In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [2]:
```
import os

file_path = r"C:\Users\91863\Downloads\titanic3.xls"

if os.path.exists(file_path):
    titanic_data = pd.read_excel(file_path)
    print("File loaded successfully.")
else:
    print("File not found. Please check the file path.")
```

```
File loaded successfully.
```

In [3]:
```
pip install xlrd
```

```
^C
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 24.1.2 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Requirement already satisfied: xlrd in c:\users\91863\anaconda3\lib\site-packages
(2.0.1)
```

In [10]:
```
file_path = r"C:\Users\91863\Downloads\titanic3.xls"
titanic_data = pd.read_excel(file_path)
```

In [12]:
```
print(titanic_data.head())
print(titanic_data.info())
print(titanic_data.describe())
```

```
   pclass  survived                                             name     sex  \
0       1         1                    Allen, Miss. Elisabeth Walton  female
1       1         1                   Allison, Master. Hudson Trevor    male
2       1         0                     Allison, Miss. Helen Loraine  female
3       1         0             Allison, Mr. Hudson Joshua Creighton    male
4       1         0  Allison, Mrs. Hudson J C (Bessie Waldo Daniels)  female

       age  sibsp  parch  ticket      fare    cabin embarked boat   body  \
0  29.0000      0      0   24160  211.3375       B5        S    2    NaN
1   0.9167      1      2  113781  151.5500  C22 C26        S   11    NaN
2   2.0000      1      2  113781  151.5500  C22 C26        S  NaN    NaN
3  30.0000      1      2  113781  151.5500  C22 C26        S  NaN  135.0
4  25.0000      1      2  113781  151.5500  C22 C26        S  NaN    NaN

                        home.dest
0                     St Louis, MO
1  Montreal, PQ / Chesterville, ON
2  Montreal, PQ / Chesterville, ON
3  Montreal, PQ / Chesterville, ON
4  Montreal, PQ / Chesterville, ON
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     1309 non-null   int64
 1   survived   1309 non-null   int64
 2   name       1309 non-null   object
 3   sex        1309 non-null   object
 4   age        1046 non-null   float64
 5   sibsp      1309 non-null   int64
 6   parch      1309 non-null   int64
 7   ticket     1309 non-null   object
 8   fare       1308 non-null   float64
 9   cabin      295 non-null    object
 10  embarked   1307 non-null   object
 11  boat       486 non-null    object
 12  body       121 non-null    float64
 13  home.dest  745 non-null    object
dtypes: float64(3), int64(4), object(7)
memory usage: 143.3+ KB
None
            pclass     survived          age        sibsp        parch  \
count  1309.000000  1309.000000  1046.000000  1309.000000  1309.000000
mean      2.294882     0.381971    29.881135     0.498854     0.385027
std       0.837836     0.486055    14.413500     1.041658     0.865560
min       1.000000     0.000000     0.166700     0.000000     0.000000
25%       2.000000     0.000000    21.000000     0.000000     0.000000
50%       3.000000     0.000000    28.000000     0.000000     0.000000
75%       3.000000     1.000000    39.000000     1.000000     0.000000
max       3.000000     1.000000    80.000000     8.000000     9.000000

              fare        body
count  1308.000000  121.000000
mean     33.295479  160.809917
std      51.758668   97.696922
```

```
min       0.000000    1.000000
25%       7.895800   72.000000
50%      14.454200  155.000000
75%      31.275000  256.000000
max     512.329200  328.000000
```

In [14]:
```
# Fill missing values
titanic_data['age'].fillna(titanic_data['age'].median(), inplace=True)
titanic_data['embarked'].fillna(titanic_data['embarked'].mode()[0], inplace=True)
titanic_data['fare'].fillna(titanic_data['fare'].median(), inplace=True)

# Convert categorical to numerical
titanic_data['sex'] = titanic_data['sex'].map({'male': 0, 'female': 1})
titanic_data['embarked'] = titanic_data['embarked'].map({'S': 0, 'C': 1, 'Q': 2})
```

```
C:\Users\91863\AppData\Local\Temp\ipykernel_17360\2340976590.py:2: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  titanic_data['age'].fillna(titanic_data['age'].median(), inplace=True)
C:\Users\91863\AppData\Local\Temp\ipykernel_17360\2340976590.py:3: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  titanic_data['embarked'].fillna(titanic_data['embarked'].mode()[0], inplace=True)
C:\Users\91863\AppData\Local\Temp\ipykernel_17360\2340976590.py:4: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  titanic_data['fare'].fillna(titanic_data['fare'].median(), inplace=True)
```

In [16]:
```
features = ['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked']
X = titanic_data[features]
y = titanic_data['survived']
```

In [24]:
```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state
```

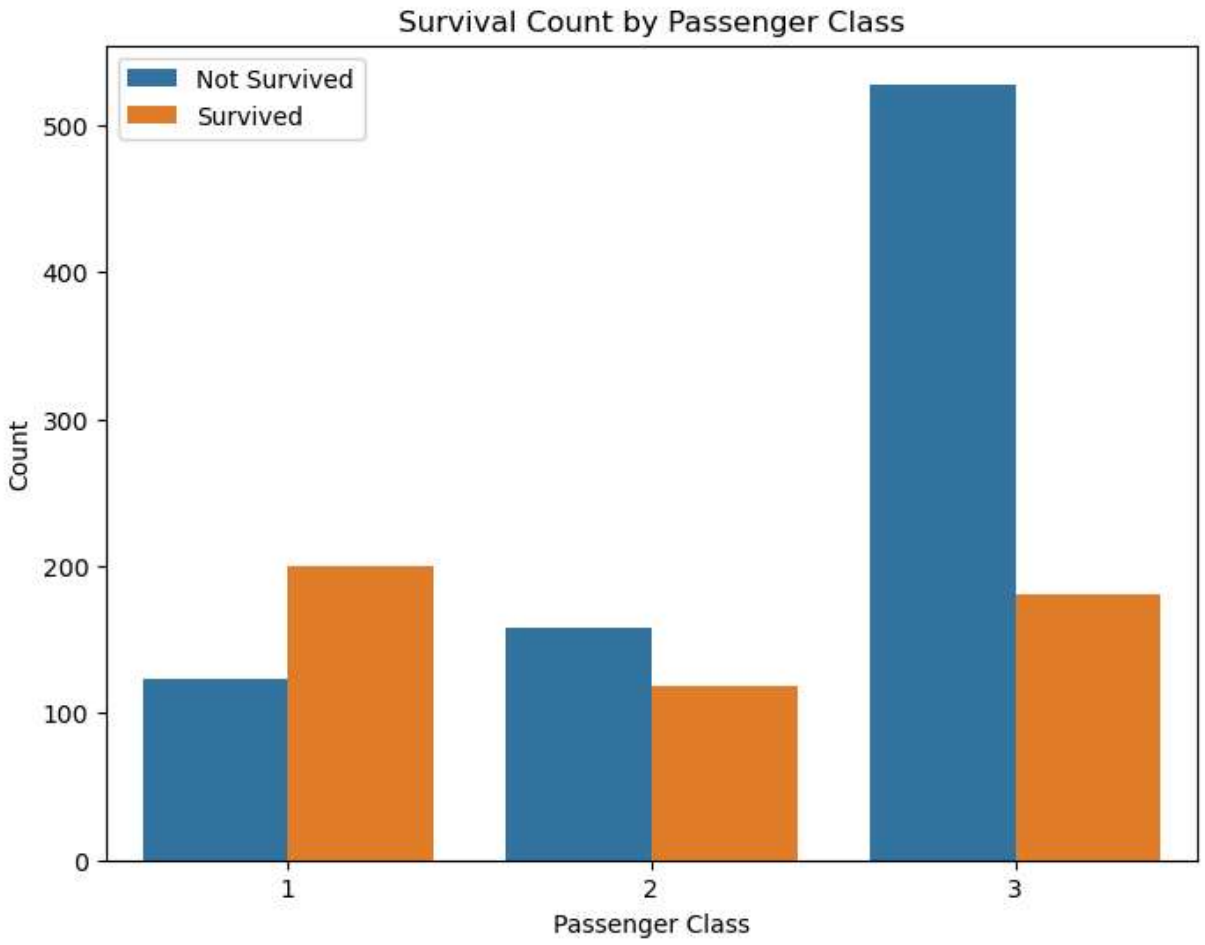
In [20]:
```
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[20]:
```
RandomForestClassifier(random_state=42)
```

In [22]:
```
y_pred = model.predict(X_val)
accuracy = accuracy_score(y_val, y_pred)
print(f'Validation Accuracy: {accuracy}')
```

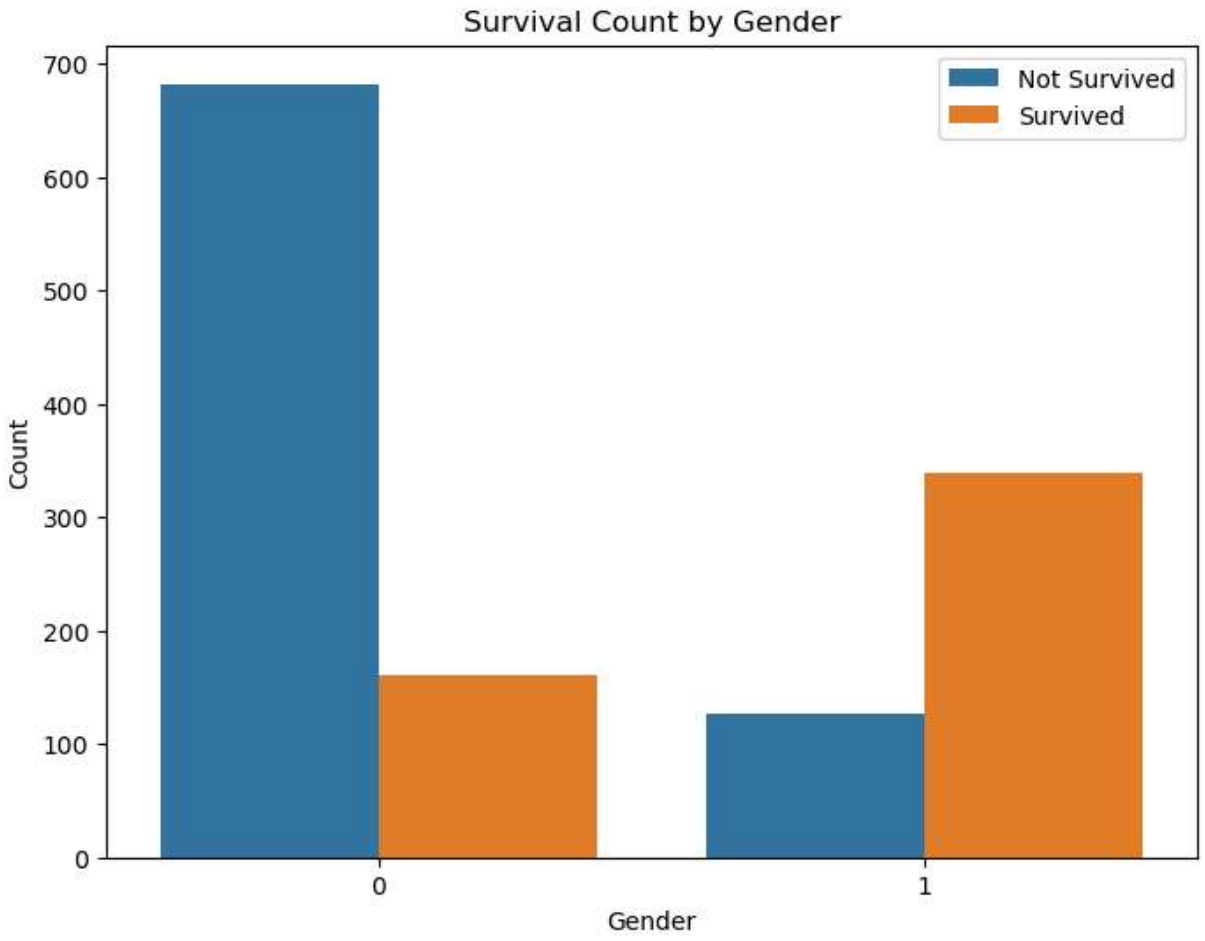
```
Validation Accuracy: 0.7900763358778626
```

In [28]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='pclass', hue='survived', data=titanic_data)
plt.title('Survival Count by Passenger Class')
plt.xlabel('Passenger Class')
plt.ylabel('Count')
plt.legend(['Not Survived', 'Survived'])
plt.show()
```

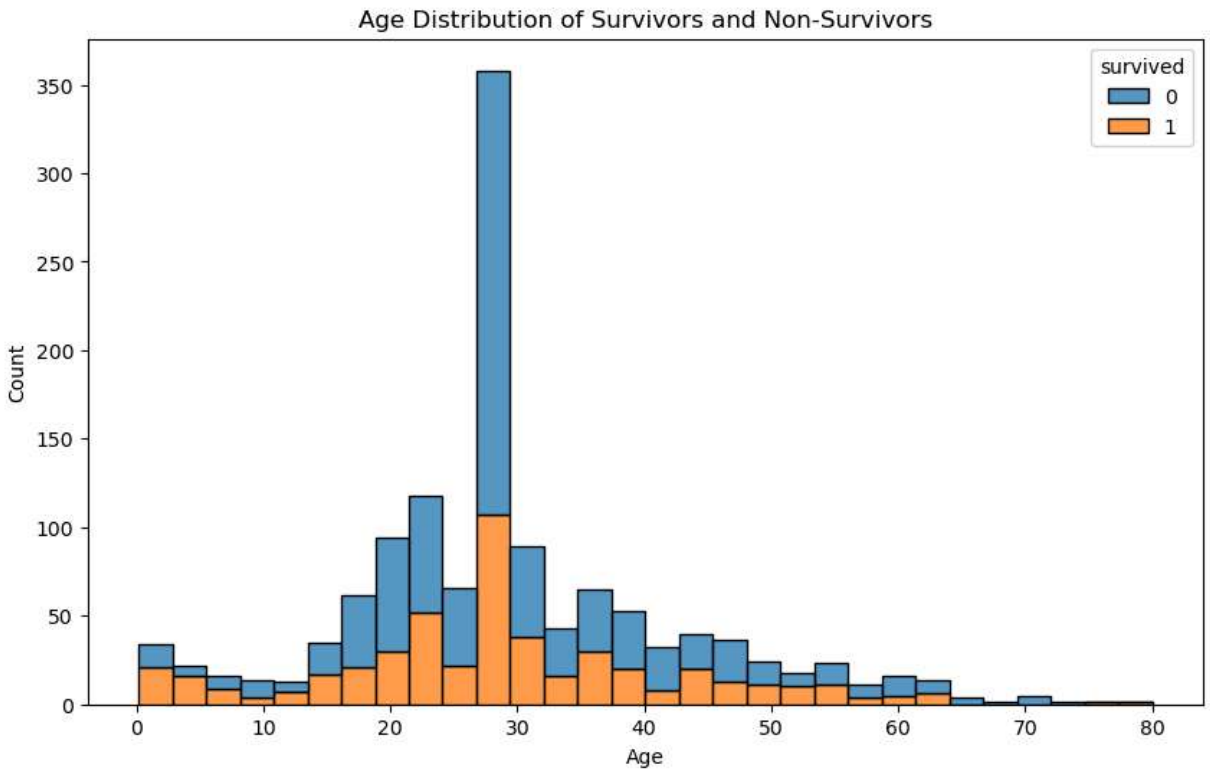


In [30]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='sex', hue='survived', data=titanic_data)
```

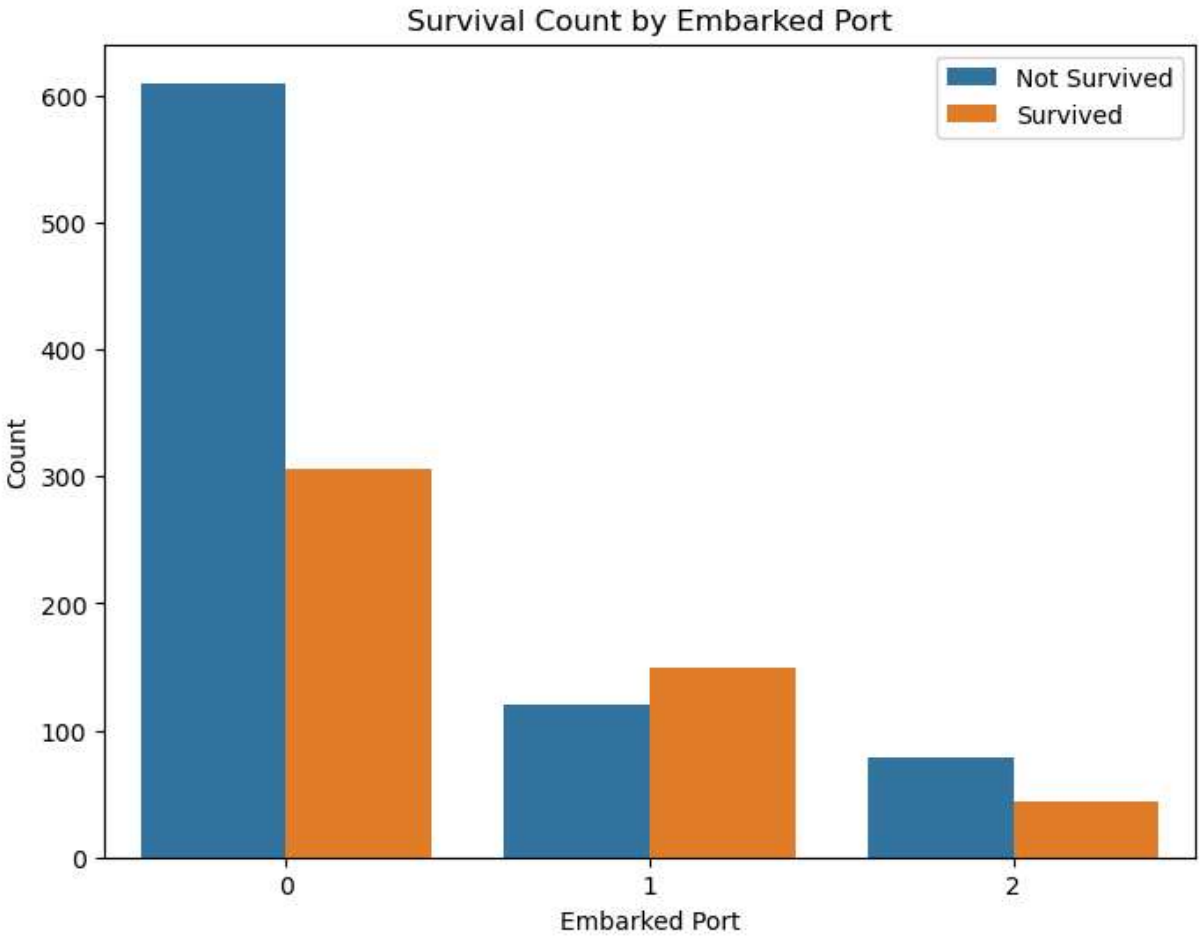
```
plt.title('Survival Count by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(['Not Survived', 'Survived'])
plt.show()
```



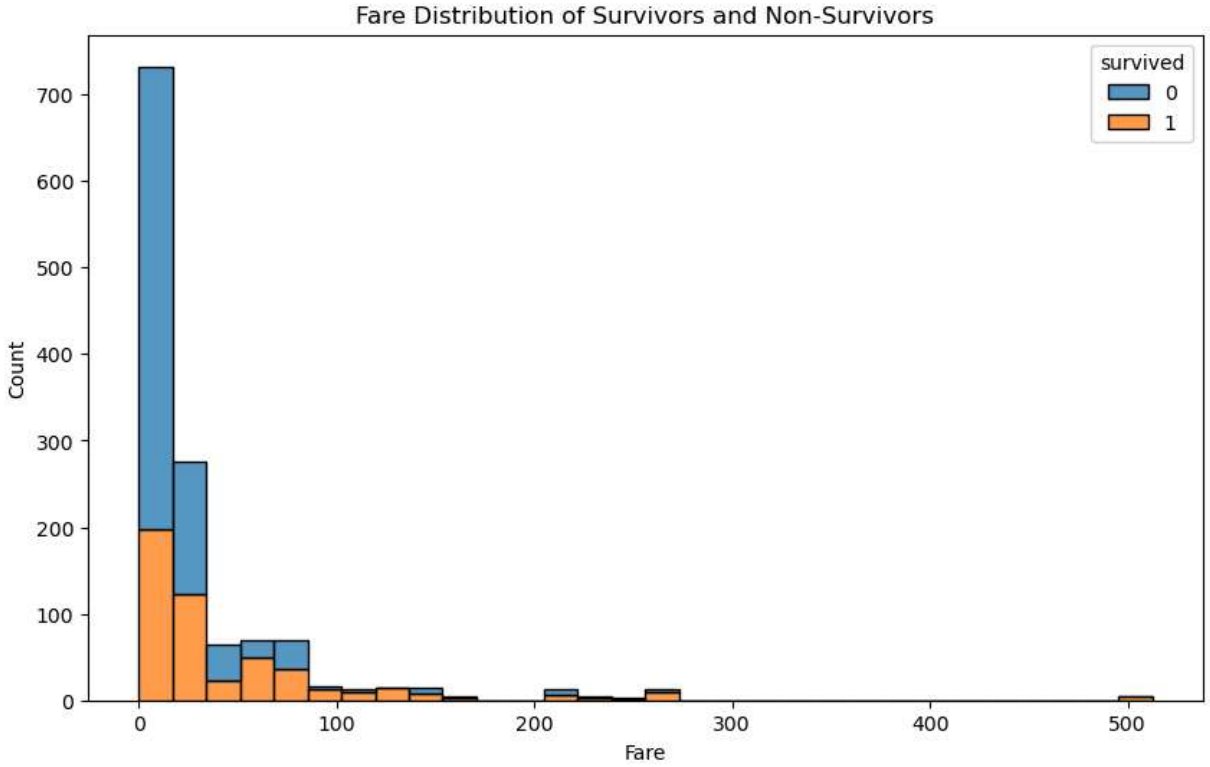
In [32]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data=titanic_data, x='age', hue='survived', multiple='stack', bins=30)
plt.title('Age Distribution of Survivors and Non-Survivors')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



In [34]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='embarked', hue='survived', data=titanic_data)
plt.title('Survival Count by Embarked Port')
plt.xlabel('Embarked Port')
plt.ylabel('Count')
plt.legend(['Not Survived', 'Survived'])
plt.show()
```



In [36]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data=titanic_data, x='fare', hue='survived', multiple='stack', bins=30
plt.title('Fare Distribution of Survivors and Non-Survivors')
plt.xlabel('Fare')
plt.ylabel('Count')
plt.show()
```


Fare Distribution of Survivors and Non-Survivors
survived
0
1
Count
Fare
0
100
200
300
400
500
0
100
200
300
400
500
600
700

In [ ]: