```
from google.colab import drive
drive.mount('/content/drive')
"""AIzaSyC43MtwN679nqybolxH4WoDd7EKUfcWNgc"""
```

```
Mounted at /content/drive
'AIzaSyC43MtwN679nqybolxH4WoDd7EKUfcWNgc'
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,ConfusionMatrixDisplay,classification_report
from prettytable import PrettyTable
import re
rcParams['figure.figsize'] = 10,8
```

```
import warnings
warnings.filterwarnings("ignore", message="use_inf_as_na option is deprecated and will be removed in a future version")
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
df_train = pd.read_csv('/content/drive/MyDrive/Titanic ML/titanic/train.csv')
```

```
df_test = pd.read_csv('/content/drive/MyDrive/Titanic ML/titanic/test.csv')
```

```
print("train data______________")
print(df_train.info())
```

```
train data______________
<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
None
```

```
print("----------test data----------")
print(df_test.info())
```

```
----------test data----------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 36.1+ KB
None
```

```
def clean_data(df):
    df['Title'] = df['Name'].str.extract(r',\s*([^\.]+)\.', expand=False)
    df = df.drop(columns=['PassengerId', 'Name', 'Cabin', 'Ticket'], errors='ignore')

    df['FamilySize'] = df['SibSp'] + df['Parch'] + 1
    df['IsAlone'] = (df['FamilySize'] == 1).astype(int)

    df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
    df['Fare'] = df['Fare'].fillna(df['Fare'].median())
    df['Age'] = df['Age'].fillna(df['Age'].median())

    df['FareBin'] = pd.qcut(df['Fare'], 4, labels=False)
    df['AgeBin'] = pd.cut(df['Age'].astype(int), 5, labels=False)

    stat_min = 10
    title_counts = df['Title'].value_counts()
    df['Title'] = df['Title'].apply(lambda x: 'Misc' if title_counts.get(x, 0) < stat_min else x)

    return df

df_train = clean_data(df_train)
df_test = clean_data(df_test)

print("----------Train Data----------")
print(df_train.info())
print("\n\n----------Test Data----------")
print(df_test.info())
```

```
----------Train Data----------
<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   Sex         891 non-null    object 
 3   Age         891 non-null    float64
 4   SibSp       891 non-null    int64  
 5   Parch       891 non-null    int64  
 6   Fare        891 non-null    float64
 7   Embarked    891 non-null    object 
 8   Title       891 non-null    object 
 9   FamilySize  891 non-null    int64  
 10  IsAlone     891 non-null    int64  
 11  FareBin     891 non-null    int64  
 12  AgeBin      891 non-null    int64  
dtypes: float64(2), int64(8), object(3)
memory usage: 90.6+ KB
None


----------Test Data----------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Pclass      418 non-null    int64  
 1   Sex         418 non-null    object 
 2   Age         418 non-null    float64
 3   SibSp       418 non-null    int64  
 4   Parch       418 non-null    int64  
 5   Fare        418 non-null    float64
 6   Embarked    418 non-null    object 
 7   Title       418 non-null    object 
 8   FamilySize  418 non-null    int64  
 9   IsAlone     418 non-null    int64  
 10  FareBin     418 non-null    int64  
 11  AgeBin      418 non-null    int64  
dtypes: float64(2), int64(7), object(3)
memory usage: 39.3+ KB
None
```

```
df_train.head()
```

```
   Survived  Pclass     Sex   Age  SibSp  Parch     Fare Embarked Title  FamilySize  IsAlone  FareBin  AgeBin
0         0       3    male  22.0      1      0   7.2500        S    Mr           2        0        0       1
1         1       1  female  38.0      1      0  71.2833        C   Mrs           2        0        3       2
2         1       3  female  26.0      0      0   7.9250        S  Miss           1        1        1       1
3         1       1  female  35.0      1      0  53.1000        S   Mrs           2        0        3       2
4         0       3    male  35.0      0      0   8.0500        S    Mr           1        1        1       2
```

```
df_train.tail()
```

```
     Survived  Pclass     Sex   Age  SibSp  Parch   Fare Embarked Title  FamilySize  IsAlone  FareBin  AgeBin
886         0       2    male  27.0      0      0  13.00        S  Misc           1        1        1       1
887         1       1  female  19.0      0      0  30.00        S  Miss           1        1        2       1
888         0       3  female  28.0      1      2  23.45        S  Miss           4        0        2       1
889         1       1    male  26.0      0      0  30.00        C    Mr           1        1        2       1
890         0       3    male  32.0      0      0   7.75        Q    Mr           1        1        0       1
```

```
df_test.head()
```

```
   Pclass     Sex   Age  SibSp  Parch     Fare Embarked Title  FamilySize  IsAlone  FareBin  AgeBin
0       3    male  34.5      0      0   7.8292        Q    Mr           1        1        0       2
1       3  female  47.0      1      0   7.0000        S   Mrs           2        0        0       3
2       2    male  62.0      0      0   9.6875        Q    Mr           1        1        1       4
3       3    male  27.0      0      0   8.6625        S    Mr           1        1        1       1
4       3  female  22.0      1      1  12.2875        S   Mrs           3        0        1       1
```

```
df_test.tail()
```

```
     Pclass     Sex   Age  SibSp  Parch      Fare Embarked   Title  FamilySize  IsAlone  FareBin  AgeBin
413       3    male  27.0      0      0    8.0500        S      Mr           1        1        1       1
414       1  female  39.0      0      0  108.9000        C    Misc           1        1        3       2
415       3    male  38.5      0      0    7.2500        S      Mr           1        1        0       2
416       3    male  27.0      0      0    8.0500        S      Mr           1        1        1       1
417       3    male  27.0      1      1   22.3583        C  Master           3        0        2       1
```

```
df_train.describe()
```

```
         Survived      Pclass         Age       SibSp       Parch        Fare  FamilySize     IsAlone     FareBin      AgeBin
count  891.000000  891.000000  891.000000  891.000000  891.000000  891.000000  891.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.361582    0.523008    0.381594   32.204208    1.904602    0.602694    1.497194    1.288440
std      0.486592    0.836071   13.019697    1.102743    0.806057   49.693429    1.613459    0.489615    1.118156    0.812038
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000    1.000000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   22.000000    0.000000    0.000000    7.910400    1.000000    0.000000    0.500000    1.000000
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200    1.000000    1.000000    1.000000    1.000000
75%      1.000000    3.000000   35.000000    1.000000    0.000000   31.000000    2.000000    1.000000    2.000000    2.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200   11.000000    1.000000    3.000000    4.000000
```

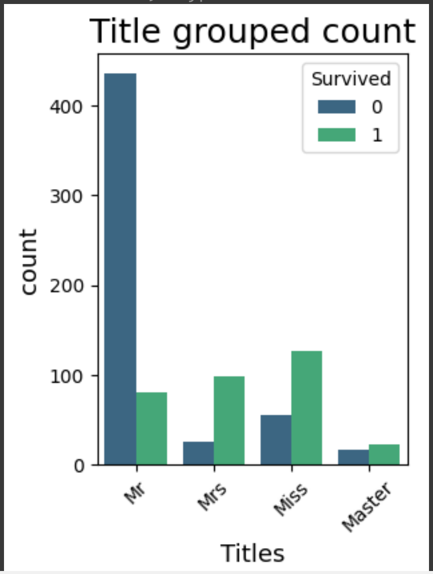
```
df_test.describe()
```

	Pclass	Age	SibSp	Parch	Fare	FamilySize	IsAlone	FareBin	AgeBin
count	418.000000	418.000000	418.000000	418.000000	418.000000	418.000000	418.000000	418.000000	418.000000
mean	2.265550	29.599282	0.447368	0.392344	35.576535	1.839713	0.605263	1.473684	1.387560
std	0.841838	12.703770	0.896760	0.981429	55.850103	1.519072	0.489380	1.140292	0.858328
min	1.000000	0.170000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	1.000000	23.000000	0.000000	0.000000	7.895800	1.000000	0.000000	0.000000	1.000000
50%	3.000000	27.000000	0.000000	0.000000	14.454200	1.000000	1.000000	1.000000	1.000000
75%	3.000000	35.750000	1.000000	0.000000	31.471875	2.000000	1.000000	2.750000	2.000000
max	3.000000	76.000000	8.000000	9.000000	512.329200	11.000000	1.000000	3.000000	4.000000

```
Title_Dictionary = {
        "Capt":       "Officer",
        "Col":        "Officer",
        "Major":      "Officer",
        "Dr":         "Officer",
        "Rev":        "Officer",
        "Jonkheer":   "Royalty",
        "Don":        "Royalty",
        "Sir" :       "Royalty",
        "the Countess":"Royalty",
        "Dona":       "Royalty",
        "Lady" :      "Royalty",
        "Mme":        "Mrs",
        "Ms":         "Mrs",
        "Mrs" :       "Mrs",
        "Mlle":       "Miss",
        "Miss" :      "Miss",
        "Mr" :        "Mr",
        "Master" :    "Master"
                   }

df_train['Title'] = df_train.Title.map(Title_Dictionary)
df_test['Title'] = df_test.Title.map(Title_Dictionary)
```

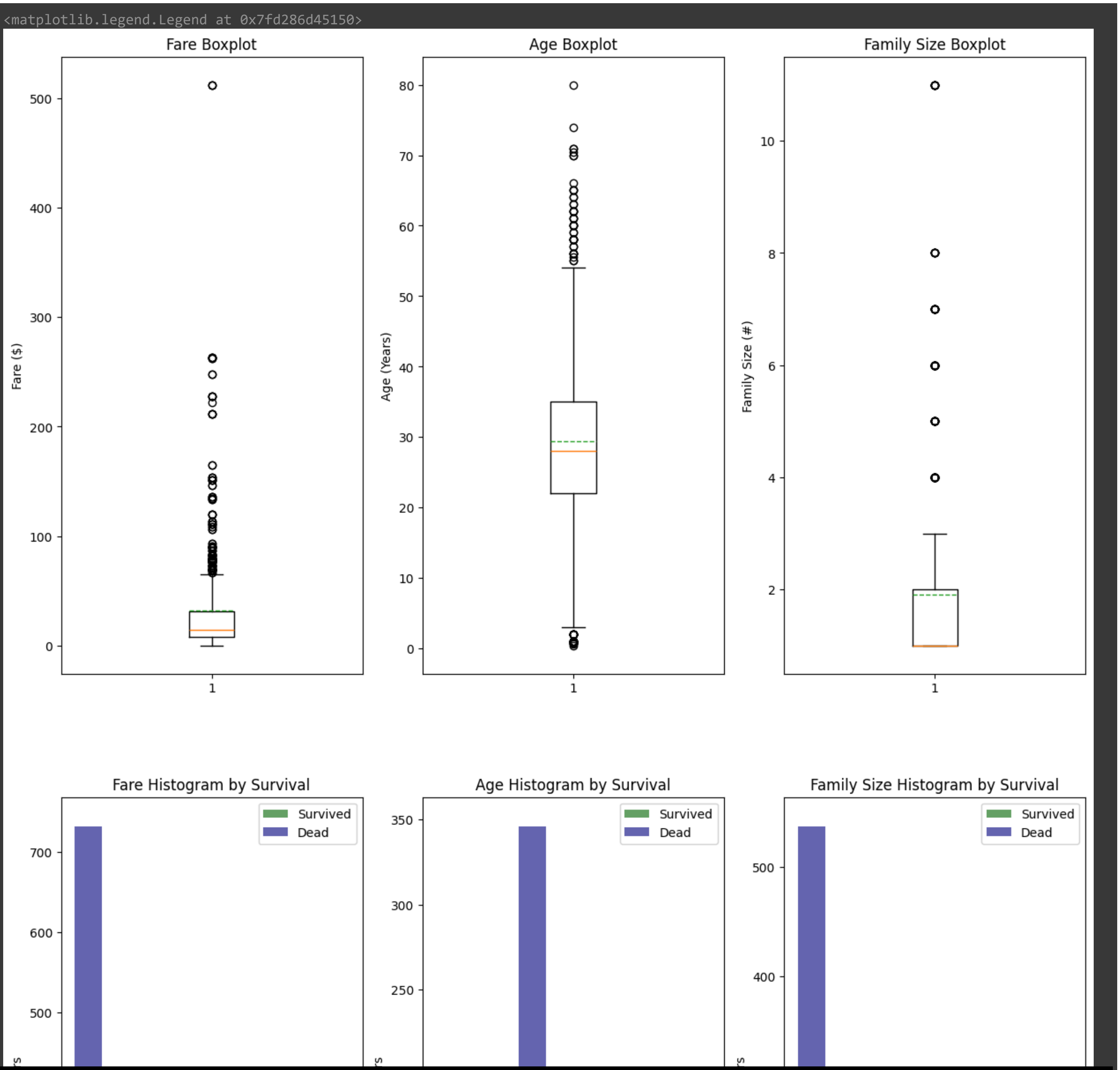
```
print("changes to survival based on titles:")
print(df_train.groupby("Title")["Survived"].mean())

plt.figure(figsize = (3,4))
sns.countplot(x='Title', data=df_train, palette="viridis",
              hue="Survived")
plt.xlabel("Titles",fontsize = 13)
plt.ylabel("count",fontsize = 13)
plt.title("Title grouped count",fontsize = 18)
plt.xticks(rotation=45)
plt.show()
```

```
changes to survival based on titles:
Title
Master    0.575000
Miss      0.697802
Mr        0.156673
Mrs       0.792000
Name: Survived, dtype: float64
```



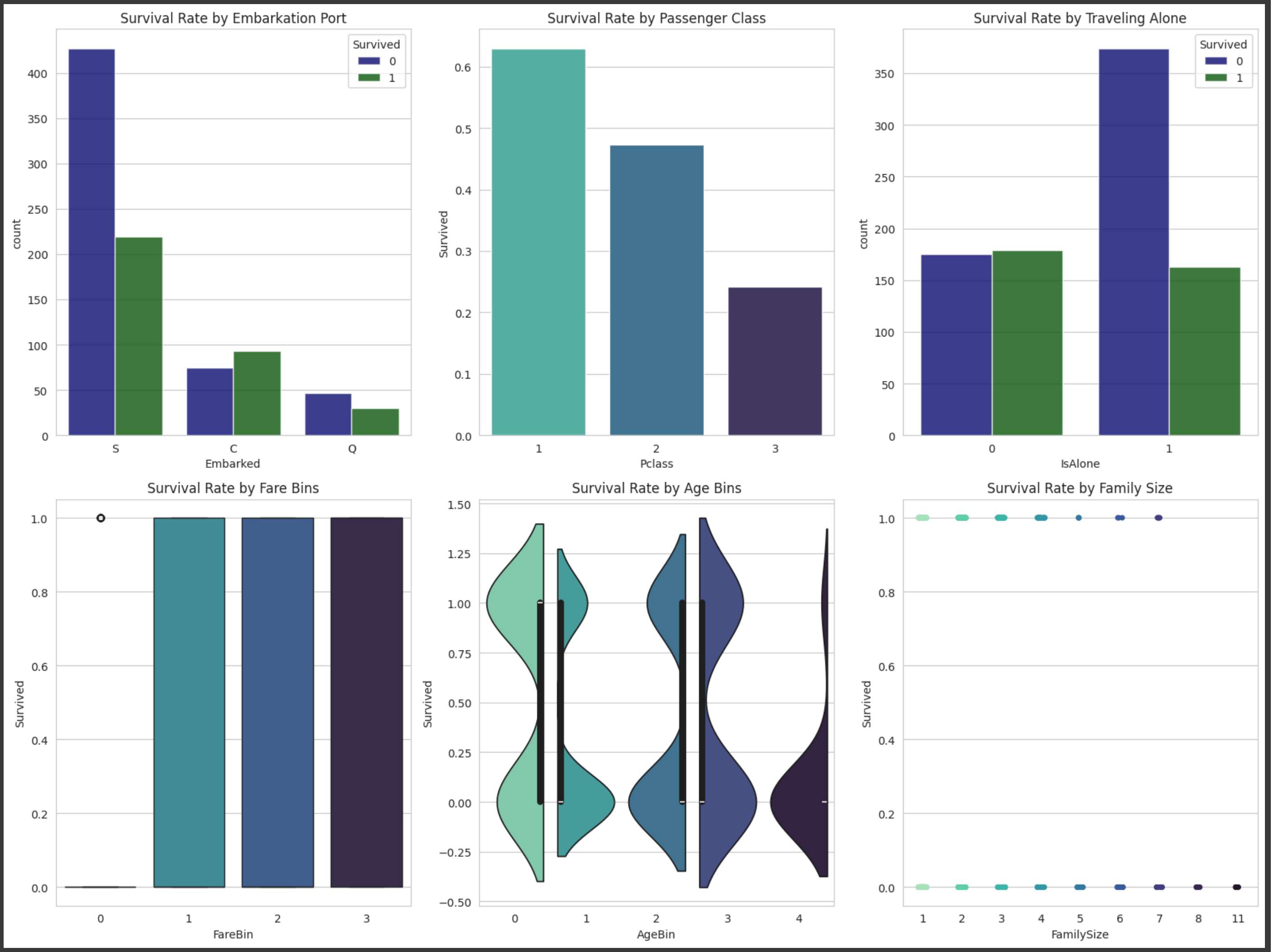
```
plt.figure(figsize=[15,20])

plt.subplot(231)
plt.boxplot(x=df_train['Fare'], showmeans = True, meanline = True)
plt.title('Fare Boxplot')
plt.ylabel('Fare ($)')

plt.subplot(232)
plt.boxplot(df_train['Age'], showmeans = True, meanline = True)
plt.title('Age Boxplot')
plt.ylabel('Age (Years)')

plt.subplot(233)
plt.boxplot(df_train['FamilySize'], showmeans = True, meanline = True)
plt.title('Family Size Boxplot')
plt.ylabel('Family Size (#)')

plt.subplot(234)
plt.hist(x = [df_train[df_train['Survived']==1]['Fare'], df_train[df_train['Survived']==0]['Fare']],
         stacked=True, color = ['darkgreen','navy'],label = ['Survived','Dead'], alpha=0.6)
plt.title('Fare Histogram by Survival')
plt.xlabel('Fare ($)')
plt.ylabel('# of Passengers')
plt.legend()

plt.subplot(235)
plt.hist(x = [df_train[df_train['Survived']==1]['Age'], df_train[df_train['Survived']==0]['Age']],
         stacked=True, color = ['darkgreen','navy'],label = ['Survived','Dead'], alpha=0.6)
plt.title('Age Histogram by Survival')
plt.xlabel('Age (Years)')
plt.ylabel('# of Passengers')
plt.legend()

plt.subplot(236)
plt.hist(x = [df_train[df_train['Survived']==1]['FamilySize'], df_train[df_train['Survived']==0]['FamilySize']],
         stacked=True, color = ['darkgreen','navy'],label = ['Survived','Dead'], alpha=0.6)
plt.title('Family Size Histogram by Survival')
plt.xlabel('Family Size (#)')
plt.ylabel('# of Passengers')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fd286d45150>
```

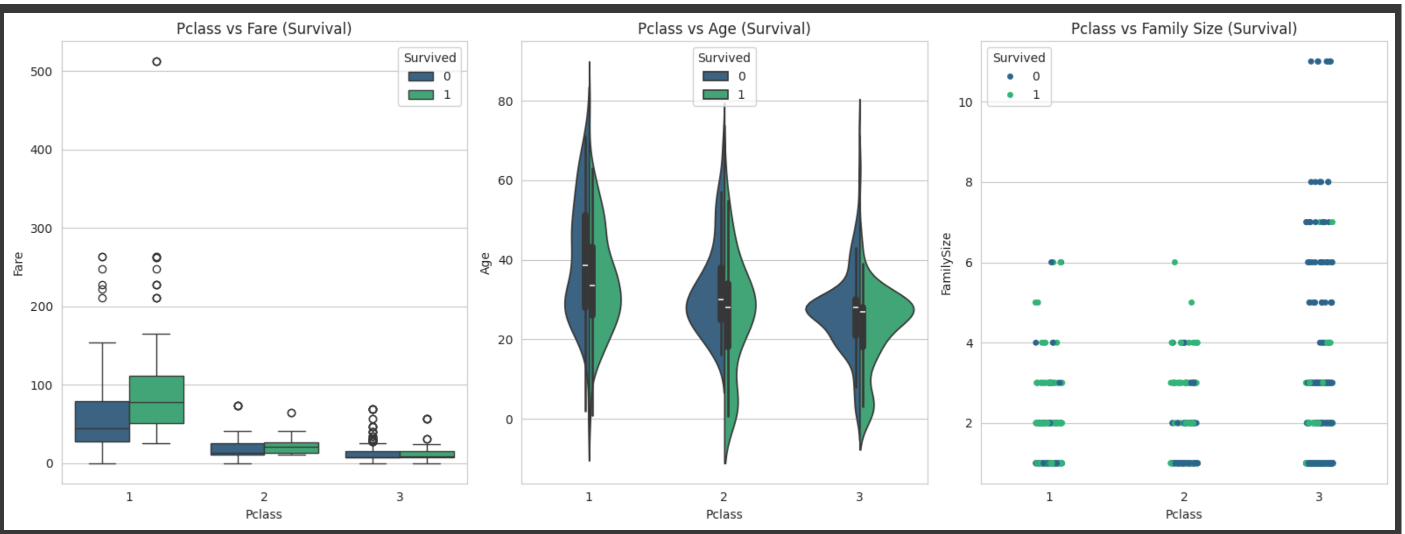


```
sns.set_style("whitegrid")
sns.set_palette("muted")


fig, axes = plt.subplots(2, 3, figsize=(16, 12))

sns.countplot(x='Embarked', hue='Survived', data=df_train, ax=axes[0, 0], palette={0: "navy", 1: "darkgreen"}, alpha=0.8)
axes[0, 0].set_title('Survival Rate by Embarkation Port')

sns.barplot(x='Pclass', y='Survived', order=[1, 2, 3], data=df_train, ax=axes[0, 1], ci=None, palette="mako_r")
axes[0, 1].set_title('Survival Rate by Passenger Class')

sns.countplot(x='IsAlone', hue='Survived', data=df_train, ax=axes[0, 2], palette={0: "navy", 1: "darkgreen"}, alpha=0.8)
axes[0, 2].set_title('Survival Rate by Traveling Alone')

sns.boxplot(x='FareBin', y='Survived', data=df_train, ax=axes[1, 0], palette="mako_r")
axes[1, 0].set_title('Survival Rate by Fare Bins')

sns.violinplot(x='AgeBin', y='Survived', data=df_train, ax=axes[1, 1], split=True, palette="mako_r")
axes[1, 1].set_title('Survival Rate by Age Bins')

sns.stripplot(x='FamilySize', y='Survived', data=df_train, ax=axes[1, 2], jitter=True, palette="mako_r")
axes[1, 2].set_title('Survival Rate by Family Size')
```

```
plt.tight_layout()
plt.show()
```

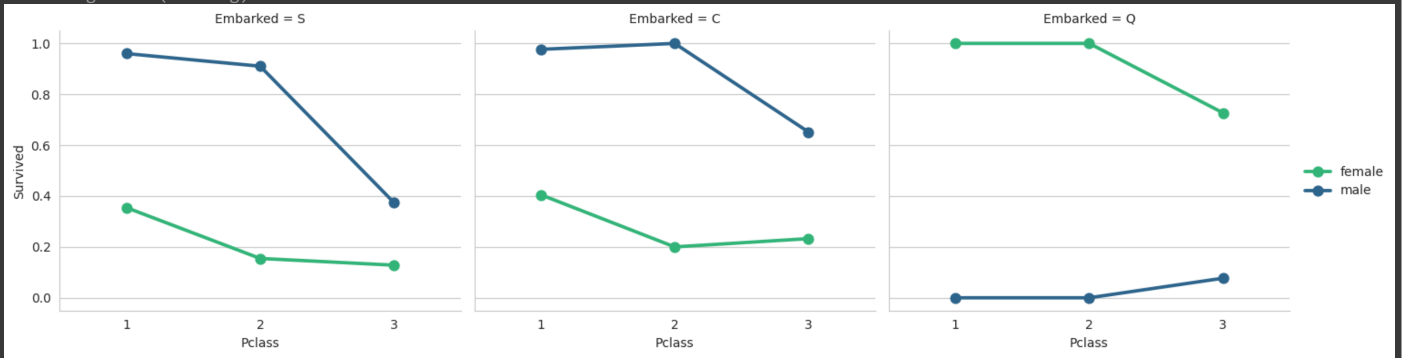


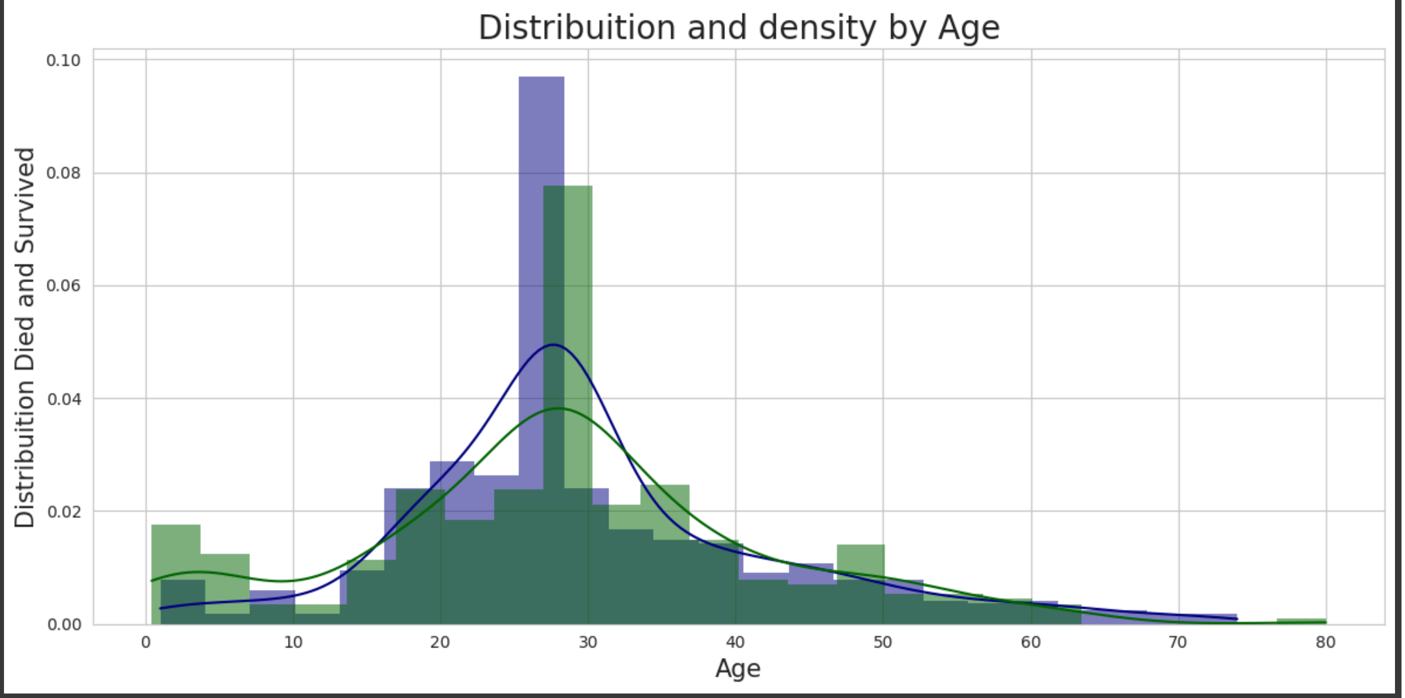
```
fig, axes = plt.subplots(1, 3, figsize=(16, 6))

sns.boxplot(x='Pclass', y='Fare', hue='Survived', data=df_train, ax=axes[0], palette="viridis")
axes[0].set_title('Pclass vs Fare (Survival)')

sns.violinplot(x='Pclass', y='Age', hue='Survived', data=df_train, split=True, ax=axes[1], palette="viridis")
axes[1].set_title('Pclass vs Age (Survival)')

sns.stripplot(x='Pclass', y='FamilySize', hue='Survived', data=df_train, jitter=True, ax=axes[2], palette="viridis")
axes[2].set_title('Pclass vs Family Size (Survival)')

plt.tight_layout()
plt.show()
```



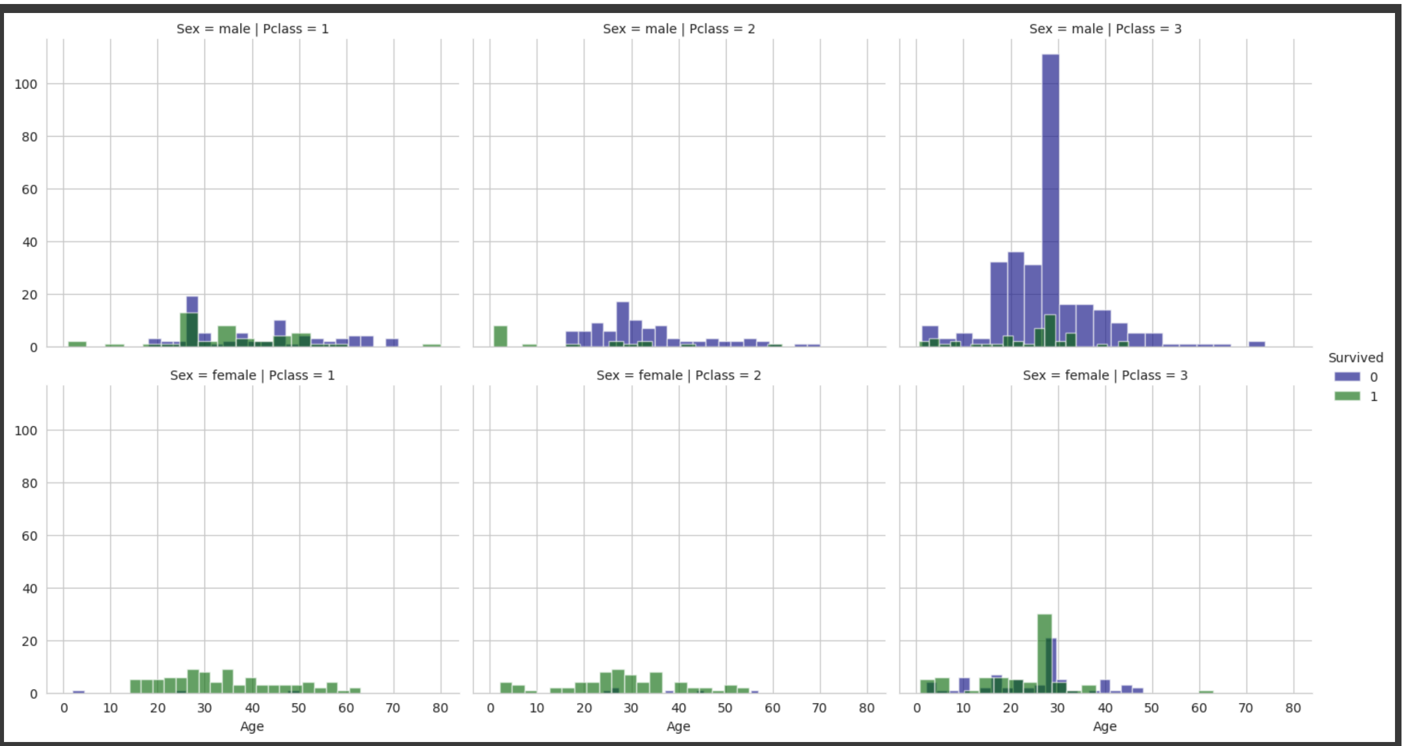
```
embark_grid = sns.FacetGrid(df_train, col='Embarked', height=4, aspect=1.2)
embark_grid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', ci=None, palette="viridis")
embark_grid.add_legend()

fig, ax = plt.subplots(figsize=(12, 6))
age_high_zero_died = df_train[(df_train["Age"] > 0) &
                              (df_train["Survived"] == 0)]
age_high_zero_surv = df_train[(df_train["Age"] > 0) &
                              (df_train["Survived"] == 1)]

sns.histplot(age_high_zero_died["Age"], color="navy", bins=24, kde=True, stat="density", linewidth=0)
sns.histplot(age_high_zero_surv["Age"], color="darkgreen", bins=24, kde=True, stat="density", linewidth=0)
plt.title("Distribuition and density by Age",fontsize=20)
plt.xlabel("Age",fontsize=15)
plt.ylabel("Distribuition Died and Survived",fontsize=15)
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py:718: UserWarning: Using the pointplot function without specifying `order` is
  warnings.warn(warning)
/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py:723: UserWarning: Using the pointplot function without specifying `hue_order
  warnings.warn(warning)
```





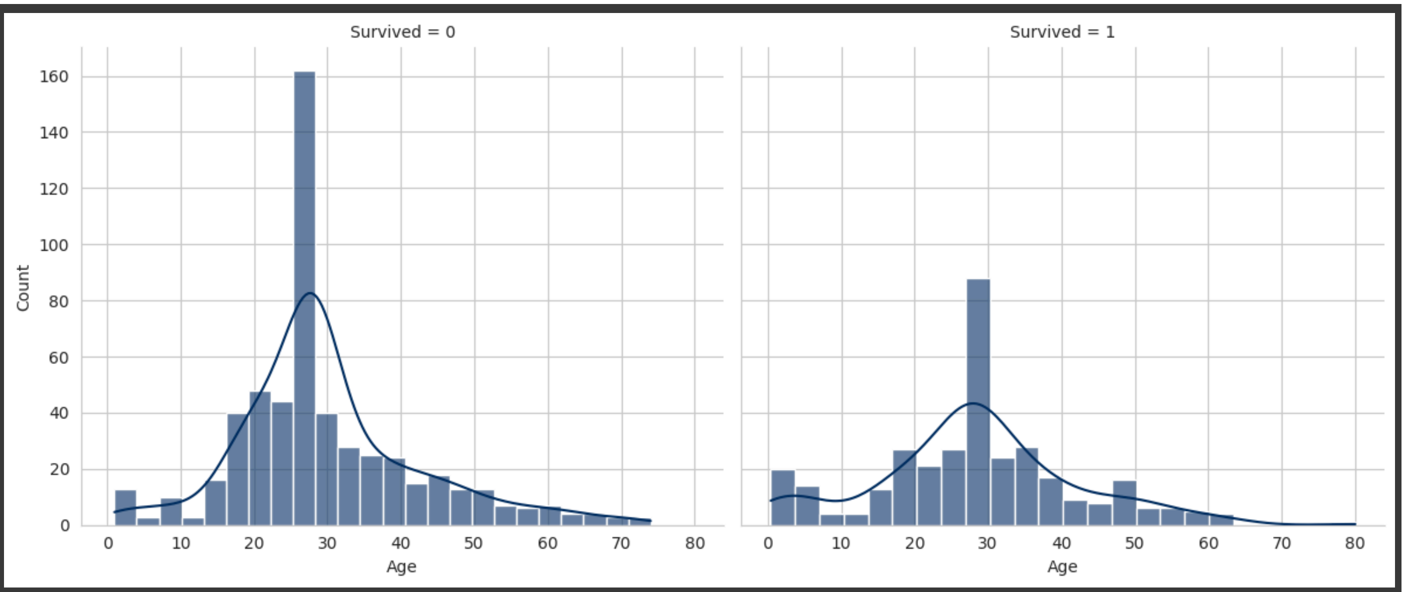
```
hist_grid = sns.FacetGrid(df_train, row='Sex', col='Pclass', hue='Survived', height=4, aspect=1.2, palette={0: "navy", 1: "darkgreen"})
hist_grid.map(plt.hist, 'Age', alpha=0.6, bins=20)
hist_grid.add_legend()
plt.show()
```



```
age_group = df_train.groupby(['Sex','Pclass','Title'])['Age']
print(age_group.median())
```

```
Sex     Pclass  Title 
female  1       Miss      30.0
                Mrs       38.5
        2       Miss      24.0
                Mrs       32.0
        3       Miss      22.0
                Mrs       29.0
male    1       Master     4.0
                Mr        36.0
        2       Master     1.0
                Mr        30.0
        3       Master     6.5
                Mr        28.0
Name: Age, dtype: float64
```

```
g = sns.FacetGrid(df_train, col="Survived", height=5, aspect=1.2)
g.map_dataframe(sns.histplot, x="Age", bins=24, kde=True, color="#002D62", alpha=0.6)

plt.show()
```



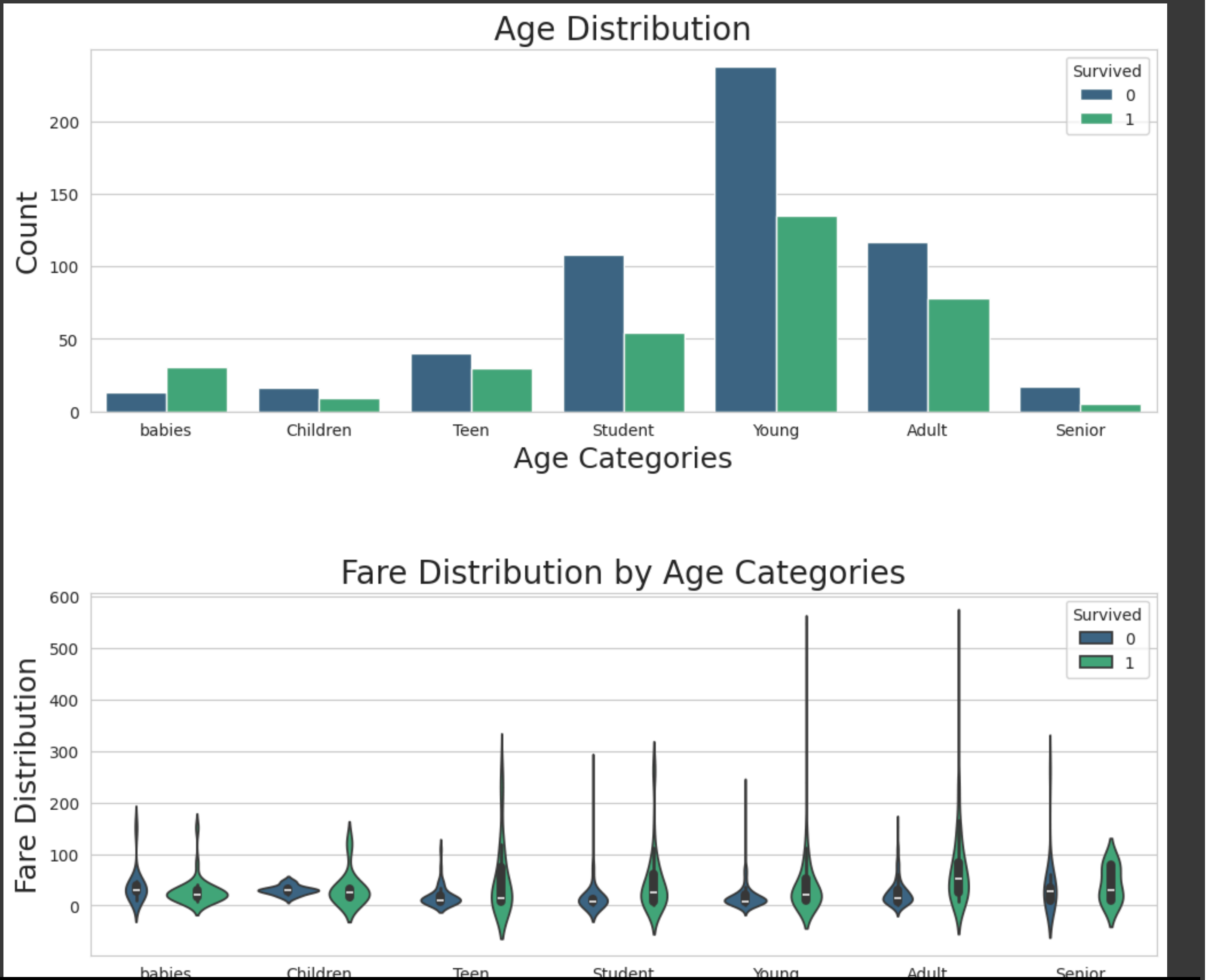
```
interval = (0, 5, 12, 18, 25, 35, 60, 120)

cats = ['babies', 'Children', 'Teen', 'Student', 'Young', 'Adult', 'Senior']

df_train["Age_cat"] = pd.cut(df_train.Age, interval, labels=cats)
df_test["Age_cat"] = pd.cut(df_test.Age, interval, labels=cats)

df_train["Age_cat"].head()
```

	Age_cat
0	Student
1	Adult
2	Young
3	Young
4	Young

dtype: category

```
print(pd.crosstab(df_train.Age_cat, df_train.Survived))

# Setting the figure size
plt.figure(figsize=(12,10))

# Plotting the result
plt.subplot(2,1,1)
sns.countplot(x="Age_cat", data=df_train, hue="Survived", palette="viridis")
plt.ylabel("Count", fontsize=18)
plt.xlabel("Age Categories", fontsize=18)
plt.title("Age Distribution", fontsize=20)

plt.subplot(2,1,2)
sns.violinplot(x='Age_cat', y="Fare", data=df_train, hue="Survived", palette="viridis")
plt.ylabel("Fare Distribution", fontsize=18)
plt.xlabel("Age Categories", fontsize=18)
plt.title("Fare Distribution by Age Categories", fontsize=20)

plt.subplots_adjust(hspace=0.5, top=0.9)

plt.show()
```

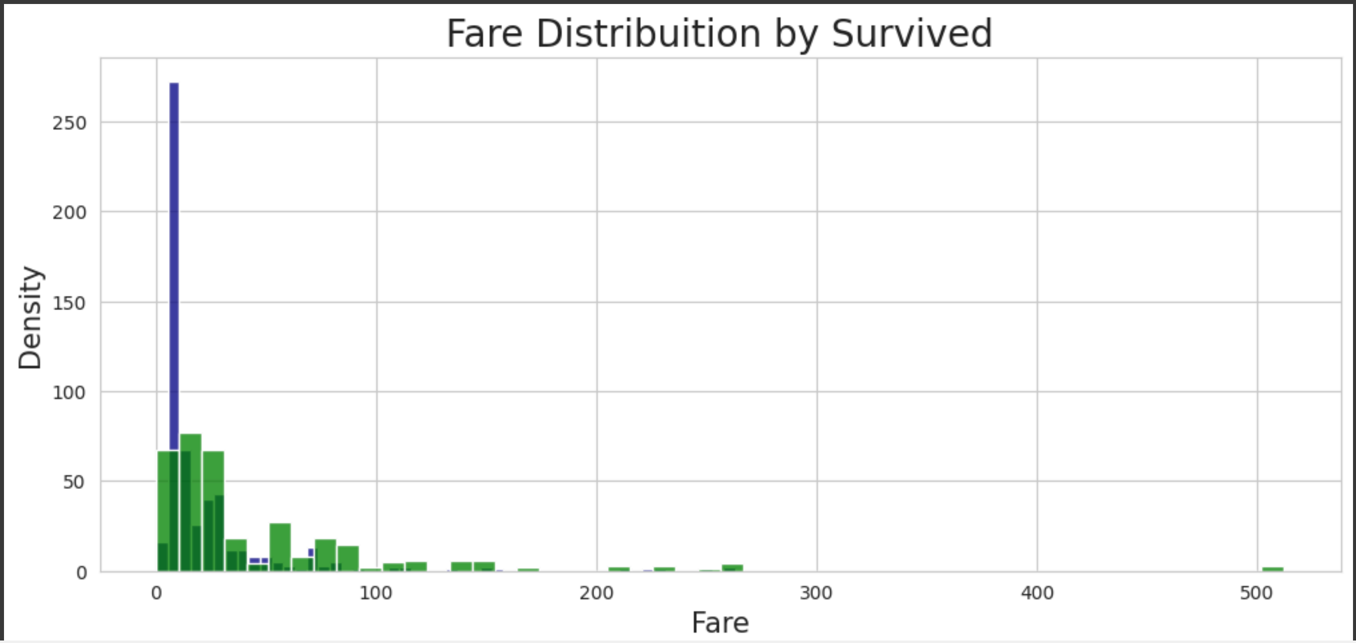
```
Survived    0    1
Age_cat
babies     13   31
Children   16    9
Teen       40   30
Student   108   54
Young     238  135
Adult     117   78
Senior     17    5
```



```
Age_fare = ['Pclass', 'Age_cat']

cm = sns.light_palette("purple", as_cmap=True)
pd.crosstab(df_train[Age_fare[0]], df_train[Age_fare[1]],
            values=df_train['Fare'], aggfunc=['mean']).style.background_gradient(cmap = cm)
```

	mean						
Age_cat	babies	Children	Teen	Student	Young	Adult	Senior
Pclass							
1	128.319433	120.000000	122.537500	113.002081	78.512619	76.983334	59.969050
2	28.179492	30.562500	21.172567	24.694331	17.535386	19.760638	10.500000
3	22.712200	27.326250	13.414589	8.903373	13.727935	13.334195	7.820000

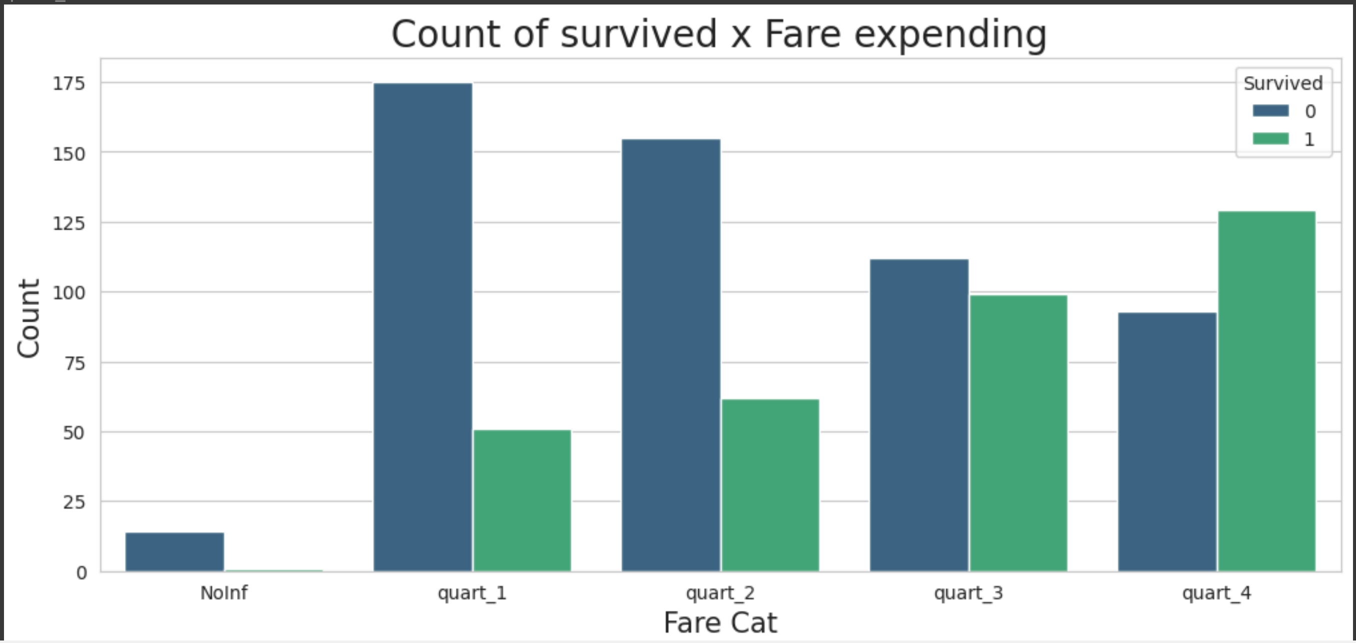
```
plt.figure(figsize=(12,5))

sns.histplot(df_train[df_train.Survived == 0]["Fare"], bins=50, color='navy')
sns.histplot(df_train[df_train.Survived == 1]["Fare"], bins=50, color='g')

plt.title("Fare Distribuition by Survived", fontsize=20)
plt.xlabel("Fare", fontsize=15)
plt.ylabel("Density",fontsize=15)
plt.show()
```



```
df_train.Fare = df_train.Fare.fillna(-0.5)
df_test.Fare = df_test.Fare.fillna(-0.5)


quant = (-1, 0, 8, 15, 31, 600)

label_quants = ['NoInf', 'quart_1', 'quart_2', 'quart_3', 'quart_4']


df_train["Fare_cat"] = pd.cut(df_train.Fare, quant, labels=label_quants)
df_test["Fare_cat"] = pd.cut(df_test.Fare, quant, labels=label_quants)
```

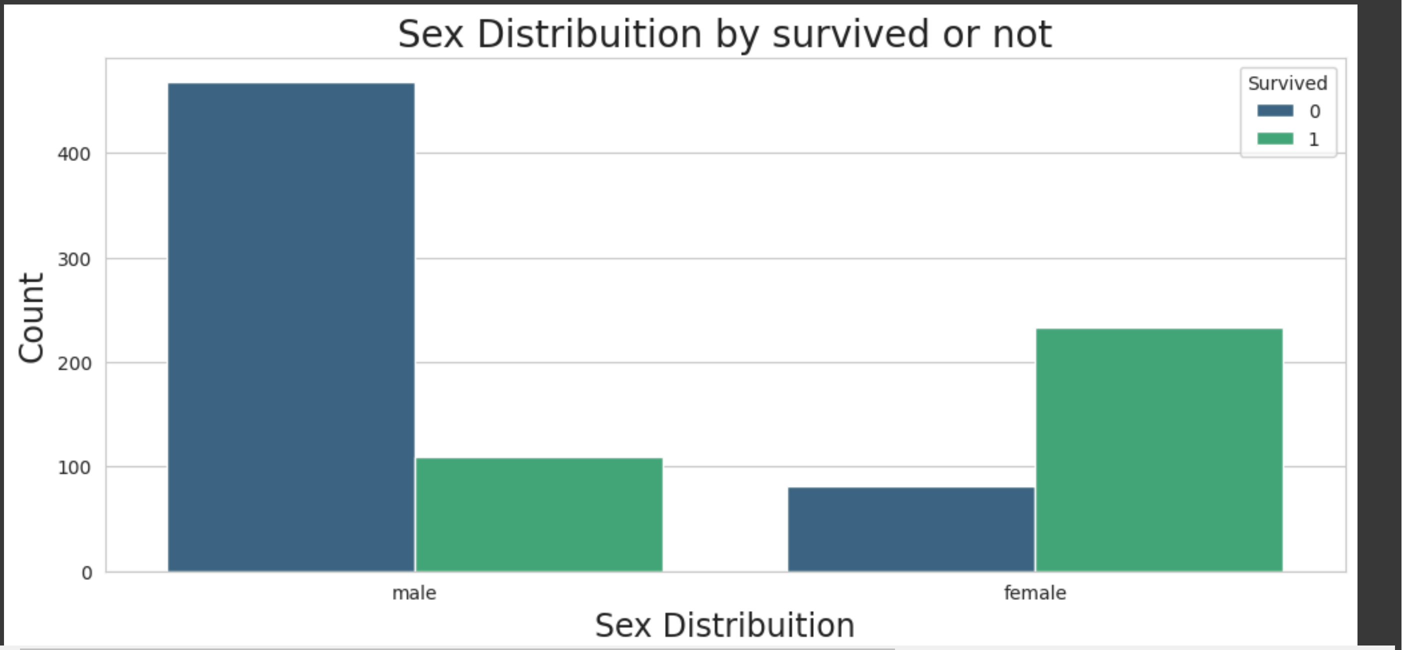
```
print(pd.crosstab(df_train.Fare_cat, df_train.Survived))

plt.figure(figsize=(12,5))


sns.countplot(x="Fare_cat", hue="Survived", data=df_train, palette="viridis")
plt.title("Count of survived x Fare expending",fontsize=20)
plt.xlabel("Fare Cat",fontsize=15)
plt.ylabel("Count",fontsize=15)

plt.show()
```

```
Survived    0    1
Fare_cat
NoInf      14    1
quart_1   175   51
quart_2   155   62
quart_3   112   99
quart_4    93  129
```



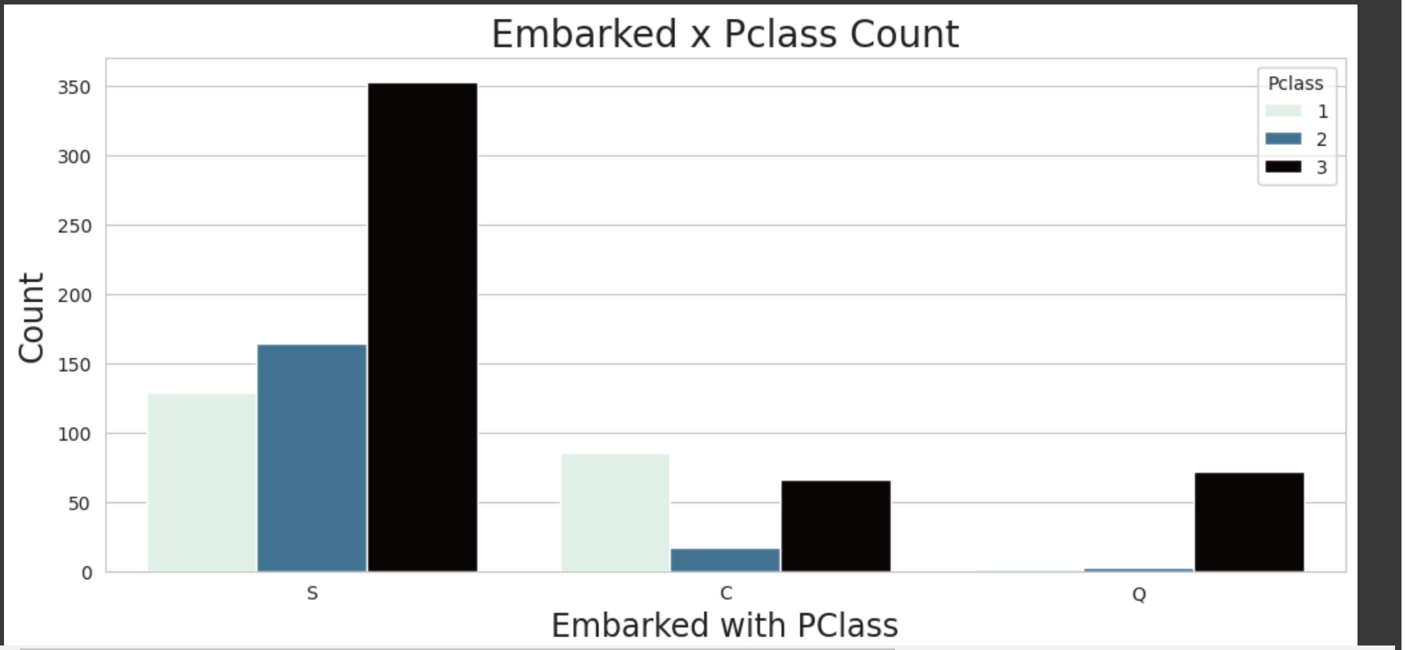
```
print(pd.crosstab(df_train.Survived, df_train.Sex))

plt.figure(figsize=(12,5))
sns.countplot(x="Sex", data=df_train, hue="Survived",palette="viridis")
plt.title('Sex Distribuition by survived or not', fontsize=20)
plt.xlabel('Sex Distribuition',fontsize=17)
plt.ylabel('Count', fontsize=17)

plt.show()
```

```
Sex       female  male
Survived
0             81   468
1            233   109
```



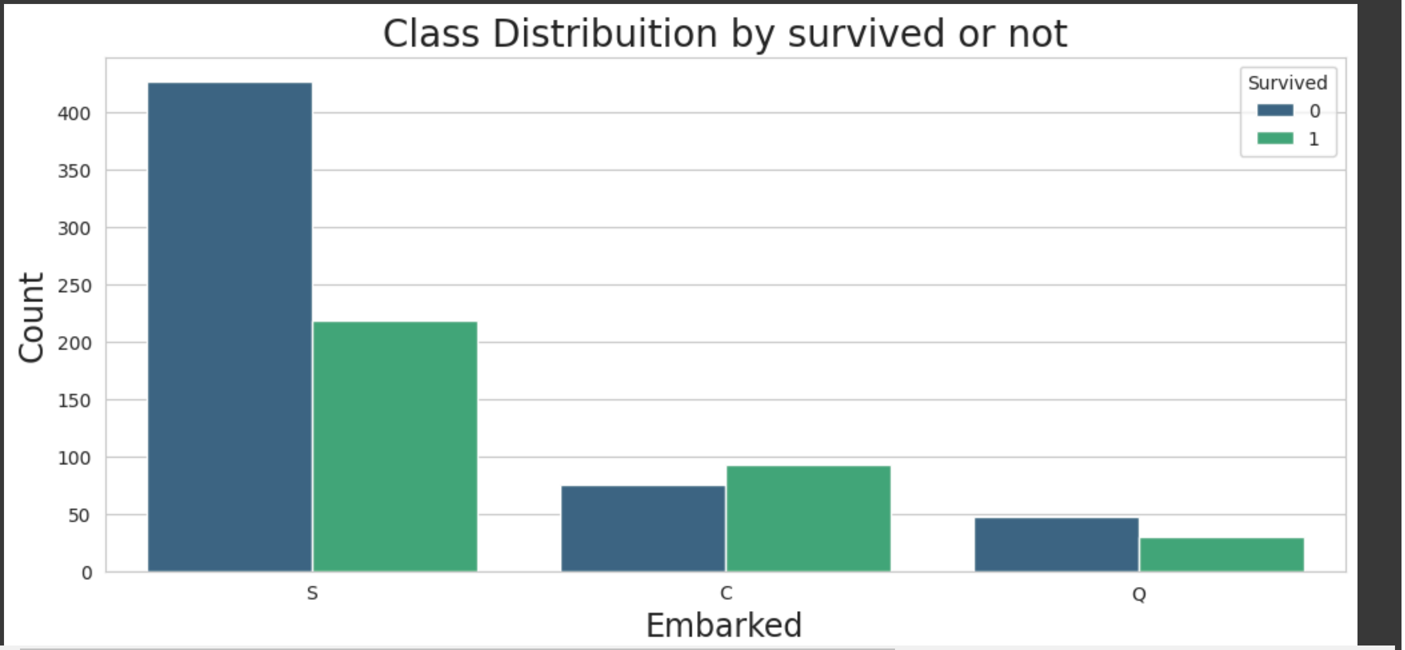
```
plt.figure(figsize=(12,5))
```

```
sns.countplot(x="Embarked", data=df_train, hue="Pclass",palette="mako_r")
plt.title('Embarked x Pclass Count', fontsize=20)
plt.xlabel('Embarked with PClass',fontsize=17)
plt.ylabel('Count', fontsize=17)

plt.show()
```



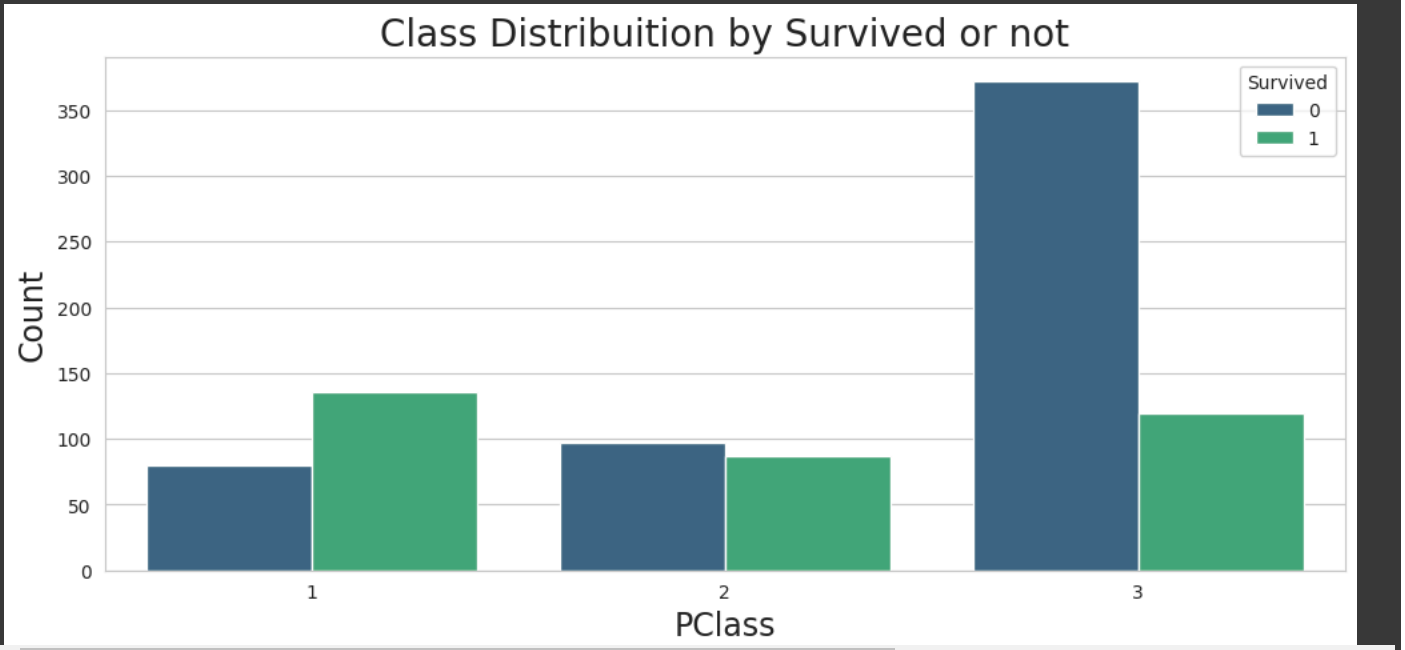
```
df_train["Embarked"] = df_train["Embarked"].fillna('S')
df_test["Embarked"] = df_test["Embarked"].fillna('S')
```

```
plt.figure(figsize=(12,5))

sns.countplot(x="Embarked", data=df_train, hue="Survived",palette="viridis")
plt.title('Class Distribuition by survived or not',fontsize=20)
plt.xlabel('Embarked',fontsize=17)
plt.ylabel('Count', fontsize=17)

plt.show()
```

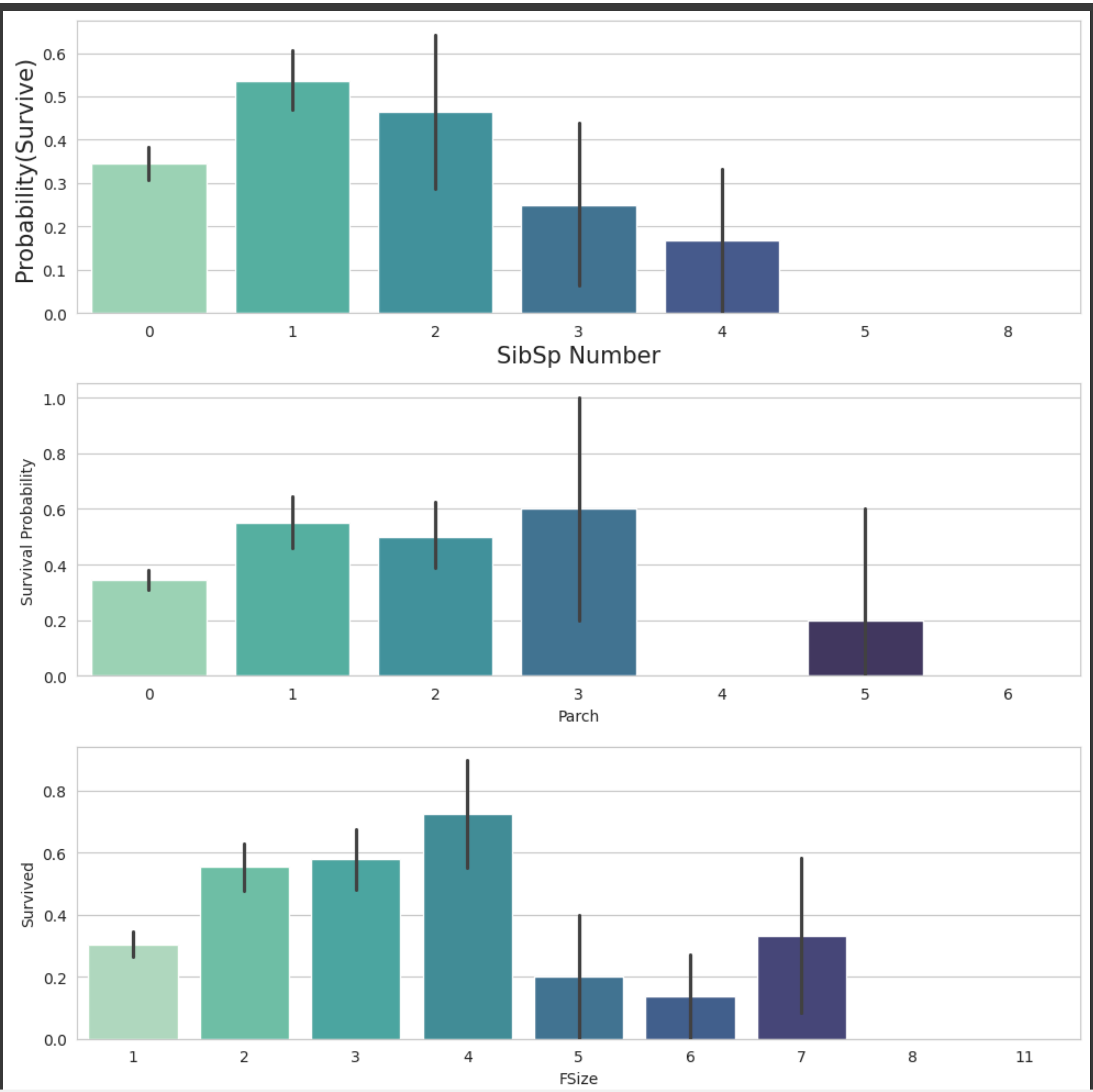


```
plt.figure(figsize=(12,5))

sns.countplot(x="Pclass", data=df_train, hue="Survived",palette="viridis")
```

```
plt.xlabel('PClass',fontsize=17)
plt.ylabel('Count', fontsize=17)
plt.title('Class Distribuition by Survived or not', fontsize=20)

plt.show()
```



```
fig, axes = plt.subplots(3, 1, figsize=(10, 10))


sns.barplot(x="SibSp", y="Survived", data=df_train,
            palette="mako_r", ax=axes[0])
axes[0].set_ylabel("Probability(Survive)", fontsize=15)
axes[0].set_xlabel("SibSp Number", fontsize=15)


sns.barplot(x="Parch", y="Survived", data=df_train,
            palette="mako_r", ax=axes[1])
axes[1].set_ylabel("Survival Probability")


df_train["FSize"] = df_train["Parch"] + df_train["SibSp"] + 1
df_test["FSize"] = df_test["Parch"] + df_test["SibSp"] + 1


sns.barplot(x="FSize", y="Survived", data=df_train,
            palette="mako_r", ax=axes[2])

plt.tight_layout()
plt.show()
```



```
del df_train["SibSp"]
del df_train["Parch"]

del df_test["SibSp"]
del df_test["Parch"]
```

```
train_column = df_train.columns
test_column = df_test.columns
print(train_column)
print(test_column)
```

```
Index(['Survived', 'Pclass', 'Sex', 'Age', 'Fare', 'Embarked', 'Title',
       'FamilySize', 'IsAlone', 'FareBin', 'AgeBin', 'Age_cat', 'Fare_cat',
       'FSize'],
      dtype='object')
Index(['Pclass', 'Sex', 'Age', 'Fare', 'Embarked', 'Title', 'FamilySize',
       'IsAlone', 'FareBin', 'AgeBin', 'Age_cat', 'Fare_cat', 'FSize'],
      dtype='object')
```

```
train = df_train.drop(['Age', 'IsAlone','FareBin','AgeBin'],axis=1)
test = df_test.drop(['Age', 'IsAlone','FareBin','AgeBin'],axis=1)
```

```
train.head()
```

	Survived	Pclass	Sex	Fare	Embarked	Title	FamilySize	Age_cat	Fare_cat	FSize
0	0	3	male	7.2500	S	Mr	2	Student	quart_1	2
1	1	1	female	71.2833	C	Mrs	2	Adult	quart_4	2
2	1	3	female	7.9250	S	Miss	1	Young	quart_1	1
3	1	1	female	53.1000	S	Mrs	2	Young	quart_4	2
4	0	3	male	8.0500	S	Mr	1	Young	quart_2	1

```
test.head()
```

	Pclass	Sex	Fare	Embarked	Title	FamilySize	Age_cat	Fare_cat	FSize
0	3	male	7.8292	Q	Mr	1	Young	quart_1	1
1	3	female	7.0000	S	Mrs	2	Adult	quart_1	2
2	2	male	9.6875	Q	Mr	1	Senior	quart_2	1
3	3	male	8.6625	S	Mr	1	Young	quart_2	1
4	3	female	12.2875	S	Mrs	3	Student	quart_2	3

```
categorical_cols = ["Sex", "Embarked", "Age_cat", "Fare_cat", "Title"]


label_encoders = {}

for col in categorical_cols:
    le = LabelEncoder()
    train[col] = le.fit_transform(train[col])
    test[col] = le.transform(test[col])
    label_encoders[col] = le
```

```
train.head()
```

	Survived	Pclass	Sex	Fare	Embarked	Title	FamilySize	Age_cat	Fare_cat	FSize
0	0	3	1	7.2500	2	2	2	3	1	2
1	1	1	0	71.2833	0	3	2	0	4	2
2	1	3	0	7.9250	2	1	1	5	1	1
3	1	1	0	53.1000	2	3	2	5	4	2
4	0	3	1	8.0500	2	2	1	5	2	1

```
test.head()
```

	Pclass	Sex	Fare	Embarked	Title	FamilySize	Age_cat	Fare_cat	FSize
0	3	1	7.8292	1	2	1	5	1	1
1	3	0	7.0000	2	3	2	0	1	2
2	2	1	9.6875	1	2	1	2	2	1
3	3	1	8.6625	2	2	1	5	2	1
4	3	0	12.2875	2	3	3	3	2	3

```
print(f" train shape {train.shape} test shape {test.shape}")
```

```
 train shape (891, 10) test shape (418, 9)
```

```
x_train = train.drop(["Survived"],axis=1)
y_train = train["Survived"]

print(f" x_train shape {x_train.shape} y_train shape {y_train.shape}")
```

```
 x_train shape (891, 9) y_train shape (891,)
```

```
X = x_train.values
y = y_train.values

x_test_without_target = test.values
x_test_without_target = test.astype(np.float64, copy=False)
```

```
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
print(f" After Spliting:\n x_train shape {x_train.shape} y_train shape {y_train.shape}\n x_test shape {x_test.shape} y_test shape {y_test.sh
```

```
 After Spliting:
 x_train shape (712, 9) y_train shape (712,)
 x_test shape (179, 9) y_test shape (179,)
```

```
class StandardScaler:
    def __init__(self):
        self.mean = None
        self.std = None

    def fit(self, X):
        self.mean = np.mean(X, axis=0)
        self.std = np.std(X, axis=0, ddof=0)

    def transform(self, X):

        if self.mean is None or self.std is None:
            raise ValueError("Scaler has not been fitted yet. Call fit(X) first.")
        return (X - self.mean) / (self.std + 1e-9)

    def fit_transform(self, X):
        self.fit(X)
        return self.transform(X)

    def inverse_transform(self, X_scaled):
        return (X_scaled * self.std) + self.mean
```

```

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

```
def initialize_weights(layers):
    weights = []
    for i in range(len(layers) - 1):
        w = np.random.uniform(-1, 1, (layers[i + 1], layers[i] + 1))
        weights.append(w)
    return weights

def sigmoid(x):
    return 1 / (1 + np.exp(-np.clip(x, -500, 500)))

def sigmoid_derivative(x):
    return x * (1 - x)

def forward_propagation(x, weights):
    activations = []
    input_layer = np.append(1, x)
    activations.append(input_layer)

    for w in weights:
        net_input = np.dot(w, input_layer)
        activation = sigmoid(net_input)
        input_layer = np.append(1, activation)
        activations.append(input_layer)

    return activations

def back_propagation(y, activations, weights, lr):
    error = y - activations[-1][1:]

    for i in range(len(weights) - 1, -1, -1):
        delta = error * sigmoid_derivative(activations[i + 1][1:])

        prev_activation = activations[i].reshape(-1, 1)
        delta = delta.reshape(-1, 1)
        weights[i][:, 1:] += lr * np.dot(delta, prev_activation[1:].T)
        weights[i][:, 0] += lr * delta.flatten()

        error = np.dot(weights[i][:, 1:].T, delta).flatten()

    return weights


def train(X, Y, weights, lr, epochs):
    losses = []
    accuracies = []

    for epoch in range(epochs):
        epoch_loss = 0
        for i in range(len(X)):
            activations = forward_propagation(X[i], weights)
            weights = back_propagation(Y[i], activations, weights, lr)


            y_predict = activations[-1]
            epoch_loss += -Y[i] * np.log(y_predict + 1e-9) - (1 - Y[i]) * np.log(1 - y_predict + 1e-9)


        losses.append(epoch_loss / len(X))


        acc = accuracy(X, Y, weights)
        accuracies.append(acc)
```