```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score


# Load Dataset
# which is a classic dataset used for classification tasks

iris = load_iris()
X = iris.data
y = iris.target
```

just to view the dataset

```
import pandas as pd

# Convert the data to a pandas DataFrame for easier viewing
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

# Add the target column to the DataFrame
df['target'] = iris.target

# Display the first few rows of the DataFrame
print(df.head())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0
```

```
#Split Dataset
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2, random_state=42)



# Train Logistic Regression Model

logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train, y_train)
```

```
LogisticRegression
```

```
# Evaluate the Model
y_pred = logistic_model.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='macro')

print("Accuracy:", accuracy)
print("F1-Score:", f1)
```

```
Accuracy: 1.0
F1-Score: 1.0
```

NEW DATASET

```
from sklearn.datasets import load_breast_cancer
```

just for viewing the dataset

```
import pandas as pd

# Load the Breast Cancer Wisconsin dataset
breast_cancer = load_breast_cancer()

# Convert the data to a pandas DataFrame for easier viewing
df = pd.DataFrame(data=breast_cancer.data, columns=breast_cancer.feature_names)

# Add the target column to the DataFrame
df['target'] = breast_cancer.target

# Display the first few rows of the DataFrame
print(df.head(8))
```

```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030
5        12.45         15.70           82.57      477.1          0.12780
6        18.25         19.98          119.60     1040.0          0.09463
7        13.71         20.83           90.20      577.9          0.11890

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760         0.30010              0.14710         0.2419
1           0.07864         0.08690              0.07017         0.1812
2           0.15990         0.19740              0.12790         0.2069
3           0.28390         0.24140              0.10520         0.2597
4           0.13280         0.19800              0.10430         0.1809
5           0.17000         0.15780              0.08089         0.2087
6           0.10900         0.11270              0.07400         0.1794
7           0.16450         0.09366              0.05985         0.2196

   mean fractal dimension  ...  worst texture  worst perimeter  worst area  \
0                 0.07871  ...          17.33           184.60      2019.0
1                 0.05667  ...          23.41           158.80      1956.0
2                 0.05999  ...          25.53           152.50      1709.0
3                 0.09744  ...          26.50            98.87       567.7
4                 0.05883  ...          16.67           152.20      1575.0
5                 0.07613  ...          23.75           103.40       741.6
6                 0.05742  ...          27.66           153.20      1606.0
7                 0.07451  ...          28.14           110.60       897.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625
5            0.1791             0.5249           0.5355                0.1741
6            0.1442             0.2576           0.3784                0.1932
7            0.1654             0.3682           0.2678                0.1556

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0
5          0.3985                  0.12440       0
6          0.3063                  0.08368       0
7          0.3196                  0.11510       0

[8 rows x 31 columns]
```

```
breast_cancer = load_breast_cancer()
X = breast_cancer.data
y = breast_cancer.target

#  Split Dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#  Train Logistic Regression Model
logistic_model = LogisticRegression(max_iter=1000)  # Increase max_iter if needed
logistic_model.fit(X_train, y_train)

#  Evaluate the Model
y_pred = logistic_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("F1-Score:", f1)
```

```
Accuracy: 0.956140350877193
F1-Score: 0.9655172413793103
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```