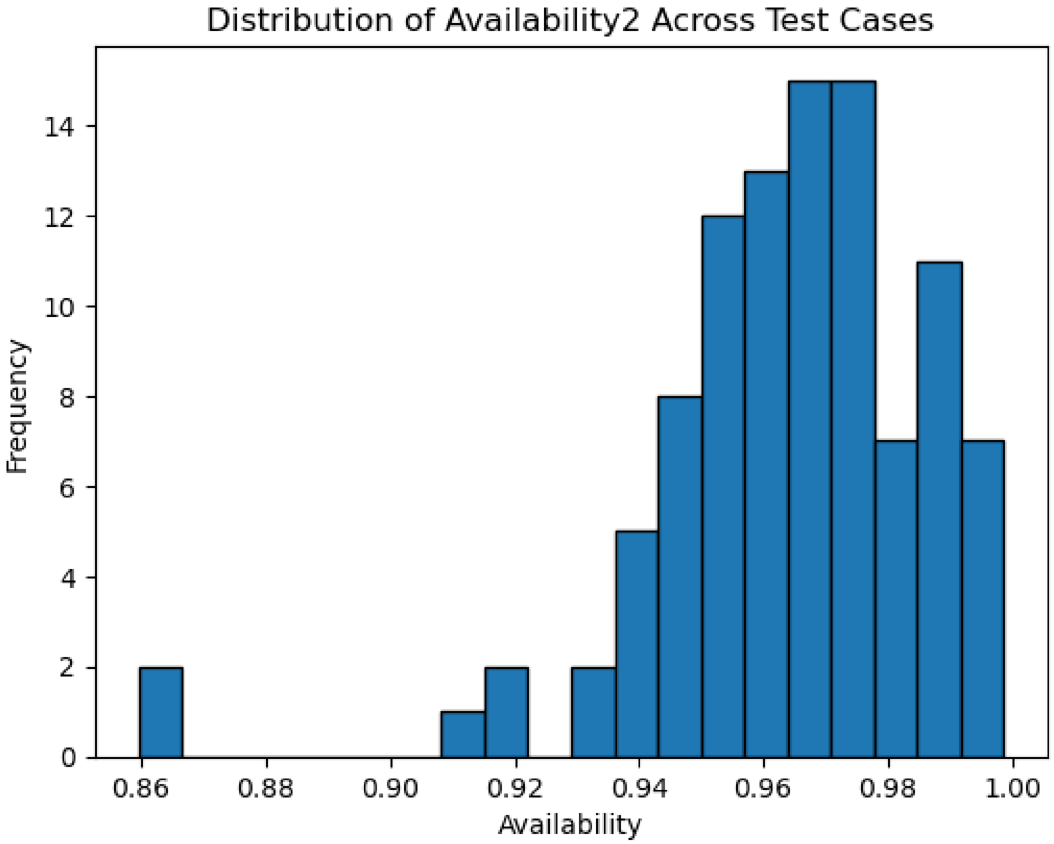
```
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Initialize lists to store test case data
test_cases = []

for _ in range(100):
    # Generate random values within specified ranges
    UT = random.uniform(500, 1000)    # Total Uptime in minutes
    MDT = random.uniform(5, 60)       # Mean Down Time in minutes
    OH = random.uniform(3, 15)        # Headway in minutes
    dt = random.uniform(0, 30)        # Other Downtime in minutes

    # Generate random values for D, V1, and V_reduced
    D = random.uniform(1, 10)           # Distance in kilometers
    V1 = random.uniform(60, 120)       # Normal speed in km/h
    V_reduced = random.uniform(20, V1 - 10)  # Reduced speed in km/h (ensure V_reduced < V1)

    # Calculate TSR using the formula and convert to minutes
    TSR = D * (1 / V_reduced - 1 / V1) * 60  # Time delay in minutes

    # Calculate N_pd
    N_pd = int(MDT / OH)

    # Ensure N_pd is at least 1
    N_pd = max(1, N_pd)

    # Calculate DT
    DT = (N_pd * TSR) + dt

    # Ensure DT does not exceed UT
    # DT = min(DT, UT)

    # Calculate Availability
    Availability = (UT - DT) / UT

    Availability2 = UT / (UT + DT)

    # Store the test case data
    test_cases.append({
        'UT (min)': UT,
        'MDT (min)': MDT,
        'OH (min)': OH,
        'D (km)': D,
        'V1 (km/h)': V1,
        'V_reduced (km/h)': V_reduced,
        'TSR (min)': TSR,
        'dt (min)': dt,
        'N_pd': N_pd,
        'DT (min)': DT,
        'Availability': Availability,
        'Availability2': Availability2
    })

# Convert the list of test cases into a DataFrame
df = pd.DataFrame(test_cases)

# Perform checks for 'Availability'
availability_check = df['Availability'].between(0, 1).all()

# Perform checks for 'Availability2'
```

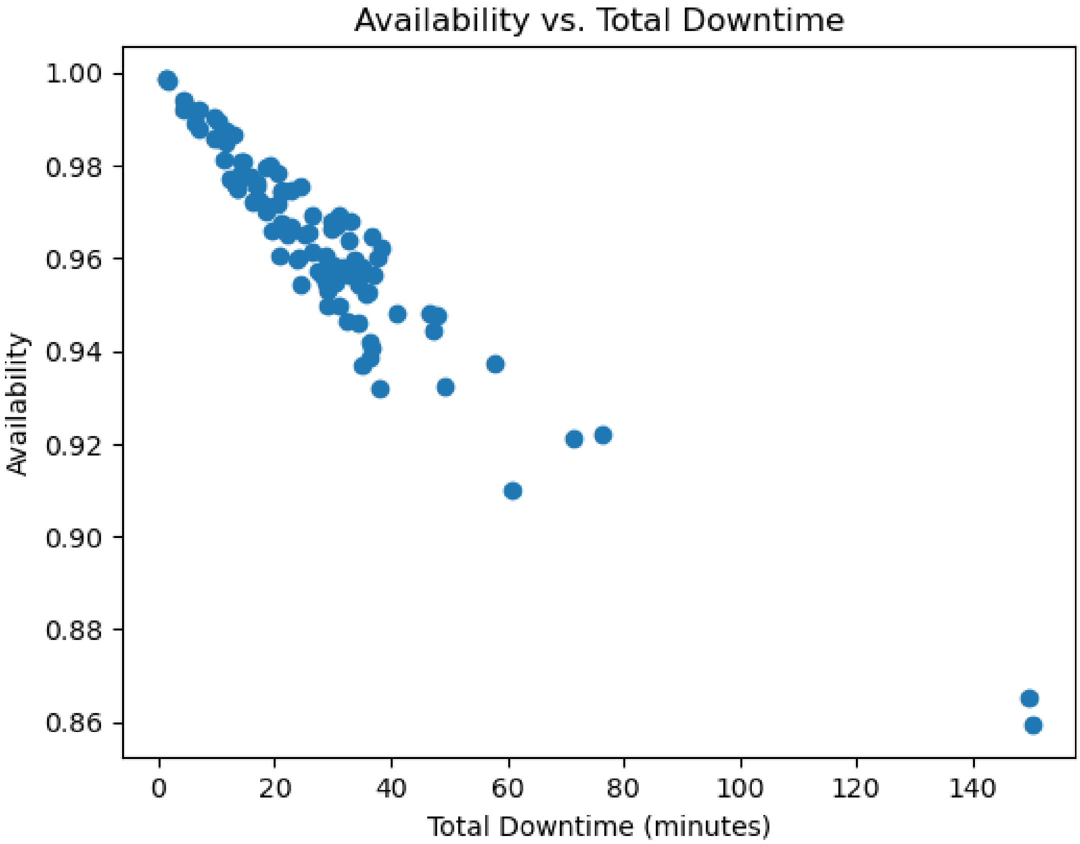
```
availability2_check = df['Availability2'].between(0, 1).all()

# Display first few test cases and availability checks
df_head = df.head()

availability_check, availability2_check, df_head
df.to_csv('output.csv', index=False)
```

[5]:

```
correlation = df['DT (min)'].corr(df['Availability2'])
print(f"Correlation between DT and Availability: {correlation}")

# Plot Availability Distribution
plt.hist(df['Availability2'], bins=20, edgecolor='black')
plt.xlabel('Availability')
plt.ylabel('Frequency')
plt.title('Distribution of Availability2 Across Test Cases')
plt.show()

# Plot Availability vs. Total Downtime
plt.scatter(df['DT (min)'], df['Availability2'])
plt.xlabel('Total Downtime (minutes)')
plt.ylabel('Availability')
plt.title('Availability vs. Total Downtime')
plt.show()
```

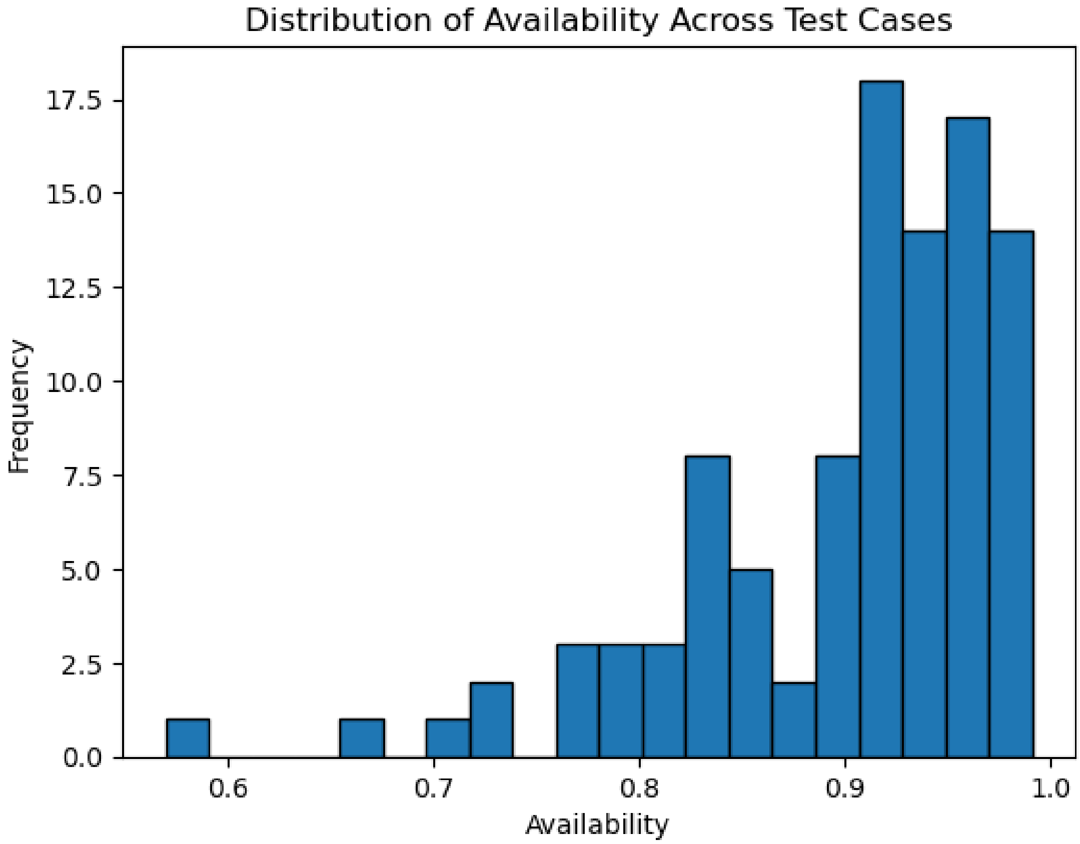
```
Correlation between DT and Availability: -0.9401271409721554
```

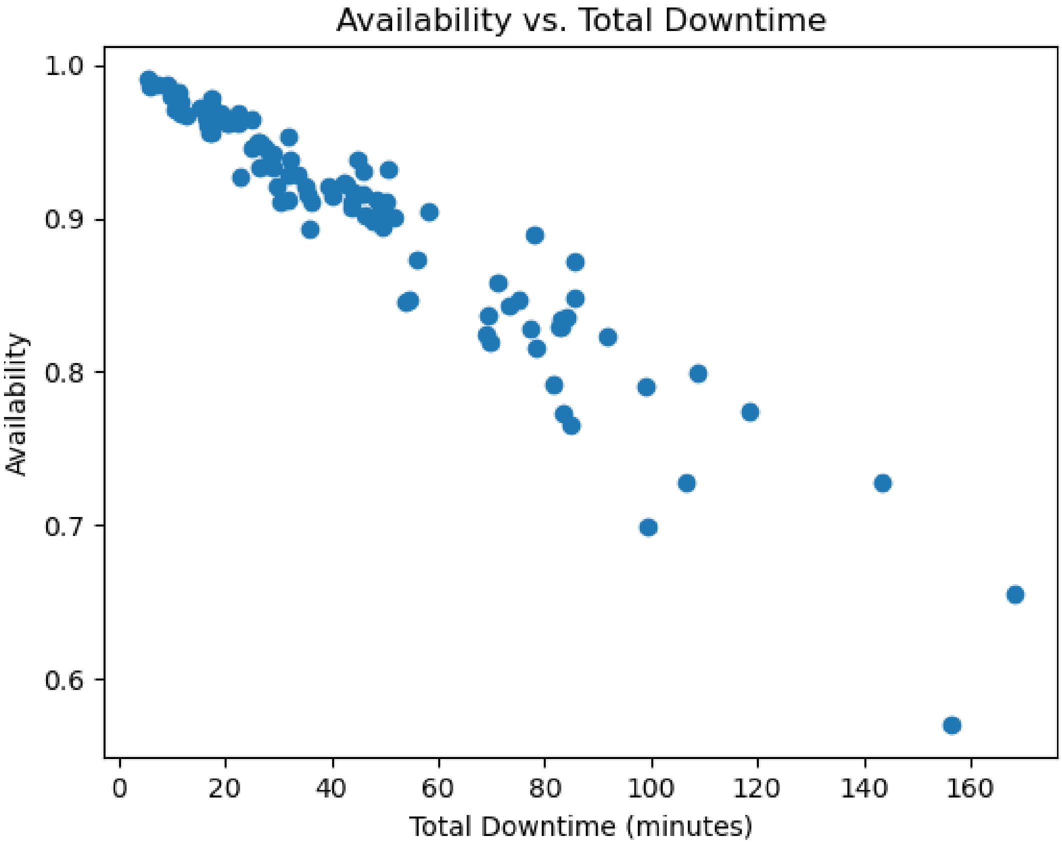

Distribution of Availability2 Across Test Cases
Frequency
0
2
4
6
8
10
12
14
0.86
0.88
0.90
0.92
0.94
0.96
0.98
1.00
Availability



```
[4]: correlation = df['DT (min)'].corr(df['Availability'])
     print(f"Correlation between DT and Availability: {correlation}")

     # Plot Availability Distribution
     plt.hist(df['Availability'], bins=20, edgecolor='black')
     plt.xlabel('Availability')
     plt.ylabel('Frequency')
     plt.title('Distribution of Availability Across Test Cases')
     plt.show()

     # Plot Availability vs. Total Downtime
     plt.scatter(df['DT (min)'], df['Availability'])
     plt.xlabel('Total Downtime (minutes)')
     plt.ylabel('Availability')
     plt.title('Availability vs. Total Downtime')
     plt.show()
```

```
Correlation between DT and Availability: -0.9550949075710531
```


Distribution of Availability Across Test Cases
Frequency
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.6
0.7
0.8
0.9
1.0
Availability



```
[3]: # Availability2 = UT / (UT + DT)
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```