📚 Using `StandardScaler` with Train-Test Split

1. Import Libraries

First, import the necessary libraries:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

2. Create a NumPy Array

Let's create a sample NumPy array:

```
# Sample data as a NumPy array
data = np.array([[1, 10],
                 [2, 20],
                 [3, 30],
                 [4, 40],
                 [5, 50],
                 [6, 60],
                 [7, 70],
                 [8, 80],
                 [9, 90],
                 [10, 100]])

# Labels (for example purposes)
labels = np.array([0, 0, 0, 1, 1, 1, 1, 1, 0, 0])
```

3. Split the Data into Train and Test Sets

Use `train_test_split` to create training and testing datasets:

```
X_train, X_test, y_train, y_test = train_test_split(data, labels,
test_size=0.2, random_state=42)

print("Training Data:\n", X_train)
print("Testing Data:\n", X_test)
```

4. Instantiate the `StandardScaler`

Create an instance of `StandardScaler`:

```
scaler = StandardScaler()
```

5. Fit the Scaler to the Training Data

Fit the scaler only on the training data:

```
scaler.fit(X_train)
```

6. Transform Both Training and Testing Data

Transform both the training and testing datasets:

```
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

print("Scaled Training Data:\n", X_train_scaled)
print("Scaled Testing Data:\n", X_test_scaled)
```

7. Using `fit_transform` on Training Data

You can also use `fit_transform` for the training data:

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)  # Use transform for test data
```

8. Inverse Transform (Optional)

If you need to revert the scaled data back to its original form, you can use `inverse_transform`:

```
original_train_data = scaler.inverse_transform(X_train_scaled)
original_test_data = scaler.inverse_transform(X_test_scaled)

print("Original Training Data (after inverse transform):\n",
original_train_data)
print("Original Testing Data (after inverse transform):\n", original_test_data)
```

📝 Summary

- **Train-Test Split**: Use `train_test_split` to create separate training and testing datasets.
- **Fit on Training Data**: Fit the `StandardScaler` only on the training data to avoid data leakage.
- **Transform Both Sets**: Use `transform` to standardize both the training and testing datasets.
- **Inverse Transform**: You can revert the scaled data back to its original values if needed.

This approach ensures that your model is trained on scaled data while still being evaluated on unseen data in a consistent manner.