PROJECT: TITANIC SURVIVAL PREDITION

Import Libraries

In [114]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_ma
%matplotlib inline
```

In [115]:
```
df = pd.read_csv('titanic_train.csv')
df.head()
```

Out[115]:

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

In [116]:
```
df.shape
```

Out[116]: (891, 12)

In [117]:
```
df.columns
```

Out[117]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [118]:
```
df.dtypes
```

Out[118]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [119]:
```
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
```

Data cleaning and preprocessing

Handling Null Values and Duplicates.

In [120]:
```
# Checking for null values
df.isnull().sum()
```

Out[120]:
```
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
```

In [121]:
```
df.drop(["PassengerId","Name","Ticket","Cabin"],axis=1,inplace=True)
```

Dealing with Null values

In [122]:
```
#Data Imputation in Age Column
df['Age']=df['Age'].fillna(df['Age'].mean())
```

In [123]:
```
df.dropna(inplace=True)
```

In [124]:
```
#Check to understand if there are any null values left in the dataset.
df.isnull().sum()
```

Out[124]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

In [125]:
```
#Value count for Survived
df['Survived'].value_counts()
```

Out[125]:
```
0    549
1    340
Name: Survived, dtype: int64
```

In [126]:
```
df.describe()
```

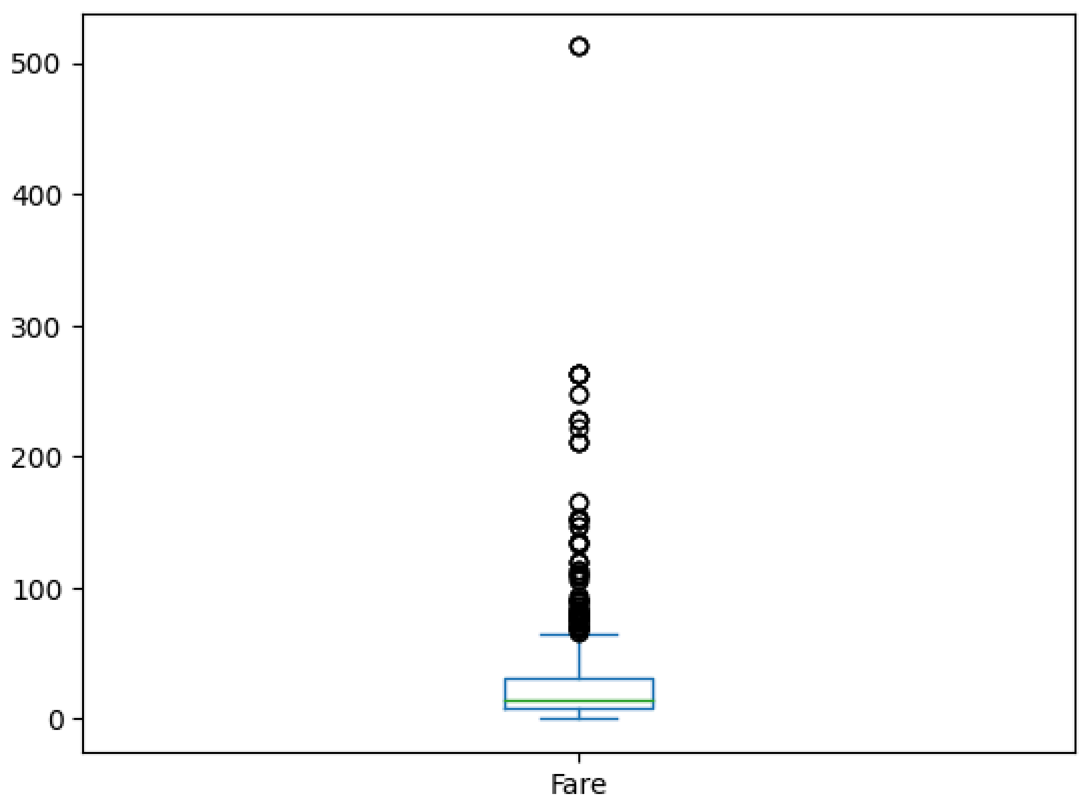
Out[126]:

	Survived	Pclass	Age	SibSp	Parch	Fare
count	889.000000	889.000000	889.000000	889.000000	889.000000	889.000000
mean	0.382452	2.311586	29.653446	0.524184	0.382452	32.096681
std	0.486260	0.834700	12.968366	1.103705	0.806761	49.697504
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	22.000000	0.000000	0.000000	7.895800
50%	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

As we can se above the standard deviation of Fare is very high,so it says that there are outliers present in the column.

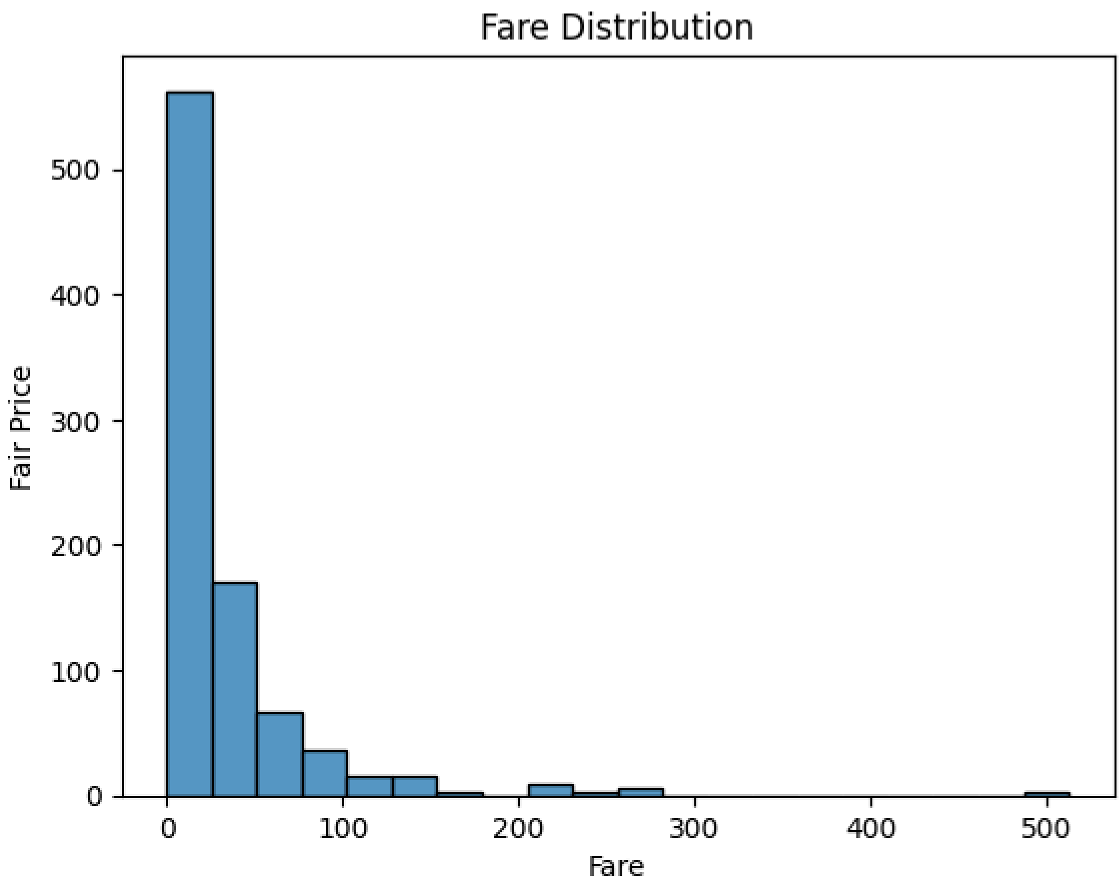
In [127]:
```
df['Fare'].plot(kind='box')
```

Out[127]: <Axes: >



In [128]:
```
sns.histplot(df['Fare'], bins=20)
plt.title("Fare Distribution")
plt.xlabel("Fare")
plt.ylabel("Fair Price")
```

Out[128]:
```
Text(0, 0.5, 'Fair Price')
```



The most fare from 0 to 100 but there is alot of outliers so we must detect the outliers

In [129]:
```
#Correlation coefficients.
df.corr()
```

```
C:\Users\99Minds-1\AppData\Local\Temp\ipykernel_3224\1095431909.py:2: FutureW
arning: The default value of numeric_only in DataFrame.corr is deprecated. In
a future version, it will default to False. Select only valid columns or spec
ify the value of numeric_only to silence this warning.
  df.corr()
```

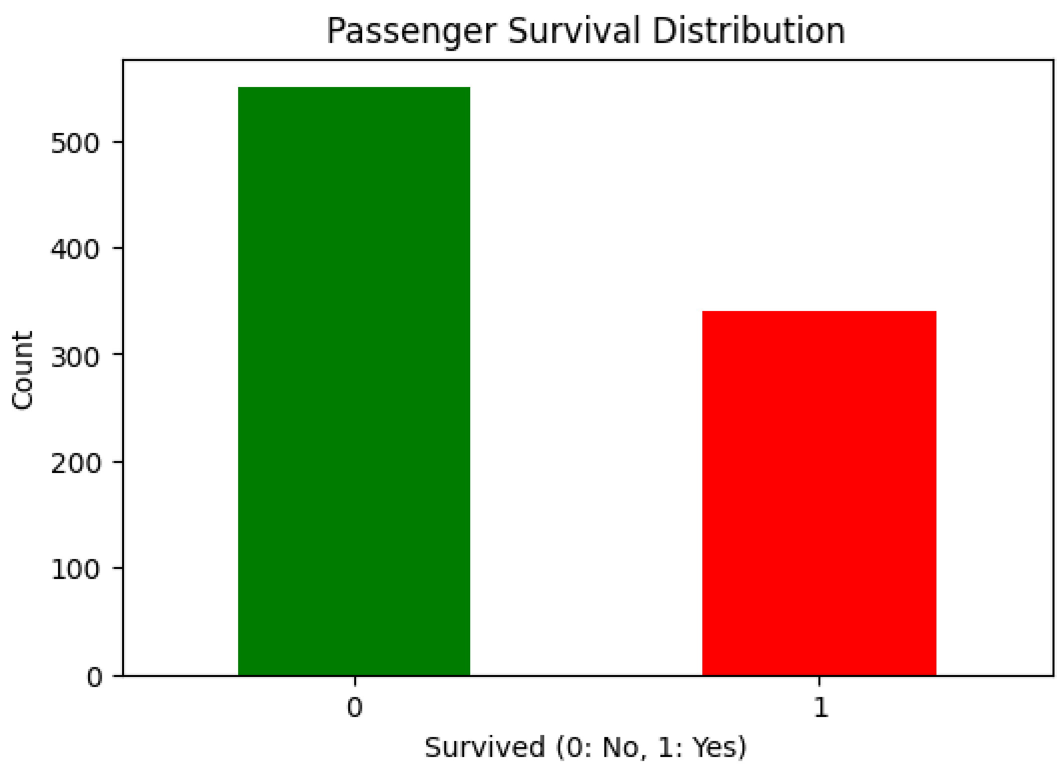
Out[129]:

	Survived	Pclass	Age	SibSp	Parch	Fare
Survived	1.000000	-0.335549	-0.074673	-0.034040	0.083151	0.255290
Pclass	-0.335549	1.000000	-0.327954	0.081656	0.016824	-0.548193
Age	-0.074673	-0.327954	1.000000	-0.231875	-0.178232	0.088604
SibSp	-0.034040	0.081656	-0.231875	1.000000	0.414542	0.160887
Parch	0.083151	0.016824	-0.178232	0.414542	1.000000	0.217532
Fare	0.255290	-0.548193	0.088604	0.160887	0.217532	1.000000

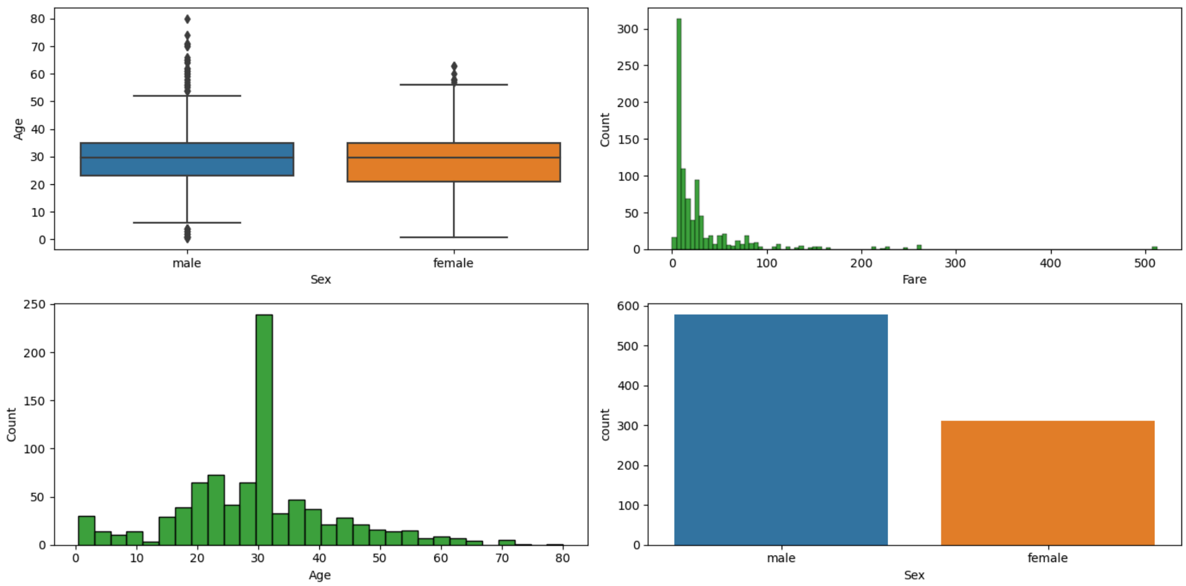
Correlation coefficients is applicable for numerical variables only - Age, SibSp, Parch and Fare. Pclass is an ordinal variable whereas Gender and Embarked are nominal variables. All numerical variables - Age, SibSp, Parch and Fare have weak correlation with target 'Survived'. However, Parch and Fare are better than other two numerical variables as these two have correlation coefficients greater than 0.1

Data Analysis & Visualization

In [130]:
```
plt.figure(figsize=(6, 4))
df['Survived'].value_counts().plot(kind='bar', color=['green', 'Red'])
plt.title('Passenger Survival Distribution')
plt.xlabel('Survived (0: No, 1: Yes)')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()
```

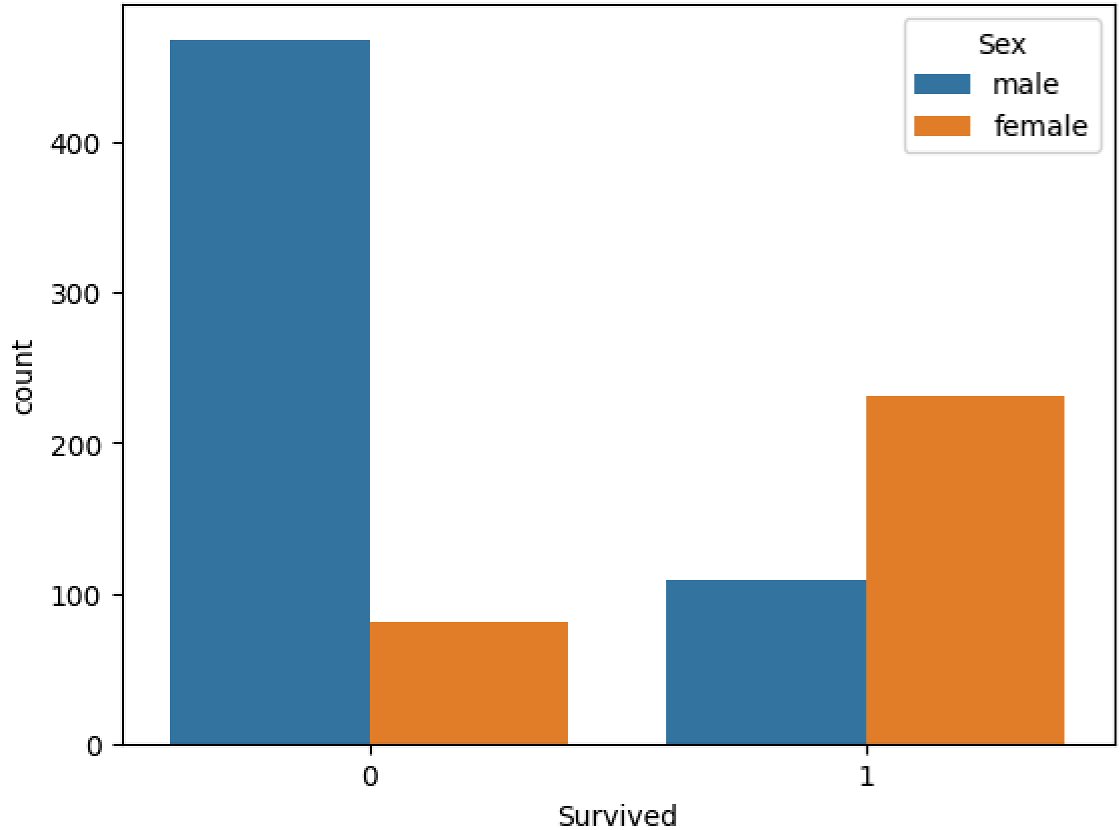


In [131]:
```
plt.figure(figsize=(14,7))
plt.subplot(2,2,1)
sns.boxplot(x='Sex', y = 'Age',data= df)

plt.subplot(2,2,2)
sns.histplot(df['Fare'],color='g')

plt.subplot(2,2,3)
sns.histplot(df['Age'],color='g')

plt.subplot(2,2,4)
sns.countplot(x='Sex', data=df)

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



In [132]:
```
#Countplot for Survived
sns.countplot(data=df,x="Survived",hue="Sex")
```

Out[132]: <Axes: xlabel='Survived', ylabel='count'>



In [133]:
```
df['Embarked'] = df['Embarked'].map( {'Q': 0,'S':1,'C':2}).astype(int)
df['Sex'] = df['Sex'].map( {'female': 1,'male':0}).astype(int)
```

In [134]:
```
df['Age'] = df['Age'].astype(int)
df['Fare'] = df['Fare'].astype(int)
```

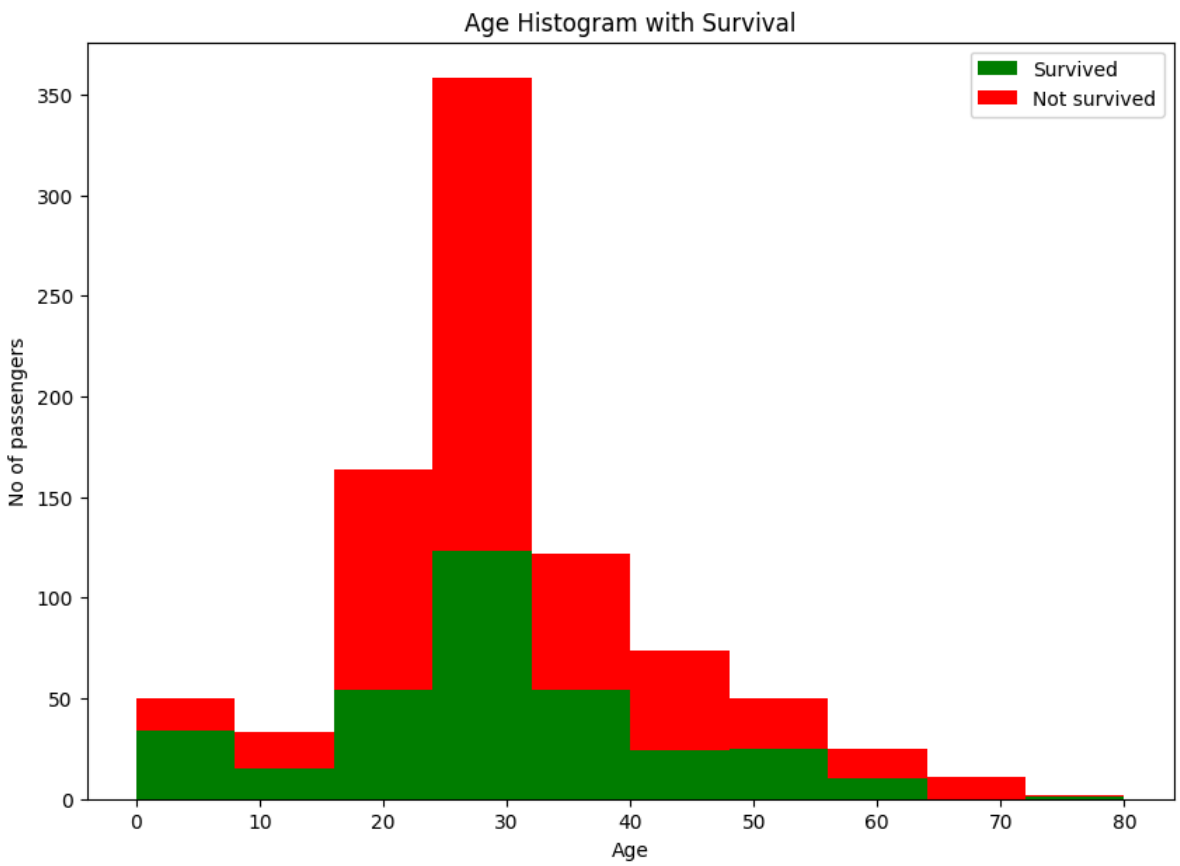
In [135]:
```
df.head()
```

Out[135]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22	1	0	7	1
1	1	1	1	38	1	0	71	2
2	1	3	1	26	0	0	7	1
3	1	1	1	35	1	0	53	1
4	0	3	0	35	0	0	8	1

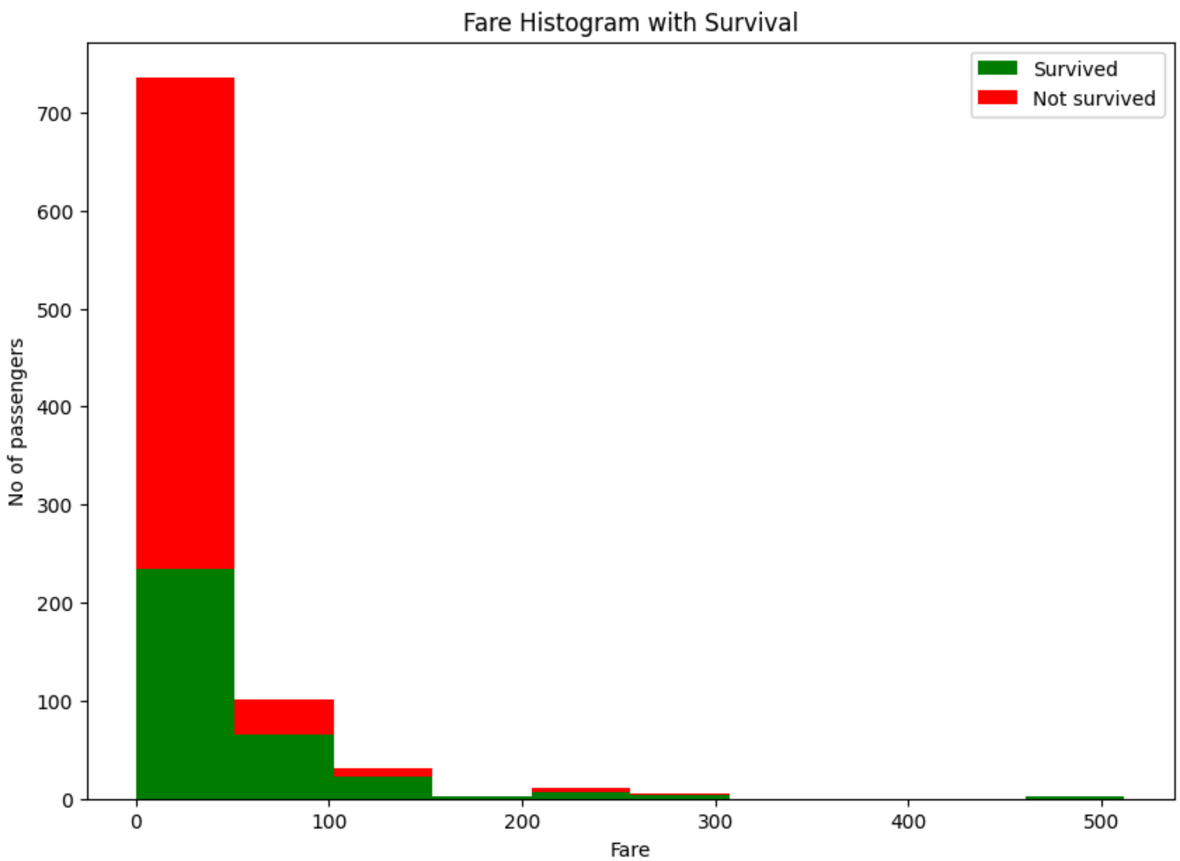
In [136]:
```
fig = plt.figure(figsize =(10, 7))
plt.hist(x = [df[df['Survived']==1]['Age'], df[df['Survived']==0]['Age']],stac
plt.title('Age Histogram with Survival')
plt.xlabel('Age')
plt.ylabel('No of passengers')
plt.legend()
```

Out[136]: <matplotlib.legend.Legend at 0x1b49d481890>

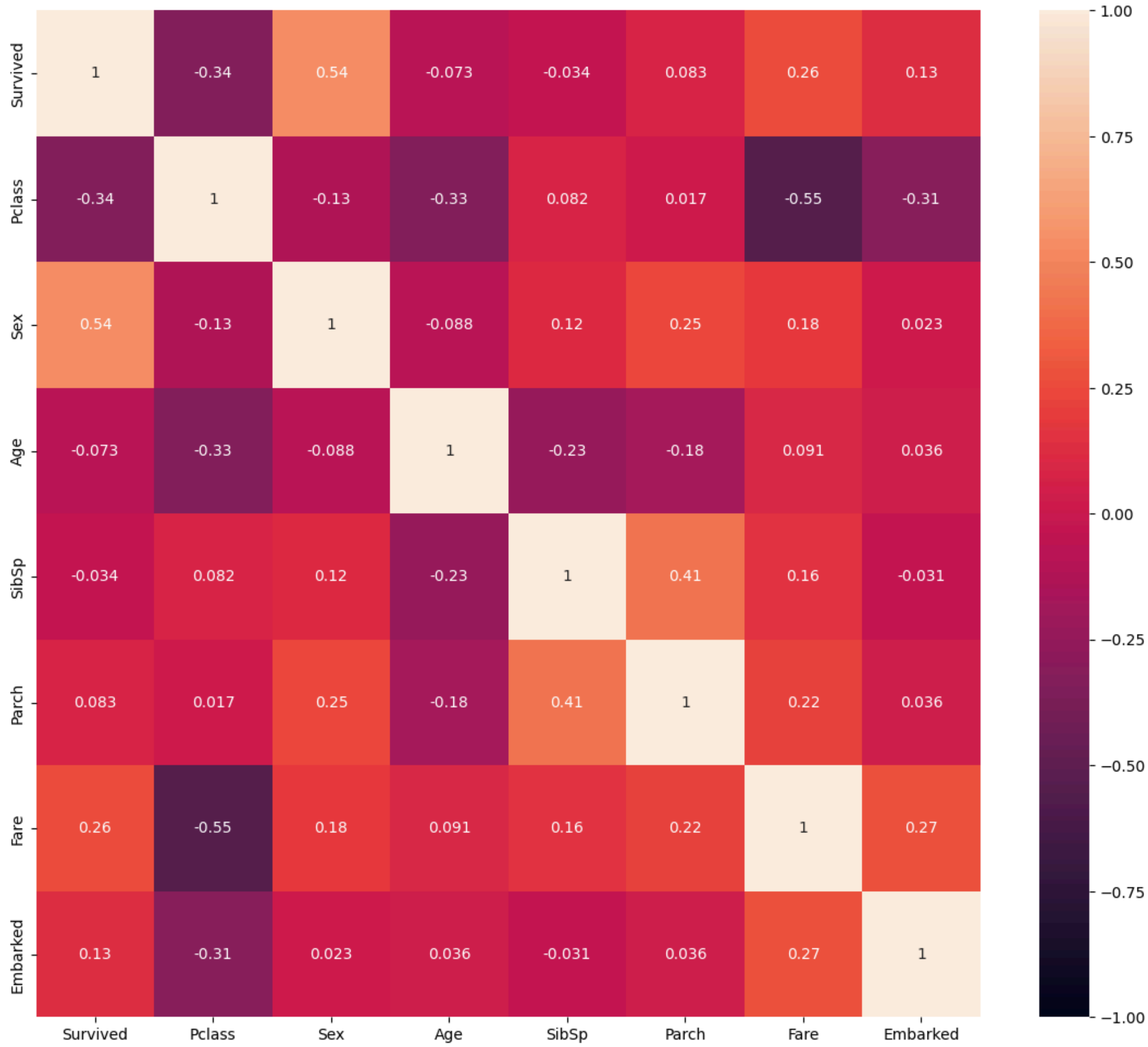


In [137]:
```
fig = plt.figure(figsize =(10, 7))
plt.hist(x = [df[df['Survived']==1]['Fare'], df[df['Survived']==0]['Fare']], s
plt.title('Fare Histogram with Survival')
plt.xlabel('Fare')
plt.ylabel('No of passengers')
plt.legend()
```

Out[137]: <matplotlib.legend.Legend at 0x1b49d312e90>



In [138]:
```
plt.figure(figsize=(14,12))
sns.heatmap(df.corr(), annot=True, vmin=-1.0, vmax=1.0)
plt.show()
```



Detecting & Removing Outliers

In [139]:
```
Q1= df['Fare'].quantile(0.25)
Q3=df['Fare'].quantile(0.75)

IQR=Q3-Q1

upper=Q3 + 1.5*IQR
Lower=Q1 - 1.5*IQR

print(upper)
print(Lower)
```

```
67.0
-29.0
```

There is alot of upper outliers and we are going to point them out

In [140]:
```
df[df["Fare"]>65]
```

Out[140]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
1	1	1	1	38	1	0	71	2
27	0	1	0	19	3	2	263	1
31	1	1	1	29	1	0	146	2
34	0	1	0	28	1	0	82	2
52	1	1	1	49	1	0	76	2
...	...	...	...	...	...	...	...	...
846	0	3	0	29	8	2	69	1
849	1	1	1	29	1	0	89	2
856	1	1	1	45	1	1	164	1
863	0	3	1	29	8	2	69	1
879	1	1	1	56	0	1	83	2

114 rows × 8 columns

There are 114 People that paid higher than average amount of Fare which can be understood because people who paid alot more than average may be because they reserved later and so on to get all survival facilities.

In [141]:
```
new_df = df[df['Fare'] < upper]
new_df
```

Out[141]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22	1	0	7	1
2	1	3	1	26	0	0	7	1
3	1	1	1	35	1	0	53	1
4	0	3	0	35	0	0	8	1
5	0	3	0	29	0	0	8	0
...	...	...	...	...	...	...	...	...
886	0	2	0	27	0	0	13	1
887	1	1	1	19	0	0	30	1
888	0	3	1	29	1	2	23	1
889	1	1	0	26	0	0	30	2
890	0	3	0	32	0	0	7	0

777 rows × 8 columns

Data Modeling

In [142]:
```
X = new_df.drop(['Survived'], axis=1)
y = new_df.iloc[:,1]
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, ran
```

Logistic Regression

In [143]:
```
LR = LogisticRegression(solver='liblinear', max_iter=200)
LR.fit(x_train, y_train)
y_pred = LR.predict(x_test)
LRAcc = accuracy_score(y_pred,y_test)
print('Logistic regression accuracy: {:.2f}%'.format(LRAcc*100))
```

```
Logistic regression accuracy: 95.51%
```

In [144]:
```
confuion_matrix = confusion_matrix(y_test, y_pred)
confuion_matrix
```

Out[144]:
```
array([[21,  1,  0],
       [ 0, 29,  6],
       [ 0,  0, 99]], dtype=int64)
```

Evaluation of Model

In [145]:
```
accuracy = accuracy_score(y_test, y_pred)

classification_rep = classification_report(y_test, y_pred)

print("Accuracy:", accuracy," - ",round(accuracy*100,2),"%")

print("Classification Report:\n", classification_rep)
```

```
Accuracy: 0.9551282051282052  -  95.51 %
Classification Report:
               precision    recall  f1-score   support

           1       1.00      0.95      0.98        22
           2       0.97      0.83      0.89        35
           3       0.94      1.00      0.97        99

    accuracy                           0.96       156
   macro avg       0.97      0.93      0.95       156
weighted avg       0.96      0.96      0.95       156
```

Building a Predictive System

In [146]:
```
input_data = (3,0,22,1,0,7,1)

# changing the input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the data as we are predicting the label for only one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = LR.predict(input_data_reshaped)
print(prediction)

if (prediction[0]==1):
  print('Survived')
else:
  print('Not Survived')
```

```
[3]
Not Survived

C:\Users\99Minds-1\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\base.py:439: UserWarning: X does not have valid feature names, but L
ogisticRegression was fitted with feature names
  warnings.warn(
```

Thankyou

In [ ]: