STATISTICS & PROBABILITY

Topic: Descriptive Statistics and Inferential Statistics

source code https://github.com/nitishbuzzpro/Statistics-and-Hypothesis-Tetsing--Titanic-Dataset---Data-Science.git (https://github.com/nitishbuzzpro/Statistics-and-Hypothesis-Tetsing--Titanic-Dataset---Data-Science.git)

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

Task 1

In [2]:

```
# Tasks:
# Load the Titanic dataset using Python and perform
# the following descriptive statistics:
# Calculate the mean, median, and mode of the age
# of passengers.
# Calculate the range, variance, and standard
# deviation of the fare paid by passengers.
# Calculate the correlation coefficient between the
# age and fare columns.
# Compute the quartiles (25th, 50th, and 75th
# percentiles) of the age and fare columns.
# Generate a histogram and box plot for the age distribution of passeng
# Explain Box plot, and How Box plot could be useful in a Real-world Sc
```

In [3]:

```python
# Load the Titanic dataset using Python and perform the following descr
df = pd.read_excel('../input/train-dataset/train.xlsx')
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

In [4]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [5]:

```python
# Calculate the mean, median, and mode of the age of passengers.
print('mean:', df['Age'].mean())
print('median:',df['Age'].median())
print('mode',df['Age'].mode()[0])
```

```
mean: 29.69911764705882
median: 28.0
mode 24.0
```

In [6]:

```
# Calculate the range, variance, and standard deviation of the fare pai
print('range: ',(df['Fare'].max(),df['Fare'].min()))
print('variance: ', df['Fare'].var())
print('Std Deviation', df['Fare'].std())
```

```
range:  (512.3292, 0.0)
variance:  2469.436845743116
Std Deviation 49.6934285971809
```

In [7]:

```
# Calculate the correlation coefficient between the age and fare column
print('correlation coefficient between the age and fare columns',df['Ag
```

```
correlation coefficient between the age and fare columns 0.096066691769038
9
```

In [8]:

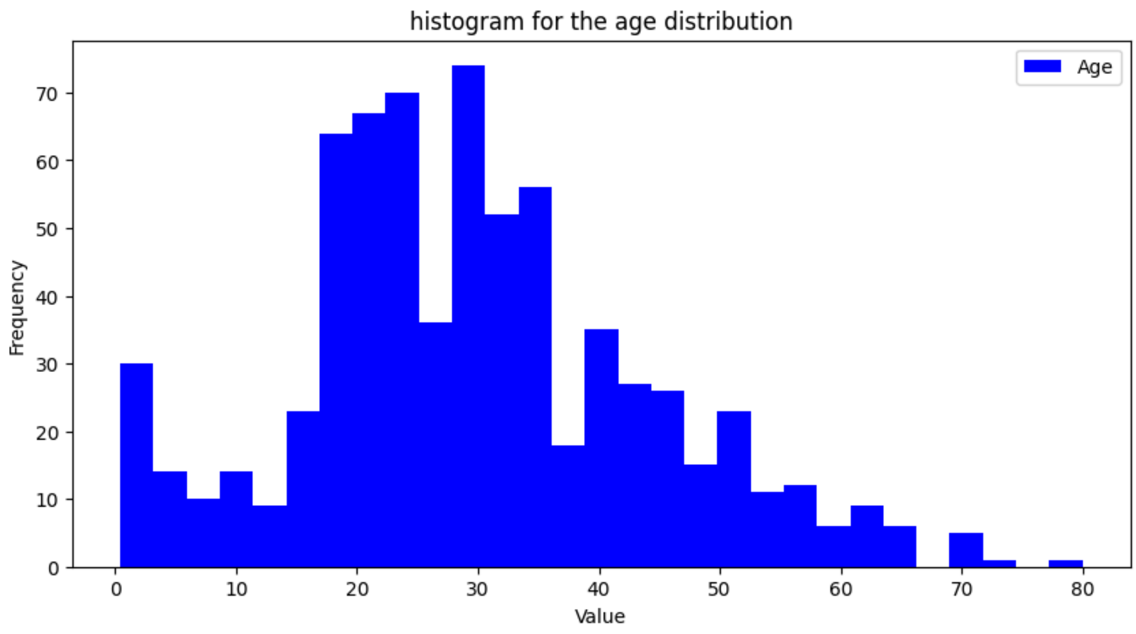
```
# Compute the quartiles (25th, 50th, and 75th percentiles) of the age a
print('quartiles (25th, 50th, and 75th percentiles) of the age: \n',df[
print('\n')
print('quartiles (25th, 50th, and 75th percentiles) of the age: \n',df[
```

```
quartiles (25th, 50th, and 75th percentiles) of the age: 
 0.25    20.125
0.50    28.000
0.75    38.000
Name: Age, dtype: float64


quartiles (25th, 50th, and 75th percentiles) of the age: 
 0.25     7.9104
0.50    14.4542
0.75    31.0000
Name: Fare, dtype: float64
```

In [9]:

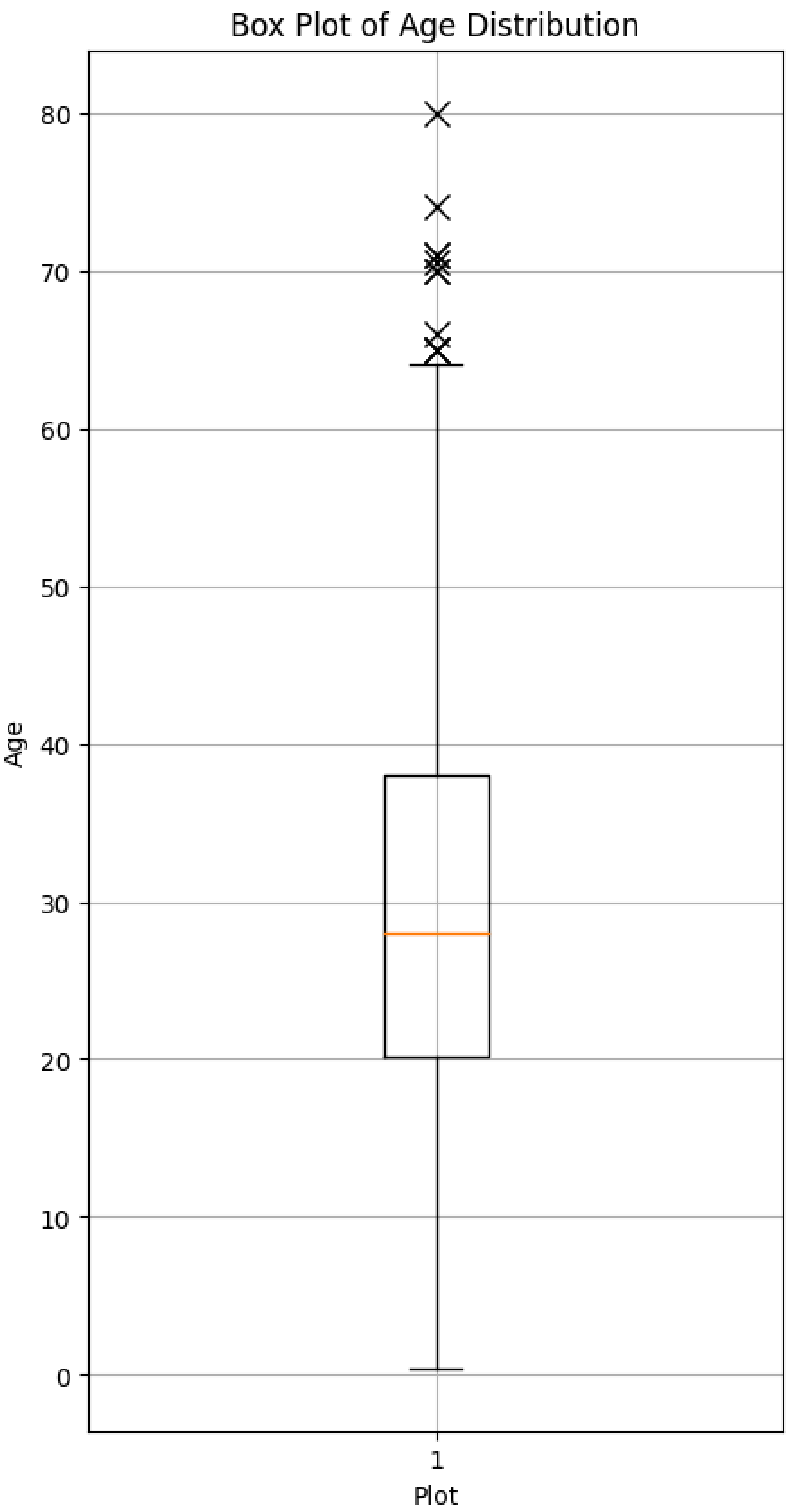
```python
# Generate a histogram and box plot for the age distribution of passeng
plt.figure(figsize=(10,5))
plt.hist(
    df['Age'],
    bins=int(np.sqrt(len(df['Age']))),
    range=None,
    density=False,
    weights=None,
    cumulative=False,
    bottom=None,
    histtype='stepfilled',
    align='mid',
    orientation='vertical',
    rwidth=None,
    log=False,
    color='blue',
    label='Age',
    stacked=False,
)
plt.title('histogram for the age distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.legend()
```

Out[9]: <matplotlib.legend.Legend at 0x7c3e5b93fd00>



In [10]:

```
# box plot for the age distribution of passengers
plt.figure(figsize=(5, 10))
plt.boxplot(df['Age'].dropna(),flierprops=dict(marker='x', color='red',
plt.title('Box Plot of Age Distribution')
plt.ylabel('Age')
plt.xlabel('Plot')
plt.grid(True)  # Add grid lines for better readability
plt.show()
```



In [11]:

```
# Explain Box plot, and How Box plot could be useful in a Real-world Sc


# Answer:- A box plot, also known as a whisker plot, is a graphical rep
# of a dataset that displays its distribution through five main summary
# statistics: minimum, first quartile (Q1), median (Q2), third quartile
# and maximum. It's a powerful tool for visualizing the spread and skew
# of the data, identifying outliers, and comparing different datasets.

# Benefits:
# Identifying Variability: By comparing the box plots of different batc

# Detecting Outliers: Outliers in the box plot can indicate defective c

# Comparing Batches: Multiple box plots allow for direct comparison of

# Ensuring Consistency: Regularly creating box plots for production bat

# Additional Applications:
# Healthcare: Monitoring patient vital signs (e.g., blood pressure) to
# Finance: Analyzing the distribution of daily stock returns to assess
# Education: Comparing student test scores across different classes or
```

In [ ]:

```

```

Task 2

In [12]:

```
# Analyze the categorical variables using the following descriptive sta

# Calculate the frequency distribution of the passenger classes (Pclass
# Calculate the percentage distribution of survival (Survived column).
# Determine the mode and percentage distribution of the passenger title
```

In [13]:

```
# Calculate the frequency distribution of the passenger classes (Pclass
print('frequency distribution of the passenger classes (Pclass column):
```

```
frequency distribution of the passenger classes (Pclass column):
 Pclass
1    216
2    184
3    491
Name: count, dtype: int64
```

In [14]:

```
# Calculate the percentage distribution of survival (Survived column).
print('the percentage distribution of survival (Survived column):\n',df
```

```
the percentage distribution of survival (Survived column):
 Survived
0    61.616162
1    38.383838
Name: proportion, dtype: float64
```

In [15]:

```
# Determine the mode and percentage distribution of the passenger title
df['title'] = df['Name'].apply(lambda name : name.split(',')[1].split('
print('the mode of the passenger titles (Name column):.\n', df['title']
print('\n')
print('the  percentage distribution of the passenger titles (Name colum
```

```
the mode of the passenger titles (Name column):.
 Mr


the  percentage distribution of the passenger titles (Name column):.
 title
Mr              58.024691
Miss            20.426487
Mrs             14.029181
Master           4.489338
Dr               0.785634
Rev              0.673401
Mlle             0.224467
Major            0.224467
Col              0.224467
the Countess     0.112233
Capt             0.112233
Ms               0.112233
Sir              0.112233
Lady             0.112233
Mme              0.112233
Don              0.112233
Jonkheer         0.112233
Name: proportion, dtype: float64
```

In [ ]:

```

```

Task 3

In [16]:

```
# Analyze the following statistical distributions:

# Determine the distribution of the passenger ages and visualize it usi
# Determine the distribution of the fares paid by passengers and visual
```

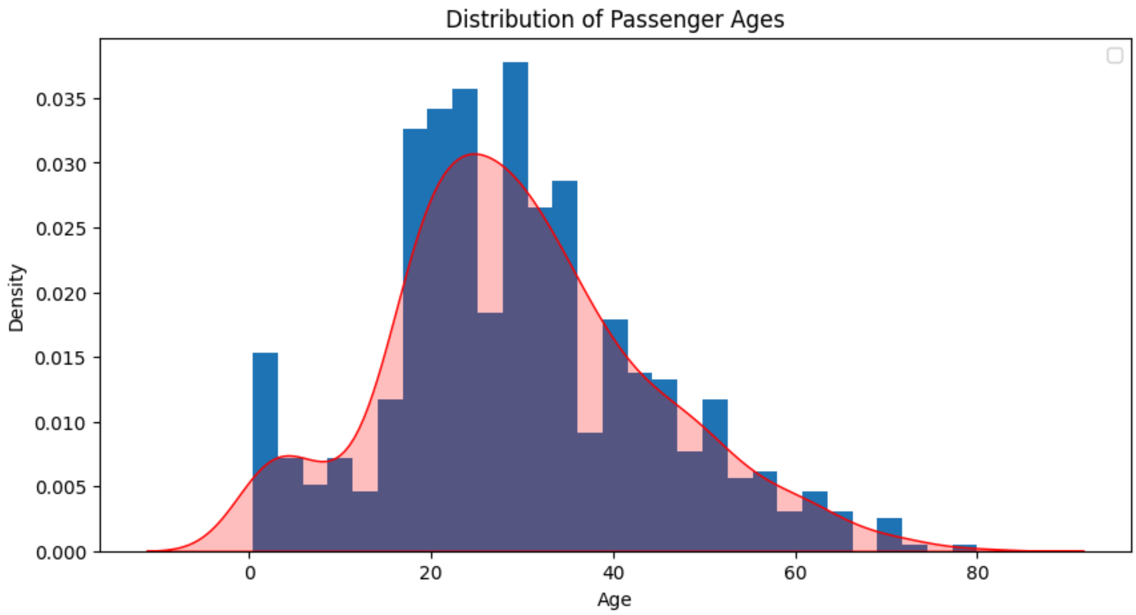
In [17]:

```python
# Determine the distribution of the passenger ages and visualize it usi
plt.figure(figsize=(10,5))
plt.hist(
    df['Age'],
    bins=int(np.sqrt(len(df['Age']))),
    range=None,
    density=True,
    weights=None,
    cumulative=False,
    bottom=None,
    histtype='bar',
    align='mid',
    orientation='vertical',
    rwidth=None,
    log=False,
    color=None,
    label=None,
    stacked=False,
)
sns.kdeplot(
    data=df['Age'],
    hue=None,
    weights=None,
    palette=None,
    hue_order=None,
    hue_norm=None,
    color='Red',
    fill=True,
    multiple='layer',
    common_norm=True,
    common_grid=True,
    cumulative=False,
    bw_method='scott',
    bw_adjust=1,
    warn_singular=True,
    log_scale=None,
    levels=10,
    thresh=0.05,
    gridsize=200,
    cut=3,
    clip=None,
    legend=True,
    cbar=False,
    cbar_ax=None,
    cbar_kws=None,
    ax=None,
)

plt.xlabel('Age')
plt.ylabel('Density')
plt.title('Distribution of Passenger Ages')
plt.legend()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWa
rning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[17]:

```
<matplotlib.legend.Legend at 0x7c3e5b999060>
```



Distribution of Passenger Ages
Density
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
Age

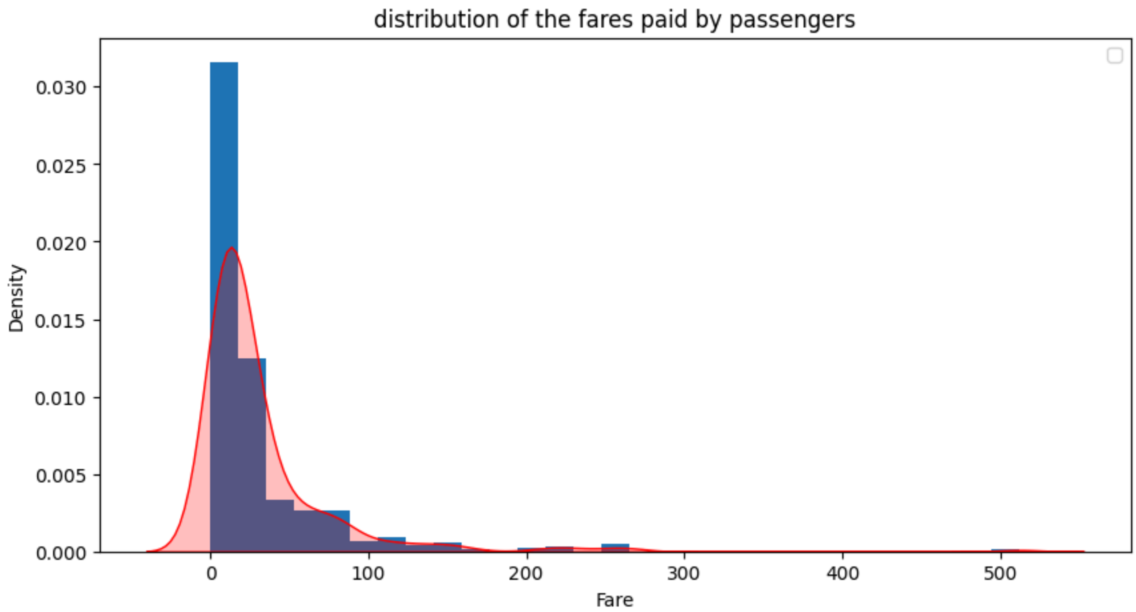
In [18]:

```python
# Determine the distribution of the fares paid by passengers and visual
plt.figure(figsize=(10,5))
plt.hist(
    df['Fare'],
    bins=int(np.sqrt(len(df['Fare']))),
    range=None,
    density=True,
    weights=None,
    cumulative=False,
    bottom=None,
    histtype='bar',
    align='mid',
    orientation='vertical',
    rwidth=None,
    log=False,
    color=None,
    label=None,
    stacked=False,
)
sns.kdeplot(
    data=df['Fare'],
    hue=None,
    weights=None,
    palette=None,
    hue_order=None,
    hue_norm=None,
    color='Red',
    fill=True,
    multiple='layer',
    common_norm=True,
    common_grid=True,
    cumulative=False,
    bw_method='silverman',
    bw_adjust=1,
    warn_singular=True,
    log_scale=None,
    levels=10,
    thresh=0.05,
    gridsize=200,
    cut=3,
    clip=None,
    legend=True,
    cbar=False,
    cbar_ax=None,
    cbar_kws=None,
    ax=None,
)

plt.xlabel('Fare')
plt.ylabel('Density')
plt.title('distribution of the fares paid by passengers')
plt.legend()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWa
rning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[18]:

```
<matplotlib.legend.Legend at 0x7c3e5b7bc5e0>
```



In [ ]:

```
```

Task 4

In [19]:

```
# Perform statistical transformations on the data:

# Apply logarithmic transformation to the fare data
# and visualize the transformed distribution using a
# histogram and a kernel density plot.

# Apply square root transformation to the age data
# and visualize the transformed distribution using a
# histogram and a kernel density plot
```

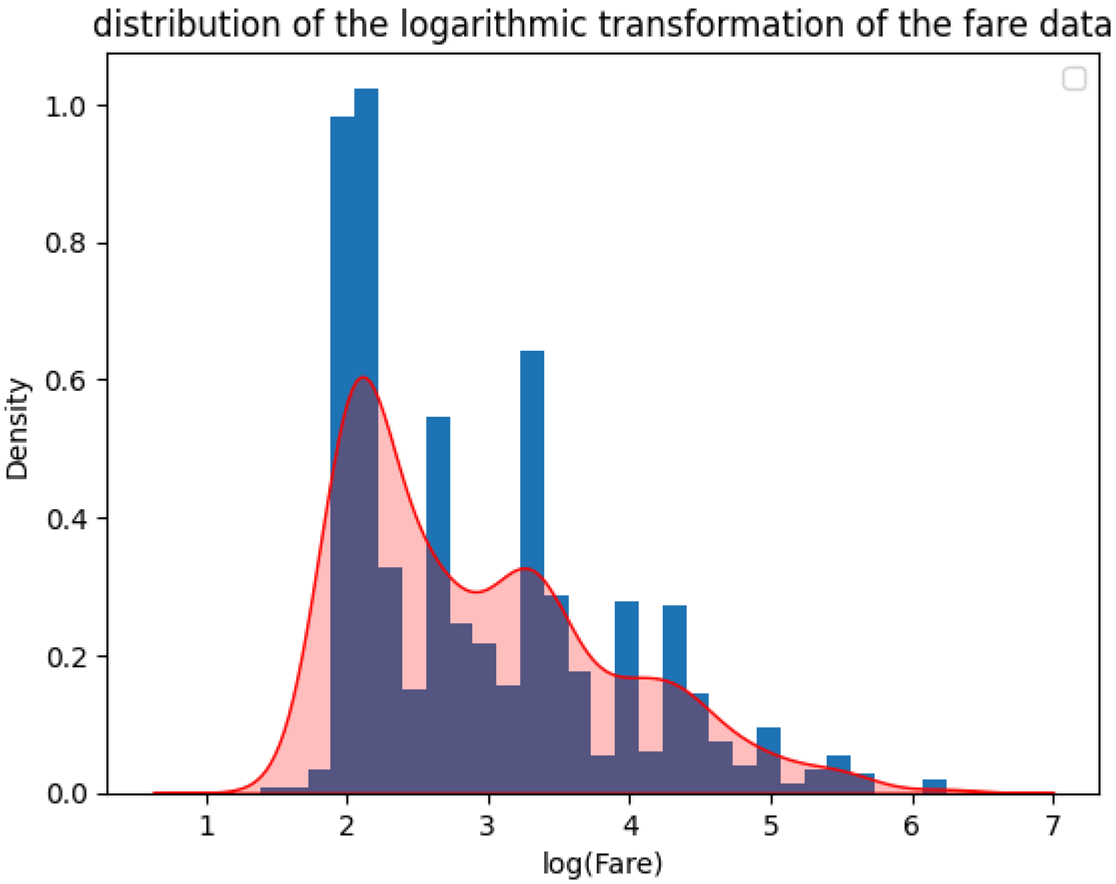
In [20]:

```
# Apply logarithmic transformation to the fare data
# and visualize the transformed distribution using a
# histogram and a kernel density plot.

log_fare = np.log(df['Fare'])
log_fare = log_fare[log_fare>0]

plt.hist(
    log_fare,
    bins=int(np.sqrt(len(log_fare))),
    range=None,
    density=True,
    weights=None,
    cumulative=False,
    bottom=None,
    histtype='bar',
    align='mid',
    orientation='vertical',
    rwidth=None,
    log=False,
    color=None,
    label=None,
    stacked=False,
)
sns.kdeplot(
    data=log_fare,
    hue=None,
    weights=None,
    palette=None,
    hue_order=None,
    hue_norm=None,
    color='Red',
    fill=True,
    multiple='layer',
    common_norm=True,
    common_grid=True,
    cumulative=False,
    bw_method='silverman',
    bw_adjust=1,
    warn_singular=True,
    log_scale=None,
    levels=10,
    thresh=0.05,
    gridsize=200,
    cut=3,
    clip=None,
    legend=True,
    cbar=False,
    cbar_ax=None,
    cbar_kws=None,
    ax=None,
)

plt.xlabel('log(Fare)')
plt.ylabel('Density')
plt.title('distribution of the logarithmic transformation of the fare d
plt.legend()
```

58

```
/opt/conda/lib/python3.10/site-packages/pandas/core/arraylike.py:399: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[20]: <matplotlib.legend.Legend at 0x7c3e5965e230>

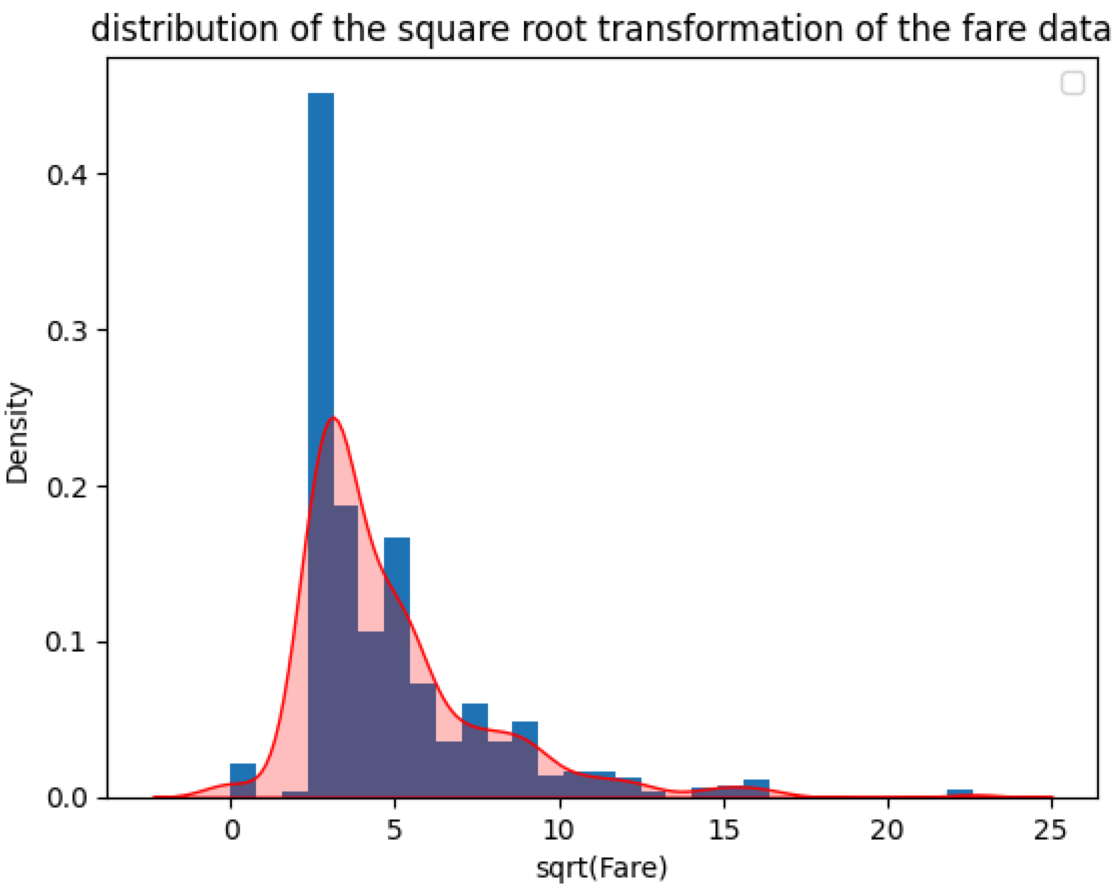


In [21]:

```
# Apply square root transformation to the age data
# and visualize the transformed distribution using a
# histogram and a kernel density plot

sqroot_fare = np.sqrt(df['Fare'])

plt.hist(
    sqroot_fare,
    bins=int(np.sqrt(len(sqroot_fare))),
    range=None,
    density=True,
    weights=None,
    cumulative=False,
    bottom=None,
    histtype='bar',
    align='mid',
    orientation='vertical',
    rwidth=None,
    log=False,
    color=None,
    label=None,
    stacked=False,
)
sns.kdeplot(
    data=sqroot_fare,
    hue=None,
    weights=None,
    palette='viridis',
    hue_order=None,
    hue_norm=None,
    color='Red',
    fill=True,
    multiple='layer',
    common_norm=True,
    common_grid=True,
    cumulative=False,
    bw_method='silverman',
    bw_adjust=1,
    warn_singular=True,
    log_scale=None,
    levels=10,
    thresh=0.05,
    gridsize=200,
    cut=3,
    clip=None,
    legend=True,
    cbar=False,
    cbar_ax=None,
    cbar_kws=None,
    ax=None,
)

plt.xlabel('sqrt(Fare)')
plt.ylabel('Density')
plt.title('distribution of the square root transformation of the fare d
plt.legend()
```

```
/tmp/ipykernel_33/2867368794.py:24: UserWarning: Ignoring `palette` becaus
e no `hue` variable has been assigned.
  sns.kdeplot(
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWa
rning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[21]: <matplotlib.legend.Legend at 0x7c3e596f89a0>



In [ ]:

```
 
```

Task 5

In [22]:

```python
# Perform Statistical sampling and calculate confidence
# intervals:

# a. Randomly sample 200 passengers' ages from the
# dataset and calculate the 95% confidence interval for
# the population mean age.

# b. Repeat step a multiple times (e.g., 1000 times) and
# calculate the confidence intervals each time.

# c. Calculate the proportion of confidence intervals
# that contain the true population mean.
```

In [23]:

```python
# a. Randomly sample 200 passengers' ages from the
# dataset and calculate the 95% confidence interval for
# the population mean age.

sample_size = 200
sample_passanger_age = np.random.choice(df['Age'].dropna(),size=sample_
sample_mean = np.mean(sample_passanger_age)

sample_std = np.std(sample_passanger_age,ddof=1)

z_critical = stats.norm.ppf(0.975) # 0.975 for 95% confidence level, tw

standard_error = sample_std/ np.sqrt(sample_size)

confidence_interval = ((sample_mean - z_critical*standard_error),(sampl
confidence_interval
```

Out[23]: (28.145661282601434, 32.272638717398564)

In [24]:

```python
# b. Repeat step a multiple times (e.g., 1000 times) and
# calculate the confidence intervals each time.
intervals = []
sample_size = 200
for i in range(1000):
    sample_passanger_age = np.random.choice(df['Age'].dropna(),size=sam
    sample_mean = np.mean(sample_passanger_age)

    sample_std = np.std(sample_passanger_age,ddof=1)

    z_critical = stats.norm.ppf(0.975) # 0.975 for 95% confidence level

    standard_error = sample_std/ np.sqrt(sample_size)

    confidence_interval = ((sample_mean - z_critical*standard_error),(s
    print(i+1, confidence_interval)
    intervals.append( confidence_interval)
```

```
1 (28.609789308949175, 32.59021069105083)
2 (27.920187241428007, 31.561512758571997)
3 (29.014251272361328, 33.109948727638674)
4 (26.824110228893957, 30.93758977110604)
5 (26.2453433493244, 30.304656650675597)
6 (28.062349257411075, 32.261850742588926)
7 (27.447698956807624, 31.693201043192378)
8 (27.48440494085015, 31.503095059149846)
9 (28.819987443301024, 32.98831255669898)
10 (28.148161964726384, 32.22773803527362)
11 (27.30904037179329, 31.094359628206714)
12 (28.319443209419223, 32.19055679058077)
13 (27.8502636874476, 31.8730363125524)
14 (27.338091741862886, 31.215308258137117)
15 (26.3291645749079, 30.3058354250921)
16 (27.804042013740244, 32.06095798625976)
17 (27.77145723280544, 32.02955427671946)
18 (27.536339990857567, 31.897860009142438)
19 (26.877493903777193, 30.687506096222805)
```

In [25]:
```
# c. Calculate the proportion of confidence intervals
# that contain the true population mean.

true_population_mean_age = df['Age'].mean()
count_mean = 0
for interval in range(len(intervals)):
    if (true_population_mean_age >= intervals[0][0]) and (true_populati
        count_mean += 1
print(f'the proportion of confidence intervals that contain the true po
```

```
the proportion of confidence intervals that contain the true population me
an: 100.0 %
```

In [ ]:
```

```

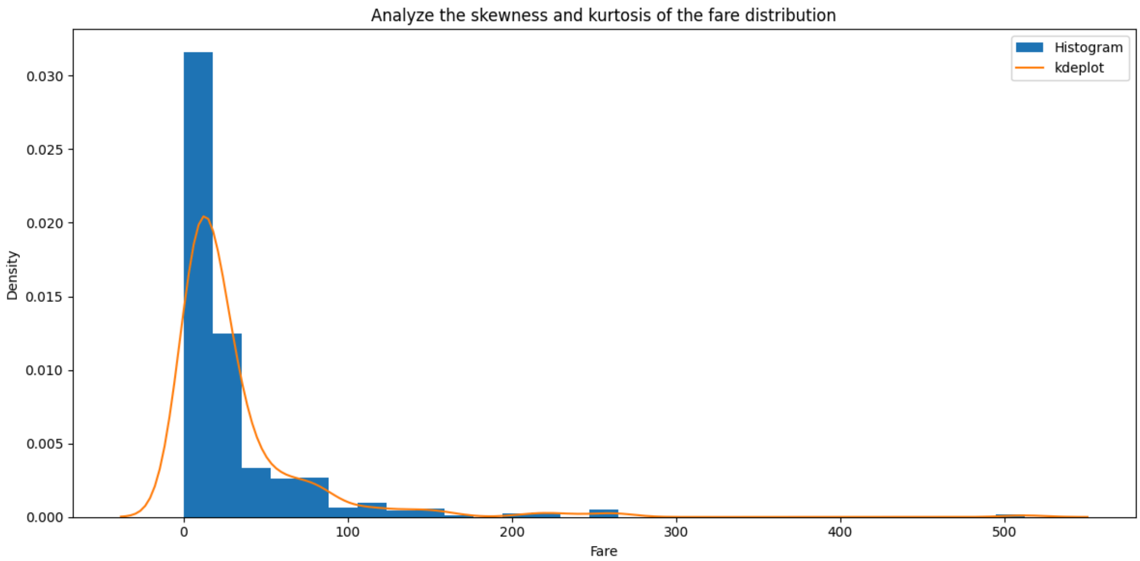
Task 5

In [26]:
```
# Analyze the skewness and kurtosis of the fare
# distribution:

# Calculate the skewness and kurtosis of the fare
# data.

# Interpret the results and discuss the shape of the
# distribution
```

In [32]:

```python
# Analyze the skewness and kurtosis of the fare
# distribution:

plt.figure(figsize=(12, 6))
plt.hist(df['Fare'],bins=int(np.sqrt(len(df['Fare']))),histtype='bar',d
sns.kdeplot(df['Fare'],bw_method='scott', label='kdeplot')

plt.xlabel('Fare')
plt.ylabel('Density')
plt.title('Analyze the skewness and kurtosis of the fare distribution')
plt.legend()

plt.tight_layout()
plt.show()


# Shape of the Distribution:

# The distribution is heavily skewed to the right, meaning most of the 
# The high kurtosis indicates that the distribution has a sharp peak an

```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWa
rning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [28]:

```python
# Calculate the skewness and kurtosis of the fare data.
fare_skewness = stats.skew(df['Fare'].dropna())
fare_kurtosis = stats.kurtosis(df['Fare'].dropna())

print(f'skewness of the fare distribution: {fare_skewness}')
print(f'kurtosis of the fare distribution: {fare_kurtosis}')
```

```
skewness of the fare distribution: 4.7792532923723545
kurtosis of the fare distribution: 33.20428925264474
```

In [29]:

```
# Interpret the results and discuss the shape of the distribution

# Skewness : Skewness measures the asymmetry of the distribution around

# Positive Skewness: A skewness value of 4.779 indicates a highly posit
# Implication: This suggests that most passengers paid relatively low f

# Kurtosis : Kurtosis measures the "tailedness" of the distribution.

# High Kurtosis (Leptokurtic): A kurtosis value of 33.204 is very high,
# Implication: The fare data has many extreme values, indicating a dist
```

In [ ]:

```

```

Task 6

In [33]:

```
# Calculate confidence intervals for population parameters:

# Calculate a 95% confidence interval for the population mean fare.

# Calculate a 95% confidence interval for the population proportion of
```

In [50]:

```
# Calculate a 95% confidence interval for the population mean fare.

population_mean = df['Fare'].mean() #population mean
population_std = df['Fare'].std() #population std

z_score = stats.norm.ppf(0.975) #for 95% confidence interval

standard_error = population_std/np.sqrt(len(df['Fare']))

confidence_interval = (population_mean - z_score*standard_error,populat
print("95% confidence interval for the population mean fare: ",confiden

count_fare=0
for fare in df['Fare']:
    if (fare >= confidence_interval[0]) and (fare <= confidence_interva
        count_fare+=1

print("Proportion of fares lie in the above mentioned confidence interv

```

```
95% confidence interval for the population mean fare:  (28.94127463271884,
35.46714130443043)
Proportion of fares lie in the above mentioned confidence interval:  0.06
```

In [56]:

```
# Calculate a 95% confidence interval for the population proportion of

sample_proportion = df['Survived'].mean()

z_score = stats.norm.ppf(0.975) #for 95% confidence interval

standard_error = np.sqrt((sample_proportion * (1-sample_proportion))/le

margin_of_error = z_score * standard_error

confidence_interval = (sample_proportion - margin_of_error,sample_propo

sample_proportion

print('95% confidence interval for the population proportion of survivo
```

```
95% confidence interval for the population proportion of survivors (Surviv
ed column):  (0.3519060427032577, 0.4157707249735099)
```