Titanic Survival Prediction

This project involves building a model to predict the likelihood of a passenger surviving the Titanic disaster based on various factors like age, gender, passenger class, and other features. The goal is to analyze the data and create a predictive model that can accurately classify whether a given passenger survived or not.

1.Import required libraries

In [1]:
```
import numpy as np   #for multidimensional arrays
import pandas as pd   #for data manipulation and cleaning
import matplotlib.pyplot as plt  #for data visualization
import seaborn as sns  #for statistical visualization

%matplotlib inline
```

2.Load the data

In [2]:
```
titanic_data = pd.read_csv(r'C:\Users\91939\Desktop\AI&DS\CODSOFT\Titanic-Datase
```

In [3]:
```
titanic_data.head(5) #returns the first five rows from the dataframe
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	I
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0

3.Analyse the data

In [4]:
```
titanic_data.columns #returns all the columns from the dataframe
```

Out[4]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [5]:
```
len(titanic_data.columns) #returns the length of columns of dataframe
```

Out[5]:
```
12
```

There are 12 columns in total

In [6]:
```
titanic_data.info() #returns the summary of the dataframe
```

```
<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
```

There are float64(2), int64(5), object(5) datatypes.

In [7]:
```
titanic_data.isna() #returns True if there is null value else False
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	False	False	False	False	False	False	False	False	False	False	T
1	False	False	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	False	T
3	False	False	False	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	False	False	False	T
...	...	...	...	...	...	...	...	...	...	...	
886	False	False	False	False	False	False	False	False	False	False	T
887	False	False	False	False	False	False	False	False	False	False	Fa
888	False	False	False	False	False	True	False	False	False	False	T
889	False	False	False	False	False	False	False	False	False	False	Fa
890	False	False	False	False	False	False	False	False	False	False	T

891 rows × 12 columns

In [8]:
```
titanic_data[titanic_data.isna()]
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cab
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	
886	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
887	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
888	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
889	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
890	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

891 rows × 12 columns

In [9]:
```
titanic_data.isna().sum() #returns sum of null values in each attribute
```

Out[9]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

There are 177 null values and 687 null values in Age and Cabin columns

4.Drop the irrelavant Attributes

In [10]:
```
titanic_data.Cabin
```

Out[10]:
```
0        NaN
1        C85
2        NaN
3       C123
4        NaN
       ...  
886      NaN
887      B42
888      NaN
889     C148
890      NaN
Name: Cabin, Length: 891, dtype: object
```

In [11]:
```
titanic_data = titanic_data.drop(columns='Cabin',axis=1)  #drop the Cabin attrib
```

In [12]:
```
titanic_data.columns
```

Out[12]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Embarked'],
      dtype='object')
```

In [13]:
```
titanic_data.head()
```

Out[13]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0

5.Missing Value Treatment

In [14]:
```
titanic_data.fillna(titanic_data['Age'].mean(),inplace=True)  #fill the age attr
```

In [15]:
```
titanic_data['Age'].isnull().sum()      #checks whether or not there are null va
```

Out[15]: 0

In [16]:
```
titanic_data.fillna(titanic_data['Embarked'].mode(),inplace=True)  #fill the age
```

In [17]:
```
titanic_data['Embarked'].isnull().sum()   #checks whether or not there are null
```

Out[17]: 0

In [18]:
```
titanic_data.isnull().sum().sum()   #checks whether or not there are null values
```

Out[18]: 0

6.Check the survived Attribute

In [19]:
```
titanic_data['Survived'].unique()   #checks the unique values from the Survived
```

Out[19]: array([0, 1], dtype=int64)

In [20]:
```
titanic_data['Survived'].nunique()   #checks the no.of unique values from the Su
```

Out[20]: 2

In [21]:
```
titanic_data['Survived'].value_counts()    #Return a Series containing the frequ
```

Out[21]:
```
Survived
0    549
1    342
Name: count, dtype: int64
```

In [22]:
```
titanic_data.describe().T  #Generate descriptive statistics.
```

Out[22]:

	count	mean	std	min	25%	50%	75%	max
PassengerId	891.0	446.000000	257.353842	1.00	223.5000	446.000000	668.5	891.0000
Survived	891.0	0.383838	0.486592	0.00	0.0000	0.000000	1.0	1.0000
Pclass	891.0	2.308642	0.836071	1.00	2.0000	3.000000	3.0	3.0000
Age	891.0	29.699118	13.002015	0.42	22.0000	29.699118	35.0	80.0000
SibSp	891.0	0.523008	1.102743	0.00	0.0000	0.000000	1.0	8.0000
Parch	891.0	0.381594	0.806057	0.00	0.0000	0.000000	0.0	6.0000
Fare	891.0	32.204208	49.693429	0.00	7.9104	14.454200	31.0	512.3292

In [23]:
```
titanic_data.head()
```

Out[23]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	I
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0

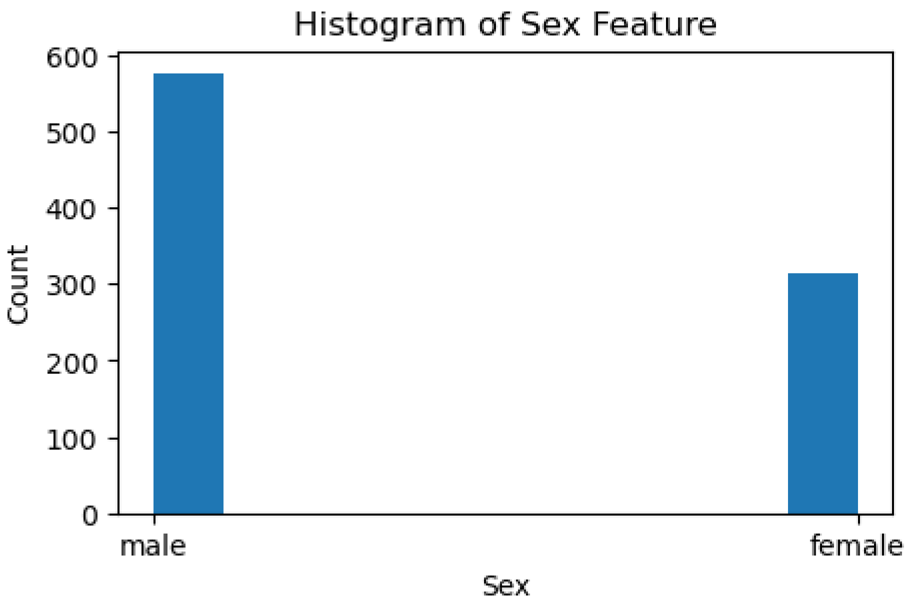
7.Univariate analysis on Sex Attribute

In [24]:
```
titanic_data.Sex.value_counts()
```

Out[24]:
```
Sex
male      577
female    314
Name: count, dtype: int64
```

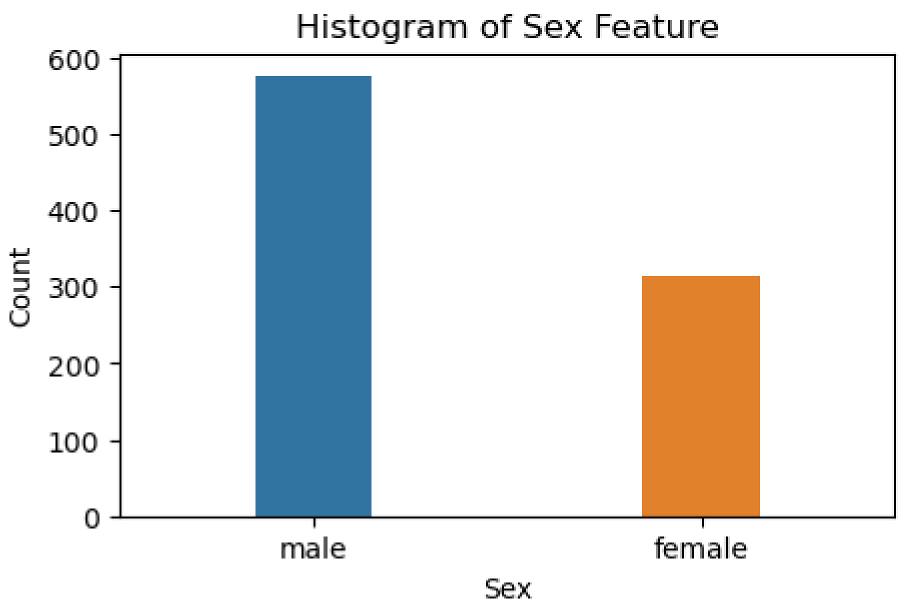
In [25]:
```
plt.figure(figsize=(5,3))
plt.title('Histogram of Sex Feature')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.hist(titanic_data.Sex,bins=10)   #returns the histogram of sex feature in fo
```

Out[25]:
```
(array([577.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 314.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



In [26]:
```
plt.figure(figsize=(5,3))
plt.title('Histogram of Sex Feature')
plt.xlabel('Sex')
plt.ylabel('Count')
sns.countplot(x=titanic_data.Sex,width=0.3,hue= 'Sex',data =titanic_data, orient
```

Out[26]:
```
<Axes: title={'center': 'Histogram of Sex Feature'}, xlabel='Sex', ylabel='Coun
t'>
```



There are significantly more male passengers than female passengers on the Titanic. The count of male passengers is over 500, while the count of female passengers is slightly above 300.

8.Univariate analysis on Age Attribute

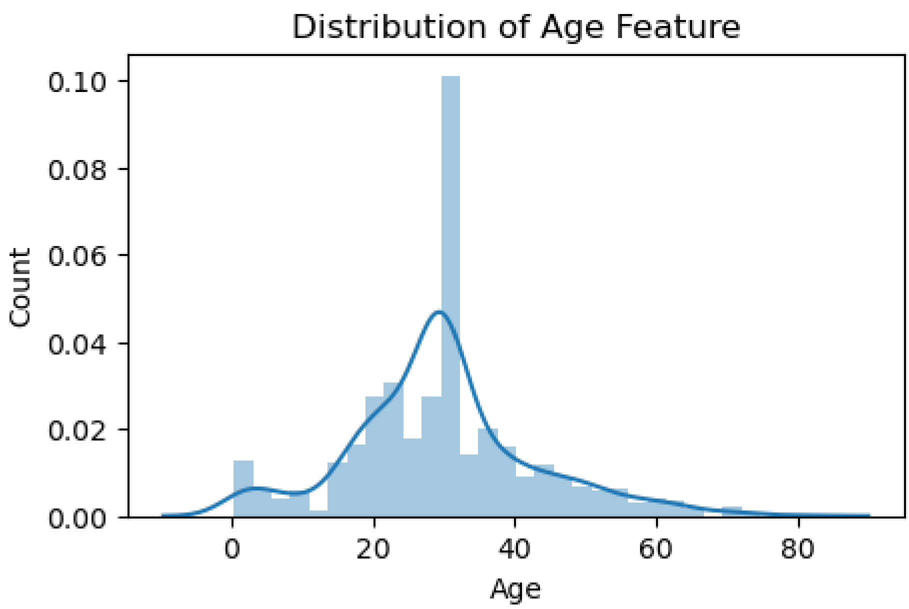
In [27]:
```
titanic_data.Age.value_counts()
```

Out[27]:
```
Age
29.699118    177
24.000000     30
22.000000     27
18.000000     26
28.000000     25
            ... 
36.500000      1
55.500000      1
0.920000       1
23.500000      1
74.000000      1
Name: count, Length: 89, dtype: int64
```

In [28]:
```
import warnings
warnings.filterwarnings('ignore')
```

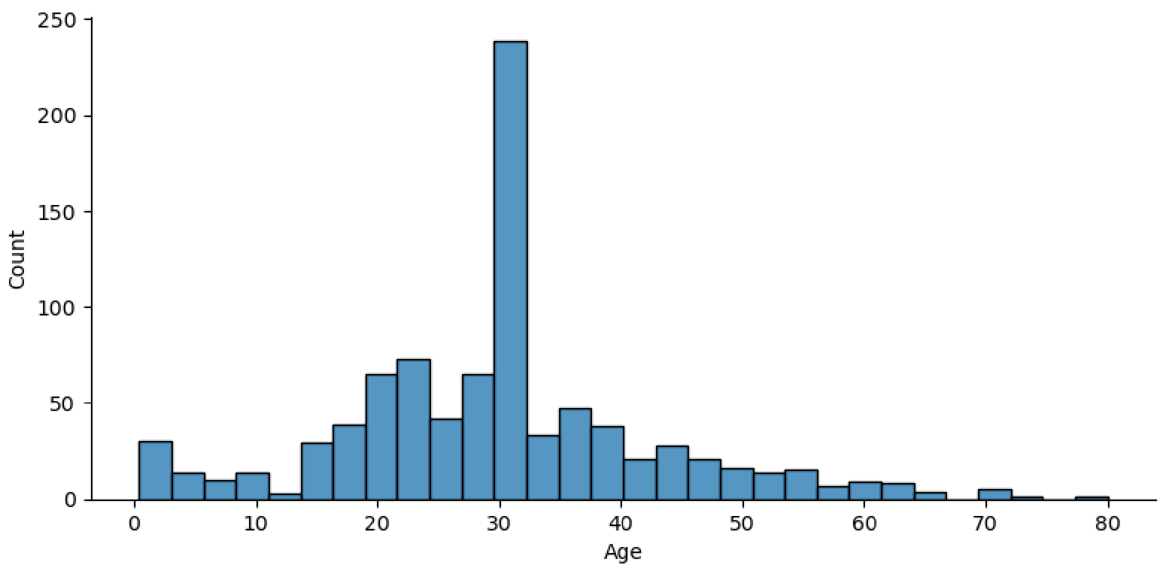
In [29]:
```
plt.figure(figsize=(5,3))
plt.title('Distribution of Age Feature')
plt.xlabel('Age')
plt.ylabel('Count')
sns.distplot(titanic_data['Age'],bins=30)
```

Out[29]:
```
<Axes: title={'center': 'Distribution of Age Feature'}, xlabel='Age', ylabel='C
ount'>
```



In [30]:
```
sns.displot(titanic_data['Age'],bins=30,height=4,aspect=2)
```

Out[30]:
```
<seaborn.axisgrid.FacetGrid at 0x2923b468e00>
```



The majority of the Titanic passengers were in their 20s and 30s, with the most common age being around 30.

9.Univariate analysis on Survived Attribute

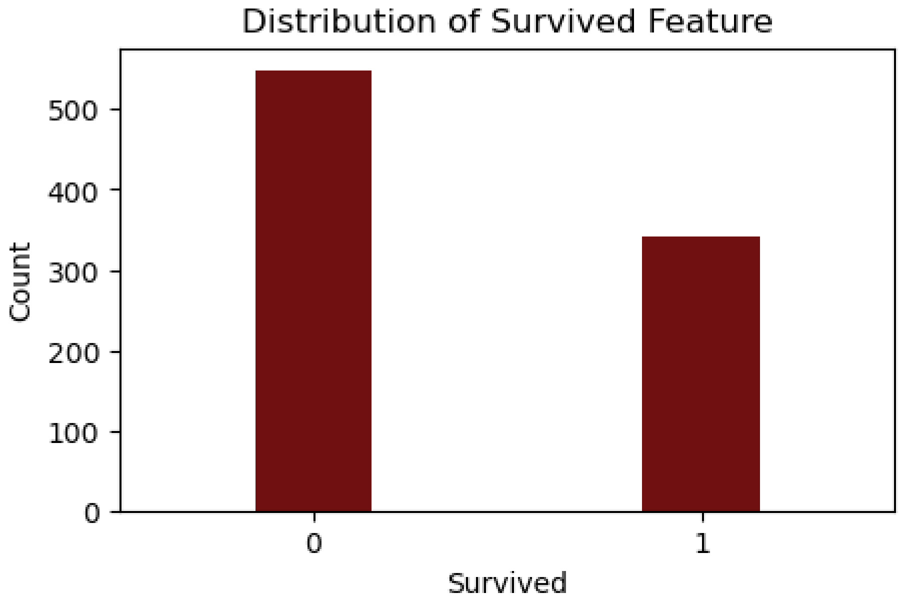
In [31]:
```
titanic_data['Survived'].value_counts()
```

Out[31]:
```
Survived
0    549
1    342
Name: count, dtype: int64
```

In [32]:
```
plt.figure(figsize=(5,3))
plt.title('Distribution of Survived Feature')
plt.xlabel('Survived')
```

```
plt.ylabel('Count')
sns.countplot(x='Survived',data=titanic_data,width=0.3,color='maroon')
```

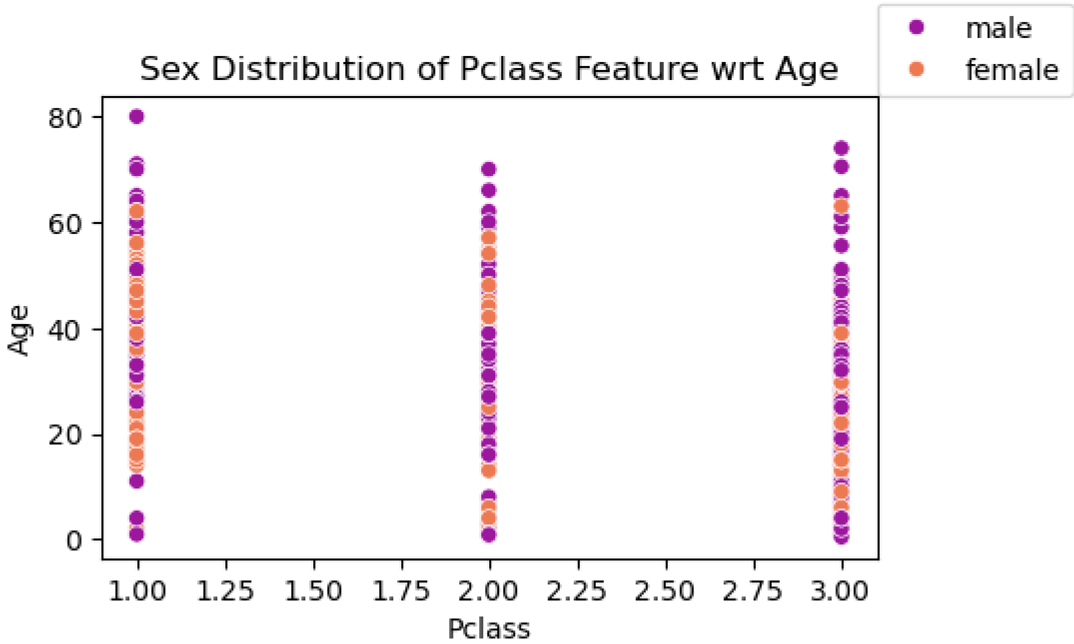
Out[32]: <Axes: title={'center': 'Distribution of Survived Feature'}, xlabel='Survived', ylabel='Count'>



10.Bivariate Analysis

In [33]:
```
plt.figure(figsize=(5,3))
plt.title('Sex Distribution of Pclass Feature wrt Age')
plt.xlabel('Pclass')
plt.ylabel('Age')
sns.scatterplot(x ='Pclass',y='Age',hue='Sex',data=titanic_data,palette='plasma'
plt.legend(loc=(1,1))
```

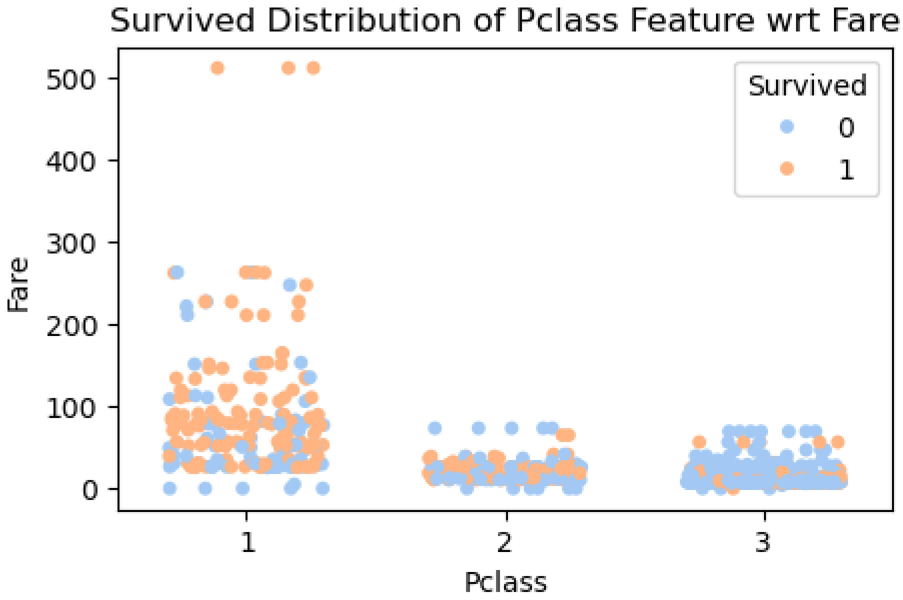
Out[33]: <matplotlib.legend.Legend at 0x2923b469ac0>



The scatter plot shows the distribution of passenger class (Pclass) with respect to age, color-coded by gender. It reveals that older passengers were more likely to be in higher-paying classes, while younger passengers were more likely to be in lower-paying classes. Additionally, the plot suggests a slight tendency for males to be older than females, especially in Pclass 3.

In [34]:
```
plt.figure(figsize=(5,3))
plt.title('Survived Distribution of Pclass Feature wrt Fare')
plt.xlabel('Pclass')
plt.ylabel('Fare')
sns.stripplot(x ='Pclass',y='Fare',data=titanic_data,hue='Survived', palette='pa
```

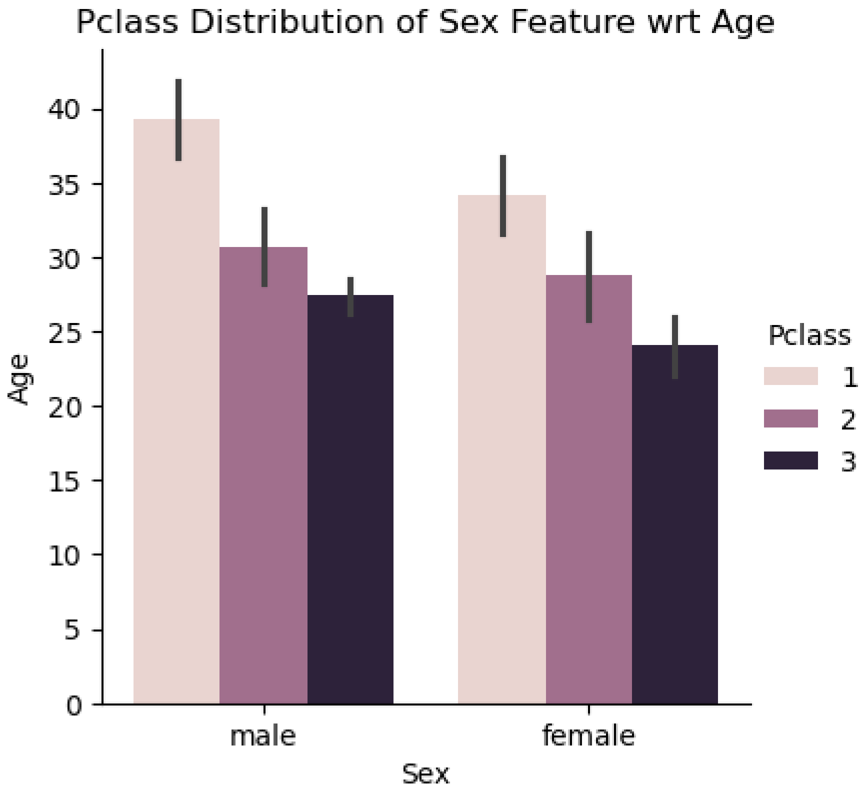
Out[34]:
```
<Axes: title={'center': 'Survived Distribution of Pclass Feature wrt Fare'}, xl
abel='Pclass', ylabel='Fare'>
```



There seems to be a general trend where passengers in higher-paying classes (Pclass 1 and 2) had a higher likelihood of survival compared to those in the lowest class (Pclass 3). This is suggested by the greater proportion of blue dots (survivors) associated with higher fare ranges.

In [35]:
```
sns.catplot(x ='Sex',y='Age',hue='Pclass',data=titanic_data,kind='bar',height=4,
plt.title('Pclass Distribution of Sex Feature wrt Age')
```

Out[35]:
```
Text(0.5, 1.0, 'Pclass Distribution of Sex Feature wrt Age')
```



The plot shows that the average age of passengers varies across different passenger classes and sexes. Passengers in higher classes tend to be older, and males are slightly older than females.

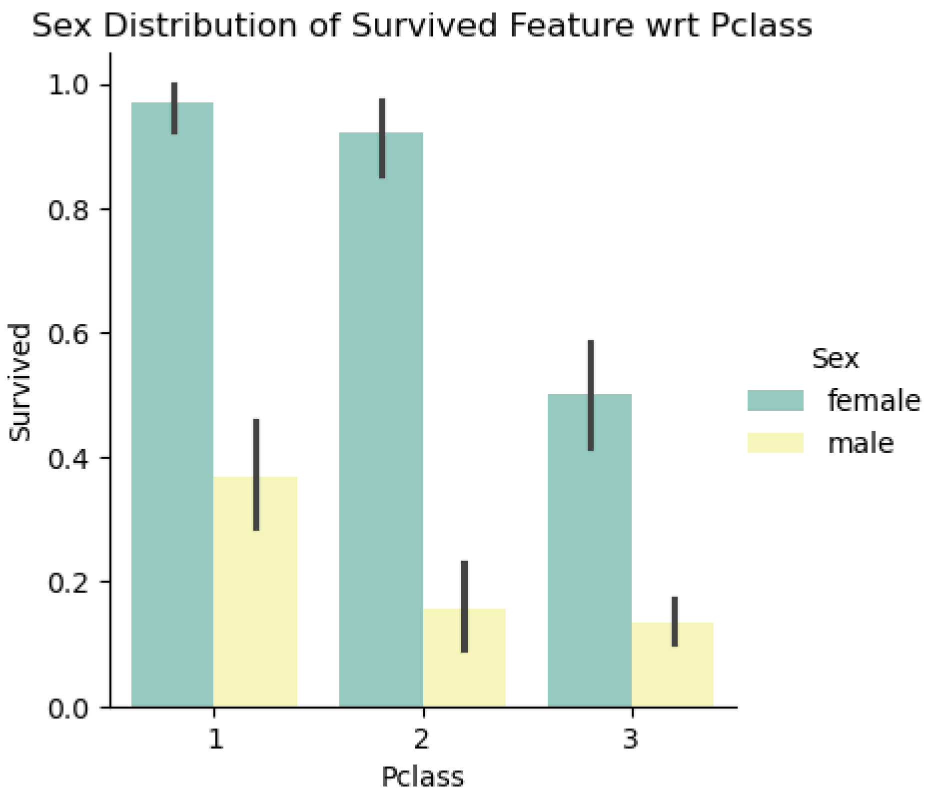
In [36]:
```
sns.catplot(x ='Pclass',y='Fare',hue='Sex',data=titanic_data,kind='bar',height=4
plt.title('Sex Distribution of Pclass Feature wrt Fare')
```

Out[36]:
```
Text(0.5, 1.0, 'Sex Distribution of Pclass Feature wrt Fare')
```

The plot shows that the average fare paid varies across different passenger classes (Pclass) and sexes. Passengers in Pclass 1 generally paid higher fares compared to those in Pclass 2 and 3. Additionally, there were slight differences in average fare between males and females within each Pclass.

In [37]:
```
sns.catplot(x ='Pclass',y='Survived',hue='Sex',data=titanic_data,kind='bar',heig
plt.title('Sex Distribution of Survived Feature wrt Pclass')
```

Out[37]: `Text(0.5, 1.0, 'Sex Distribution of Survived Feature wrt Pclass')`

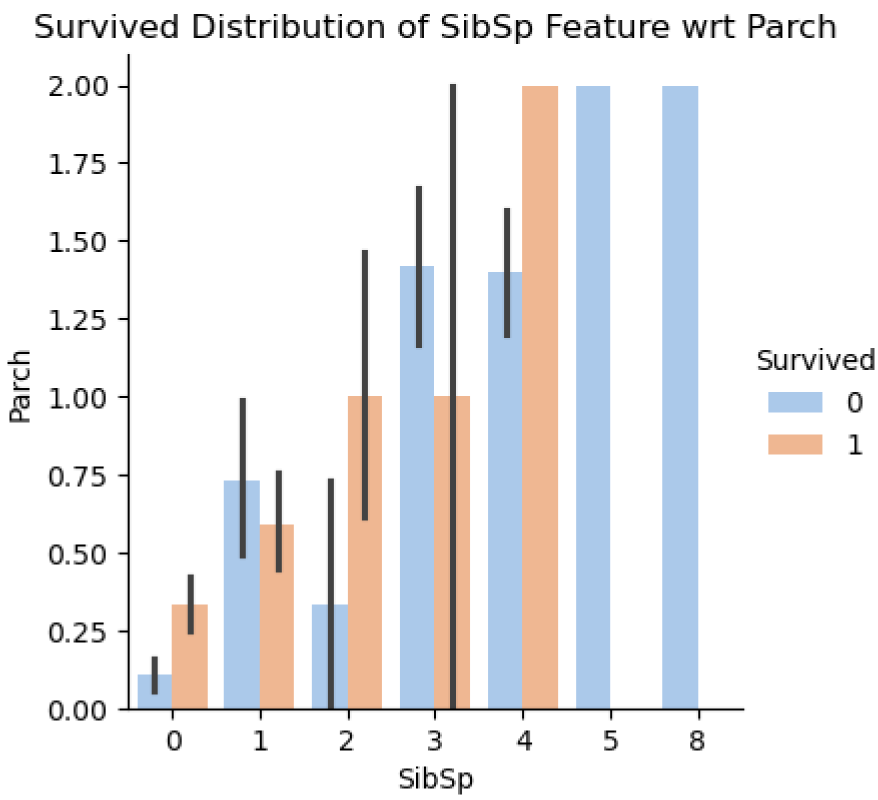


Effect of Pclass on Survived: Passengers in Pclass 1 had a higher survival rate compared to those in Pclass 2 and 3.

Impact of Sex on Survived: Within each Pclass, females had a higher survival rate than males.

In [38]:
```
sns.catplot(x ='SibSp',y='Parch',hue='Survived',data=titanic_data,kind='bar',hei
plt.title('Survived Distribution of SibSp Feature wrt Parch')
```

Out[38]: `Text(0.5, 1.0, 'Survived Distribution of SibSp Feature wrt Parch')`



The plot indicates that passengers with larger families (higher combined values of SibSp and Parch) might have had a slightly lower survival rate compared to those with smaller families.

11.Variable Transformation

In [39]:
```
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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          891 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  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

In [40]:
```
titanic_data['Sex'].value_counts()
```

Out[40]:
```
Sex
male      577
female    314
Name: count, dtype: int64
```

In [41]:
```
titanic_data['Embarked'].value_counts()
```

Out[41]:
```
Embarked
S            644
C            168
Q             77
29.699118      2
Name: count, dtype: int64
```

In [42]:
```
titanic_data.replace({'Sex':{'male':0,'female':1},'Embarked':{'S':0,'C':1,'Q':2}
```

In [43]:
```
titanic_data['Embarked'].value_counts()
```

Out[43]:
```
Embarked
0.000000     644
1.000000     168
2.000000      77
29.699118      2
Name: count, dtype: int64
```

In [44]:
```
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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    int64
 5   Age          891 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  Embarked     891 non-null    float64
dtypes: float64(3), int64(6), object(2)
memory usage: 76.7+ KB
```

12.Variable Identification

In [45]:
```
X = titanic_data.drop(columns=['PassengerId','Name','Ticket'],axis=1)
```

In [46]:
```
X.head()    #independent variables
```

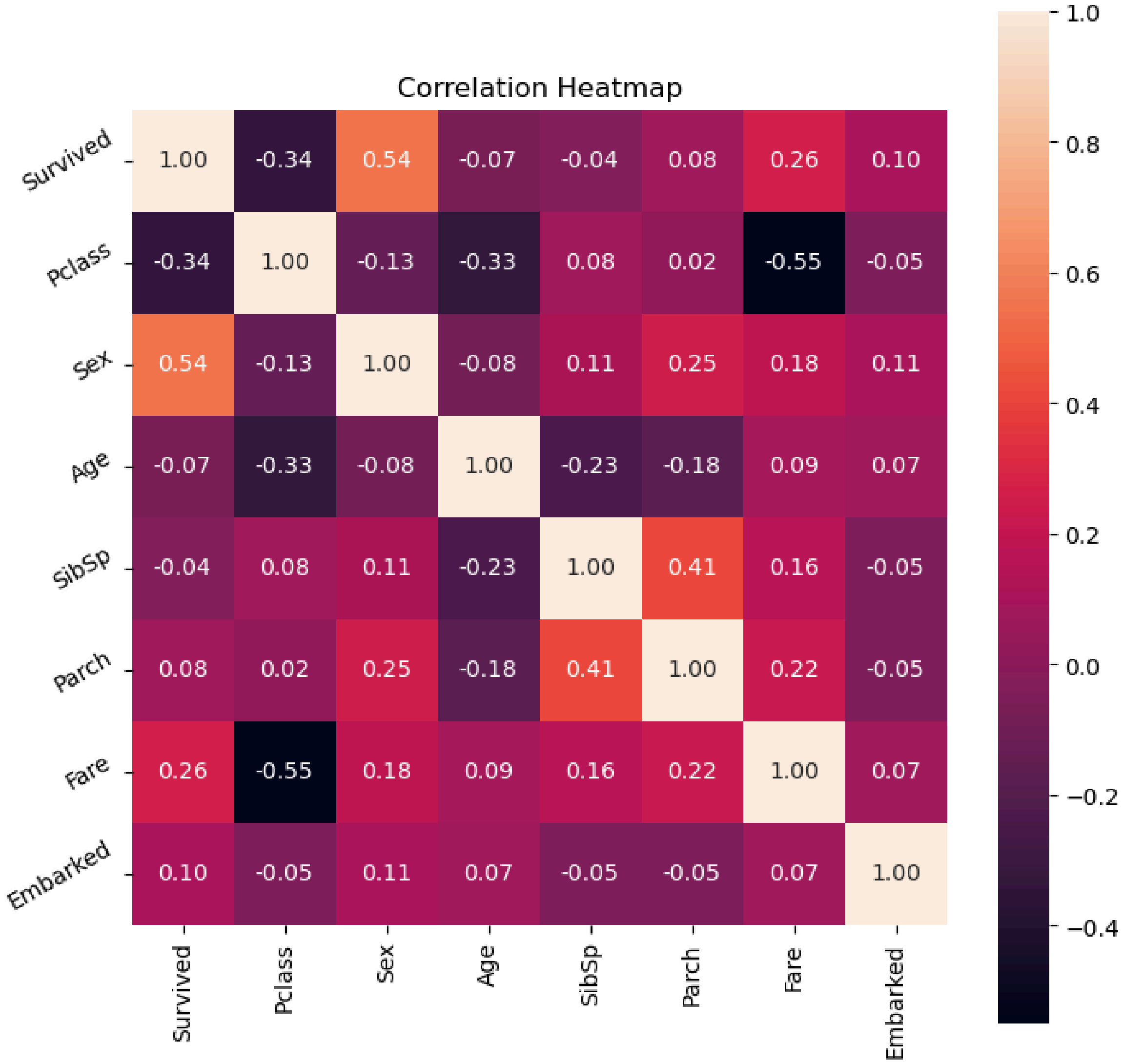
Out[46]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22.0	1	0	7.2500	0.0
1	1	1	1	38.0	1	0	71.2833	1.0
2	1	3	1	26.0	0	0	7.9250	0.0
3	1	1	1	35.0	1	0	53.1000	0.0
4	0	3	0	35.0	0	0	8.0500	0.0

In [47]:
```
Y = titanic_data['Survived']      #dependent variable
```

In [48]:
```
correlation = X.corr()
```

In [49]:
```
plt.figure(figsize=(8,8))
plt.title('Correlation Heatmap')
a = sns.heatmap(correlation, square=True, annot=True, fmt='.2f', linecolor='whit
a.set_xticklabels(a.get_xticklabels(), rotation=90)
a.set_yticklabels(a.get_yticklabels(), rotation=30)
plt.show()
```



Social Status and Survival: The strong negative correlation between Pclass and Survived suggests that social status (as indicated by passenger class) played a significant role in survival.

Family Ties: The moderate positive correlation between SibSp and Parch indicates that passengers traveling with family members were more likely to be accompanied by others.

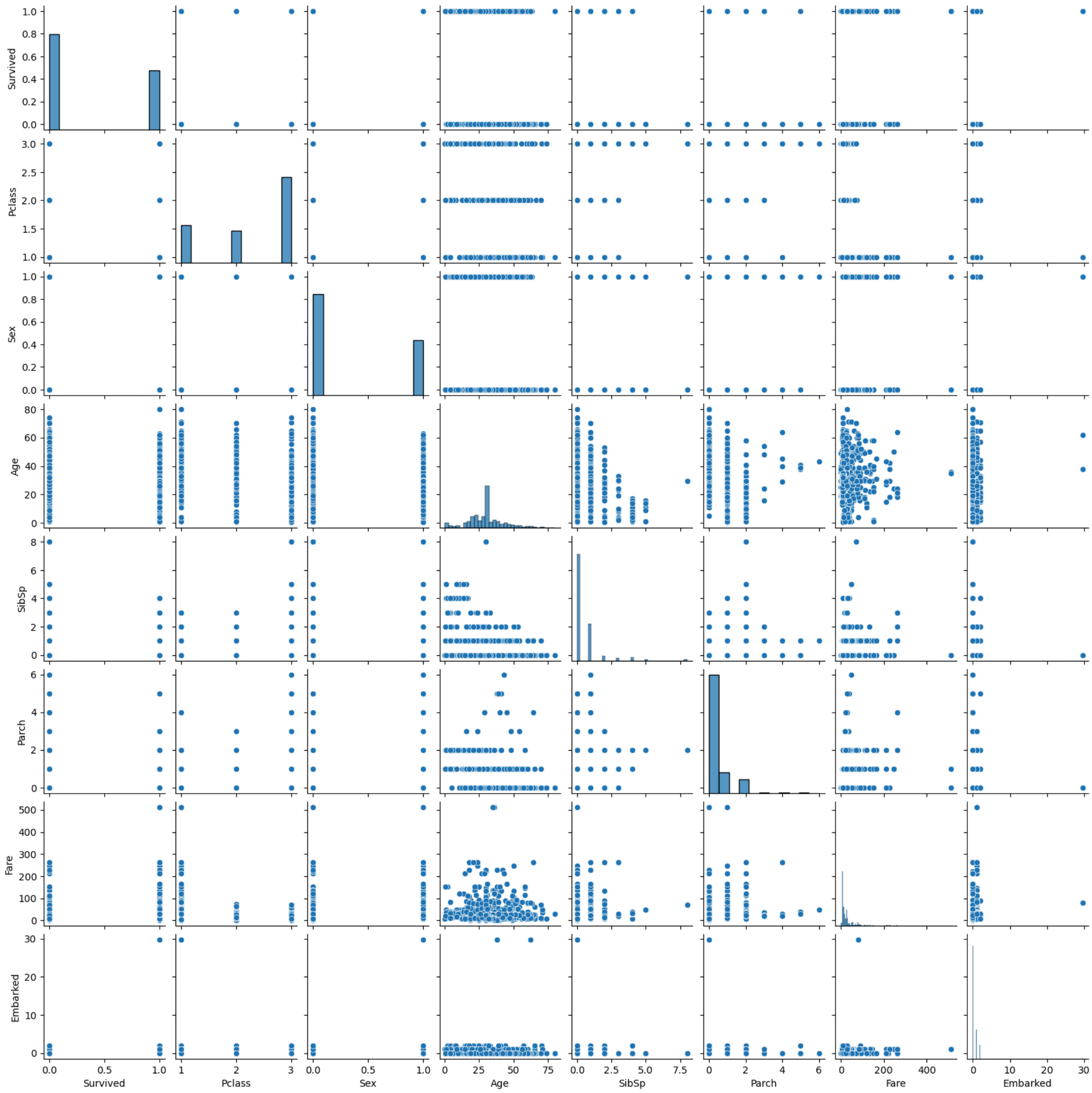
Gender Bias: The strong positive correlation between Sex and Survived suggests that females were more likely to survive compared to males, possibly due to societal norms and practices at the time.

In [50]:
```
titanic_data.head()
```

Out[50]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.050

In [51]:
```
a = sns.pairplot(X,kind='scatter',palette='pastel', diag_kind='hist',height = 2,
plt.show()
```

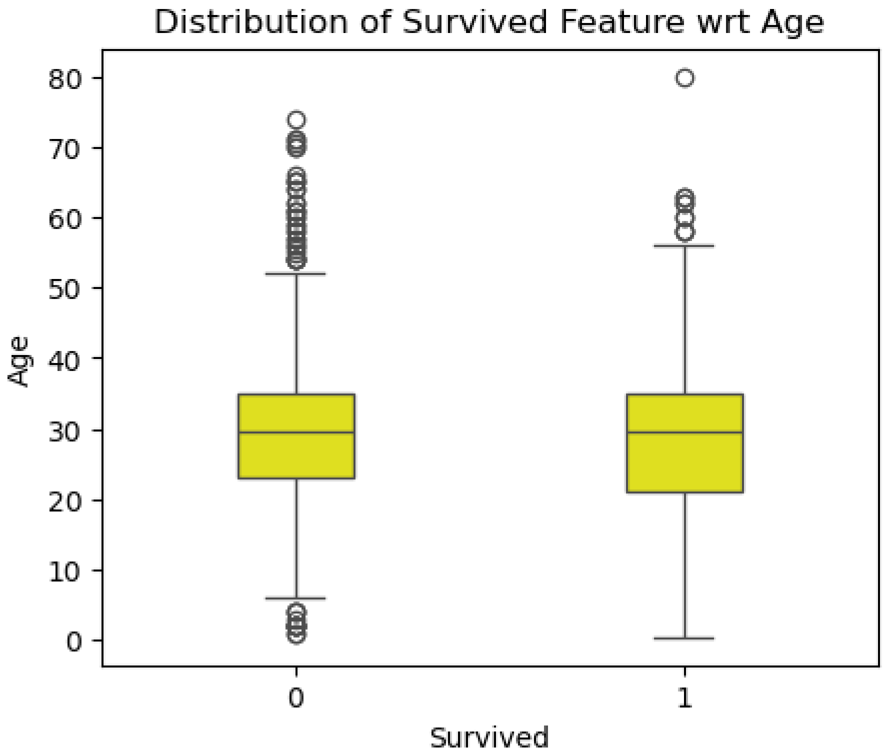


The pair plots have revealed valuable insights into the relationships between different variables in the dataset.

13.Outlier detection

In [52]:
```
plt.figure(figsize=(5,4))
plt.title('Distribution of Survived Feature wrt Age')
plt.xlabel('Survived')
plt.ylabel('Age')
sns.boxplot(x ='Survived',y='Age',data=titanic_data,width=0.3,color='yellow')
```

Out[52]:
```
<Axes: title={'center': 'Distribution of Survived Feature wrt Age'}, xlabel='Survived', ylabel='Age'>
```



The median age of survivors (Survived = 1) is slightly lower than that of non-survivors (Survived = 0). This suggests that Younger passengers had a better chance of survival on the Titanic. The distribution of ages among survivors and non-survivors is similar, but there are subtle differences in medians and the presence of older outliers.

In [53]:
```
X.head()
```

Out[53]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22.0	1	0	7.2500	0.0
1	1	1	1	38.0	1	0	71.2833	1.0
2	1	3	1	26.0	0	0	7.9250	0.0
3	1	1	1	35.0	1	0	53.1000	0.0
4	0	3	0	35.0	0	0	8.0500	0.0

In [54]:
```
Y.head()
```

Out[54]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

14.Applying Machine Learning

In [55]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

```
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import r2_score,confusion_matrix
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
```

In [56]:
```
X=titanic_data.drop('Survived',axis=1)
```

In [57]:
```
X = titanic_data.drop(columns=['PassengerId','Name','Ticket'],axis=1)
```

In [58]:
```
Y = titanic_data['Survived']
```

In [59]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=0)
```

In [60]:
```
print(X.shape,X_train.shape,X_test.shape)
```

```
(891, 8) (623, 8) (268, 8)
```

In [61]:
```
X.head()
```

Out[61]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22.0	1	0	7.2500	0.0
1	1	1	1	38.0	1	0	71.2833	1.0
2	1	3	1	26.0	0	0	7.9250	0.0
3	1	1	1	35.0	1	0	53.1000	0.0
4	0	3	0	35.0	0	0	8.0500	0.0

In [62]:
```
Y.head()
```

Out[62]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

In [63]:
```
model=LogisticRegression()
```

In [64]:
```
model.fit(X_train,Y_train)
```

Out[64]:
```
LogisticRegression
LogisticRegression()
```

In [65]:
```
X_train_prediction=model.predict(X_train)
```

In [66]:
```
print(X_train_prediction)
```

```
[1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0 1 0 0 1 1
 1 0 0 1 0 1 0 0 0 0 1 0 1 0 1 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0
 1 0 1 1 1 0 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 1
 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0
 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1
 1 0 0 0 1 0 1 0 0 1 0 0 1 1 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1
 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 1 0 0 0 1 1 1 1 0 0 0
 0 0 0 0 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 1 1 0 1 0 1
 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 0
 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1
 0 0 0 1 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 0
 0 0 1 0 0 0 0 1 1 1 0 0 1 1 1 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 1 0 1 1
 1 0 1 1 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 0 1 1 0 0 0 0
 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 0 0 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0
 0 0 0 0 1 1 1 1 1 1 1 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 1 1 0 0 0 1 1 0 1 1 0
 1 0 0 1 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1 0]
```

In [67]:
```
train_data_accuracy=r2_score(Y_train,X_train_prediction)
```

In [68]:
```
print("Accuracy Score of training data: ",train_data_accuracy)
```

```
Accuracy Score of training data:  1.0
```

In [69]:
```
X_test_prediction=model.predict(X_test)
```

In [70]:
```
print(X_test_prediction)
```

```
[0 0 0 1 1 1 1 1 1 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 0 1 0 1 1 0 0 1 0 1 0 1 0
 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 1 1 1 1 0 0
 0 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0 0 1 0
 1 1 0 1 1 1 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1
 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0
 0 1 0 0 1 0 0 1 1 0 0 0 1 1 0 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0 1
 1 0 1 0 0 1 1 0 0 1 1 0 0 0 1 1 1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1 1
 0 0 0 0 0 0 0 1 0]
```

In [71]:
```
test_data_accuracy=r2_score(Y_test,X_test_prediction)
```

In [72]:
```
print("Accuracy score of testing data:",test_data_accuracy)
```

```
Accuracy score of testing data: 1.0
```