Import Necessary Packages

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
data = pd.read_csv("titanic_data.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	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 [4]:
```
data.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 [5]:
```
data.describe()
```

Out[5]:

	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

Find The NULL Values

In [6]:
```
data.isna().sum()
```

Out[6]:
```
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
```

Clean NULL values

In [7]:
```
age_mean = data["Age"].mean()
```

In [8]:
```
data["Age"] = data["Age"].fillna(age_mean)
```

In [9]:
```
data.isna().sum()
```

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

In [10]:
```
data["Embarked"] = data["Embarked"].fillna("S")
```

In [11]:
```
data.isna().sum()
```

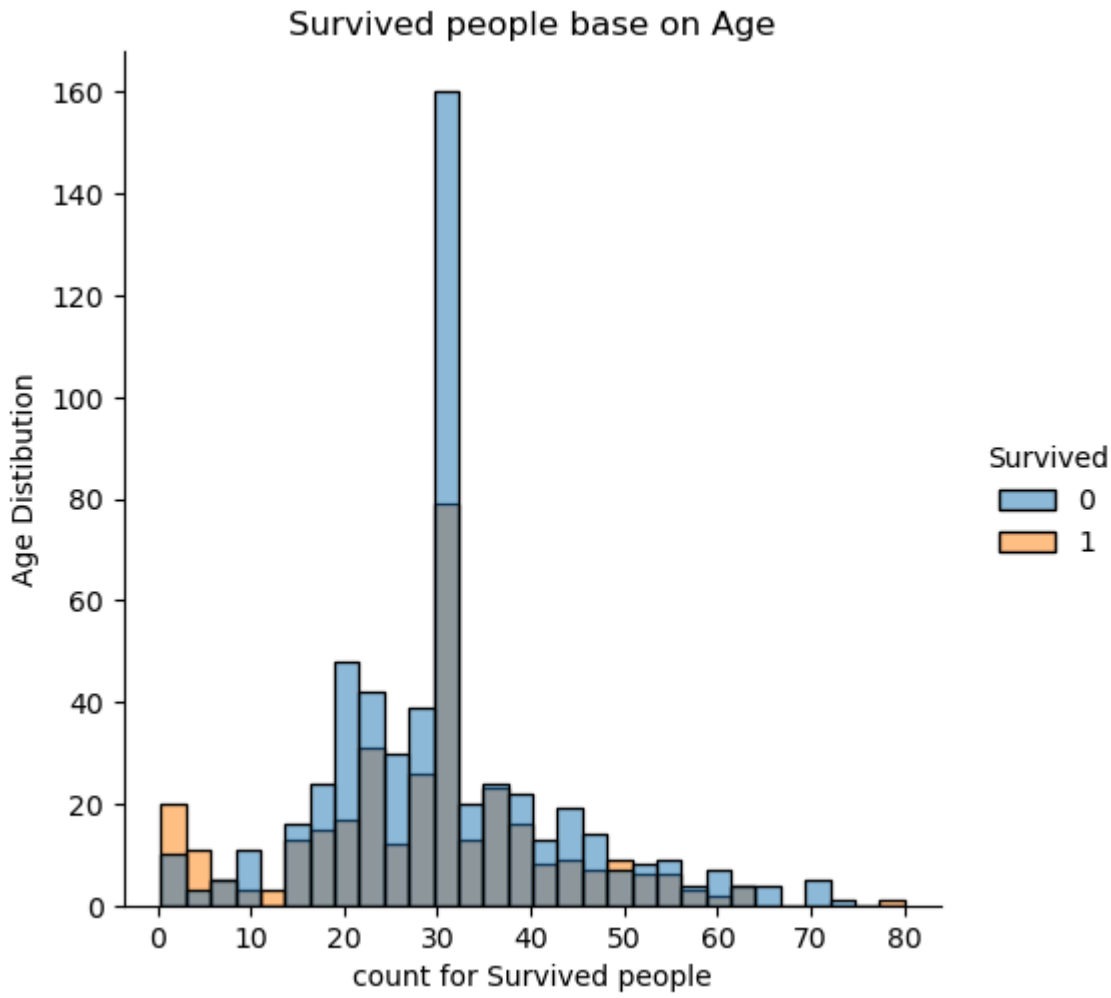
Out[11]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         0
dtype: int64
```

Perform Analysis

In [12]:
```
sns.displot(x = data["Age"] , hue = data["Survived"] , color = "red")
plt.xlabel("count for Survived people")
plt.ylabel("Age Distibution")
plt.title("Survived people base on Age")
```

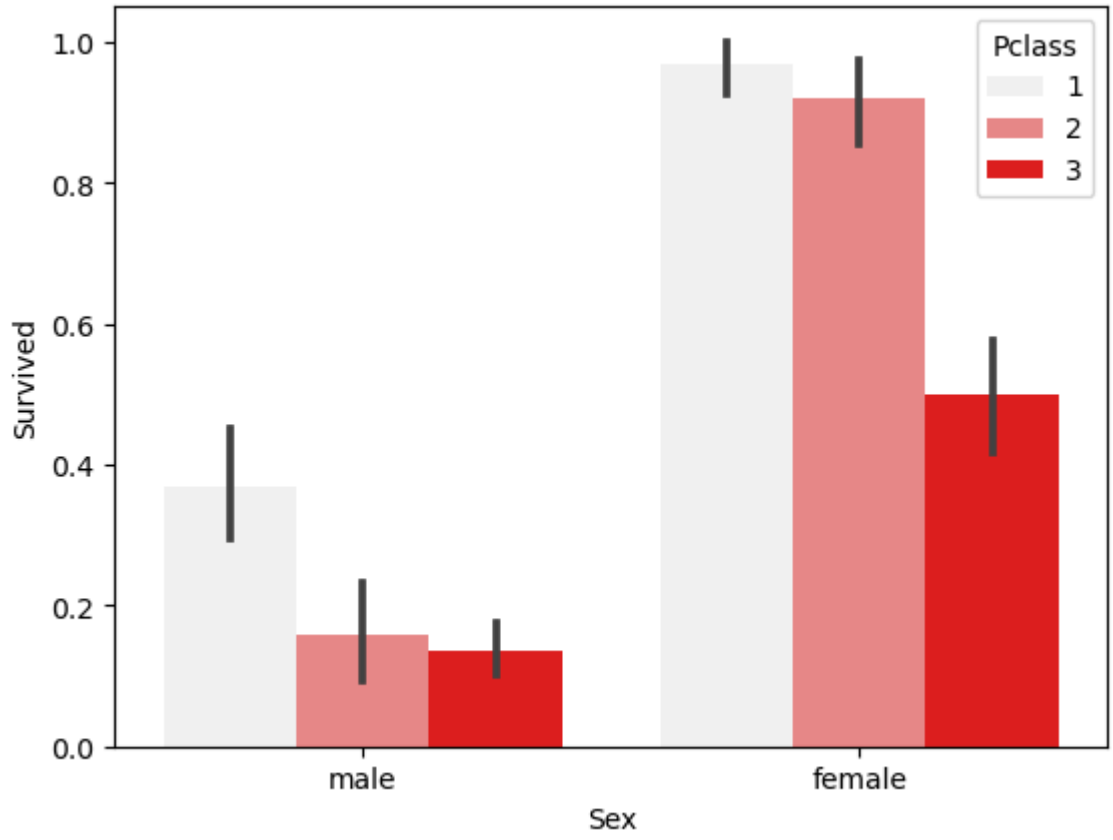
```
C:\Users\godde\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: T
he figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[12]:
```
Text(0.5, 1.0, 'Survived people base on Age')
```



In [13]:
```
sns.barplot(data = data , x = "Sex" , y = "Survived" , hue = "Pclass" , color = "re
```

Out[13]:
```
<Axes: xlabel='Sex', ylabel='Survived'>
```



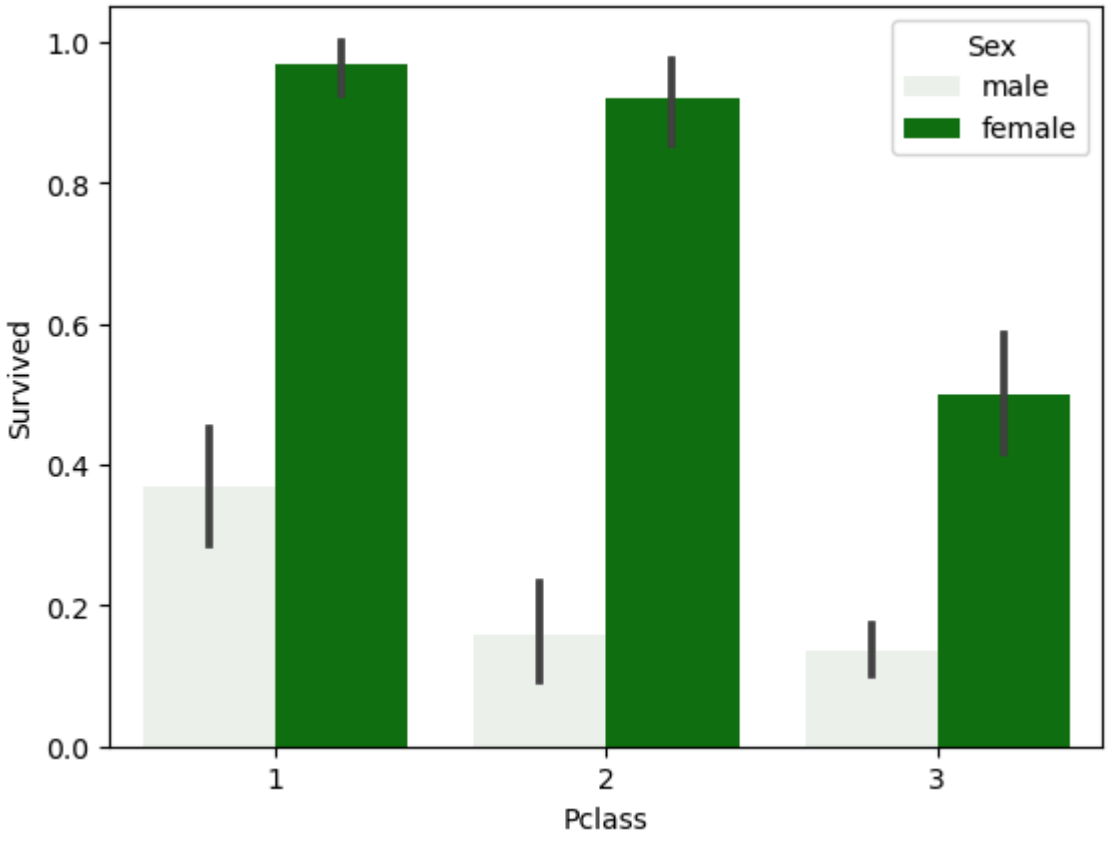
In [14]:
```
data.head()
```

Out[14]:

	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

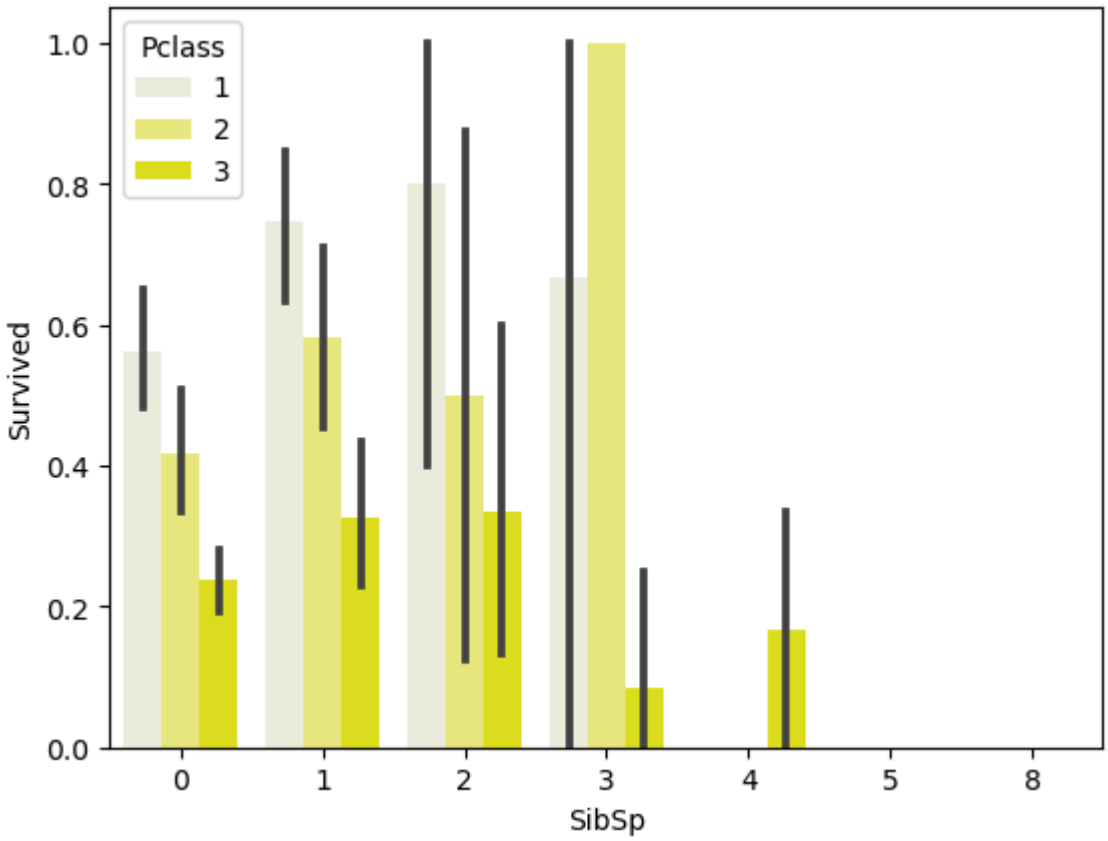
In [15]:
```
sns.barplot(data = data , x = "Pclass" , y = "Survived" , hue = "Sex" , color = "gr
```

Out[15]:
```
<Axes: xlabel='Pclass', ylabel='Survived'>
```



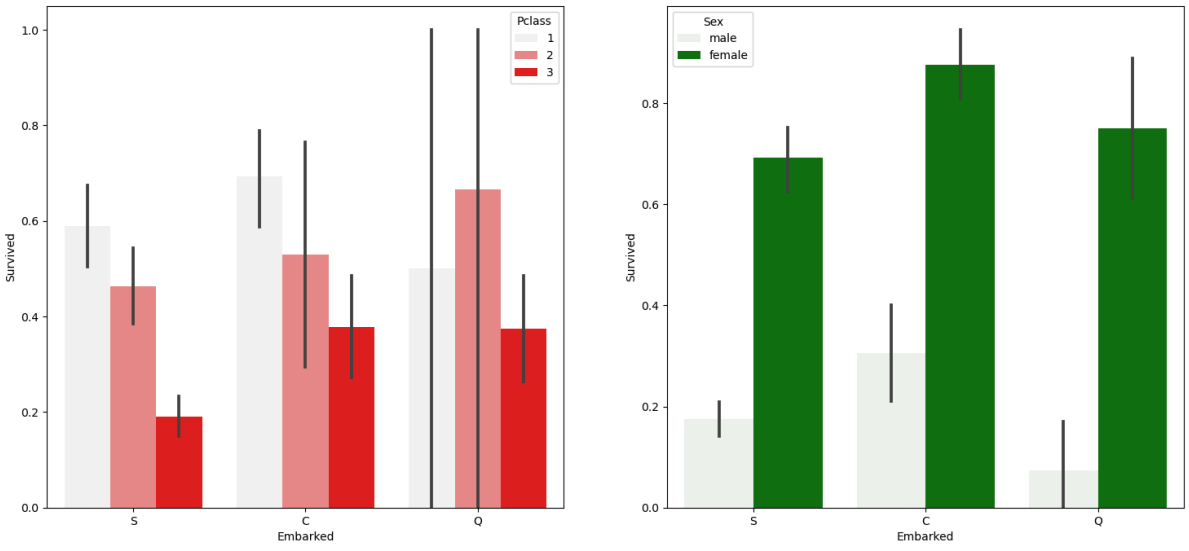
In [16]:
```
sns.barplot(data = data , x = "SibSp" , y = "Survived" , hue = "Pclass" , color = '
```

Out[16]:
```
<Axes: xlabel='SibSp', ylabel='Survived'>
```



In [17]:
```
plt.figure(figsize = (18,8))
plt.subplot(1,2,1)
sns.barplot(data = data , x = "Embarked" , y = "Survived" , hue = "Pclass" , color
plt.subplot(1,2,2)
sns.barplot(data = data , x = "Embarked" , y = "Survived" , hue = "Sex" , color = '
```

Out[17]:
```
<Axes: xlabel='Embarked', ylabel='Survived'>
```



In [18]:
```
data.head()
```

Out[18]:

	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 [19]:
```
sex = {"male" : 1 , "female" : 0}
data["Sex"] = data["Sex"].map(sex)
Embarked = {"S" : 1 , "C" : 2 , "Q" : 3}
data["Embarked"] = data ["Embarked"].map(Embarked)
```

In [20]:
```
data.head()
```

Out[20]:

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

In [21]:
```
sns.heatmap(data[["Survived" , "Pclass" , "Sex" , "Age" , "SibSp" , "Fare" , "Embar
```

Out[21]:
```
<Axes: >
```

In [22]:
```
data = data.drop(["PassengerId" , "Name" , "Parch" , "Ticket" , "Cabin"] , axis = 1
```

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

Out[23]:

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

Spliting Data

In [26]:
```
x = data.drop("Survived" , axis = 1)
y = data[["Survived"]]
```

In [27]:
```
x
```

Out[27]:

	Pclass	Sex	Age	SibSp	Fare	Embarked
0	3	1	22.000000	1	7.2500	1
1	1	0	38.000000	1	71.2833	2
2	3	0	26.000000	0	7.9250	1
3	1	0	35.000000	1	53.1000	1
4	3	1	35.000000	0	8.0500	1
...	...	...	...	...	...	...
886	2	1	27.000000	0	13.0000	1
887	1	0	19.000000	0	30.0000	1
888	3	0	29.699118	1	23.4500	1
889	1	1	26.000000	0	30.0000	2
890	3	1	32.000000	0	7.7500	3

891 rows × 6 columns

In [28]:
```
y
```

Out[28]:

	Survived
0	0
1	1
2	1
3	1
4	0
...	...
886	0
887	1
888	0
889	1
890	0

891 rows × 1 columns

In [29]:
```
x_train , x_test , y_train , y_test = train_test_split(x , y , test_size = 0.2 , ra
```

In [30]:
```
x_train
```

Out[30]:

	Pclass	Sex	Age	SibSp	Fare	Embarked
711	1	1	29.699118	0	26.5500	1
525	3	1	40.500000	0	7.7500	3
76	3	1	29.699118	0	7.8958	1
626	2	1	57.000000	0	12.3500	3
159	3	1	29.699118	8	69.5500	1
...	...	...	...	...	...	...
742	1	0	21.000000	2	262.3750	2
528	3	1	39.000000	0	7.9250	1
74	3	1	32.000000	0	56.4958	1
176	3	1	29.699118	3	25.4667	1
338	3	1	45.000000	0	8.0500	1

712 rows × 6 columns

In [31]:
```
x_test
```

Out[31]:

	Pclass	Sex	Age	SibSp	Fare	Embarked
736	3	0	48.000000	1	34.3750	1
421	3	1	21.000000	0	7.7333	3
442	3	1	25.000000	1	7.7750	1
196	3	1	29.699118	0	7.7500	3
200	3	1	28.000000	0	9.5000	1
...	...	...	...	...	...	...
603	3	1	44.000000	0	8.0500	1
750	2	0	4.000000	1	23.0000	1
185	1	1	29.699118	0	50.0000	1
644	3	0	0.750000	2	19.2583	2
815	1	1	29.699118	0	0.0000	1

179 rows × 6 columns

In [32]:
```
y_train
```

Out[32]:

	Survived
711	0
525	0
76	0
626	0
159	0
...	...
742	1
528	0
74	1
176	0
338	1

712 rows × 1 columns

In [33]:
```
y_test
```

Out[33]:

	Survived
736	0
421	0
442	0
196	0
200	0
...	...
603	0
750	1
185	0
644	1
815	0

179 rows × 1 columns

In [34]:
```
scalar = StandardScaler()
scalar
```

Out[34]:
```
StandardScaler
StandardScaler()
```

In [35]:
```
x_train_scalar = scalar.fit_transform(x_train)
x_test_scalar = scalar.transform(x_test)
```

In [37]:
```
x_train_scalar
```

Out[37]:
```
array([[-1.53184784,  0.71980808, -0.00277682, -0.45812021, -0.12304699,
        -0.56364944],
       [ 0.84435186,  0.71980808,  0.8219131 , -0.45812021, -0.48463015,
         2.63410674],
       [ 0.84435186,  0.71980808, -0.00277682, -0.45812021, -0.48182595,
        -0.56364944],
       ...,
       [ 0.84435186,  0.71980808,  0.17290461, -0.45812021,  0.45290496,
        -0.56364944],
       [ 0.84435186,  0.71980808, -0.00277682,  2.1237921 , -0.14388226,
        -0.56364944],
       [ 0.84435186,  0.71980808,  1.16550582, -0.45812021, -0.4788602 ,
        -0.56364944]])
```

In [38]:
```
x_test_scalar
```

Out[38]:
```
array([[ 0.84435186, -1.38925921,  1.39456764,  0.40251723,  0.02745238,
        -0.56364944],
       [ 0.84435186,  0.71980808, -0.66698872, -0.45812021, -0.48495134,
         2.63410674],
       [ 0.84435186,  0.71980808, -0.36157296,  0.40251723, -0.48414932,
        -0.56364944],
       ...,
       [-1.53184784,  0.71980808, -0.00277682, -0.45812021,  0.32797029,
        -0.56364944],
       [ 0.84435186, -1.38925921, -2.21315599,  1.26315467, -0.26328933,
         1.03522865],
       [-1.53184784,  0.71980808, -0.00277682, -0.45812021, -0.63368703,
        -0.56364944]])
```

Define Models

In [57]:
```
logistic = LogisticRegression()
Support_vector = SVC()
Tree = DecisionTreeClassifier()
Random_forest = RandomForestClassifier(n_estimators=1000)
KNN = KNeighborsClassifier()
```

In [58]:
```
print("model for LogisticRegression                    =  " , logistic)
print("model for support vector machine                =  " ,Support_vector)
print("model for Decision tree                         =  " , Tree)
print("model for Random forest                         =  " , Random_forest)
print("model for K nearst neghbor                      =  " , KNN)
```

```
model for LogisticRegression                    =   LogisticRegression()
model for support vector machine                =   SVC()
model for Decision tree                         =   DecisionTreeClassifier()
model for Random forest                         =   RandomForestClassifier(n_estim
ators=1000)
model for K nearst neghbor                      =   KNeighborsClassifier()
```

In [59]:
```
logistic_model = logistic.fit(x_train_scalar , y_train)
SVC_model = Support_vector.fit(x_train_scalar, y_train)
tree_model = Tree.fit(x_train_scalar , y_train)
random_forest_model = Random_forest.fit(x_train , y_train)
KNN_model = KNN.fit(x_train_scalar , y_train)
```

```
C:\Users\godde\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Pleas
e change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\godde\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Pleas
e change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\godde\anaconda3\Lib\site-packages\sklearn\base.py:1151: DataConversionWar
ning: A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\godde\anaconda3\Lib\site-packages\sklearn\neighbors\_classification.py:22
8: DataConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

Predictions

In [60]:
```
logistic_prediction = logistic_model.predict(x_test_scalar)
SVC_prediction = SVC_model.predict(x_test_scalar)
tree_prediction = tree_model.predict(x_test_scalar)
random_forest_prediction = random_forest_model.predict(x_test_scalar)
KNN_model_prediction = KNN_model.predict(x_test_scalar)
```

```
C:\Users\godde\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X doe
s not have valid feature names, but RandomForestClassifier was fitted with feature
names
  warnings.warn(
```

LogisticRegression Prediction

In [61]:
```
logistic_prediction
```

Out[61]:
```
array([0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 1, 0], dtype=int64)
```

SVC Algoritham prediction

In [62]:
```
SVC_prediction
```

Out[62]:
```
array([0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 1, 0], dtype=int64)
```

Decision Tree Algoritham prediction

In [63]:
```
tree_prediction
```

Out[63]:
```
array([1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       0, 1, 0], dtype=int64)
```

Random Forest Algoritham Prediction

In [64]:
```
random_forest_prediction
```

Out[64]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1], dtype=int64)
```

KNN Algoritham prediction

In [65]:
```
KNN_model_prediction
```

Out[65]:
```
array([0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
       0, 1, 0], dtype=int64)
```

Cost Function for All Above algorithams

In [69]:
```
logistic_model_error = mean_squared_error(logistic_prediction , y_test)
SVC_model_error = mean_squared_error(SVC_prediction , y_test)
random_forest_error = mean_squared_error(random_forest_prediction , y_test)
decision_tree_error = mean_squared_error(tree_prediction , y_test)
KNN_model_error = mean_squared_error(KNN_model_prediction , y_test)
```

In [71]:
```
print("Cost Function For Logistic regression                   = "  ,logistic_model_
print("Cost Function For  SVC ALgoritham Prediction            ="  ,SVC_model_error)
print("Cost Function For Random Forest Algoritham              = "  ,random_forest_e
print("Cost Function For Decision Tree Algoritham              ="  ,decision_tree_er
print("Cost Function For KNN Algoritham                        =" ,KNN_model_error)
```

```
Cost Function For Logistic regression                   =  0.16759776536312848
Cost Function For  SVC ALgoritham Prediction            = 0.1564245810055866
Cost Function For Random Forest Algoritham              =  0.6089385474860335
Cost Function For Decision Tree Algoritham              = 0.2569832402234637
Cost Function For KNN Algoritham                        = 0.16759776536312848
```

In [ ]:

In [72]:
```
data.head()
```

Out[72]:

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

In [77]:
```
x_test_scalar
```

Out[77]:
```
array([[ 0.84435186, -1.38925921,  1.39456764,  0.40251723,  0.02745238,
        -0.56364944],
       [ 0.84435186,  0.71980808, -0.66698872, -0.45812021, -0.48495134,
         2.63410674],
       [ 0.84435186,  0.71980808, -0.36157296,  0.40251723, -0.48414932,
        -0.56364944],
       ...,
       [-1.53184784,  0.71980808, -0.00277682, -0.45812021,  0.32797029,
        -0.56364944],
       [ 0.84435186, -1.38925921, -2.21315599,  1.26315467, -0.26328933,
         1.03522865],
       [-1.53184784,  0.71980808, -0.00277682, -0.45812021, -0.63368703,
        -0.56364944]])
```

In [ ]:

In [ ]: