1-
PCA should be faster than both t-SNE and JL lemma + t-SNE because of the computational complexity of these methods. However, PCA might not capture the structure as well if there are non-linear relationships in the data. On the other hand, t-SNE might capture these relationships better, but at a cost of higher computational resources. Adding the JL lemma to t-SNE might help with the computational cost but could also potentially lose some important information due to the randomness of the projection.

For better understanding to our dataset i am going use to distinguish between the different classes . this will give a better visualization of the scatter plot

```
plt.scatter(digits_t2[:,0], digits_t2[:,1], c=digits.target,
cmap=plt.get_cmap('nipy_spectral'), alpha=.5)
```

```
plt.colorbar(label='digit label', ticks=range(10))
```

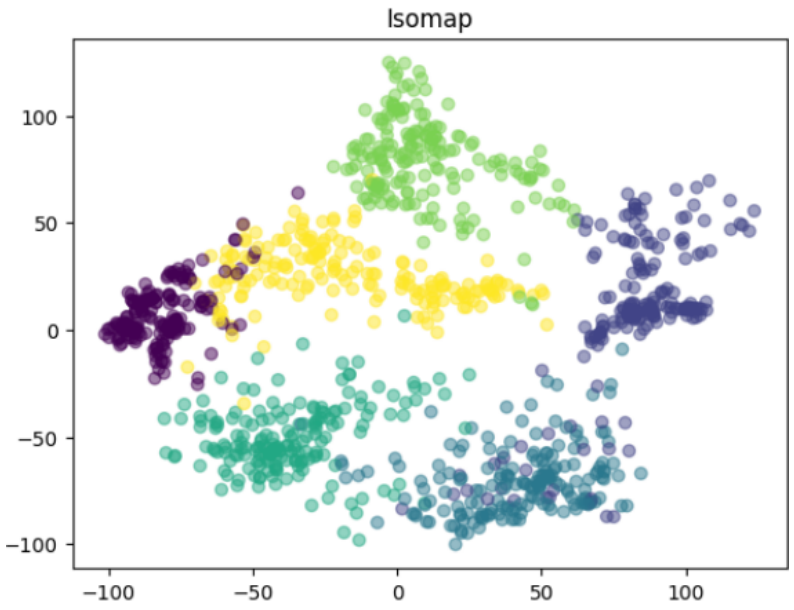
2-
MDS reduces dimensions while trying to keep the original distances between points, making it good for visualizing data but it might struggle with nonlinear data.while Isomap, like MDS, also keeps the original distances but uses a graph-based approach to deal with both linear and nonlinear data, making it more versatile.and LLE focuses on preserving relationships between neighboring points instead of all points, which allows it to capture nonlinear structures well but it might lose some global relationships.

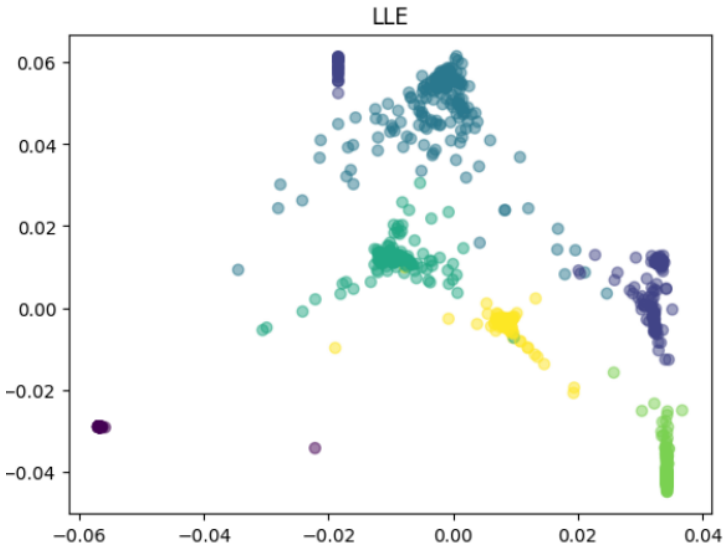
find the best k(n_neighbor) for Isomap and LLE

```
from sklearn.metrics import silhouette_score

best_score_iso = -1
best_score_lle = -1
best_neighbors_iso = None
best_neighbors_lle = None

for n_neighbors in range(10, 31):  # Try values from 10 to 30
    #... rest of your code
    # assuming reasonable values between 2 and 50
    iso = Isomap(n_components=2, n_neighbors=n_neighbors)
    lle = LocallyLinearEmbedding(n_components=2, method="standard",
n_neighbors=n_neighbors)

    # For Isomap
    iso.fit(digits.data)
    digits_projected_iso = iso.transform(digits.data)
    score_iso = silhouette_score(digits_projected_iso, digits.target)
```

```
    if score_iso > best_score_iso:
        best_score_iso = score_iso
        best_neighbors_iso = n_neighbors

    # For LLE
    lle.fit(digits.data)
    digits_projected_lle = lle.transform(digits.data)
    score_lle = silhouette_score(digits_projected_lle, digits.target)

    if score_lle > best_score_lle:
        best_score_lle = score_lle
        best_neighbors_lle = n_neighbors

print(f"Best n_neighbors for Isomap: {best_neighbors_iso}")
print(f"Best n_neighbors for LLE: {best_neighbors_lle}")
```

Best n_neighbors for Isomap: 13
Best n_neighbors for LLE: 10




LLE
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.04
−0.02
0.00
0.02
0.04