Experiment No.1

AIM: Getting data to work with:

i. Download the sample dataset locally for any application (Kaggle)

ii. Setting up the working directory.

iii. Unpacking the data. Decompress the file locally.

iv. Looking at the data. Display the top (10) and bottom (10) of the file.

v. Measuring the length of the data set. Count the number of lines in the file.

vi. Encode the categorical data

vi. Plot a graph and give your insights for the application selected cases.

Theory:

i. Download the sample dataset locally for any application (Kaggle)

Step 1: Create a Kaggle Account

1. Go to Kaggle.
2. Sign up for an account or log in if you already have one.

Step 2: Install Kaggle API

1. Ensure you have Python installed on your computer.
2. Install the Kaggle API using pip:

```
bash
Copy code
pip install kaggle
```

Step 3: Get Your Kaggle API Key

1. Go to your Kaggle account settings by clicking on your profile picture in the top right corner and selecting "My Account".
2. Scroll down to the "API" section and click "Create New API Token". This will download a file named `kaggle.json`.

Step 4: Configure Kaggle API

1. Place the `kaggle.json` file in a folder named `.kaggle` in your home directory.
   - For Windows: `C:\Users\<Your Username>\.kaggle\kaggle.json`
   - For Mac/Linux: `/home/<Your Username>/.kaggle/kaggle.json`
2. Ensure the file has the correct permissions:

```
bash
Copy code
chmod 600 ~/.kaggle/kaggle.json
```

Step 5: Find the Dataset

1. Browse or search for the dataset you want to download on the Kaggle website.
2. Go to the dataset's page and copy the API command for downloading it. It will look something like this:

```
bash
Copy code
kaggle datasets download -d <dataset-owner>/<dataset-name>
```

Step 6: Download the Dataset

1. Open a terminal or command prompt.
2. Run the command you copied in Step 5 to download the dataset. For example:

```
bash
Copy code
kaggle datasets download -d zillow/zecon
```

Step 7: Extract the Dataset

1. The dataset will be downloaded as a zip file in your current directory.
2. Extract the contents of the zip file:
    - For Windows: Right-click the file and select "Extract All".
    - For Mac/Linux: Use the terminal:

```
bash
Copy code
unzip <dataset-file>.zip
```

Example Walkthrough

Let's go through an example to download a sample dataset called "Titanic - Machine Learning from Disaster".

1. **Find the Dataset**:
    - Go to the Titanic dataset page on Kaggle.
2. **Copy the API Command**:
    - The API command to download this dataset is:

```
bash
Copy code
kaggle competitions download -c titanic
```

3. **Download and Extract**:

```
bash
Copy code
kaggle competitions download -c titanic
unzip titanic.zip
```

Additional Notes

- You can specify the download location by changing the directory before running the download command:

```
bash
Copy code
cd /path/to/your/directory
```

```
kaggle datasets download -d <dataset-owner>/<dataset-name>
```

ii. Setting up the working directory.

Step-by-Step Guide

Step 1: Create a Working Directory

Choose a location on your computer where you want to store your datasets. For example, you can create a directory named `KaggleDatasets`.

1. Open your terminal (Command Prompt on Windows, Terminal on Mac/Linux).
2. Create a directory:

```
bash
Copy code
mkdir -p /path/to/your/directory/KaggleDatasets
```

Replace `/path/to/your/directory` with the path where you want to create the directory.

Step 2: Navigate to the Working Directory

Move into the directory you just created:

```
bash
Copy code
cd /path/to/your/directory/KaggleDatasets
```

Step 3: Set Up Kaggle API Key

Ensure you have your `kaggle.json` API key file configured properly in your home directory as mentioned in the previous steps.

For Windows:

- `C:\Users\<Your Username>\.kaggle\kaggle.json`

For Mac/Linux:

- `/home/<Your Username>/.kaggle/kaggle.json`

Step 4: Download a Dataset

Now, you can download a dataset into your working directory. For example, to download the Titanic dataset:

```
bash
Copy code
kaggle competitions download -c titanic
```

Step 5: Extract the Dataset

After downloading, the dataset will be in a zip file. Extract it using the following command:

```
bash
Copy code
unzip titanic.zip
```

Example Walkthrough

1. **Create a Working Directory**:

```
bash
Copy code
mkdir -p ~/Documents/KaggleDatasets
cd ~/Documents/KaggleDatasets
```

2. **Download the Dataset**:

```
bash
Copy code
kaggle competitions download -c titanic
```

3. **Extract the Dataset**:

```
bash
Copy code
unzip titanic.zip
```

Automating with a Script

You can automate the process with a shell script. Here's an example script:

```
bash
Copy code
#!/bin/bash

# Set your working directory
WORK_DIR=~/Documents/KaggleDatasets

# Create and navigate to the working directory
mkdir -p $WORK_DIR
cd $WORK_DIR

# Download the dataset
kaggle competitions download -c titanic

# Extract the dataset
unzip titanic.zip
```

Save this script as `download_kaggle_dataset.sh`, give it executable permissions, and run it:

```
bash
Copy code
chmod +x download_kaggle_dataset.sh
./download_kaggle_dataset.sh
```

This should streamline the process of setting up your working directory and downloading datasets from Kaggle.

iii. Unpacking the data. Decompress the file locally.

Step-by-Step Guide

Step 1: Download the Dataset

Assuming you have already downloaded the dataset as described in previous steps, the file will be in a ZIP format.

Step 2: Decompress the File

Using Command Line

1. **Navigate to the Directory**: Open your terminal and navigate to the directory where your downloaded dataset is located:

```
bash
Copy code
cd /path/to/your/directory/KaggleDatasets
```

2. **Unzip the File**: Use the `unzip` command to decompress the ZIP file:

```
bash
Copy code
unzip titanic.zip
```

Replace `titanic.zip` with the name of your downloaded ZIP file.

Using Python

You can also use Python to decompress the file. This is especially useful if you're working within a Jupyter notebook or a Python script.

1. **Python Script**:

```
python
Copy code
import zipfile
import os

# Define the path to your zip file and the directory to extract to
zip_path = '/path/to/your/directory/KaggleDatasets/titanic.zip'
extract_dir = '/path/to/your/directory/KaggleDatasets'

# Unzip the file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_dir)

print("Dataset unpacked successfully!")
```

Example Walkthrough with Titanic Dataset

Command Line

1. **Navigate to the Directory**:

```
bash
Copy code
cd ~/Documents/KaggleDatasets
```

2. **Unzip the File**:

```
bash
Copy code
unzip titanic.zip
```

Python Script

1. **Unpacking Script**:

```
python
Copy code
import zipfile
import os

# Define the path to your zip file and the directory to extract to
zip_path =
os.path.expanduser('~/Documents/KaggleDatasets/titanic.zip')
extract_dir = os.path.expanduser('~/Documents/KaggleDatasets')
```

```
# Unzip the file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_dir)

print("Dataset unpacked successfully!")
```

Verification

After decompressing the file, you should see the extracted contents in your specified directory. Typically, the dataset will include CSV files and possibly other supporting files like README or documentation.

You can list the contents of the directory to verify:

```
bash
Copy code
ls /path/to/your/directory/KaggleDatasets
```

Or in Python:

```
python
Copy code
import os

# List the contents of the directory
extracted_files = os.listdir('/path/to/your/directory/KaggleDatasets')
print("Extracted files:", extracted_files)
```

This will help you confirm that the dataset has been successfully decompressed and is ready for use.

iv. Unpacking the data. Decompress the file locally.

To decompress and unpack a downloaded dataset from Kaggle, you can use command line tools or a Python script. Here's a detailed guide for both methods:

Method 1: Using Command Line

1. **Navigate to the Directory**: **Open your terminal and navigate to the directory where your downloaded dataset is located. Replace `/path/to/your/directory/KaggleDatasets` with your actual path:**

   ```
   bash
   Copy code
   cd /path/to/your/directory/KaggleDatasets
   ```

2. **Unzip the File**: Use the `unzip` command to decompress the ZIP file. Replace `titanic.zip` with the name of your downloaded ZIP file:

   ```
   bash
   Copy code
   unzip titanic.zip
   ```

Method 2: Using Python

You can also use Python to decompress the file, which is particularly useful if you are working within a Jupyter notebook or a Python script.

1. **Python Script**:

   ```
   python
   Copy code
   import zipfile
   import os
   ```

```
# Define the path to your zip file and the directory to extract to
zip_path = '/path/to/your/directory/KaggleDatasets/titanic.zip'
extract_dir = '/path/to/your/directory/KaggleDatasets'

# Unzip the file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_dir)

print("Dataset unpacked successfully!")
```

Example Walkthrough with Titanic Dataset

Using Command Line

1. **Navigate to the Directory**:

```
bash
Copy code
cd ~/Documents/KaggleDatasets
```

2. **Unzip the File**:

```
bash
Copy code
unzip titanic.zip
```

Using Python Script

1. **Unpacking Script**:

```
python
Copy code
import zipfile
import os

# Define the path to your zip file and the directory to extract to
zip_path =
os.path.expanduser('~/Documents/KaggleDatasets/titanic.zip')
extract_dir = os.path.expanduser('~/Documents/KaggleDatasets')

# Unzip the file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_dir)

print("Dataset unpacked successfully!")
```

Verification

After decompressing the file, you should see the extracted contents in your specified directory. Typically, the dataset will include CSV files and possibly other supporting files like README or documentation.

You can list the contents of the directory to verify:

Using Command Line

```
bash
Copy code
ls /path/to/your/directory/KaggleDatasets
```

Using Python

```
python
Copy code
import os
```

```
# List the contents of the directory
extracted_files = os.listdir('/path/to/your/directory/KaggleDatasets')
print("Extracted files:", extracted_files)
```

iv. Looking at the data. Display the top (10) and bottom (10) of the file.

Using Python and Pandas

1. **Install Pandas**: If you haven't already, install the pandas library:

   ```
   bash
   Copy code
   pip install pandas
   ```

2. **Load and Display the Data**: Use the following Python script to load the dataset and display the top and bottom 10 rows:

   ```
   python
   Copy code
   import pandas as pd

   # Define the path to your dataset file
   file_path = '/path/to/your/directory/KaggleDatasets/train.csv'

   # Load the dataset
   df = pd.read_csv(file_path)

   # Display the top 10 rows
   print("Top 10 rows of the dataset:")
   print(df.head(10))

   # Display the bottom 10 rows
   print("\nBottom 10 rows of the dataset:")
   print(df.tail(10))
   ```

Example with Titanic Dataset

1. **Assume your Titanic dataset is in `train.csv`.**

2. **Save and run the following script**:

   ```
   python
   Copy code
   import pandas as pd
   import os

   # Define the path to your dataset file
   file_path =
   os.path.expanduser('~/Documents/KaggleDatasets/train.csv')

   # Load the dataset
   df = pd.read_csv(file_path)

   # Display the top 10 rows
   print("Top 10 rows of the dataset:")
   print(df.head(10))

   # Display the bottom 10 rows
   print("\nBottom 10 rows of the dataset:")
   print(df.tail(10))
   ```

Explanation:

- **Importing Pandas**: The `import pandas as pd` statement imports the pandas library and gives it an alias `pd`.
- **Loading the Dataset**: The `pd.read_csv(file_path)` function reads the CSV file into a pandas DataFrame.
- **Displaying Rows**:
  - `df.head(10)` displays the first 10 rows of the DataFrame.
  - `df.tail(10)` displays the last 10 rows of the DataFrame.

Running the Script

Save the script in a `.py` file and run it using a Python interpreter:

```
bash
Copy code
python script_name.py
```

Or, if you are using a Jupyter notebook, you can simply execute the cells containing the above code.

Example Output:

You should see an output similar to the following, showing the first 10 and last 10 rows of the dataset:

```
r
Copy code
Top 10 rows of the dataset:
   PassengerId  Survived  Pclass  ...            Ticket     Fare Cabin
0            1         0       3  ...         A/5 21171   7.2500   NaN
1            2         1       1  ...          PC 17599  71.2833   C85
2            3         1       3  ...  STON/O2. 3101282   7.9250   NaN
3            4         1       1  ...            113803  53.1000  C123
4            5         0       3  ...            373450   8.0500   NaN
5            6         0       3  ...            330877   8.4583   NaN
6            7         0       1  ...             17463  51.8625   E46
7            8         0       3  ...            349909  21.0750   NaN
8            9         1       3  ...            347742  11.1333   NaN
9           10         1       2  ...            237736  30.0708   NaN

[10 rows x 12 columns]

Bottom 10 rows of the dataset:
     PassengerId  Survived  Pclass  ...  SibSp  Parch     Fare Cabin
881          882         0       3  ...      0      0  10.5167   NaN
882          883         0       2  ...      0      0  10.5000   NaN
883          884         0       3  ...      0      0   7.0500   NaN
884          885         0       3  ...      0      0  29.1250   NaN
885          886         0       3  ...      0      5   7.2250   NaN
886          887         0       2  ...      0      0  13.0000   NaN
887          888         1       1  ...      0      0  30.0000   B42
888          889         0       3  ...      1      2  23.4500   NaN
889          890         1       1  ...      0      0  30.0000  C148
890          891         0       3  ...      0      0   7.7500   NaN

[10 rows x 12 columns]
```

This will give you a quick overview of the dataset, helping you understand its structure and contents.

v. Measuring the length of the data set. Count the number of lines in the file.

To count the number of lines in a dataset file, you can use either command line tools or a Python script. Here are both methods:

Method 1: Using Command Line

Using `wc` Command on Unix/Linux/Mac

The `wc` command is used to count lines, words, and characters in a file. To count the lines, you can use the `-l` option.

```
bash
Copy code
wc -l /path/to/your/directory/KaggleDatasets/train.csv
```

This command will output the number of lines in the file.

Using `Get-Content` on Windows PowerShell

If you are using Windows, you can use PowerShell to count the lines:

```
powershell
Copy code
(Get-Content C:\path\to\your\directory\KaggleDatasets\train.csv).Count
```

Method 2: Using Python

1. **Python Script**: You can use a simple Python script to count the number of lines in the file:

```
python
Copy code
# Define the path to your dataset file
file_path = '/path/to/your/directory/KaggleDatasets/train.csv'

# Initialize a counter
line_count = 0

# Open the file and count the lines
with open(file_path, 'r') as file:
    for line in file:
        line_count += 1

print(f"Number of lines in the file: {line_count}")
```

Example with Titanic Dataset

1. **Command Line**:

```
bash
Copy code
wc -l ~/Documents/KaggleDatasets/train.csv
```

2. **Python Script**:

```
python
Copy code
import os

# Define the path to your dataset file
file_path =
os.path.expanduser('~/Documents/KaggleDatasets/train.csv')

# Initialize a counter
line_count = 0

# Open the file and count the lines
```

```
with open(file_path, 'r') as file:
    for line in file:
        line_count += 1

print(f"Number of lines in the file: {line_count}")
```

Running the Script

Save the Python script in a `.py` file and run it using a Python interpreter:

```
bash
Copy code
python script_name.py
```

Or, if you are using a Jupyter notebook, you can simply execute the cell containing the above code.

Example Output:

If the Titanic dataset file (`train.csv`) has 892 lines (including the header), the output will be:

```
javascript
Copy code
Number of lines in the file: 892
```

This method provides an accurate count of the number of lines in the dataset file, including the header row.

To count the number of lines in a dataset file, you can use either command line tools or a Python script. Here are both methods:

Method 1: Using Command Line

Using `wc` Command on Unix/Linux/Mac

The `wc` command is used to count lines, words, and characters in a file. To count the lines, you can use the `-l` option.

```
bash
Copy code
wc -l /path/to/your/directory/KaggleDatasets/train.csv
```

This command will output the number of lines in the file.

Using `Get-Content` on Windows PowerShell

If you are using Windows, you can use PowerShell to count the lines:

```
powershell
Copy code
(Get-Content C:\path\to\your\directory\KaggleDatasets\train.csv).Count
```

Method 2: Using Python

1. **Python Script**: You can use a simple Python script to count the number of lines in the file:

   ```
   python
   Copy code
   # Define the path to your dataset file
   file_path = '/path/to/your/directory/KaggleDatasets/train.csv'

   # Initialize a counter
   line_count = 0

   # Open the file and count the lines
   with open(file_path, 'r') as file:
       for line in file:
   ```

```
            line_count += 1

    print(f"Number of lines in the file: {line_count}")
```

Example with Titanic Dataset

1. **Command Line**:

```
bash
Copy code
wc -l ~/Documents/KaggleDatasets/train.csv
```

2. **Python Script**:

```
python
Copy code
import os

# Define the path to your dataset file
file_path =
os.path.expanduser('~/Documents/KaggleDatasets/train.csv')

# Initialize a counter
line_count = 0

# Open the file and count the lines
with open(file_path, 'r') as file:
    for line in file:
        line_count += 1

print(f"Number of lines in the file: {line_count}")
```

Running the Script

Save the Python script in a `.py` file and run it using a Python interpreter:

```
bash
Copy code
python script_name.py
```

Or, if you are using a Jupyter notebook, you can simply execute the cell containing the above code.

Example Output:

If the Titanic dataset file (`train.csv`) has 892 lines (including the header), the output will be:

```
javascript
Copy code
Number of lines in the file: 892
```

This method provides an accurate count of the number of lines in the dataset file, including the header row.

vi. Encode the categorical data

Encoding categorical data is an essential step in preprocessing before using the data in machine learning models. Common methods for encoding categorical data include:

1. **Label Encoding**: Assigns each unique category a different integer.
2. **One-Hot Encoding**: Creates binary columns for each category.

We'll use the Titanic dataset as an example to demonstrate both methods.

Step-by-Step Guide

Step 1: Load the Dataset

First, load the Titanic dataset into a pandas DataFrame.

```
python
Copy code
import pandas as pd
import os

# Define the path to your dataset file
file_path = os.path.expanduser('~/Documents/KaggleDatasets/train.csv')

# Load the dataset
df = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(df.head())
```

Step 2: Identify Categorical Columns

Identify the columns with categorical data. In the Titanic dataset, typical categorical columns include 'Sex', 'Embarked', and possibly 'Pclass'.

```
python
Copy code
# Display the data types of the columns
print("\nData types of the columns:")
print(df.dtypes)

# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object', 'category']).columns
print("\nCategorical columns:")
print(categorical_cols)
```

Step 3: Encode Categorical Data

Method 1: Label Encoding

Label encoding can be applied directly using pandas or `sklearn.preprocessing.LabelEncoder`. This method is straightforward but can introduce ordinal relationships that might not exist.

```
python
Copy code
from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Apply label encoding to each categorical column
for col in categorical_cols:
    df[col] = label_encoder.fit_transform(df[col].astype(str))

print("\nDataset after label encoding:")
print(df.head())
```

Method 2: One-Hot Encoding

One-hot encoding creates binary columns for each category. This method avoids ordinal relationships and is more suitable for most machine learning models.

```
python
Copy code
# Apply one-hot encoding
df_one_hot_encoded = pd.get_dummies(df, columns=categorical_cols)

print("\nDataset after one-hot encoding:")
print(df_one_hot_encoded.head())
```

Example Output

After applying label encoding and one-hot encoding, the dataset's categorical columns will be transformed.

Label Encoding Example Output:

```
python
Copy code
First few rows of the dataset:
   PassengerId  Survived  Pclass
Name  Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
0            1         0       3                                Braund, Mr.
Owen Harris    1  22.0      1      0         A/5 21171   7.2500   NaN
2
1            2         1       1  Cumings, Mrs. John Bradley (Florence
Briggs Thayer)  0  38.0      1      0          PC 17599  71.2833   C85
0
2            3         1       3                                Heikkinen,
Miss. Laina    0  26.0      0      0  STON/O2. 3101282   7.9250   NaN
2
3            4         1       1       Futrelle, Mrs. Jacques Heath (Lily
May Peel)    0  35.0      1      0            113803  53.1000  C123
2
4            5         0       3                              Allen, Mr.
William Henry    1  35.0      0      0            373450   8.0500   NaN
2

Dataset after label encoding:
   PassengerId  Survived  Pclass  Name  Sex   Age  SibSp  Parch  Ticket
Fare  Cabin  Embarked
0            1         0       3     22    1  22.0      1      0      9
7.2500      0         2
1            2         1       1     41    0  38.0      1      0     33
71.2833      2         0
2            3         1       3     51    0  26.0      0      0     48
7.9250      0         2
3            4         1       1     56    0  35.0      1      0     27
53.1000      2         2
4            5         0       3     61    1  35.0      0      0     53
8.0500      0         2
```

One-Hot Encoding Example Output:

```
python
Copy code
Dataset after one-hot encoding:
   PassengerId  Survived  Pclass
Name  Age  SibSp  Parch            Ticket     Fare  Sex_0  Sex_1
Embarked_0  Embarked_1  Embarked_2
0            1         0       3                              Braund, Mr.
Owen Harris  22.0      1      0         A/5 21171   7.2500      0      1
0           0           1
1            2         1       1  Cumings, Mrs. John Bradley (Florence
Briggs Thayer)  38.0      1      0          PC 17599  71.2833      1      0
1           0           0
```

```
2            3         1       3                               Heikkinen,
Miss. Laina  26.0      0       0  STON/O2. 3101282   7.9250      1       0
0            0          1
3            4         1       1       Futrelle, Mrs. Jacques Heath (Lily
May Peel)  35.0      1       0           113803  53.1000      1       0
0            0          1
4            5         0       3                            Allen, Mr.
William Henry  35.0      0       0            373450   8.0500      0       1
0            0          1
```

Conclusion

Label encoding is simpler but may introduce ordinal relationships. One-hot encoding is more suitable for most machine learning models as it avoids ordinal relationships. Choose the method that best fits your application.

vi. Plot a graph and give your insights for the application selected cases.

To gain insights from the Titanic dataset, let's plot some graphs and analyze them. We will use the `matplotlib` and `seaborn` libraries for visualization.

Step-by-Step Guide

Step 1: Install Required Libraries

If you haven't already, install the necessary libraries:

```
bash
Copy code
pip install pandas matplotlib seaborn
```

Step 2: Load the Dataset and Preprocess

```
python
Copy code
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Define the path to your dataset file
file_path = os.path.expanduser('~/Documents/KaggleDatasets/train.csv')

# Load the dataset
df = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(df.head())
```

Step 3: Plotting Graphs

1. Survival Rate by Gender

```
python
Copy code
# Set the style of the visualization
sns.set(style="whitegrid")

# Create a count plot
plt.figure(figsize=(10, 6))
```

```
sns.countplot(x='Sex', hue='Survived', data=df, palette='viridis')

# Set the title and labels
plt.title('Survival Rate by Gender')
plt.xlabel('Sex')
plt.ylabel('Count')

# Show the plot
plt.show()
```

2. Survival Rate by Passenger Class

```
python
Copy code
# Create a count plot
plt.figure(figsize=(10, 6))
sns.countplot(x='Pclass', hue='Survived', data=df, palette='viridis')

# Set the title and labels
plt.title('Survival Rate by Passenger Class')
plt.xlabel('Pclass')
plt.ylabel('Count')

# Show the plot
plt.show()
```

3. Age Distribution of Survivors and Non-Survivors

```
python
Copy code
# Create a histogram
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Age', hue='Survived', multiple='stack',
palette='viridis', bins=30)

# Set the title and labels
plt.title('Age Distribution of Survivors and Non-Survivors')
plt.xlabel('Age')
plt.ylabel('Count')

# Show the plot
plt.show()
```

Insights from the Plots

1. Survival Rate by Gender

- **Observation**: The count plot shows that a significantly higher proportion of females survived compared to males. This is consistent with the "women and children first" policy often followed in maritime disasters.

2. Survival Rate by Passenger Class

- **Observation**: The plot reveals that passengers in first class had a higher survival rate compared to those in second and third classes. This indicates that socioeconomic status played a significant role in survival during the Titanic disaster.

3. Age Distribution of Survivors and Non-Survivors

- **Observation**: The histogram shows that a higher proportion of younger passengers survived. This could be due to the prioritization of women and children during the evacuation.

Full Code

Here's the complete code to load the dataset, preprocess it, and generate the plots:

```
python
Copy code
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Define the path to your dataset file
file_path = os.path.expanduser('~/Documents/KaggleDatasets/train.csv')

# Load the dataset
df = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(df.head())

# Set the style of the visualization
sns.set(style="whitegrid")

# Plot 1: Survival Rate by Gender
plt.figure(figsize=(10, 6))
sns.countplot(x='Sex', hue='Survived', data=df, palette='viridis')
plt.title('Survival Rate by Gender')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.show()

# Plot 2: Survival Rate by Passenger Class
plt.figure(figsize=(10, 6))
sns.countplot(x='Pclass', hue='Survived', data=df, palette='viridis')
plt.title('Survival Rate by Passenger Class')
plt.xlabel('Pclass')
plt.ylabel('Count')
plt.show()

# Plot 3: Age Distribution of Survivors and Non-Survivors
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Age', hue='Survived', multiple='stack',
palette='viridis', bins=30)
plt.title('Age Distribution of Survivors and Non-Survivors')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```

Conclusion

These visualizations provide valuable insights into the Titanic dataset, highlighting the influence of gender, passenger class, and age on survival rates. Such analyses are crucial for understanding patterns in the data and can guide further machine learning modeling.