In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df = pd.read_csv("titanic_train.csv")
df.head()
```

Out[2]:

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

In [3]:

```
df = pd.read_csv("titanic_train.csv")
df.head()
```

Out[3]:

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

In [4]:

```
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 [5]:

```
1 df.describe()
```

Out[5]:

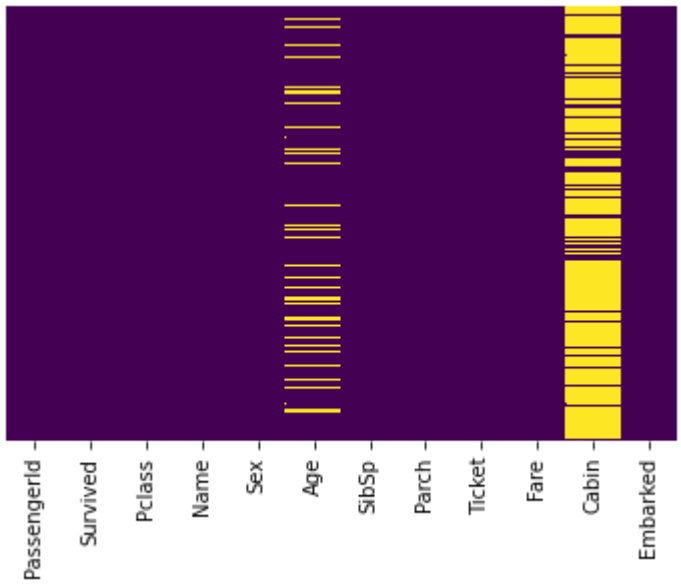
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	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

In [6]:

```
1 sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap="viridis")
```

Out[6]:

```
<AxesSubplot:>
```

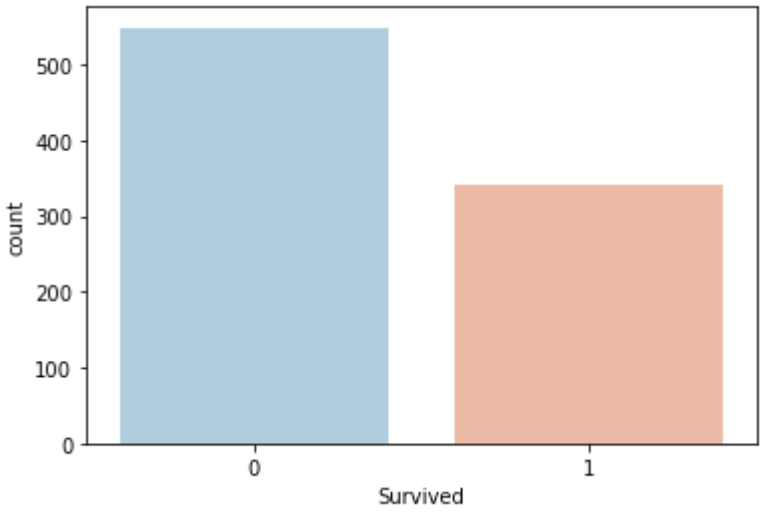


In [7]:

```
1 sns.countplot(data=df, x="Survived", palette="RdBu_r")
```

Out[7]:

```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```

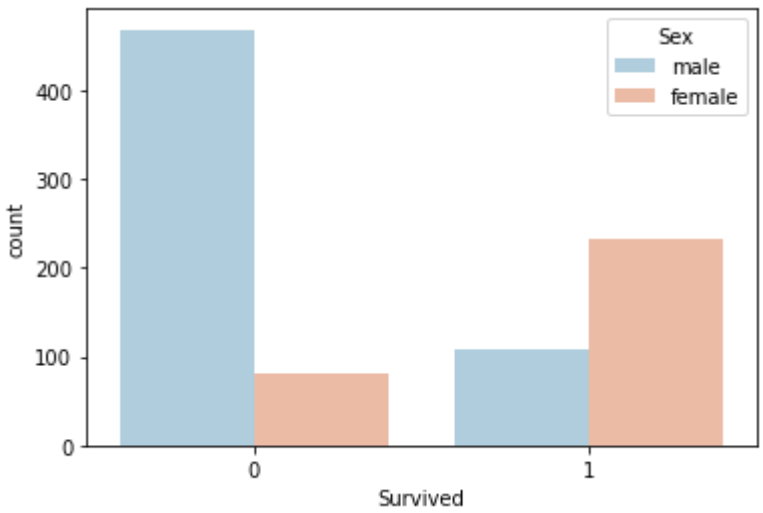


In [8]:

```
1 sns.countplot(x="Survived", hue="Sex", data=df, palette="RdBu_r")
```

Out[8]:

```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```

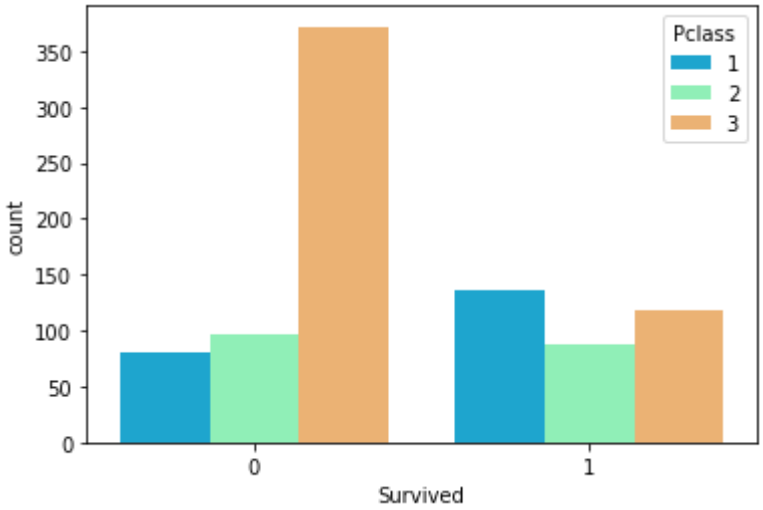


In [9]:

```
1 sns.countplot(data=df, x="Survived", hue="Pclass", palette="rainbow")
```

Out[9]:

```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```

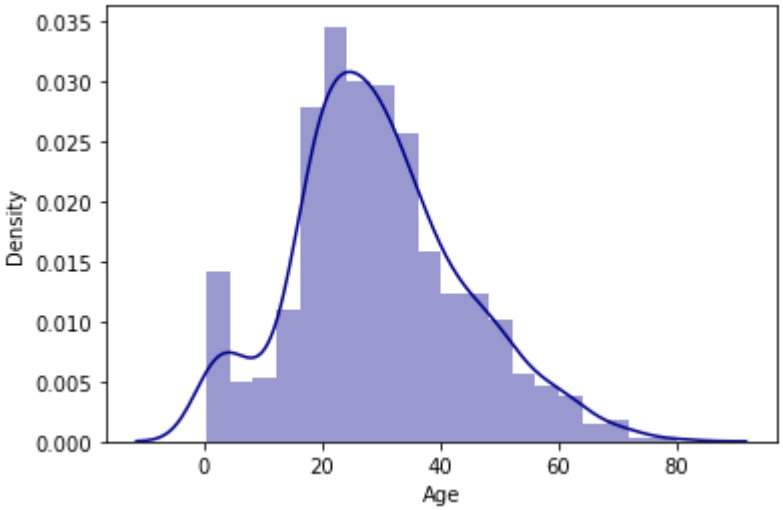


In [10]:

```
1 sns.distplot(df["Age"].dropna(), color="darkblue")
```

Out[10]:

```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```

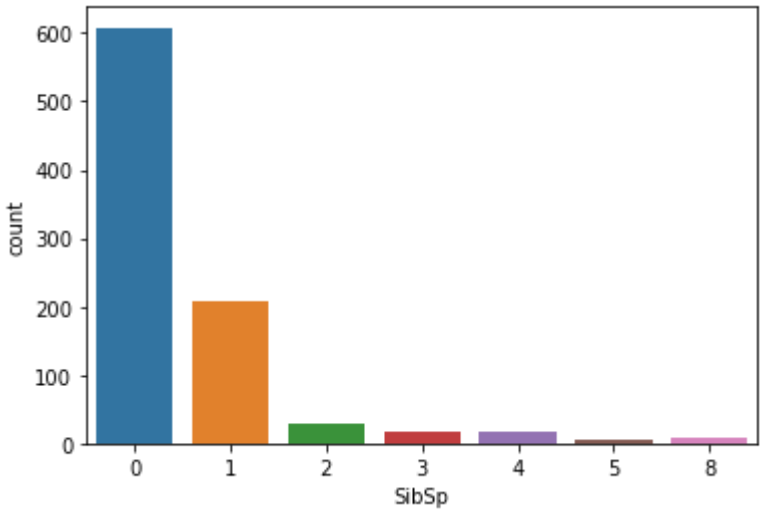


In [11]:

```
1 sns.countplot(data=df, x="SibSp")
```

Out[11]:

```
<AxesSubplot:xlabel='SibSp', ylabel='count'>
```

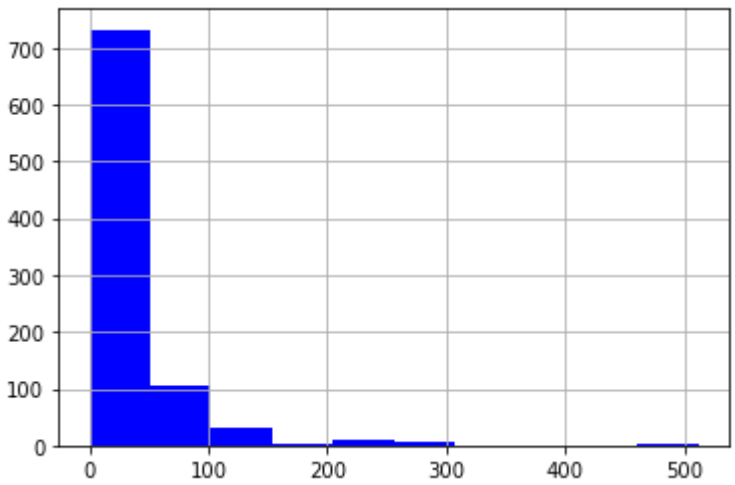


In [12]:

```
1 df["Fare"].hist(color="blue")
```

Out[12]:

```
<AxesSubplot:>
```

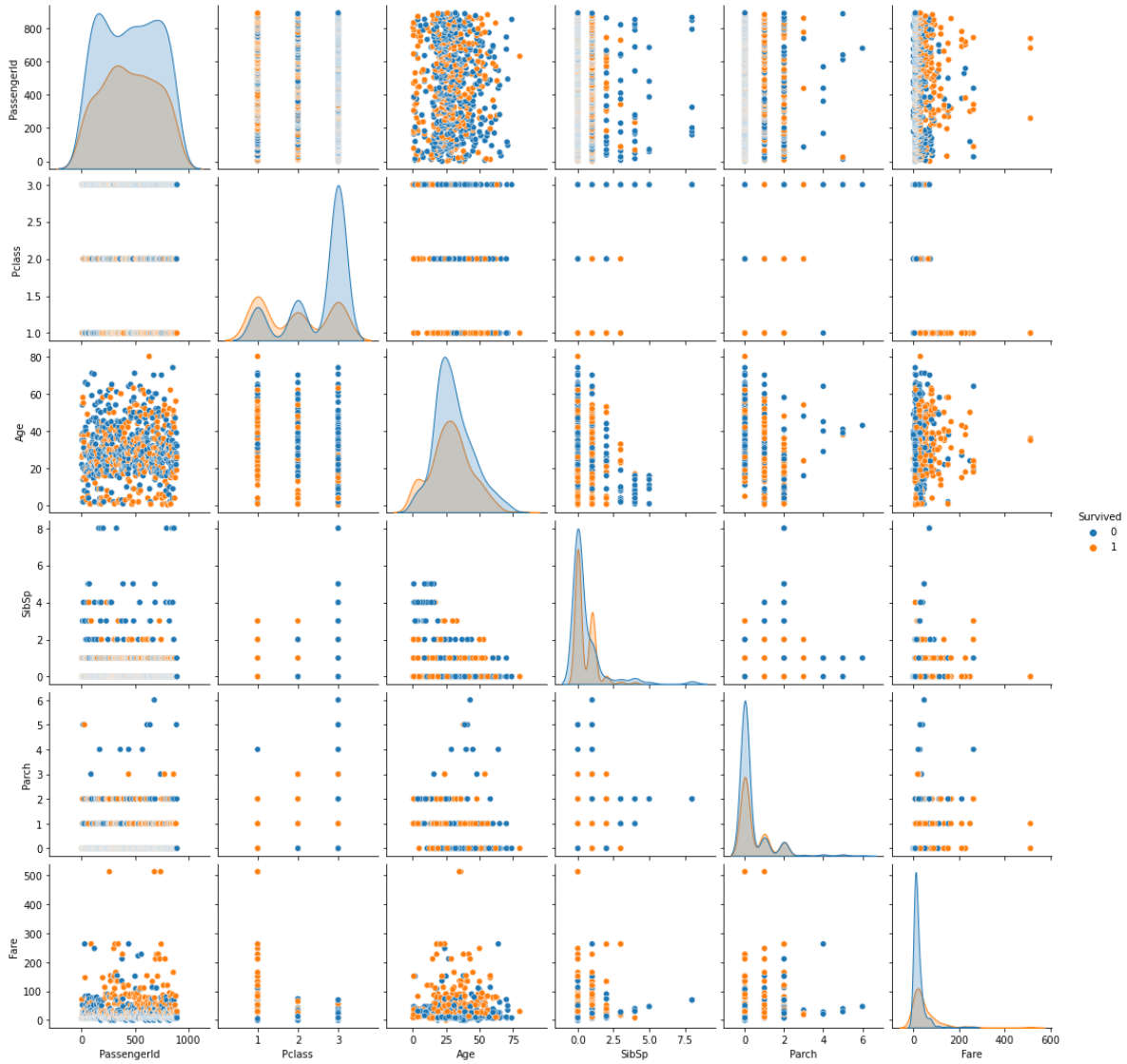


In [13]:

```
1 sns.pairplot(df, hue="Survived")
```

Out[13]:

```
<seaborn.axisgrid.PairGrid at 0x4884f543d0>
```



In [14]:

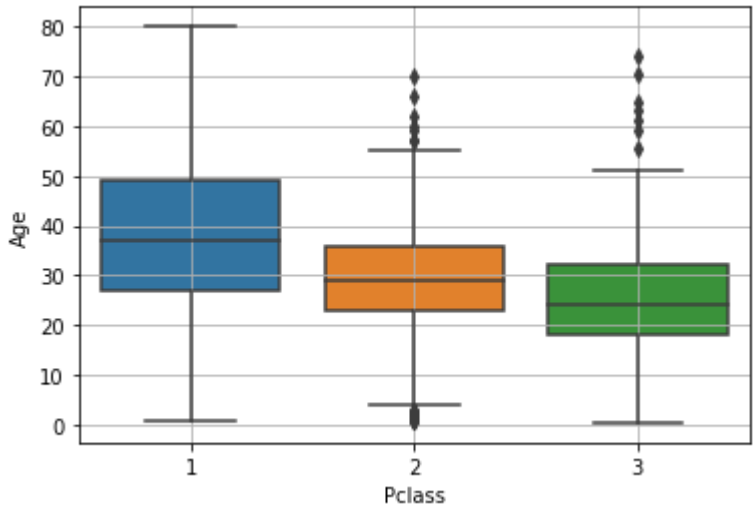
```
df.corr().style.background_gradient(cmap="coolwarm")
```

Out[14]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

In [15]:

```
sns.boxplot(data=df, x="Pclass", y="Age")
plt.grid(True)
```



In [16]:

```
def fillage(cols):
    Age = cols[0]
    Pclass = cols[0]
    if(pd.isnull(Age)):
        if(Pclass==1):
            return 38
        elif(Pclass==2):
            return 29
        else:
            return 24
    else:
        return Age
```

In [17]:

