In [1]:
```
import numpy as np

import pandas as pd

import seaborn as sns

from matplotlib import pyplot as plt

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC
```

In [2]:
```
test_df = pd.read_csv("D:/resume projects/ML Titanic Survival Prediction/titanic/te
train_df = pd.read_csv("D:/resume projects/ML Titanic Survival Prediction/titanic/t

train_df.head()
```

Out[2]:

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

In [3]:
```
train_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 [4]:
```
train_df.describe()
```

Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [5]:
```
total = train_df.isnull().sum().sort_values(ascending=False)

percent_1 = train_df.isnull().sum()/train_df.isnull().count()*100
percent_2 = (round(percent_1, 1)).sort_values(ascending=False)

missing_data = pd.concat([total, percent_2], axis=1, keys=['Total', '%'])
missing_data.head(5)
```

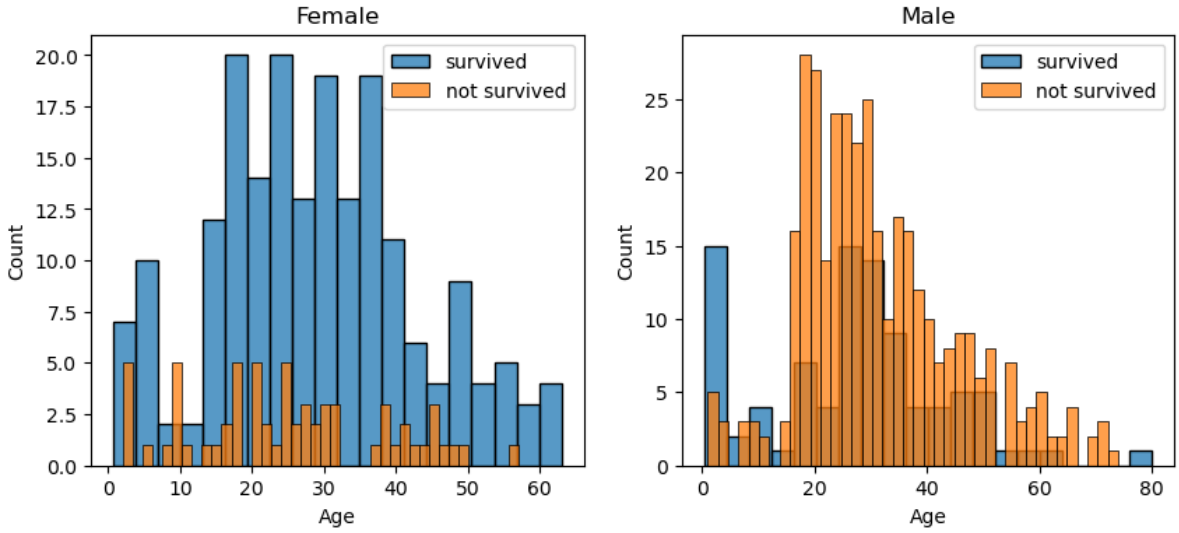
Out[5]:

	Total	%
Cabin	687	77.1
Age	177	19.9
Embarked	2	0.2
PassengerId	0	0.0
Survived	0	0.0

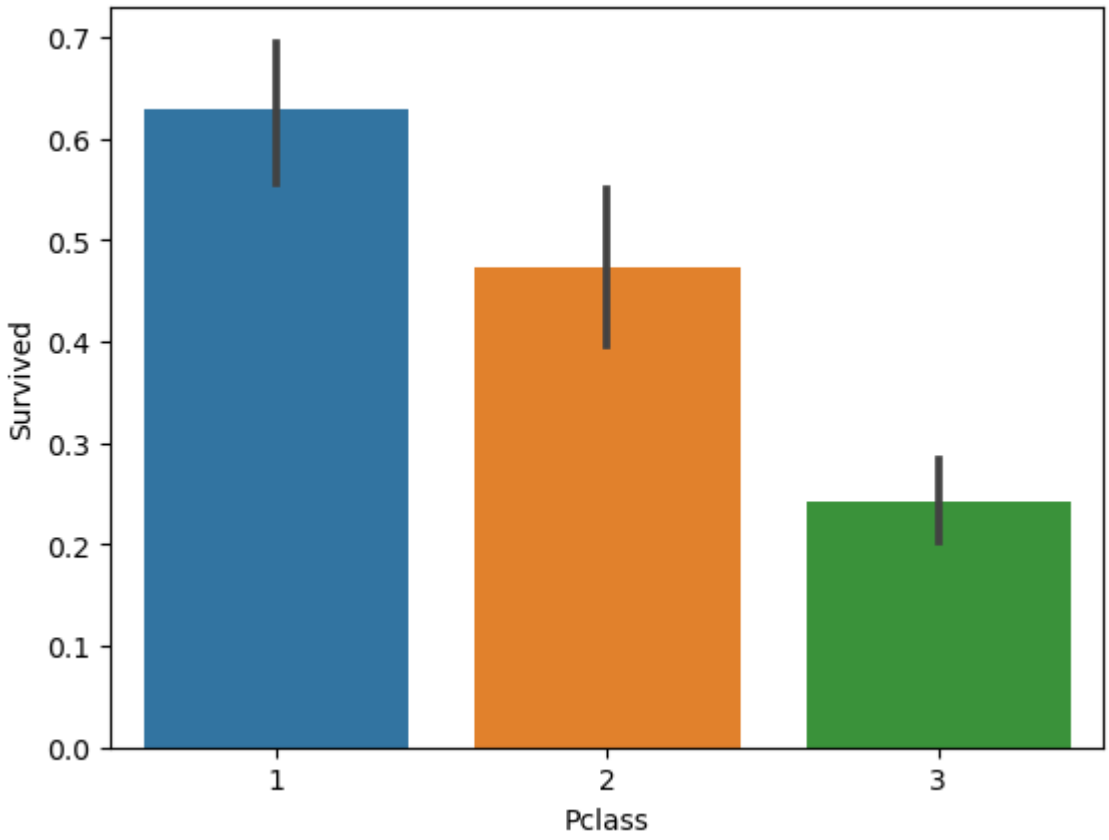
In [6]:
```
survived = 'survived'

not_survived = 'not survived'

fig, axes = plt.subplots (nrows=1, ncols=2, figsize=(10, 4))
```

```
women = train_df[train_df['Sex']=='female']

men = train_df[train_df['Sex']=='male']

ax = sns.histplot(women [women ['Survived']==1].Age.dropna(), bins=20, label= survi

ax = sns.histplot(women [women['Survived']==0].Age.dropna(), bins=40, label = not_s
ax.set_title('Female')

ax.legend()

ax = sns.histplot(men[men['Survived']==1].Age.dropna(), bins=20, label = survived,

ax = sns.histplot(men[men['Survived']==0].Age.dropna(), bins=40, label = not_surviv

ax.legend()

ax.set_title('Male')
```

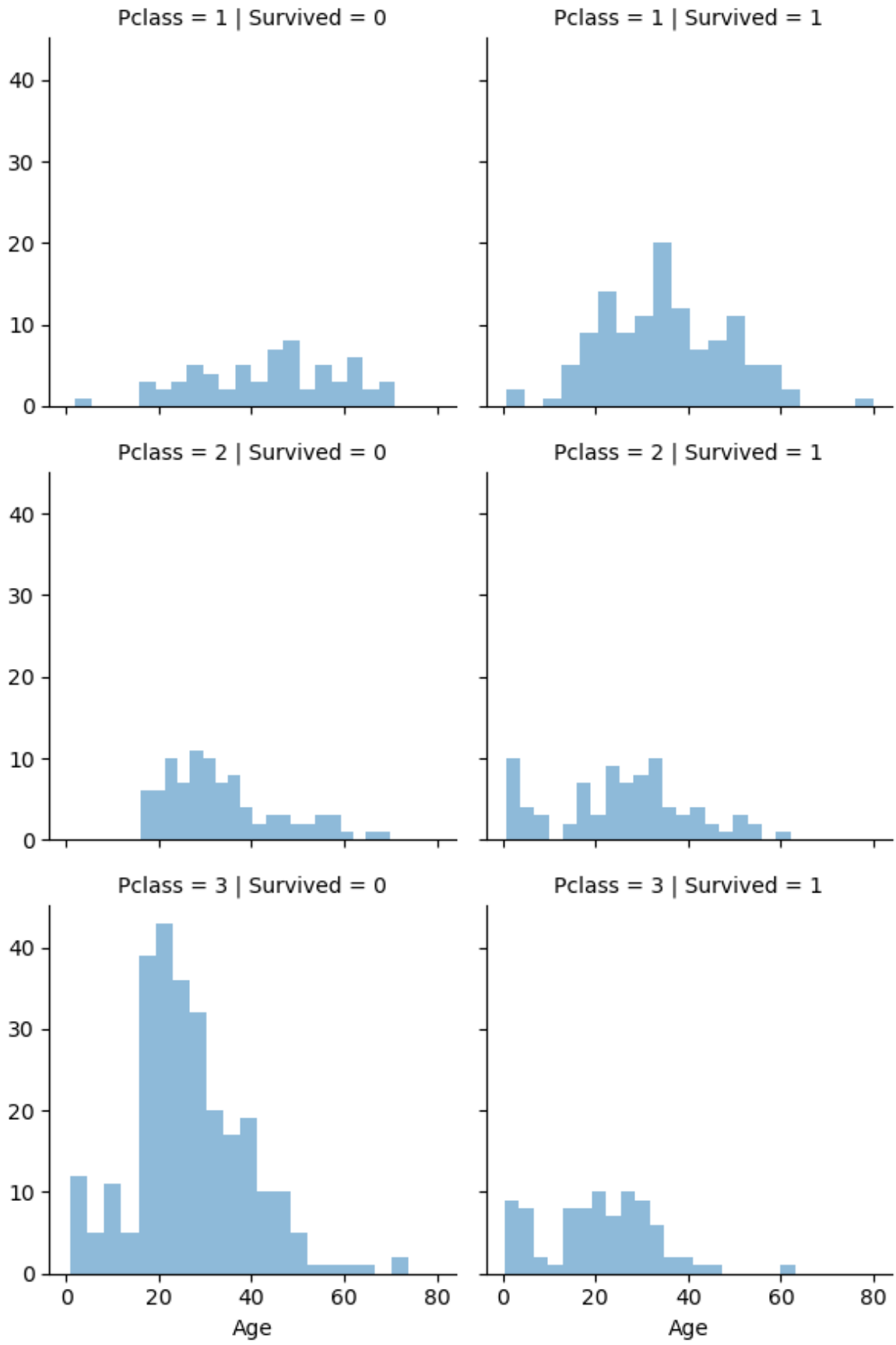
Out[6]:
```
Text(0.5, 1.0, 'Male')
```



In [7]:
```
sns.barplot(x='Pclass', y='Survived', data=train_df)
plt.show()
```



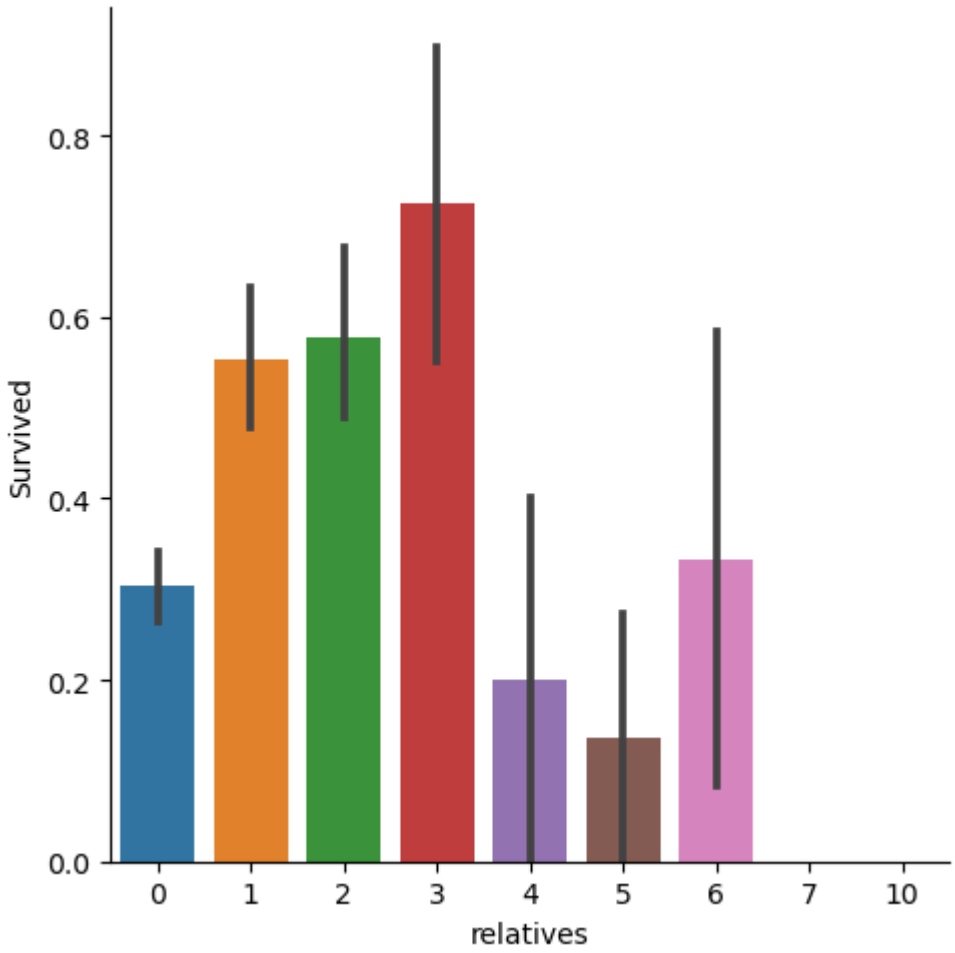
In [8]:
```
grid = sns.FacetGrid(train_df, col='Survived', row='Pclass')
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend();
```



In [9]:
```
data = [train_df, test_df]

for dataset in data:
    dataset['relatives'] = dataset['SibSp'] + dataset['Parch']
    dataset.loc[dataset['relatives'] > 0, 'not_alone'] = 0
    dataset.loc[dataset['relatives'] == 0, 'not_alone'] = 1
    dataset['not_alone'] = dataset['not_alone'].astype(int)

train_df['not_alone'].value_counts()
```

Out[9]:
```
1    537
0    354
Name: not_alone, dtype: int64
```

In [10]:
```
axes = sns.catplot(x='relatives', y='Survived', data=train_df, kind="bar")
```



In [11]:
```
train_df = train_df.drop(['PassengerId'], axis=1)
```

In [12]:
```
import re

deck = {"A": 1, "B": 2, "C": 3, "D": 4, "E": 5, "F": 6, "G": 7, "U": 8}

data = [train_df, test_df]

for dataset in data:

   dataset['Cabin'] = dataset['Cabin'].fillna("U0")
   dataset['Deck'] = dataset['Cabin'].map(lambda x: re.compile("([a-zA-Z]+)").searc
   dataset['Deck'] = dataset['Deck'].map(deck)
   dataset['Deck'] = dataset['Deck'].fillna(0)
   dataset['Deck'] = dataset['Deck'].astype(int)

train_df = train_df.drop(['Cabin'], axis=1)

test_df = test_df.drop(['Cabin'], axis=1)
```

In [13]:
```
data = [train_df, test_df]

for dataset in data:

   mean = train_df["Age"].mean()
   std = test_df["Age"].std()
```

```
   is_null = dataset["Age"].isnull().sum()
   rand_age = np.random.randint(mean- std, mean + std, size = is_null)
   age_slice = dataset["Age"].copy()
   age_slice[np.isnan(age_slice)] = rand_age
   dataset["Age"] = age_slice
   dataset["Age"] = train_df["Age"].astype(int)

train_df["Age"].isnull().sum()
```

Out[13]:
```
0
```

In [14]:
```
train_df['Embarked'].describe()
```

Out[14]:
```
count     889
unique      3
top         S
freq      644
Name: Embarked, dtype: object
```

In [15]:
```
common_value = 'S'

data = [train_df, test_df]

for dataset in data:
   dataset['Embarked'] = dataset['Embarked'].fillna(common_value)
```

In [16]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Survived   891 non-null    int64
 1   Pclass     891 non-null    int64
 2   Name       891 non-null    object
 3   Sex        891 non-null    object
 4   Age        891 non-null    int32
 5   SibSp      891 non-null    int64
 6   Parch      891 non-null    int64
 7   Ticket     891 non-null    object
 8   Fare       891 non-null    float64
 9   Embarked   891 non-null    object
 10  relatives  891 non-null    int64
 11  not_alone  891 non-null    int32
 12  Deck       891 non-null    int32
dtypes: float64(1), int32(3), int64(5), object(4)
memory usage: 80.2+ KB
```

In [17]:
```
data = [train_df, test_df]

for dataset in data:
   dataset['Fare'] = dataset['Fare'].fillna(0)
   dataset['Fare'] = dataset['Fare'].astype(int)
```

In [18]:
```
data = [train_df, test_df]

titles = {"r": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}

for dataset in data:
   dataset['Title'] = dataset.Name.str.extract(' ([A-Za-z]+)\.', expand=False)
   dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess', 'Capt', 'Col',
                                                'Rare')
```

```
   dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
   dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
   dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
   dataset['Title'] = dataset['Title'].map(titles)
   dataset['Title'] = dataset['Title'].fillna(0)
train_df=train_df.drop(['Name'], axis=1)

test_df = test_df.drop(['Name'], axis=1)
```

In [19]:
```
genders = {"male": 0, "female": 1}
data = [train_df, test_df]

for dataset in data:
   dataset['Sex'] = dataset['Sex'].map(genders)
```

In [20]:
```
train_df['Ticket'].describe()
```

Out[20]:
```
count        891
unique       681
top       347082
freq           7
Name: Ticket, dtype: object
```

In [21]:
```
train_df = train_df.drop(['Ticket'], axis=1)

test_df =test_df.drop(['Ticket'], axis=1)
```

In [22]:
```
ports = {"S": 0, "C": 1, "Q": 2}

data = [train_df, test_df]

for dataset in data:
   dataset['Embarked'] = dataset['Embarked'].map(ports)
```

In [23]:
```
data = [train_df, test_df]

for dataset in data:
   dataset['Age'] = dataset['Age'].astype(int)
   dataset.loc[ dataset['Age'] <= 11, 'Age'] = 0
   dataset.loc[(dataset['Age']> 11) & (dataset['Age'] <= 18), 'Age'] = 1
   dataset.loc[(dataset['Age']> 18) & (dataset['Age'] <= 22), 'Age'] = 2
   dataset.loc[(dataset['Age']> 22) & (dataset['Age'] <= 27), 'Age'] = 3
   dataset.loc[(dataset['Age']> 27) & (dataset['Age'] <= 33), 'Age'] = 4
   dataset.loc[(dataset['Age']> 33) & (dataset['Age'] <= 40), 'Age'] = 5
   dataset.loc[(dataset['Age']> 40) & (dataset['Age'] <= 66), 'Age'] = 6
   dataset.loc[ dataset['Age']> 66, 'Age'] = 6
train_df['Age'].value_counts()
```

Out[23]:
```
5    164
4    163
6    156
3    143
2    110
1     87
0     68
Name: Age, dtype: int64
```

In [24]:
```
data = [train_df, test_df]

for dataset in data:
   dataset.loc[ dataset ['Fare'] <= 7.91, 'Fare'] = 0
   dataset.loc[(dataset['Fare'] > 7.91) & (dataset['Fare'] <= 14.454), 'Fare'] = 1
   dataset.loc[(dataset['Fare'] > 14.454) & (dataset['Fare'] <= 31), 'Fare'] = 2
```

```
    dataset.loc[(dataset['Fare'] > 31) & (dataset['Fare'] <= 99), 'Fare'] = 3
    dataset.loc[(dataset['Fare'] > 99) & (dataset['Fare'] <= 250), 'Fare'] = 4
    dataset.loc[ dataset['Fare'] > 250, 'Fare'] = 5
    dataset['Fare'] = dataset['Fare'].astype(int)
```

In [25]:
```
data = [train_df, test_df]

for dataset in data:
  dataset['Age_Class']= dataset['Age']* dataset['Pclass']
```

In [26]:
```
for dataset in data:
  dataset['Fare_Per_Person'] = dataset['Fare']/(dataset['relatives']+1)
  dataset['Fare_Per_Person'] = dataset['Fare_Per_Person'].astype(int)

train_df.head()
```

Out[26]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	relatives	not_alone	Deck	Title	A
0	0	3	0	2	1	0	0	0	1	0	8	0.0	
1	1	1	1	5	1	0	3	1	1	0	3	3.0	
2	1	3	1	3	0	0	0	0	0	1	8	2.0	
3	1	1	1	5	1	0	3	0	1	0	3	3.0	
4	0	3	0	5	0	0	1	0	0	1	8	0.0	

In [27]:
```
X_train = train_df.drop("Survived", axis=1)
Y_train = train_df["Survived"]
X_test = test_df.drop("PassengerId", axis=1).copy()

random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)

Y_prediction = random_forest.predict(X_test)

random_forest.score(X_train, Y_train)

acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)
```

In [28]:
```
logreg =LogisticRegression(max_iter=5000)
logreg.fit(X_train, Y_train)
Y_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)

linear_svc = SVC()

linear_svc.fit(X_train, Y_train)

Y_pred =linear_svc.predict(X_test)

acc_linear_svc = round(linear_svc.score(X_train, Y_train) * 100, 2)

results = pd.DataFrame({
     'Model': ['Support Vector Machines', 'Logistic Regression', 'Random Forest'],
    'Score': [acc_linear_svc, acc_log, acc_random_forest]

})

result_df = results.sort_values(by='Score', ascending=False)
```

```
result_df = result_df.set_index('Score')

result_df
```

Out[28]:

Score	Model
92.70	Random Forest
81.71	Support Vector Machines
81.48	Logistic Regression

In [ ]: