```
import os
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from skimage.io import imread
from skimage.transform import resize

# Path to the dataset
train_fractured_path = "/kaggle/input/bone-fracture-detection-using-xrays/archive (6)/train/fractured"
train_not_fractured_path = "/kaggle/input/bone-fracture-detection-using-xrays/archive (6)/train/not fractured"
val_fractured_path = "/kaggle/input/bone-fracture-detection-using-xrays/archive (6)/val/fractured"
val_not_fractured_path = "/kaggle/input/bone-fracture-detection-using-xrays/archive (6)/val/not fractured"

# Function to load images and labels from the dataset
def load_data(data_path, target_size=(100, 100)):
    images = []
    labels = []
    classes = os.listdir(data_path)
    for class_name in classes:
        class_path = os.path.join(data_path, class_name)
        for image_name in os.listdir(class_path):
            image_path = os.path.join(class_path, image_name)
            image = imread(image_path, as_gray=True)  # Load image directly in grayscale
            # Resize images to a fixed size for uniformity
            image_resized = resize(image, target_size)
            images.append(image_resized.flatten())  # Append the flattened image
            labels.append(class_name)
    return np.array(images), np.array(labels)

# Load data
images, labels = load_data(data_path)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)

# Create and train logistic regression model with increased max_iter
logistic_regression = LogisticRegression(max_iter=5000)
logistic_regression.fit(X_train, y_train)

# Predict on the test set
y_pred = logistic_regression.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8674562887760857
```