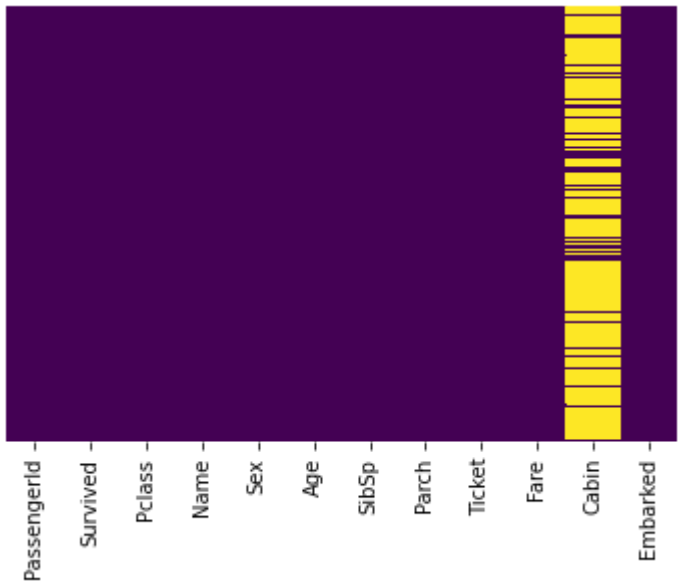
```
df["Age"] = df[["Age", "Pclass"]].apply(fillage, axis=1)
```

In [18]:

```
1 sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap="viridis")
```

Out[18]:

<AxesSubplot:>



In [19]:

```
1 df.drop("Cabin", axis=1, inplace=True)
```

In [20]:

```
1 df.dropna(inplace=True)
```

In [21]:

```
1 df.isna().sum()
```

Out[21]:

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [22]:

```
1 df.head()
```

Out[22]:

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

In [23]:

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

In [24]:

```
1 df.head()
```

Out[24]:

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

In [25]:

```
1 x= df.iloc[:, 1:]
2 y = df.iloc[:,0]
```

In [26]:

```
1 x
```

Out[26]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.0	1	0	7.2500	S
1	1	female	38.0	1	0	71.2833	C
2	3	female	26.0	0	0	7.9250	S
3	1	female	35.0	1	0	53.1000	S
4	3	male	35.0	0	0	8.0500	S
...	...	...	...	...	...	...	...
886	2	male	27.0	0	0	13.0000	S
887	1	female	19.0	0	0	30.0000	S
888	3	female	24.0	1	2	23.4500	S
889	1	male	26.0	0	0	30.0000	C
890	3	male	32.0	0	0	7.7500	Q

889 rows × 7 columns

In [27]:

```
1 y
```

Out[27]:

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 889, dtype: int64
```

In [28]:

```
from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

from sklearn.model_selection import train_test_split, cross_val_score

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification_report, accuracy_score

```

In [29]:

```
ct = ColumnTransformer(transformers=[("encoder", OneHotEncoder(),["Sex", "Embarked"])],
x = np.array(ct.fit_transform(x))
```

In [30]:

```
x
```

Out[30]:

```
array([[ 0.    ,  1.    ,  0.    , ...,  1.    ,  0.    ,  7.25  ],
       [ 1.    ,  0.    ,  1.    , ...,  1.    ,  0.    , 71.2833],
       [ 1.    ,  0.    ,  0.    , ...,  0.    ,  0.    ,  7.925 ],
       ...,
       [ 1.    ,  0.    ,  0.    , ...,  1.    ,  2.    , 23.45  ],
       [ 0.    ,  1.    ,  1.    , ...,  0.    ,  0.    , 30.    ],
       [ 0.    ,  1.    ,  0.    , ...,  0.    ,  0.    ,  7.75  ]])
```

In [31]:

```
y
```

Out[31]:

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 889, dtype: int64
```

In [32]:

```
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.30, random_state=0, st
```

In [33]:

```
def mymodel(model):
    model.fit(xtrain, ytrain)
    ypred = model.predict(xtest)
    ac = accuracy_score(ytest, ypred)
    cr = classification_report(ytest, ypred)
    print(f"Accuracy -: {ac}\n\nClassification Report-:\n{cr}")
```

In [34]:

```
logreg = LogisticRegression()
knn = KNeighborsClassifier()
svm = SVC()
dt = DecisionTreeClassifier()
```

In [35]:

```
mymodel(knn)
```

```
Accuracy -: 0.7303370786516854

Classification Report-:
              precision    recall  f1-score   support

           0       0.76      0.82      0.79       165
           1       0.67      0.59      0.62       102

    accuracy                           0.73       267
   macro avg       0.71      0.70      0.71       267
weighted avg       0.73      0.73      0.73       267
```

In [36]:

```
mymodel(logreg)
```

```
Accuracy -: 0.8089887640449438

Classification Report-:
              precision    recall  f1-score   support

           0       0.83      0.87      0.85       165
           1       0.77      0.71      0.74       102

    accuracy                           0.81       267
   macro avg       0.80      0.79      0.79       267
weighted avg       0.81      0.81      0.81       267
```

In [37]:

```
1 mymodel(svm)
```

```
Accuracy -: 0.6779026217228464

Classification Report-:
              precision    recall  f1-score   support

           0       0.68      0.90      0.78       165
           1       0.67      0.31      0.43       102

    accuracy                           0.68       267
   macro avg       0.67      0.61      0.60       267
weighted avg       0.68      0.68      0.64       267
```

In [38]:

```
1 mymodel(dt)
```

```
Accuracy -: 0.797752808988764

Classification Report-:
              precision    recall  f1-score   support

           0       0.86      0.80      0.83       165
           1       0.71      0.79      0.75       102

    accuracy                           0.80       267
   macro avg       0.79      0.80      0.79       267
weighted avg       0.80      0.80      0.80       267
```

In [39]:

```
1 logreg = LogisticRegression(solver="liblinear")
```

In [40]:

```
1 logreg.fit(xtest, ytest)
2 ypred = logreg.predict(xtest)
3 ac = accuracy_score(ytest, ypred)
4 cr = classification_report(ytest, ypred)
5 print(f"Accuracy -: {ac}\n\nClassification Report-:\n{cr}")
```

```
Accuracy -: 0.8389513108614233

Classification Report-:
              precision    recall  f1-score   support

           0       0.84      0.91      0.87       165
           1       0.83      0.73      0.77       102

    accuracy                           0.84       267
   macro avg       0.84      0.82      0.82       267
weighted avg       0.84      0.84      0.84       267
```

In [41]:

```
logreg = LogisticRegression(solver="newton-cg")
```

In [42]:

```
logreg.fit(xtest, ytest)
ypred = logreg.predict(xtest)
ac = accuracy_score(ytest, ypred)
cr = classification_report(ytest, ypred)
print(f"Accuracy -: {ac}\n\nClassification Report-:\n{cr}")
```

```
Accuracy -: 0.8314606741573034

Classification Report-:
              precision    recall  f1-score   support

           0       0.84      0.89      0.87       165
           1       0.81      0.74      0.77       102

    accuracy                           0.83       267
   macro avg       0.83      0.81      0.82       267
weighted avg       0.83      0.83      0.83       267
```

In [43]:

```
logreg = LogisticRegression(solver="saga")
```

In [44]:

```
logreg.fit(xtest, ytest)
ypred = logreg.predict(xtest)
ac = accuracy_score(ytest, ypred)
cr = classification_report(ytest, ypred)
print(f"Accuracy -: {ac}\n\nClassification Report-:\n{cr}")
```

```
Accuracy -: 0.7265917602996255

Classification Report-:
              precision    recall  f1-score   support

           0       0.71      0.95      0.81       165
           1       0.82      0.36      0.50       102

    accuracy                           0.73       267
   macro avg       0.76      0.66      0.66       267
weighted avg       0.75      0.73      0.69       267
```

In [45]:

```
cvs = cross_val_score(logreg, x, y, cv = 8)
cvs.mean()
```

Out[45]:

```
0.6918436293436294
```

Conclusion

- the best fit model for this dataset is logistic regression with solver(Liblinear) with the accuracy of 84%

In [ ]:

```
1
```