In [26]:
```
pip show my_ml_library
```

```
Name: my_ml_library
Version: 0.1.1
Summary: A simple machine learning library with regression models and metrics.
Home-page: https://github.com/bhushanzade02
Author: Bhushan Zade
Author-email: bhushanzade02@gmail.com
License:
Location: C:\Users\bhush\AppData\Local\Programs\Python\Python313\Lib\site-packages
Editable project location: D:\16_MACHINE_LEARNING\01_MY_ML_LIBRARY (FINAL DEPLOYED O
N PYPI )
Requires: numpy, scipy
Required-by:
Note: you may need to restart the kernel to use updated packages.
```

In [58]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
```

In [59]:
```
import my_ml_library
print(my_ml_library)
```

```
<module 'my_ml_library' from 'D:\\16_MACHINE_LEARNING\\01_MY_ML_LIBRARY (FINAL DEPLO
YED ON PYPI )\\my_ml_library\\__init__.py'>
```

In [ ]:
```

```

In [60]:
```
from my_ml_library import SimpleLinearRegression, MultipleLinearRegression, Logisti
from my_ml_library.metrics import mean_squared_error, mean_absolute_error, root_mea
from my_ml_library.optimizers import GradientDescent, StochasticGradientDescent
```

In [ ]:
```

```

In [ ]:
```

```

SIMPLE LINEAR REGRESSION THROUGH CUSTOM LIBRARY

In [61]:
```
from my_ml_library import SimpleLinearRegression
```

In [62]:
```
df=pd.read_csv(R"D:\16_MACHINE_LEARNING\03_MACHINE_LEARNING_USING_PYTHON\placement_
```

In [63]:
```
X=df.iloc[:,0:1]
y=df.iloc[:,-1]
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
slr=SimpleLinearRegression()
```

```
slr.fit(X_train,y_train)
y_pred=slr.predict(X_test)
```

In [64]:
```
from my_ml_library.metrics import r2_score,mean_absolute_error,mean_squared_error,r
print("R2 SCORE :",r2_score(y_test,y_pred))
print("MEAN ABSOLUTE ERROR :",mean_absolute_error(y_test,y_pred))
print("MEAN SQUARED ERROR  :",mean_squared_error(y_test,y_pred))
print("ROOT MEAN SQUARED ERROR  :",root_mean_squared_error(y_test,y_pred))
```

```
R2 SCORE : 0.780730147510384
MEAN ABSOLUTE ERROR : 0.2884710931878175
MEAN SQUARED ERROR  : 0.12129235313495527
ROOT MEAN SQUARED ERROR  : 0.34827051717731616
```

In [ ]:

In [ ]:

In [ ]:

SIMPLE LINEAR REGRESSION THROUGH SCIKIT LIBRARY

In [65]:
```
df=pd.read_csv(R"D:\16_MACHINE_LEARNING\03_MACHINE_LEARNING_USING_PYTHON\placement_
X=df.iloc[:,0:1]
y=df.iloc[:,-1]
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
```

In [66]:
```
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error,root_me
print("R2 SCORE :",r2_score(y_test,y_pred))
print("MEAN ABSOLUTE ERROR :",mean_absolute_error(y_test,y_pred))
print("MEAN SQUARED ERROR  :",mean_squared_error(y_test,y_pred))
print("ROOT MEAN SQUARED ERROR  :",root_mean_squared_error(y_test,y_pred))
```

```
R2 SCORE : 0.780730147510384
MEAN ABSOLUTE ERROR : 0.2884710931878175
MEAN SQUARED ERROR  : 0.12129235313495527
ROOT MEAN SQUARED ERROR  : 0.34827051717731616
```

In [ ]:

In [ ]:

In [ ]:

MULTIPLE LINEAR REGRESSION THROUGH CUSTOM LIBRARY

In [67]:
```
df=pd.read_csv(R"D:\16_MACHINE_LEARNING\03_MACHINE_LEARNING_USING_PYTHON\concrete_d
```

In [68]:
```
X=df.iloc[:,0:8]
y=df.iloc[:,-1]
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
slr=MultipleLinearRegression()
slr.fit(X_train,y_train)
y_pred=slr.predict(X_test)
```

In [69]:
```
from my_ml_library.metrics import r2_score,mean_absolute_error,mean_squared_error,r
print("R2 SCORE :",r2_score(y_test,y_pred))
print("MEAN ABSOLUTE ERROR :",mean_absolute_error(y_test,y_pred))
print("MEAN SQUARED ERROR  :",mean_squared_error(y_test,y_pred))
print("ROOT MEAN SQUARED ERROR  :",root_mean_squared_error(y_test,y_pred))
```

```
R2 SCORE : 0.5701142652758168
MEAN ABSOLUTE ERROR : 8.226419967035435
MEAN SQUARED ERROR  : 105.76432225736826
ROOT MEAN SQUARED ERROR  : 10.284178249007953
```

In [ ]:

In [ ]:

MULTIPLE LINEAR REGRESSION THROUGH SCIKIT LIBRARY

In [70]:
```
df=pd.read_csv(R"D:\16_MACHINE_LEARNING\03_MACHINE_LEARNING_USING_PYTHON\concrete_d
X=df.iloc[:,0:8]
y=df.iloc[:,-1]
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
```

In [71]:
```
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error,root_me
print("R2 SCORE :",r2_score(y_test,y_pred))
print("MEAN ABSOLUTE ERROR :",mean_absolute_error(y_test,y_pred))
print("MEAN SQUARED ERROR  :",mean_squared_error(y_test,y_pred))
print("ROOT MEAN SQUARED ERROR  :",root_mean_squared_error(y_test,y_pred))
```

```
R2 SCORE : 0.570114265275778
MEAN ABSOLUTE ERROR : 8.226419967037103
MEAN SQUARED ERROR  : 105.76432225737778
ROOT MEAN SQUARED ERROR  : 10.284178249008415
```

In [ ]:

In [ ]:

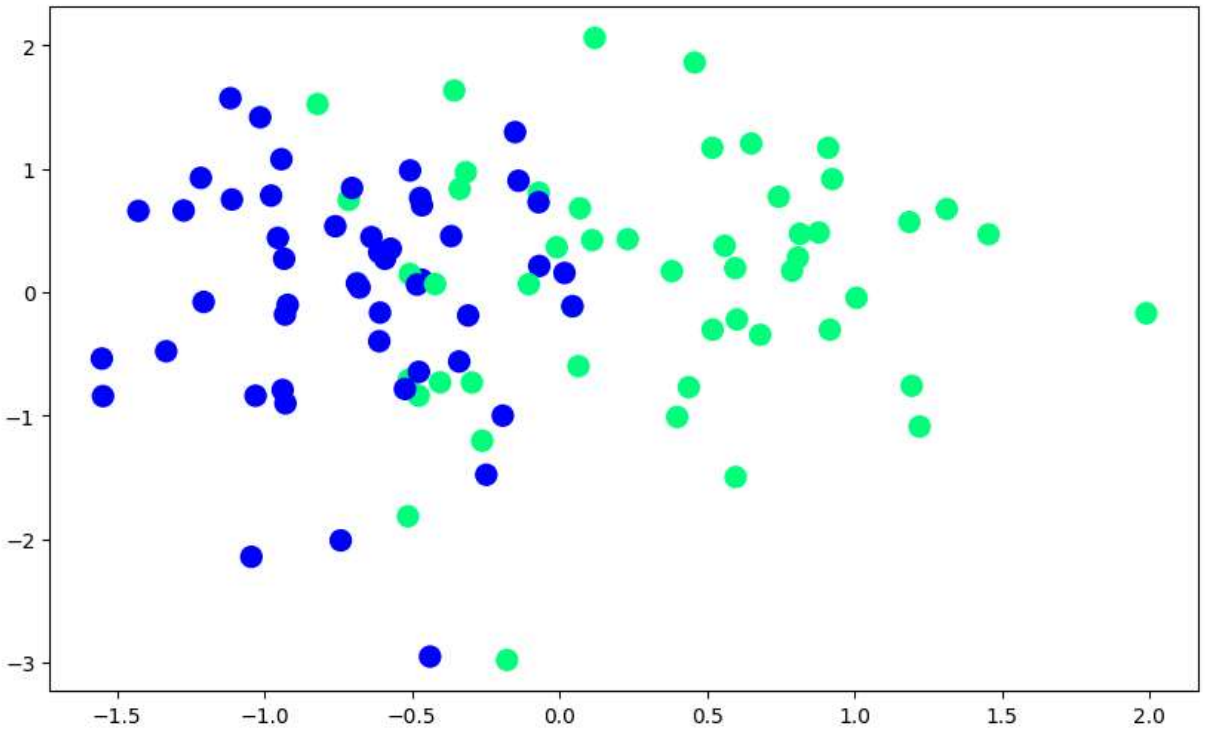
In [ ]:

LOGISTIC REGRESSIOM THROUGH CUSTOM LIBRARY

In [117…
```
X, y = make_classification(n_samples=100, n_features=2, n_informative=1,n_redundant
                           n_classes=2, n_clusters_per_class=1, random_state=41,hyp
```

In [118…
```
plt.figure(figsize=(10,6))
plt.scatter(X[:,0],X[:,1],c=y,cmap="winter",s=100)
```

Out[118… <matplotlib.collections.PathCollection at 0x22d79c94910>



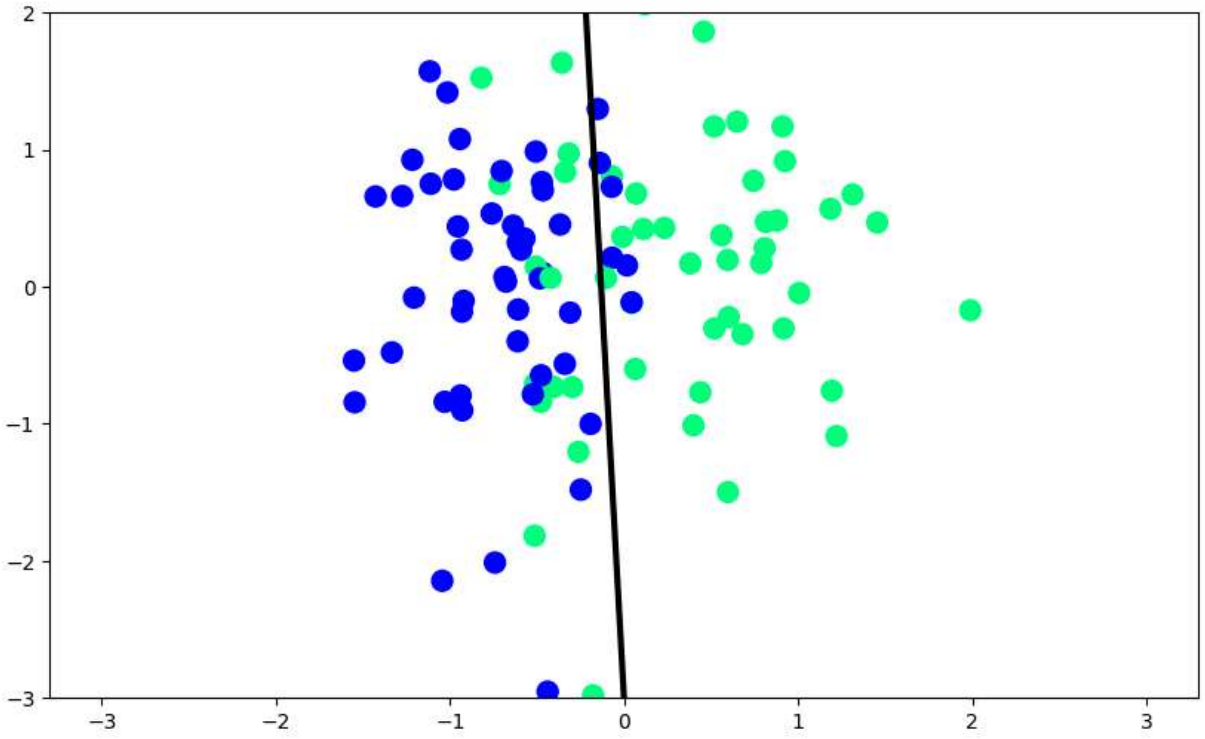
In [119…
```
from my_ml_library import LogisticRegression
lor = LogisticRegression()
lor.fit(X,y)
```

In [120…
```
m = -(lor.weights[0] / lor.weights[1])
b = -(lor.bias / lor.weights[1])
```

In [121…
```
x_input1 = np.linspace(-3,3,100)
y_input1 = m*x_input1 + b
```

In [122…
```
plt.figure(figsize=(10,6))
plt.plot(x_input1,y_input1,color='black',linewidth=3)
plt.scatter(X[:,0],X[:,1],c=y,cmap='winter',s=100)
plt.ylim(-3,2)
```

Out[122…
```
(-3.0, 2.0)
```



In [123…
```
# Evaluation metrics
print("Accuracy Score       :", accuracy_score(y, y_pred))
print("Precision Score      :", precision_score(y, y_pred))
print("Recall Score         :", recall_score(y, y_pred))
print("F1 Score             :", f1_score(y, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y, y_pred))
print("\nClassification Report:\n", classification_report(y, y_pred))
```

```
Accuracy Score       : 0.8
Precision Score      : 0.8409090909090909
Recall Score         : 0.74
F1 Score             : 0.7872340425531915

Confusion Matrix:
 [[43  7]
 [13 37]]

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.86      0.81        50
           1       0.84      0.74      0.79        50

    accuracy                           0.80       100
   macro avg       0.80      0.80      0.80       100
weighted avg       0.80      0.80      0.80       100
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

LOGISTIC REGRESSION THROUGH SCIKIT LIBRARY

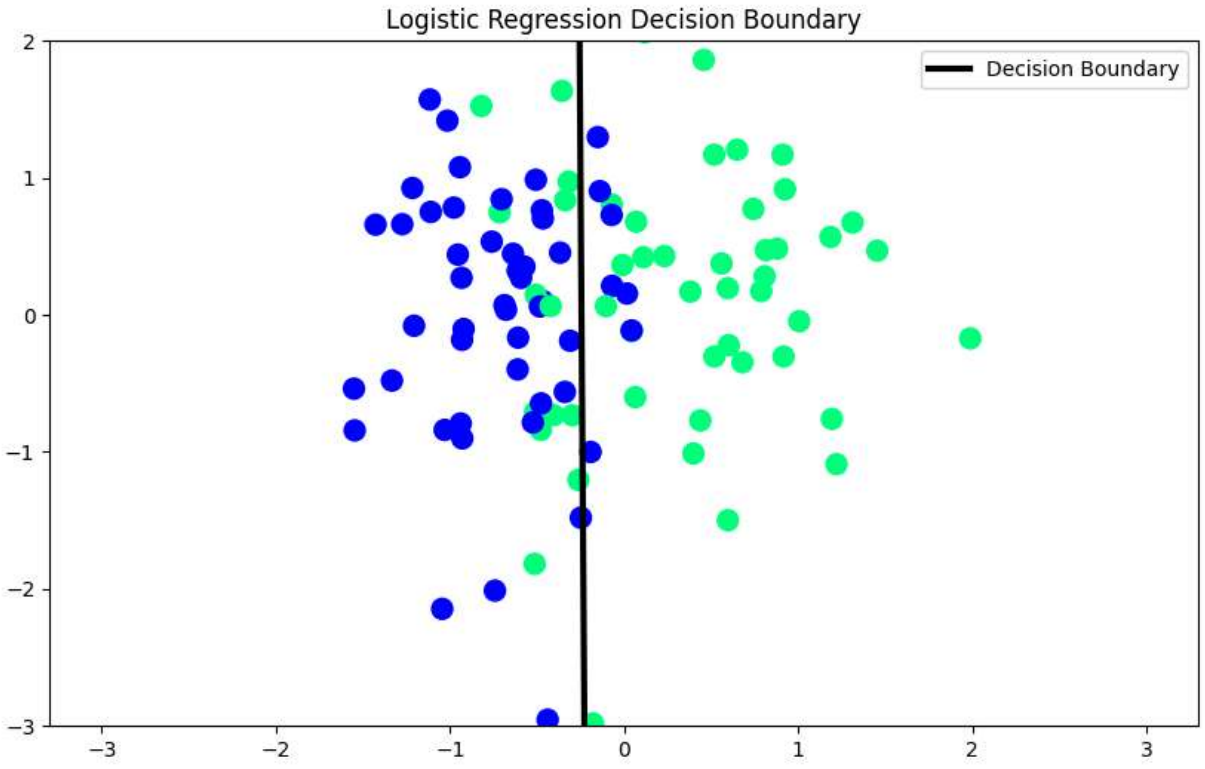
In [130…

```
X, y = make_classification(n_samples=100, n_features=2, n_informative=1,n_redundant
                           n_classes=2, n_clusters_per_class=1, random_state=41,hyp
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

lor = LogisticRegression()
lor.fit(X, y)
y_pred = lor.predict(X)
m = -(lor.coef_[0][0] / lor.coef_[0][1])
b = -(lor.intercept_[0] / lor.coef_[0][1])
x_input1 = np.linspace(-3, 3, 100)
y_input1 = m * x_input1 + b

plt.figure(figsize=(10, 6))
plt.plot(x_input1, y_input1, color='black', linewidth=3, label='Decision Boundary')
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='winter', s=100)
plt.ylim(-3, 2)
plt.title("Logistic Regression Decision Boundary")
```

```
plt.legend()
plt.show()
```



In [ ]:

In [ ]:

In [131…
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
print("Accuracy Score       :", accuracy_score(y, y_pred))
print("Precision Score      :", precision_score(y, y_pred))
print("Recall Score         :", recall_score(y, y_pred))
print("F1 Score             :", f1_score(y, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y, y_pred))
print("\nClassification Report:\n", classification_report(y, y_pred))
```

```
Accuracy Score       : 0.8
Precision Score      : 0.8409090909090909
Recall Score         : 0.74
F1 Score             : 0.7872340425531915

Confusion Matrix:
 [[43  7]
 [13 37]]

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.86      0.81        50
           1       0.84      0.74      0.79        50

    accuracy                           0.80       100
   macro avg       0.80      0.80      0.80       100
weighted avg       0.80      0.80      0.80       100
```

In [ ]: