In [28]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [29]:
```
df = pd.read_csv('titanic.csv')
```

In [30]:
```
df
```

Out[30]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25(
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28:
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92:
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10(
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05(
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00(
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00(
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45(
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00(
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75(

891 rows × 12 columns

In [31]:
```
df['Sex'] = df['Sex'].map({'female': 0, 'male': 1})
```

In [32]:
```
df.Age = df.Age.fillna(df.Age.mean())
```

In [33]:
```
columns_to_drop = ['PassengerId', 'Name', 'Ticket', 'Cabin']
df.drop(columns=columns_to_drop, inplace=True)
```

In [34]:
```
columns_to_drop = ['Pclass', 'SibSp', 'Parch', 'Fare']
df.drop(columns=columns_to_drop, inplace=True)
```

In [35]:
```
columns_to_drop = ['Embarked']
df.drop(columns=columns_to_drop, inplace=True)
```

In [36]:
```
df
```

Out[36]:

	Survived	Sex	Age
0	0	1	22.000000
1	1	0	38.000000
2	1	0	26.000000
3	1	0	35.000000
4	0	1	35.000000
...	...	...	...
886	0	1	27.000000
887	1	0	19.000000
888	0	0	29.699118
889	1	1	26.000000
890	0	1	32.000000

891 rows × 3 columns

In [37]:
```
from sklearn.model_selection import train_test_split
```

In [38]:
```
X = df.drop('Survived', axis=1)
y = df['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
```

In [39]:
```
# initialise wt,b,(lr_rate,epoch) values pass
# fit summarization z = dot [x+]+b
# y_pred = activation(z)-
# write init method
# if x has two column sthen w must have two columns
# X
# add pclass instead of sex
```

In [40]:
```
X
```

Out[40]:

	Sex	Age
0	1	22.000000
1	0	38.000000
2	0	26.000000
3	0	35.000000
4	1	35.000000
...	...	...
886	1	27.000000
887	0	19.000000
888	0	29.699118
889	1	26.000000
890	1	32.000000

891 rows × 2 columns

In [41]:
```
X.shape
```

Out[41]:
```
(891, 2)
```

In [42]:
```
y
```

Out[42]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

In [43]:
```
y.shape
```

Out[43]:
```
(891,)
```

In [44]:
```
import numpy as np
```

In [45]:
```
class Perceptron_c :
    def __init__(self,learning_rate,epochs):
        self.weights = None
        self.bias = None
        self.learning_rate = learning_rate
        self.epochs = epochs

    def activation(self,z):
        return np.heaviside(z,0)

    def fit(self,X,y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)
        self.bias = 0

        for epoch in range(self.epochs):
            for i in range(n_samples):
                z = np.dot(X.iloc[i], self.weights) + self.bias
                y_pred = self.activation(z)

                # Update weights and bias
                self.weights += self.learning_rate * (y.iloc[i]-y_pred) * X
                self.bias += self.learning_rate * (y.iloc[i]-y_pred)
        print(self.weights.shape)

    def predict(self,X):
        z = np.dot(X,self.weights) + self.bias
        return self.activation(z)
```

#titanic fit

In [46]:
```
perceptron = Perceptron_c(0.0001,10)
perceptron.fit(X_train,y_train)
pred = perceptron.predict(X_test)
# perceptron is the object of the class
from sklearn.metrics import accuracy_score

accuracy_score(y_test,pred)
```

```
(2,)
```

Out[46]:
```
0.7910447761194029
```

In [47]:
```
from sklearn.metrics import classification_report

report = classification_report(y_test,pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.80      0.85      0.83       157
           1       0.77      0.70      0.74       111

    accuracy                           0.79       268
   macro avg       0.79      0.78      0.78       268
weighted avg       0.79      0.79      0.79       268
```

GridSearchCV

In [48]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score
```

In [51]:
```
from sklearn.linear_model import Perceptron
p1 = Perceptron()
```

In [52]:
```
param_grid = {'alpha': [0.0001, 0.001, 0.01, 0.1, 1.0], 'max_iter': [10, 50
```

In [53]:
```
grid_search = GridSearchCV(p1, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
```

```
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_g
radient.py:702: ConvergenceWarning: Maximum number of iteration reached be
fore convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
```

Out[53]:

```
▸ GridSearchCV
▸ estimator: Perceptron
  ▸ Perceptron
```

In [54]:

```
best_params = grid_search.best_params_
print("Best Parameters:", best_params)
```

```
Best Parameters: {'alpha': 0.0001, 'max_iter': 50}
```

In [55]:

```
cp = Perceptron_c(learning_rate=best_params['alpha'], epochs=best_params['m
cp.fit(X_train, y_train)

cy_pred = cp.predict(X_test)

accuracy = accuracy_score(y_test, cy_pred)
print(f"Accuracy: {accuracy}")
```

```
(2,)
Accuracy: 0.6604477611940298
```

In [56]:

```
sp = Perceptron(alpha=best_params['alpha'], max_iter=best_params['max_iter'
sp.fit(X_train, y_train)

sy_pred = sp.predict(X_test)

accuracy = accuracy_score(y_test, sy_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.7910447761194029
```

In [ ]: