In [41]:
```
import pandas as pd
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score
from sklearn.impute import SimpleImputer
```

In [42]:
```
titanic = pd.read_csv('titanic_dataset.csv')
```

In [43]:
```
titanic.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 [44]:
```
age_imputer = SimpleImputer(strategy='mean')
titanic['Age'] = age_imputer.fit_transform(titanic[['Age']])
```

In [45]:
```
titanic = titanic.drop(['Name','Cabin','Ticket'], axis = 1)
titanic['Sex'] = titanic['Sex'].map({'male':1, 'female':0})
titanic['Embarked'] = titanic['Embarked'].map({'C':2,'Q':1, 'S': 0})
```

In [46]:
```
titanic.dropna(inplace = True)
```

In [47]:
```
titanic.shape
```

Out[47]:
```
(889, 9)
```

In [48]:
```
titanic.head()
```

Out[48]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	1	22.0	1	0	7.2500	0.0
1	2	1	1	0	38.0	1	0	71.2833	2.0
2	3	1	3	0	26.0	0	0	7.9250	0.0
3	4	1	1	0	35.0	1	0	53.1000	0.0
4	5	0	3	1	35.0	0	0	8.0500	0.0

In [49]:
```
X = titanic.drop(['Survived'], axis=1)
y = titanic['Survived']
```

In [50]:
```
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

In [67]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

 # we can perform hyperparameter tuning to prvent the model from overfitting

classifier = SVC(kernel='rbf', C= 1, gamma=0.1)
classifier.fit(X_train, y_train)

#test accuracy
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f' Test Accuracy: {accuracy}')
# tain accuracy
pred_train = classifier.predict(X_train)
accuracy = accuracy_score(y_train, pred_train)
print(f' Train Accuracy: {accuracy}')
```

```
Test Accuracy: 0.8202247191011236
Train Accuracy: 0.8382559774964838
```

In [ ]:

In [ ]:

In [ ]: