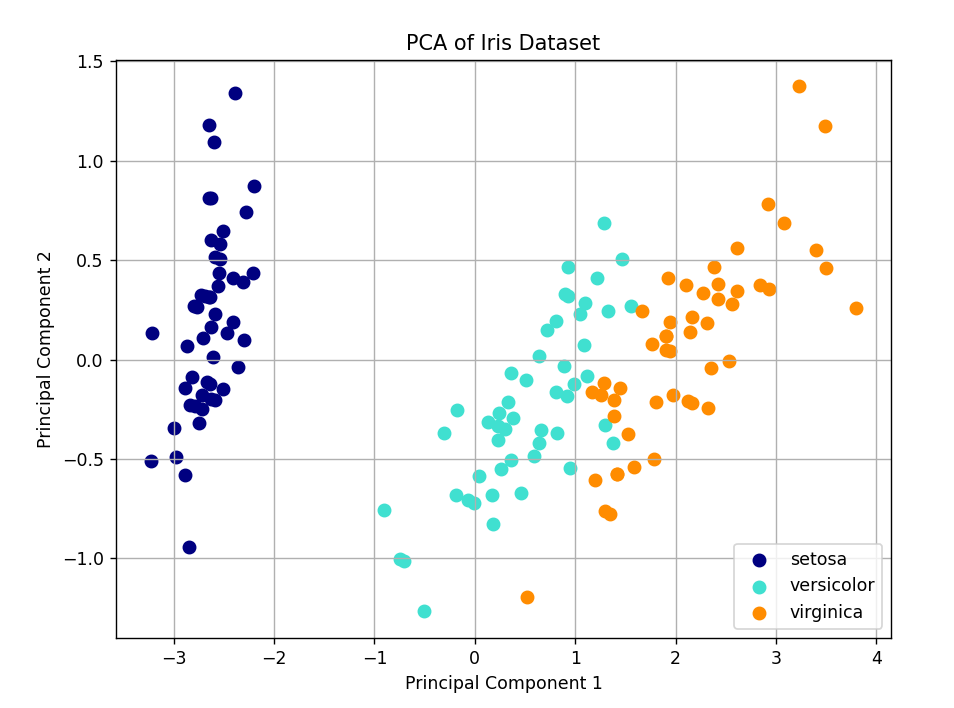
1. Задание 1: Применение метода главных компонент (PCA).

Используйте набор данных "Iris" из sklearn.datasets. Примените PCA, чтобы снизить размерность до 2-х и визуализируйте результаты.  


iris = load\_iris()

X = iris.data

y = iris.target

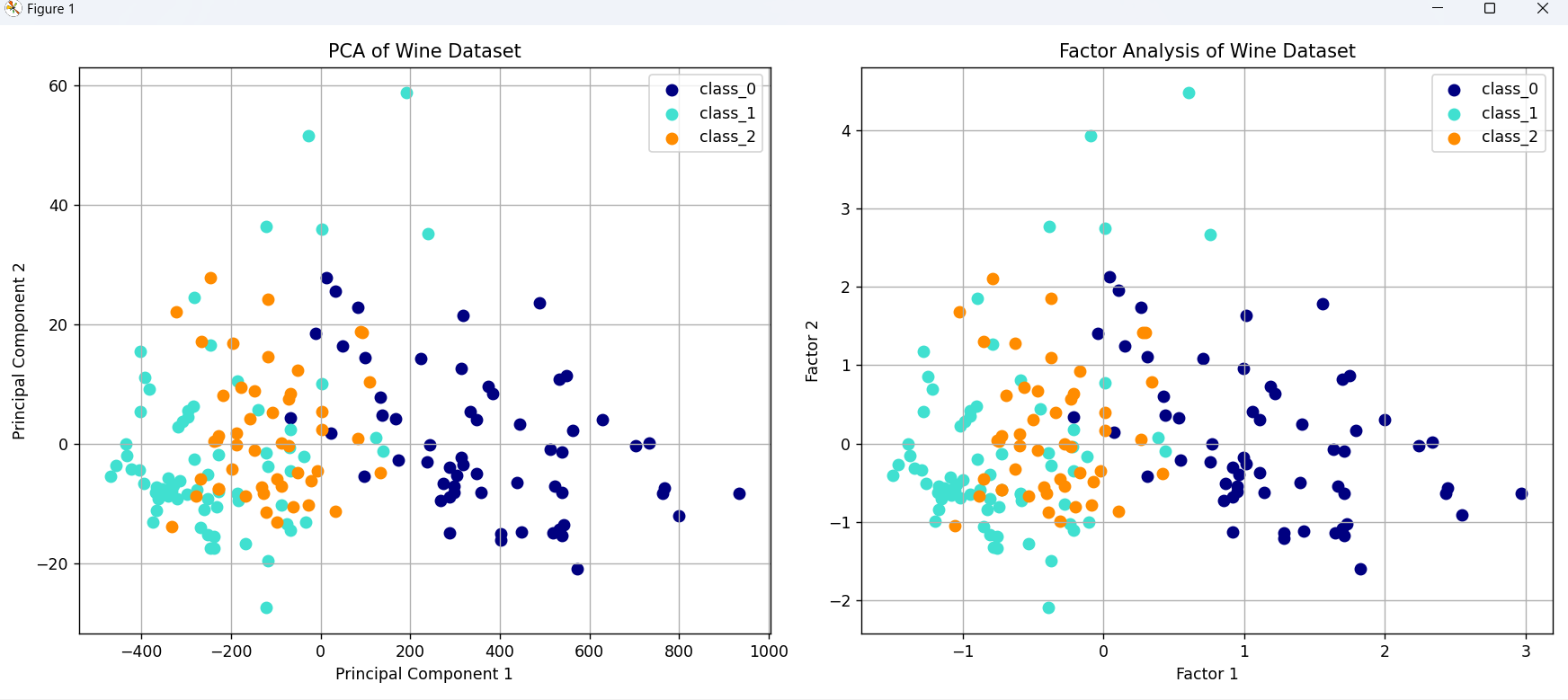
target\_names = iris.target\_names

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X)

2. Задание 2: Сравнение PCA и Factor Analysis.

Используйте набор данных "Wine" из sklearn.datasets. Примените PCA и Factor Analysis, чтобы снизить размерность до 2-х и визуализируйте различия в результатах.



wine = load\_wine()

X = wine.data

y = wine.target

target\_names = wine.target\_names

# PCA

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X)

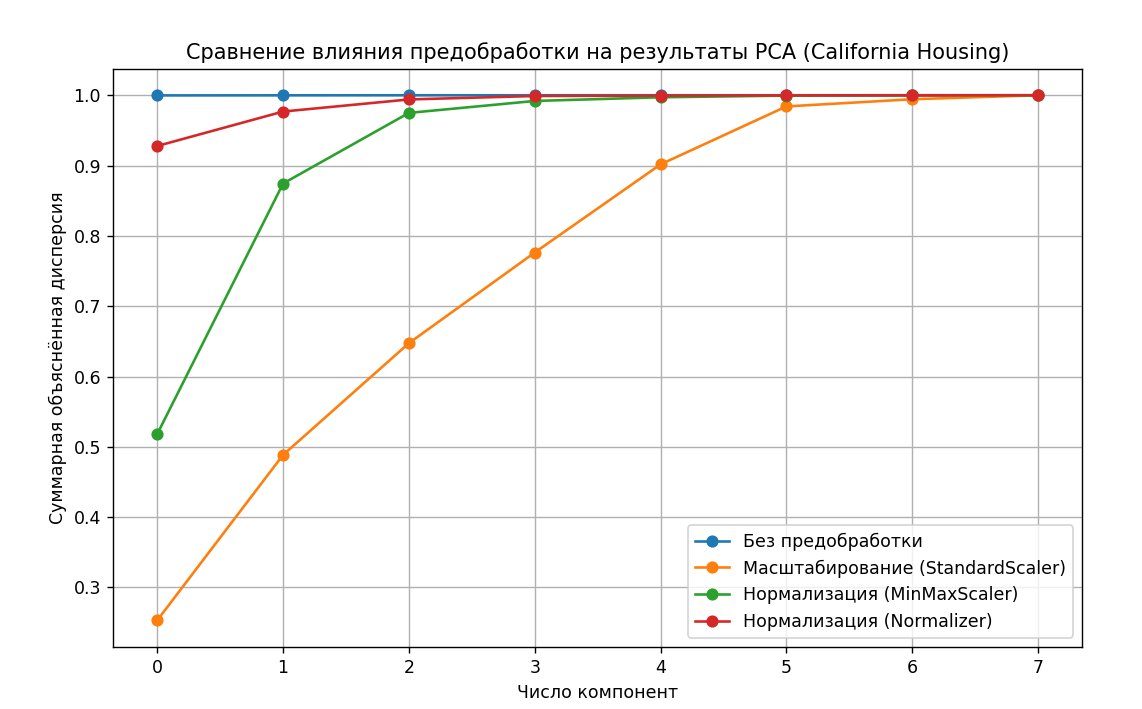
# Factor Analysis

fa = FactorAnalysis(n\_components=2, random\_state=42)

X\_fa = fa.fit\_transform(X)

3. Задание 3: Исследование влияния предобработки данных на результаты PCA.

Используйте набор данных "Boston Housing" из sklearn.datasets. Примените различные методы предобработки (например, масштабирование, нормализацию) перед применением PCA и сравните полученные результаты.



ata = fetch\_california\_housing()

X = pd.DataFrame(data.data, columns=data.feature\_names)

y = data.target

def apply\_pca(data, n\_components=8):

    pca = PCA(n\_components=n\_components)

    principal\_components = pca.fit\_transform(data)

    explained\_variance = pca.explained\_variance\_ratio\_

    return explained\_variance, principal\_components

explained\_variance\_raw, \_ = apply\_pca(X)

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

explained\_variance\_scaled, \_ = apply\_pca(X\_scaled)

minmax\_scaler = MinMaxScaler()

X\_minmax = minmax\_scaler.fit\_transform(X)

explained\_variance\_minmax, \_ = apply\_pca(X\_minmax)

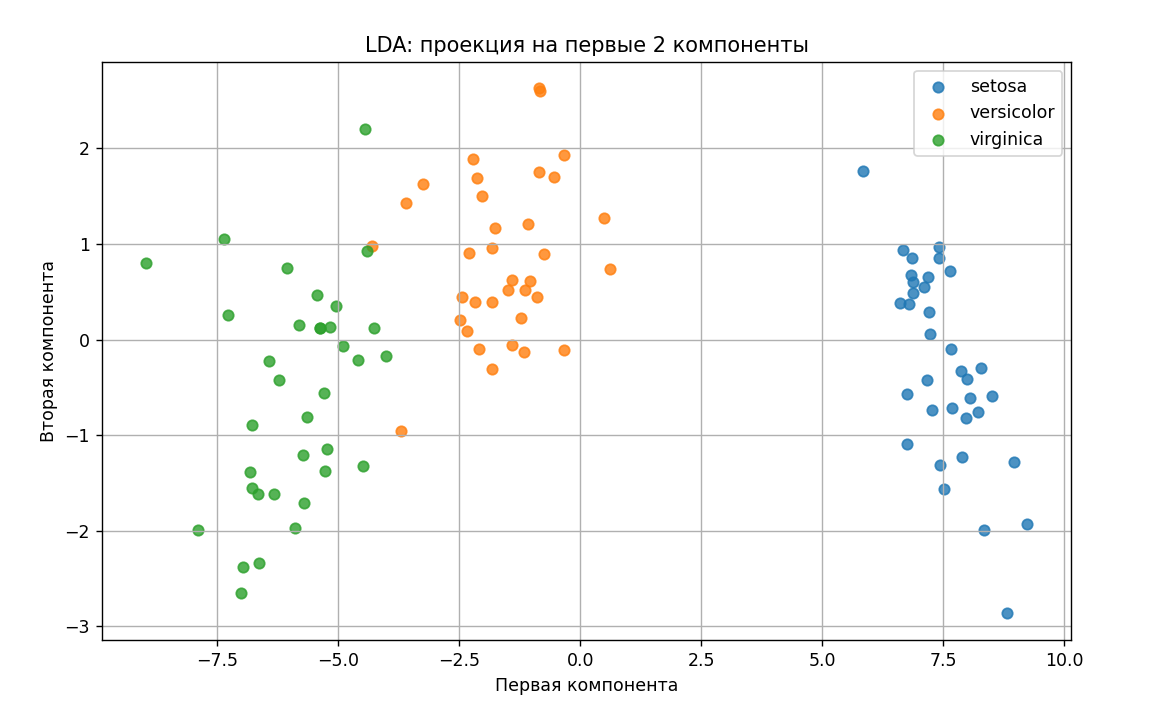
normalizer = Normalizer()

X\_normalized = normalizer.fit\_transform(X)

explained\_variance\_normalized, \_ = apply\_pca(X\_normalized)

4. Задание 4: Применение Disciminant Analysis.

Используйте набор данных "Iris" из sklearn.datasets. Примените Linear Discriminant Analysis (LDA) и Quadratic Discriminant Analysis (QDA), чтобы снизить размерность до 2-х и визуализируйте результаты.



X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42, stratify=y)

lda = LDA(n\_components=2)

X\_lda = lda.fit\_transform(X\_train, y\_train)

plt.figure(figsize=(10, 6))

for i, target\_name in enumerate(target\_names):

    plt.scatter(X\_lda[y\_train == i, 0], X\_lda[y\_train == i, 1], alpha=0.8, label=target\_name)

qda = QDA()

qda.fit(X\_train, y\_train)

y\_pred\_qda = qda.predict(X\_test)

accuracy\_qda = accuracy\_score(y\_test, y\_pred\_qda)

print(f"Точность классификации QDA: {accuracy\_qda:.2f}")

X\_test\_lda = lda.transform(X\_test)

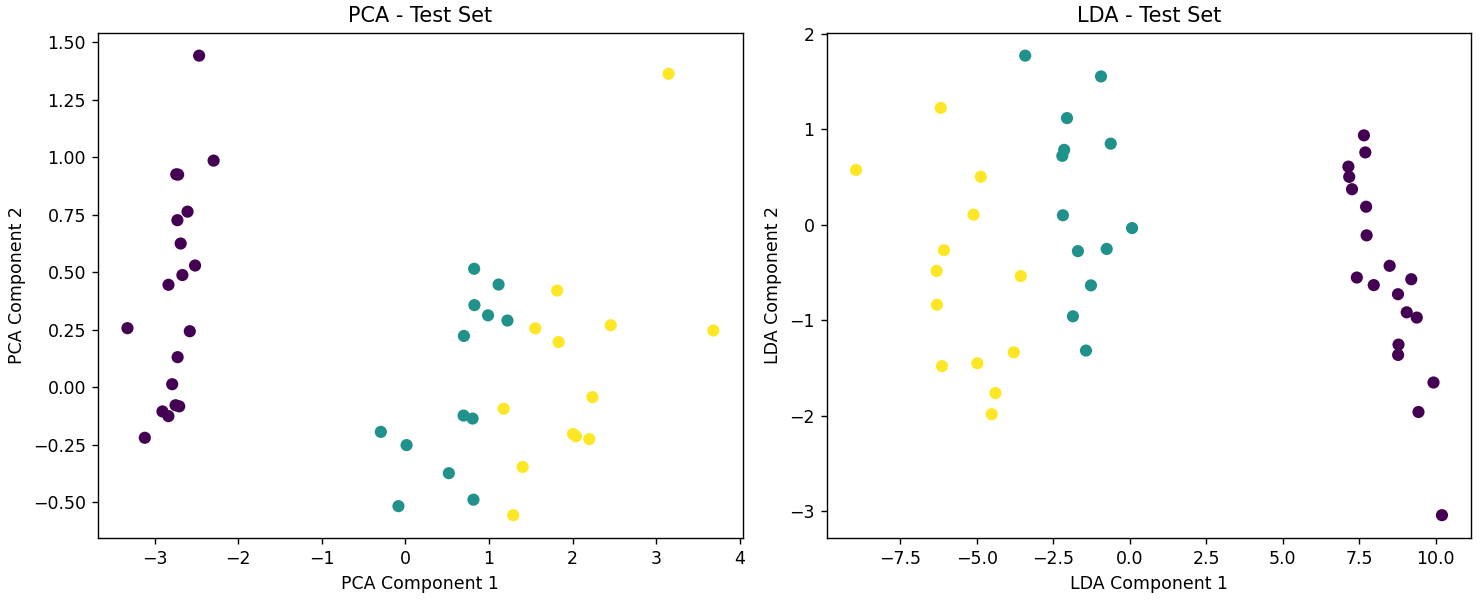
plt.figure(figsize=(10, 6))

for i, target\_name in enumerate(target\_names):

    plt.scatter(X\_test\_lda[y\_test == i, 0], X\_test\_lda[y\_test == i, 1], alpha=0.8, label=f"Истинный класс: {target\_name}")

5. Задание 5: Сравнение PCA и LDA.

Используйте любой набор данных с классифицирующей моделью. Примените PCA и LDA и сравните, как влияют эти методы снижения размерности на эффективность классификации.



X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

#PCA

pca = PCA(n\_components=2)

X\_train\_pca = pca.fit\_transform(X\_train)

X\_test\_pca = pca.transform(X\_test)

#LDA

lda = LDA(n\_components=2)

X\_train\_lda = lda.fit\_transform(X\_train, y\_train)

X\_test\_lda = lda.transform(X\_test)

logreg = LogisticRegression(max\_iter=200)

logreg.fit(X\_train, y\_train)

y\_pred = logreg.predict(X\_test)

accuracy\_original = accuracy\_score(y\_test, y\_pred)

logreg.fit(X\_train\_pca, y\_train)

y\_pred\_pca = logreg.predict(X\_test\_pca)

accuracy\_pca = accuracy\_score(y\_test, y\_pred\_pca)

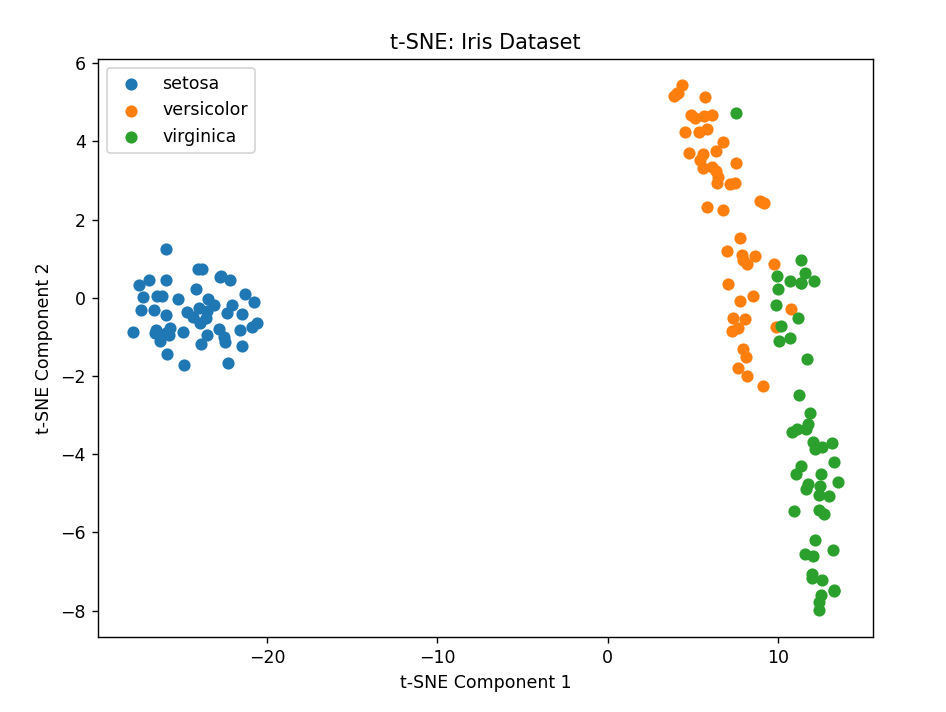
logreg.fit(X\_train\_lda, y\_train)

y\_pred\_lda = logreg.predict(X\_test\_lda)

accuracy\_lda = accuracy\_score(y\_test, y\_pred\_lda)

1. Задание 1: Применение метода t-SNE.

Используйте набор данных "Iris" из sklearn.datasets. Примените t-SNE, чтобы снизить размерность до 2-х, и визуализируйте результаты.



data = load\_iris()

X = data.data

y = data.target

target\_names = data.target\_names

#t-SNE

tsne = TSNE(n\_components=2, random\_state=42)

X\_tsne = tsne.fit\_transform(X)

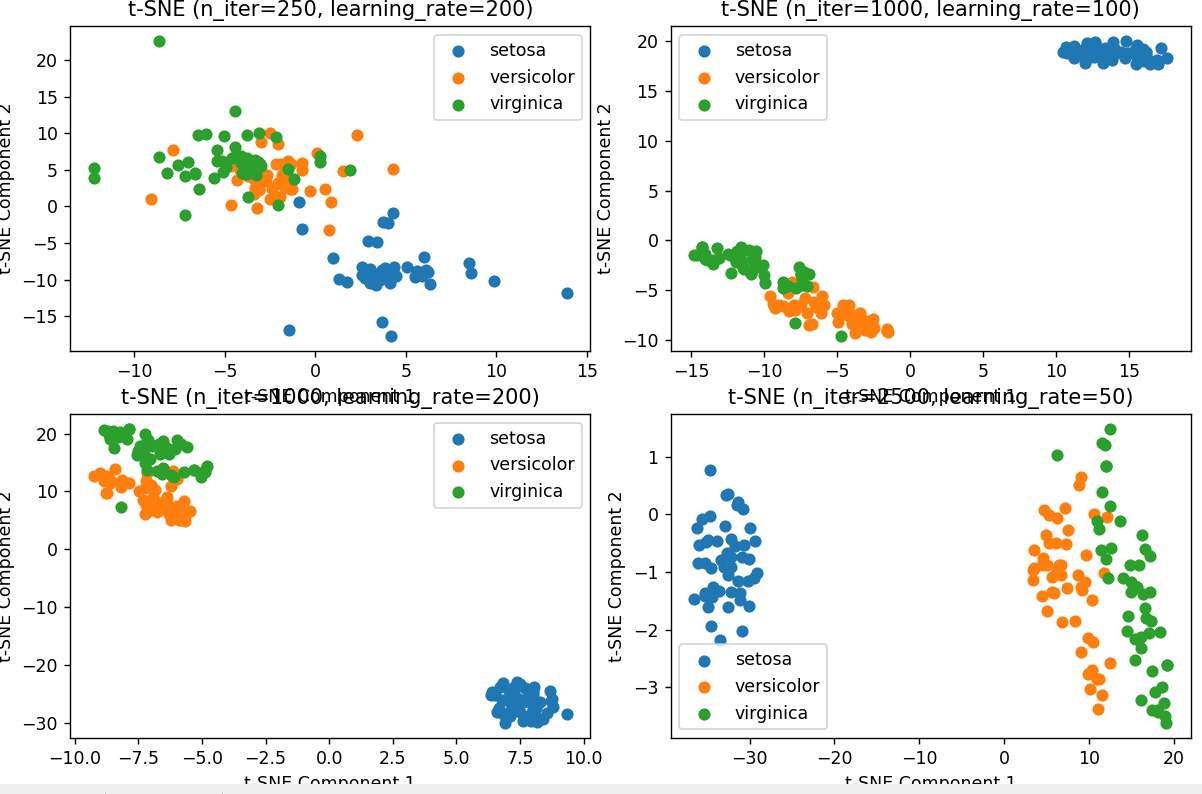
plt.figure(figsize=(8, 6))

for i, target\_name in enumerate(target\_names):

    plt.scatter(X\_tsne[y == i, 0], X\_tsne[y == i, 1], label=target\_name)

2. Задание 2: Определение влияния параметров t-SNE.

Используйте тот же набор данных "Iris". Примените t-SNE с разными значениями параметров (например, число итераций, learning rate) и сравните полученные результаты.



params = [

    {'n\_iter': 250, 'learning\_rate': 200},

    {'n\_iter': 1000, 'learning\_rate': 100},

    {'n\_iter': 1000, 'learning\_rate': 200},

    {'n\_iter': 2500, 'learning\_rate': 50}

]

fig, axes = plt.subplots(2, 2, figsize=(12, 10))

axes = axes.flatten()

for i, param in enumerate(params):

    tsne = TSNE(n\_components=2, random\_state=42, n\_iter=param['n\_iter'], learning\_rate=param['learning\_rate'])

    X\_tsne = tsne.fit\_transform(X)

    ax = axes[i]

    for j, target\_name in enumerate(target\_names):

        ax.scatter(X\_tsne[y == j, 0], X\_tsne[y == j, 1], label=target\_name)

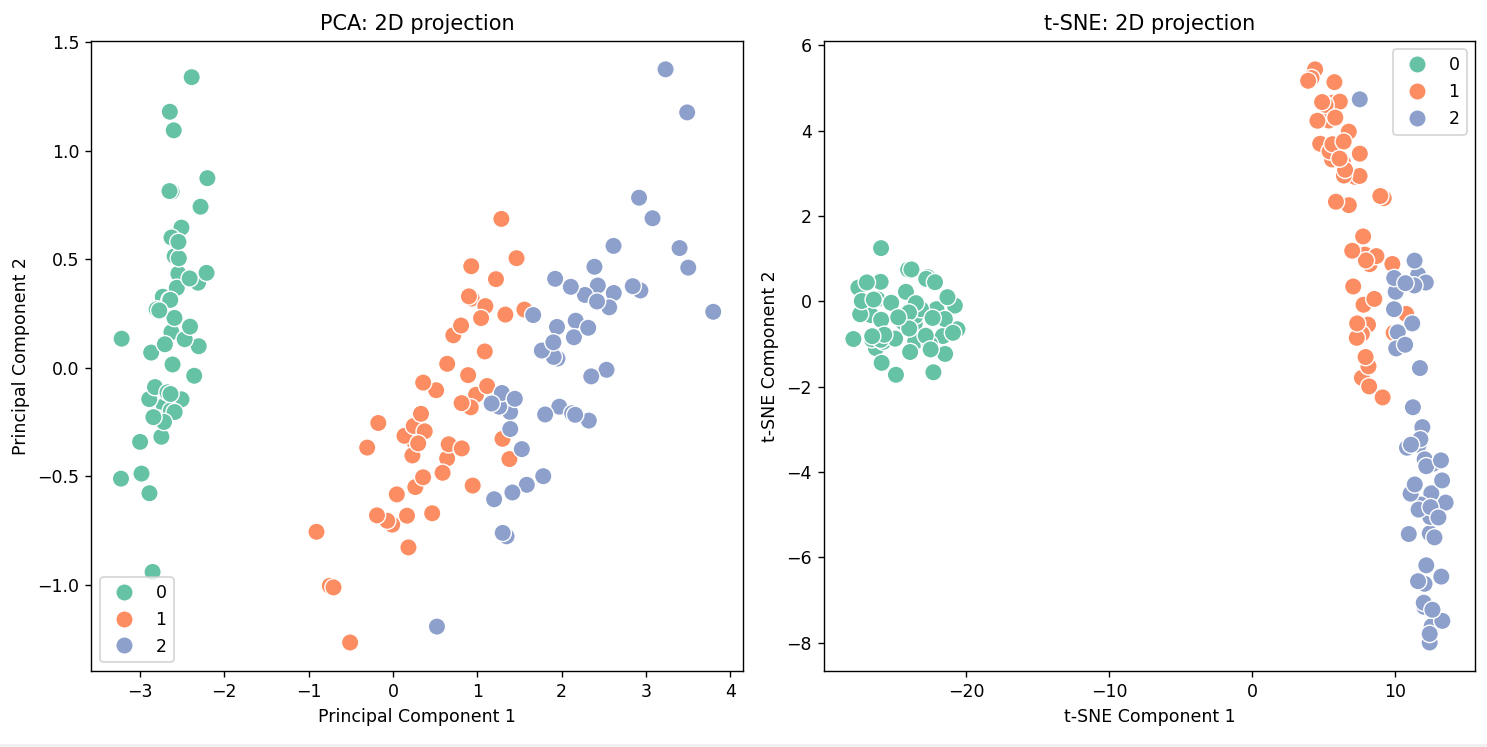
    ax.set\_title(f"t-SNE (n\_iter={param['n\_iter']}, learning\_rate={param['learning\_rate']})")

    ax.set\_xlabel('t-SNE Component 1')

    ax.set\_ylabel('t-SNE Component 2')

3. Задание 3: Сравнение t-SNE и PCA.

Используйте любой набор данных на ваше усмотрение. Примените PCA и t-SNE, чтобы снизить размерность до 2-х, и сравните разницы в результатах визуализации.



data = load\_iris()

X = data.data

y = data.target

labels = data.target\_names

#PCA

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X)

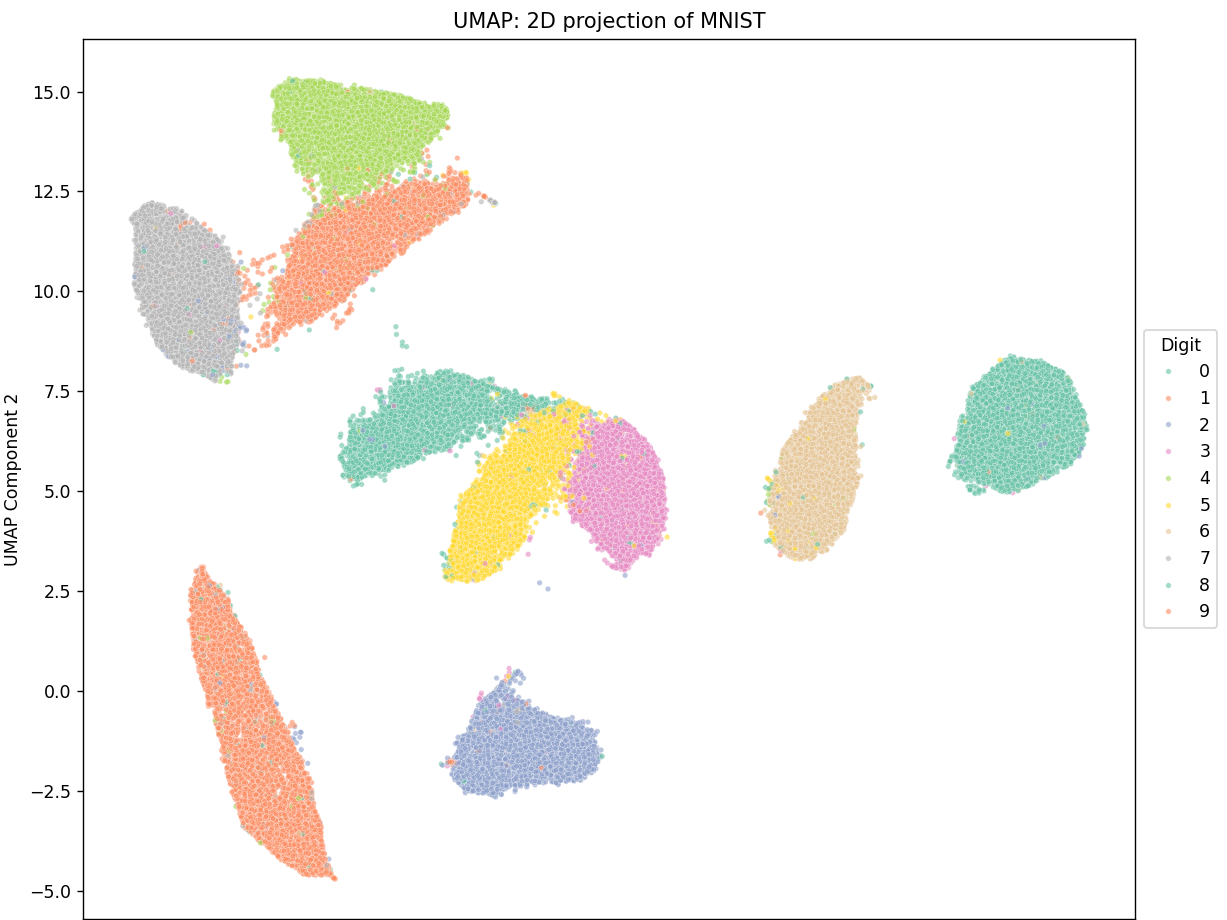
#t-SNE

tsne = TSNE(n\_components=2, random\_state=42)

X\_tsne = tsne.fit\_transform(X)

4. Задание 4: Применение UMAP.

Используйте набор данных "MNIST" (например, содержащийся в sklearn.datasets). Примените UMAP для снижения размерности и визуализируйте результаты.



mnist = fetch\_openml('mnist\_784', version=1)

X, y = mnist["data"], mnist["target"]

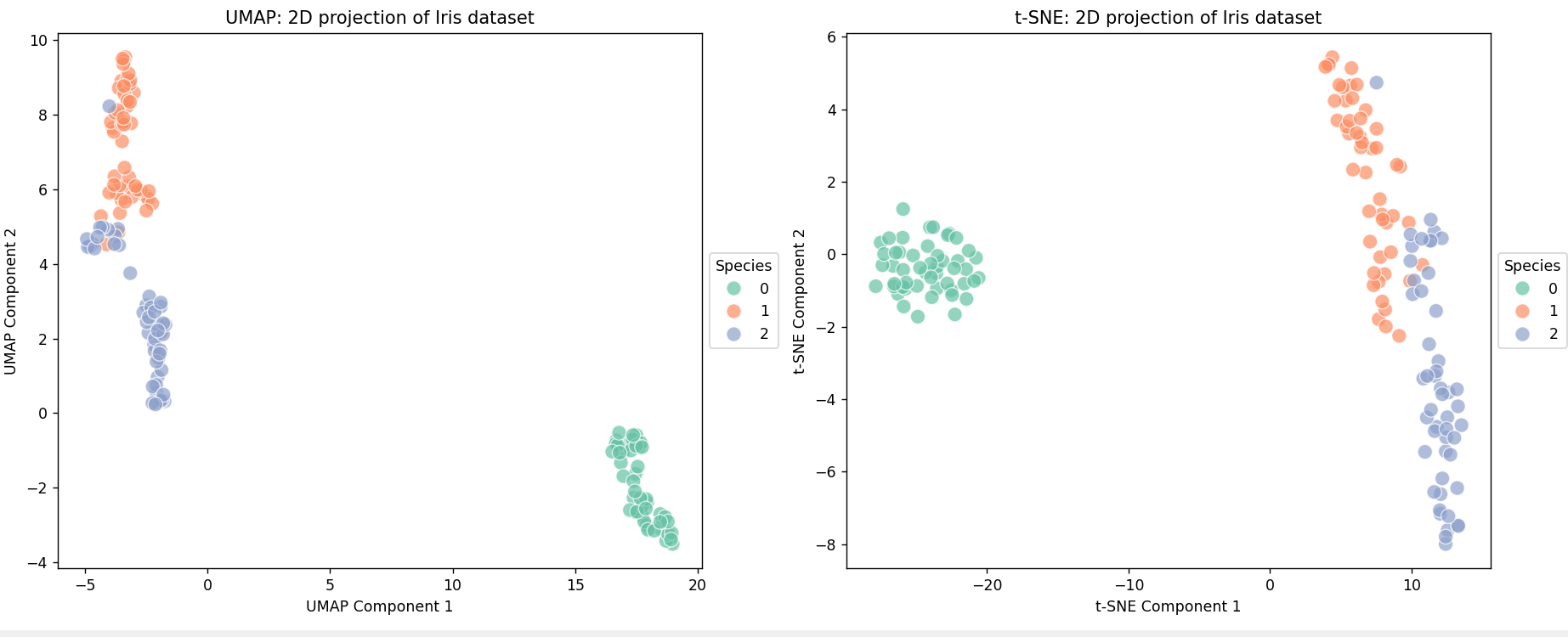
y = y.astype(int)

umap\_model = umap.UMAP(n\_components=2, random\_state=42)

X\_umap = umap\_model.fit\_transform(X)

5. Задание 5: Сравнение UMAP и t-SNE.

Используйте один и тот же набор данных для применения UMAP и t-SNE. Сравните влияние этих методов снижения размерности на визуальное разделение классов в данных.



iris = load\_iris()

X = iris.data

y = iris.target

labels = iris.target\_names

umap\_model = umap.UMAP(n\_components=2, random\_state=42)

X\_umap = umap\_model.fit\_transform(X)

tsne\_model = TSNE(n\_components=2, random\_state=42, perplexity=30, n\_iter=1000)

X\_tsne = tsne\_model.fit\_transform(X)