problem statement:

Use the Titanic dataset to build a model that predicts whether a passenger on the Titanic survived or not. This is a classic beginner project with readily available data.The dataset typically used for this project contains informationabout individual passengers, such as their age, gender, ticket class, fare, cabin, and whether or not they survived.

Import Libraries and Dataset

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv(r"C:\Users\LENOVO\Downloads\tested.csv")
df
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.829
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.000
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.687
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.662
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.287
...	...	...	...	...	...	...	...	...	...	.
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.050
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.900
415	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.250
416	1308	0	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.050
417	1309	0	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.358

418 rows × 12 columns

In [3]:
```
df.shape
```

Out[3]:
```
(418, 12)
```

In [4]:
```
df.describe()
```

Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	0.363636	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.481622	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	0.000000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	1.000000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000	512.329200

In [5]:
```
df.head()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	Nal
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	Nal
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	Nal
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	Nal
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	Nal

In [6]:
```
df.tail()
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000
415	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500
416	1308	0	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500
417	1309	0	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583

In [7]:
```
df.info()
```

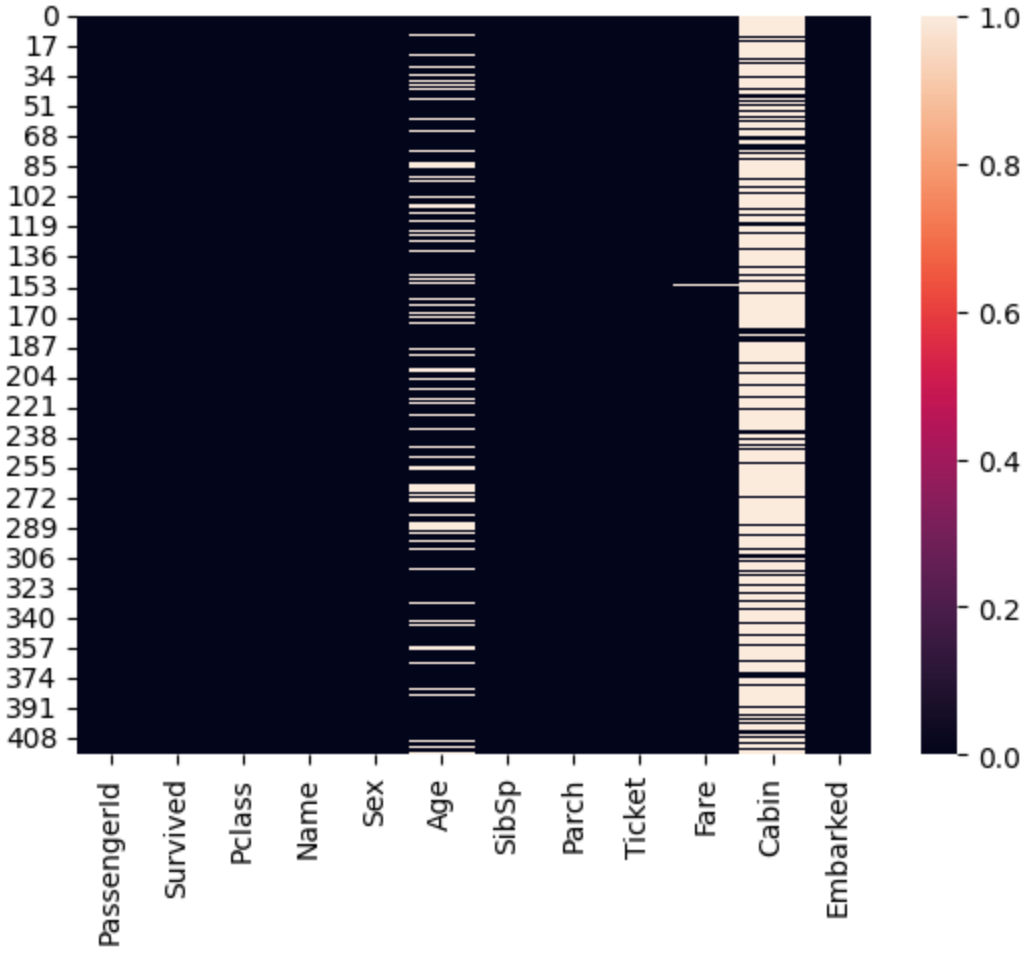
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    object 
 4   Sex          418 non-null    object 
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    object 
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object 
 11  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

Data visualization

In [8]:
```
df.isna().sum()
```

Out[8]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [9]:
```
sns.heatmap(df.isnull());
```



In [10]:
```
df['Age'].replace(np.nan, df['Age'].mean(), inplace=True)
df['Fare'].replace(np.nan, df['Fare'].mean(), inplace=True)
df.drop('Cabin', axis=1,inplace=True)
df.isna().sum()
```

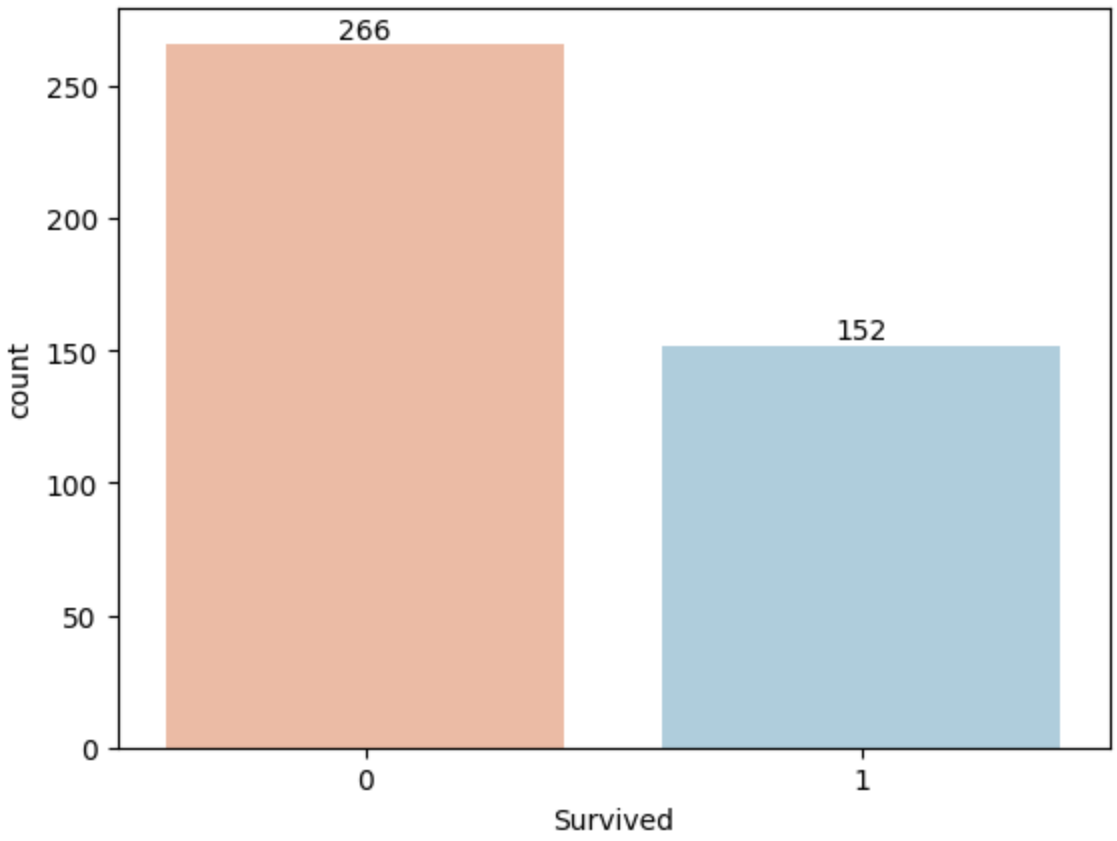
Out[10]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [11]:
```
df.nunique()
```

Out[11]:
```
PassengerId    418
Survived         2
Pclass           3
Name           418
Sex              2
Age             80
SibSp            7
Parch            8
Ticket         363
Fare           170
Embarked         3
dtype: int64
```

In [12]:
```
catvar=['Pclass', 'Sex', 'SibSp', 'Parch', 'Embarked']
numvar=['Age' ,'Fare']
```

In [13]:
```
ax=sns.countplot(x=df['Survived'], palette='RdBu')
ax.bar_label(ax.containers[0])
plt.show()
```

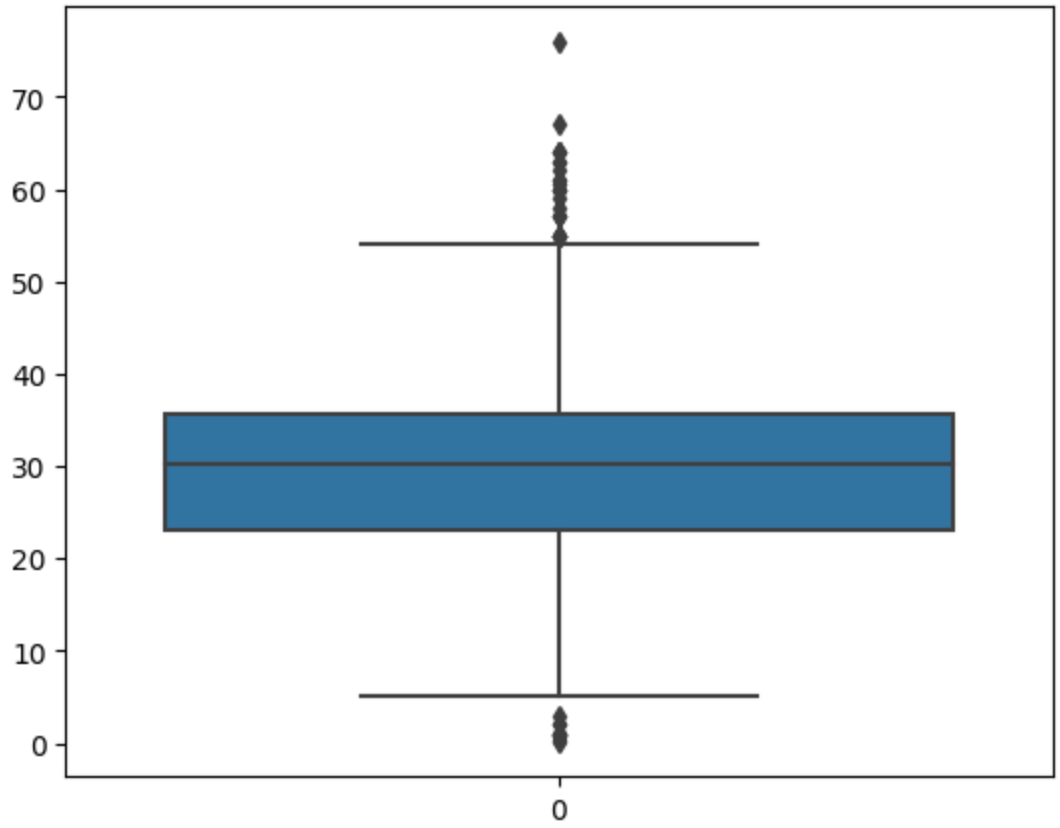


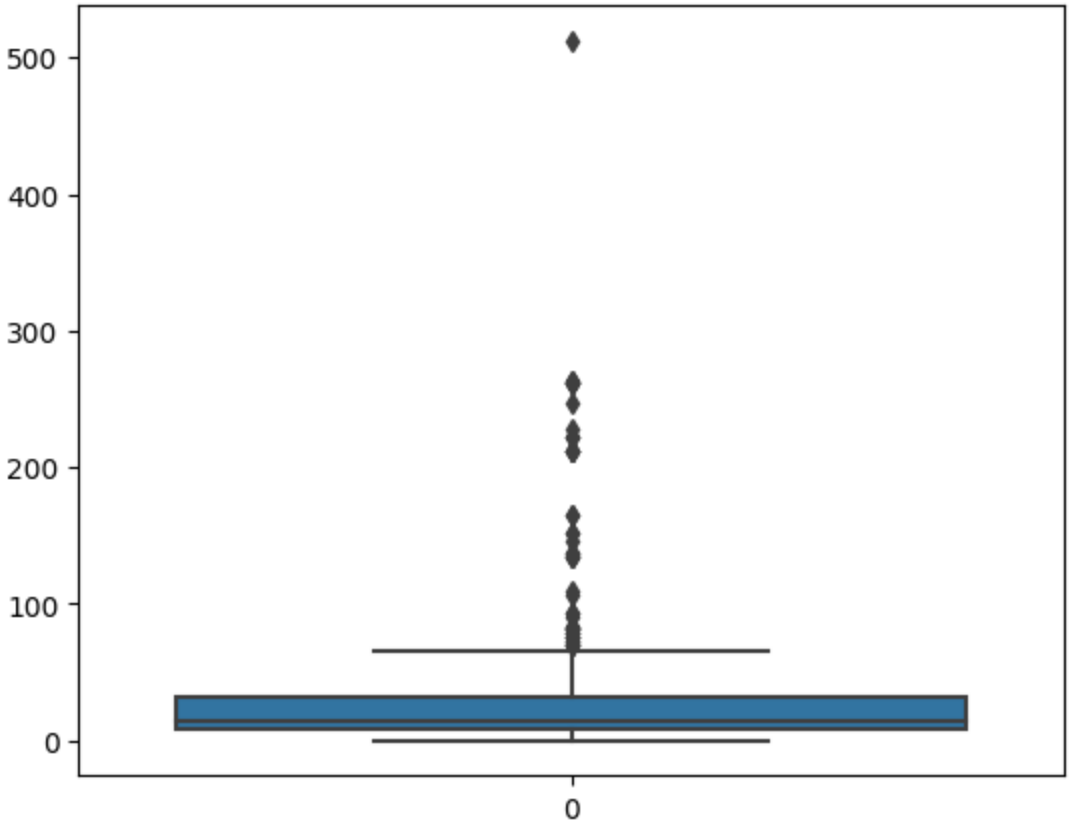
In [14]:
```
df[numvar].describe()
```

Out[14]:

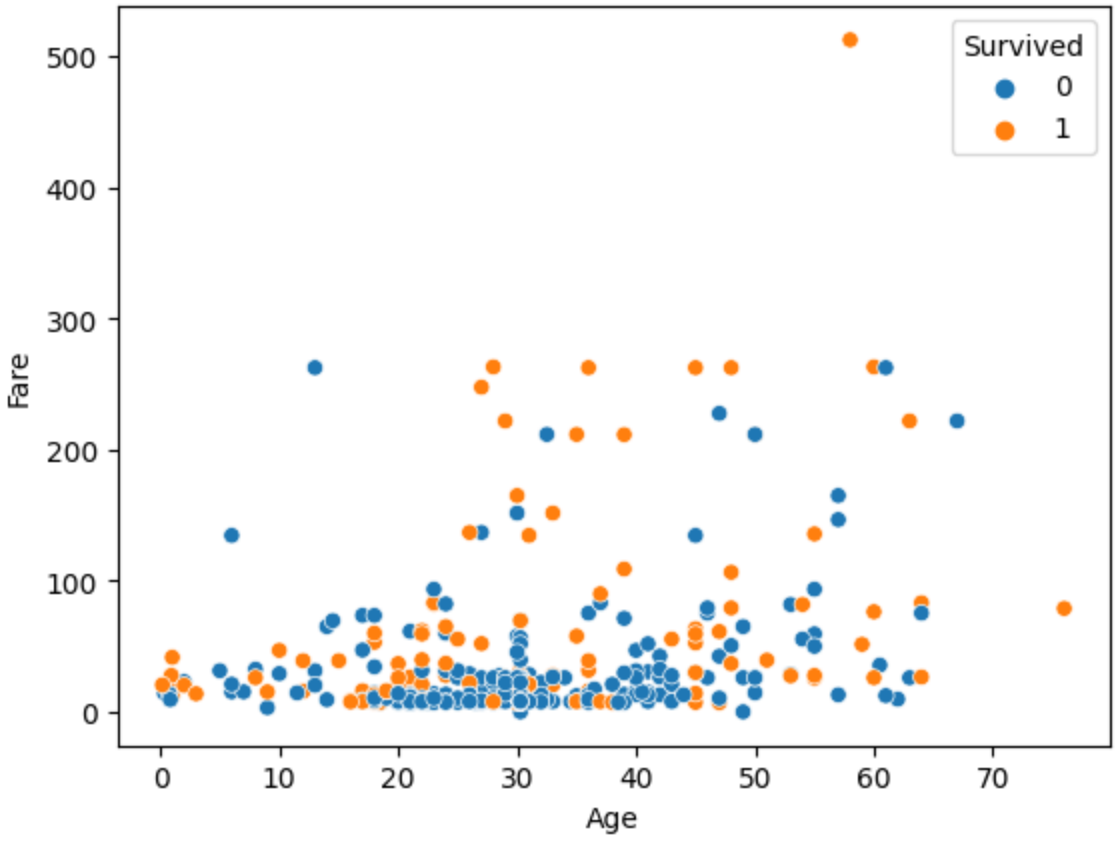
	Age	Fare
count	418.000000	418.000000
mean	30.272590	35.627188
std	12.634534	55.840500
min	0.170000	0.000000
25%	23.000000	7.895800
50%	30.272590	14.454200
75%	35.750000	31.500000
max	76.000000	512.329200

In [15]:
```
for column in numvar:
    plt.figure(figsize=(14,5))
    plt.subplot(1,2,1)
    ax=sns.boxplot(df[column])
    plt.show()
```

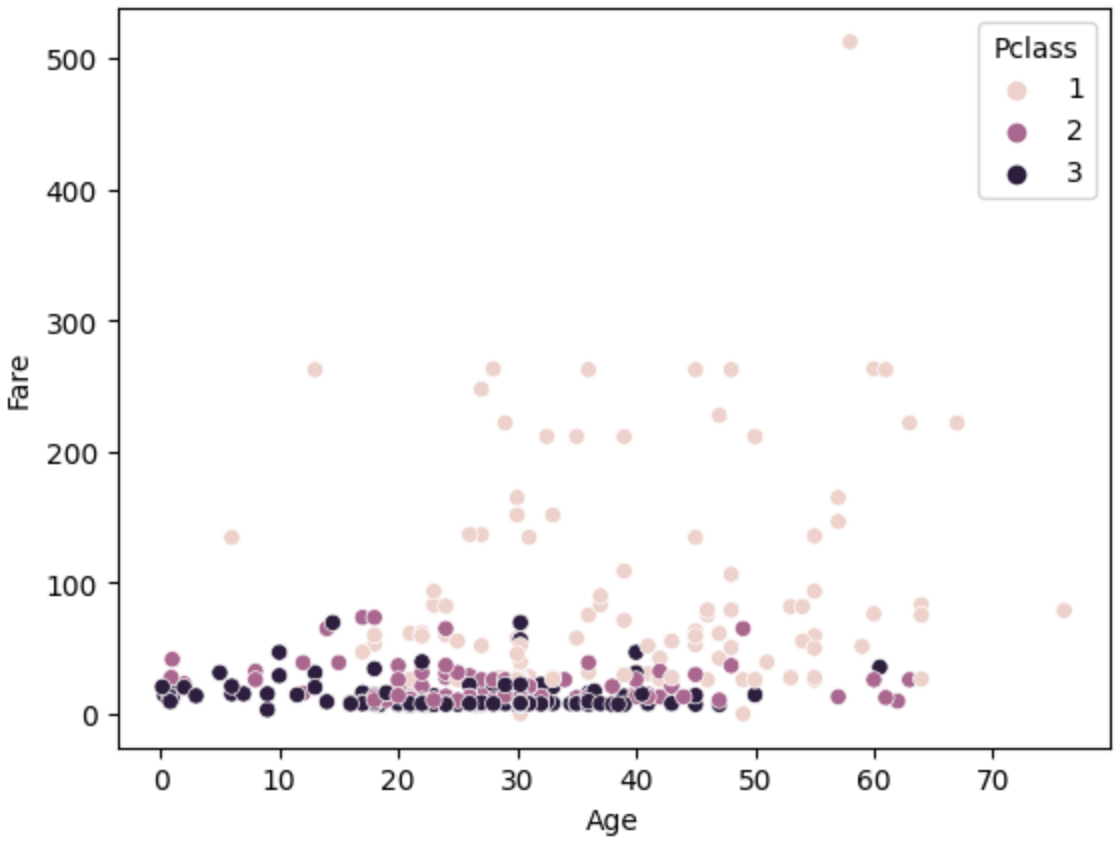




In [16]:
```
sns.scatterplot(data=df, x='Age', y='Fare', hue='Survived')
plt.show()
```



In [17]:
```
sns.scatterplot(data=df, x='Age', y='Fare', hue='Pclass')
plt.show()
```



In [18]:
```
dfupdate=df.drop(['PassengerId','Name','Ticket'], axis=1)
dfupdate.head()
```

Out[18]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	34.5	0	0	7.8292	Q
1	1	3	female	47.0	1	0	7.0000	S
2	0	2	male	62.0	0	0	9.6875	Q
3	0	3	male	27.0	0	0	8.6625	S
4	1	3	female	22.0	1	1	12.2875	S

In [19]:
```
dfupdate['Sex'].replace({'male':1,'female':0}, inplace=True)
dfupdate['Embarked'].replace({'Q':0,'S':1,'C':2}, inplace=True)
dfupdate.head()
```

Out[19]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	34.5	0	0	7.8292	0
1	1	3	0	47.0	1	0	7.0000	1
2	0	2	1	62.0	0	0	9.6875	0
3	0	3	1	27.0	0	0	8.6625	1
4	1	3	0	22.0	1	1	12.2875	1

In [20]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score ,recall_score, f1_s
confusion_matrix, ConfusionMatrixDisplay, RocCurveDisplay
```

In [21]:
```
X = dfupdate.drop('Survived', axis=1)
y = dfupdate['Survived']
X.head()
```

Out[21]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	34.5	0	0	7.8292	0
1	3	0	47.0	1	0	7.0000	1
2	2	1	62.0	0	0	9.6875	0
3	3	1	27.0	0	0	8.6625	1
4	3	0	22.0	1	1	12.2875	1

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [23]:
```
lr=LogisticRegression(random_state=40)
lr.fit(X_train, y_train)
```

```
C:\Users\LENOVO\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to conve
rge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[23]:
```
LogisticRegression
LogisticRegression(random_state=40)
```

In [24]:
```
lr.coef_
```

Out[24]:
```
array([[ 2.88212996e-02, -6.38554968e+00, -8.73466763e-03,
         4.23433418e-02,  1.08495596e-01,  4.21035471e-03,
        -1.55510791e-01]])
```

In [25]:
```
lr.intercept_
```

Out[25]:
```
array([3.07681392])
```

In [26]:
```
y_pred_train = lr.predict(X_train)
```

In [27]:
```
print('Accuracy: ', accuracy_score(y_train, y_pred_train))
print('Precision: ', precision_score(y_train, y_pred_train))
print('Recall: ', recall_score(y_train, y_pred_train))
print('F1 Score: ', f1_score(y_train, y_pred_train))
```

```
Accuracy:  1.0
Precision:  1.0
Recall:  1.0
F1 Score:  1.0
```

In [28]:
```
y_pred_lr = lr.predict(X_test)
```

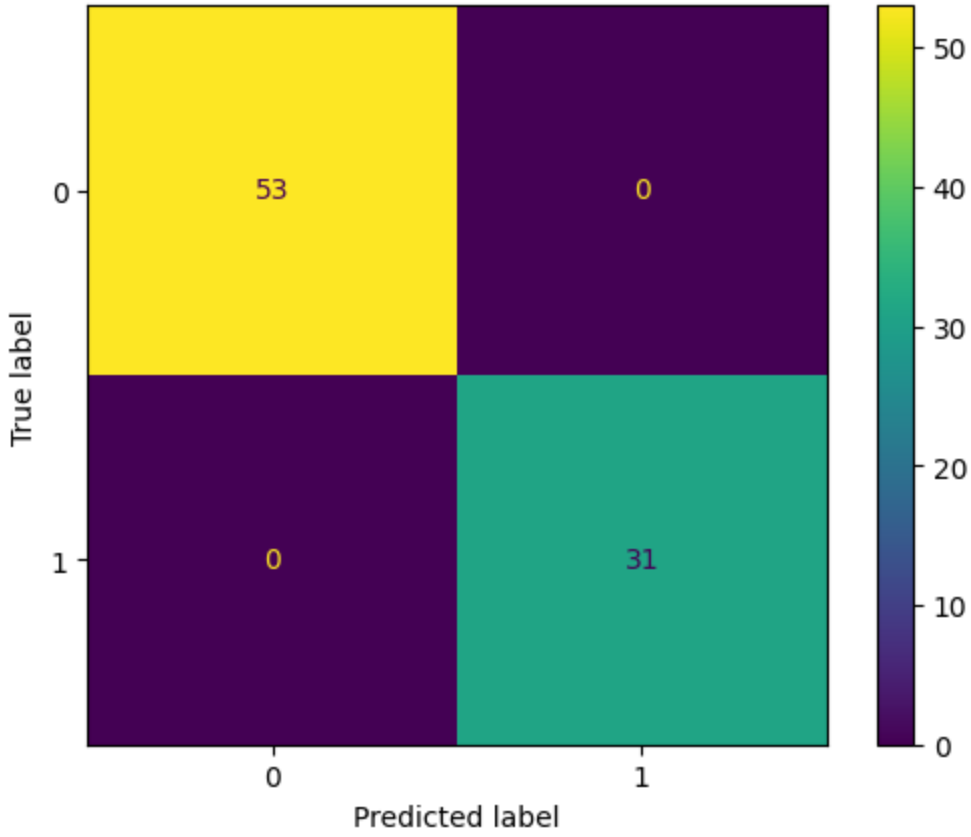
In [29]:
```
lr.predict(X_test)
```

Out[29]:
```
array([0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1], dtype=int64)
```

In [30]:
```
print('Accuracy: ', accuracy_score(y_test, y_pred_lr))
print('Precision: ', precision_score(y_test, y_pred_lr))
print('Recall: ', recall_score(y_test, y_pred_lr))
print('F1 Score: ', f1_score(y_test, y_pred_lr))
```

```
Accuracy:  1.0
Precision:  1.0
Recall:  1.0
F1 Score:  1.0
```

In [31]:
```
cm = confusion_matrix(y_test, y_pred_lr, labels=lr.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=lr.classes_)
disp.plot();
```



The logistic regression model has 100% accuracy. This model has correctly predicted all test samples.

K - Nearest Neighbour

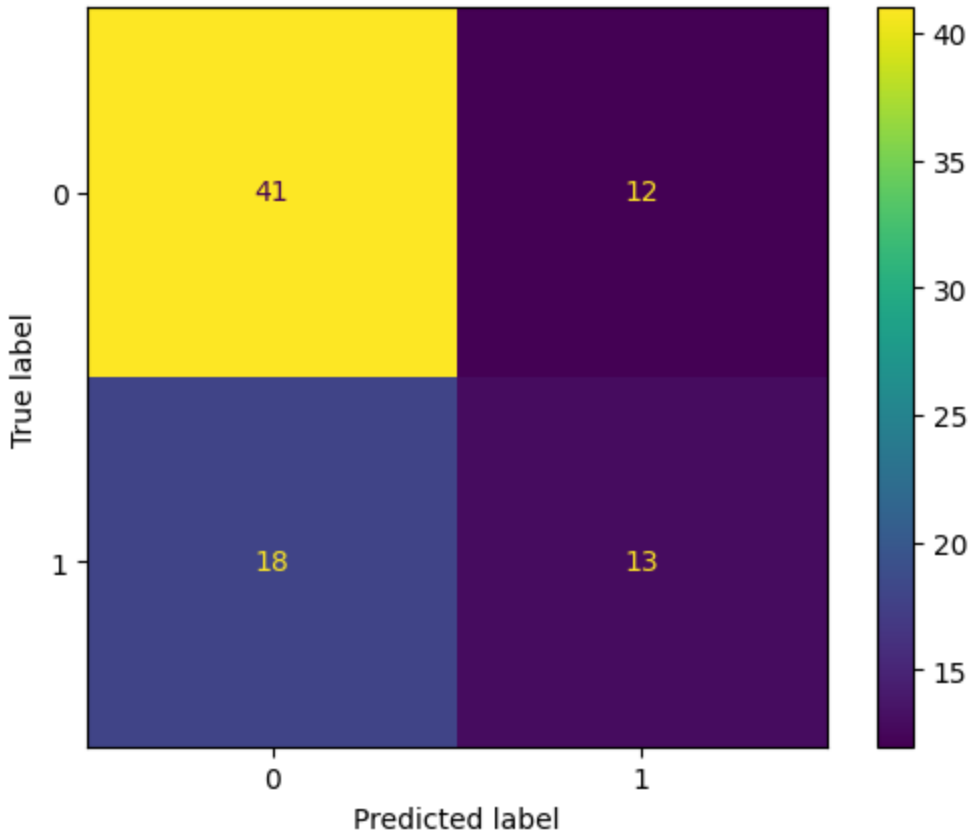
In [32]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(X_train,y_train)
y_pred_knn = knn.predict(X_test)
```

In [33]:
```
print('Accuracy: ', accuracy_score(y_test, y_pred_knn))
print('Precision: ', precision_score(y_test, y_pred_knn))
print('Recall: ', recall_score(y_test, y_pred_knn))
print('F1 Score: ', f1_score(y_test, y_pred_knn))
```

```
Accuracy:  0.6428571428571429
Precision:  0.52
Recall:  0.41935483870967744
F1 Score:  0.46428571428571425
```

In [34]:
```
cm = confusion_matrix(y_test, y_pred_knn, labels=knn.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=knn.classes_
disp.plot();
```



The K-Nearest Neighbour model has an accuracy of 64.2%. There are many false negative predictions in this model.

Decision tree

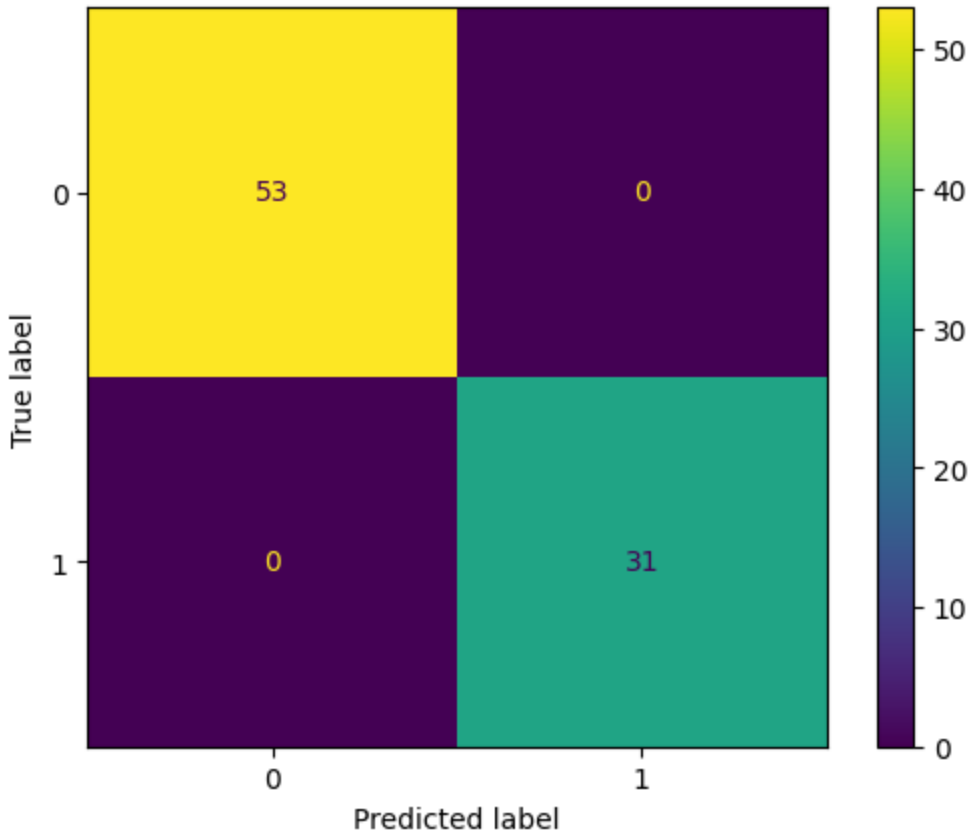
In [35]:
```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier()
tree.fit(X_train,y_train)
y_pred_tree = tree.predict(X_test)
```

In [36]:
```
print('Accuracy: ', accuracy_score(y_test, y_pred_tree))
print('Precision: ', precision_score(y_test, y_pred_tree))
print('Recall: ', recall_score(y_test, y_pred_tree))
print('F1 Score: ', f1_score(y_test, y_pred_tree))
```

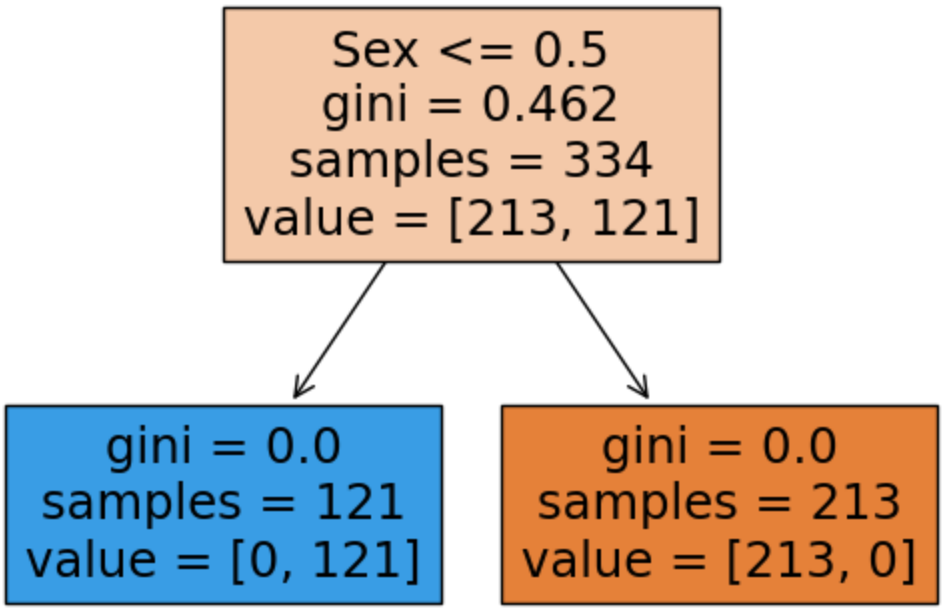
```
Accuracy:  1.0
Precision:  1.0
Recall:  1.0
F1 Score:  1.0
```

In [37]:
```
cm = confusion_matrix(y_test, y_pred_tree, labels=tree.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=tree.classes_
disp.plot();
```



The Decision Tree model also has 100% accuracy and has correctly predicted all test samples.

In [38]:
```
from sklearn.tree import plot_tree
plot_tree(tree, filled=True, feature_names=list(X.columns))
plt.show()
```



The tree map shows that the Decision tree model was able to predict passengers survival solely on passengers sex because in this dataset all of the females survived the Titanic disaster and all of the males did not survive the disaster.

Summary

There are 36.36% passengers who survived Titanic disaster and 63.64% passengers who did not survive.

All of the females survived the disaster and all of the males did not survive the disaster.The number of not surviving passengers is high for passenger class 3 and 2. Single passengers with no siblings/spouses or parents/children aboard have less chance of survival. The passngers who embarked from Queenstown have good chance of survival and the passngers who embarked from Southampton have less chance of survival.

The models Logistic regression and Decision tree resulted in 100% accuracy.