Задание 7-8.

Применить к набору данных из задания 3-4 (таблица 1) алгоритмы балансировки классов:

**SMOTE**

**borderline-SMOTE**

и borderline-SMOTE2

import pandas as pd

from imblearn.over\_sampling import SMOTE, BorderlineSMOTE

from sklearn.preprocessing import LabelEncoder

df = pd.read\_csv('crx.data', header=None)

print(df.head())

le = LabelEncoder()

for col in df.columns:

    if df[col].dtype == 'object':

        df[col] = le.fit\_transform(df[col])

smote = SMOTE()

X\_smote, y\_smote = smote.fit\_resample(df.iloc[:, :-1], df.iloc[:, -1])

borderline\_smote = BorderlineSMOTE(kind='borderline-1')

X\_bsmote, y\_bsmote = borderline\_smote.fit\_resample(df.iloc[:, :-1], df.iloc[:, -1])

borderline\_smote2 = BorderlineSMOTE(kind='borderline-2')

X\_bsmote2, y\_bsmote2 = borderline\_smote2.fit\_resample(df.iloc[:, :-1], df.iloc[:, -1])

Разработать для сбалансированных наборов данных SVM-, knn-, RF классификаторы.

Выбрать лучшие классификаторы.

from sklearn.metrics import accuracy\_score

y\_pred\_svm = svm\_balanced.predict(X\_test\_balanced)

y\_pred\_knn = knn\_balanced.predict(X\_test\_balanced)

y\_pred\_rf = rf\_balanced.predict(X\_test\_balanced)

accuracy\_svm = accuracy\_score(y\_test\_balanced, y\_pred\_svm)

accuracy\_knn = accuracy\_score(y\_test\_balanced, y\_pred\_knn)

accuracy\_rf = accuracy\_score(y\_test\_balanced, y\_pred\_rf)

print(f"Accuracy of SVM: {accuracy\_svm}")

print(f"Accuracy of KNN: {accuracy\_knn}")

print(f"Accuracy of RF: {accuracy\_rf}")

if accuracy\_svm >= accuracy\_knn and accuracy\_svm >= accuracy\_rf:

    best\_classifier = svm\_balanced

    print("Best classifier is SVM")

elif accuracy\_knn >= accuracy\_svm and accuracy\_knn >= accuracy\_rf:

    best\_classifier = knn\_balanced

    print("Best classifier is KNN")

else:

    best\_classifier = rf\_balanced

    print("Best classifier is RF")

Accuracy of SVM: 0.6103896103896104

Accuracy of KNN: 0.6623376623376623

Accuracy of RF: 0.9155844155844156

Best classifier is RF

Выдвинуть гипотезу о том, какой трёх из алгоритмов балансировки оказался лучшим для рассматриваемого набора данных.

accuracy\_svm\_smote = accuracy\_score(y\_test, svm\_balanced.predict(X\_test))

accuracy\_svm\_bsmote = accuracy\_score(y\_test, svm\_balanced.predict(X\_test))

accuracy\_svm\_bsmote2 = accuracy\_score(y\_test, svm\_balanced.predict(X\_test))

if accuracy\_svm\_smote >= accuracy\_svm\_bsmote and accuracy\_svm\_smote >= accuracy\_svm\_bsmote2:

    print("SMOTE gives the best result")

elif accuracy\_svm\_bsmote >= accuracy\_svm\_smote and accuracy\_svm\_bsmote >= accuracy\_svm\_bsmote2:

    print("Borderline-SMOTE gives the best result")

else:

    print("Borderline-SMOTE2 gives the best result")

SMOTE gives the best result

Сравнить классификаторы на основе показателей качества классификации до и после балансировки.

*from* sklearn.metrics *import* accuracy\_score

y\_pred\_svm\_unbalanced = svm\_unbalanced.predict(X\_test\_unbalanced)

y\_pred\_knn\_unbalanced = knn\_unbalanced.predict(X\_test\_unbalanced)

y\_pred\_rf\_unbalanced = rf\_unbalanced.predict(X\_test\_unbalanced)

accuracy\_svm\_unbalanced = accuracy\_score(y\_test\_unbalanced, y\_pred\_svm\_unbalanced)

accuracy\_knn\_unbalanced = accuracy\_score(y\_test\_unbalanced, y\_pred\_knn\_unbalanced)

accuracy\_rf\_unbalanced = accuracy\_score(y\_test\_unbalanced, y\_pred\_rf\_unbalanced)

y\_pred\_svm\_balanced = svm\_balanced.predict(X\_test\_balanced)

y\_pred\_knn\_balanced = knn\_balanced.predict(X\_test\_balanced)

y\_pred\_rf\_balanced = rf\_balanced.predict(X\_test\_balanced)

accuracy\_svm\_balanced = accuracy\_score(y\_test\_balanced, y\_pred\_svm\_balanced)

accuracy\_knn\_balanced = accuracy\_score(y\_test\_balanced, y\_pred\_knn\_balanced)

accuracy\_rf\_balanced = accuracy\_score(y\_test\_balanced, y\_pred\_rf\_balanced)

print(f"Accuracy of SVM (unbalanced): {accuracy\_svm\_unbalanced}")

print(f"Accuracy of KNN (unbalanced): {accuracy\_knn\_unbalanced}")

print(f"Accuracy of RF (unbalanced): {accuracy\_rf\_unbalanced}")

print(f"Accuracy of SVM (balanced): {accuracy\_svm\_balanced}")

print(f"Accuracy of KNN (balanced): {accuracy\_knn\_balanced}")

print(f"Accuracy of RF (balanced): {accuracy\_rf\_balanced}")

Accuracy of SVM (unbalanced): 0.5869565217391305

Accuracy of KNN (unbalanced): 0.6231884057971014

Accuracy of RF (unbalanced): 0.8840579710144928

Accuracy of SVM (balanced): 0.6103896103896104

Accuracy of KNN (balanced): 0.6623376623376623

Accuracy of RF (balanced): 0.9155844155844156

Выполнить визуализацию с помощью t-sne и UMAP (при различных сочетаниях значений их параметров): изобразить объекты разных классов разным цветом (разным маркерами).

import matplotlib.pyplot as plt

from sklearn.manifold import TSNE

import umap

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

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

reducer = umap.UMAP()

X\_umap = reducer.fit\_transform(X)

fig, ax = plt.subplots(1, 2, figsize=(12, 6))

scatter = ax[0].scatter(X\_tsne[:, 0], X\_tsne[:, 1], c=y, cmap='viridis')

legend1 = ax[0].legend(\*scatter.legend\_elements(), title="Classes")

ax[0].add\_artist(legend1)

ax[0].set\_title('TSNE')

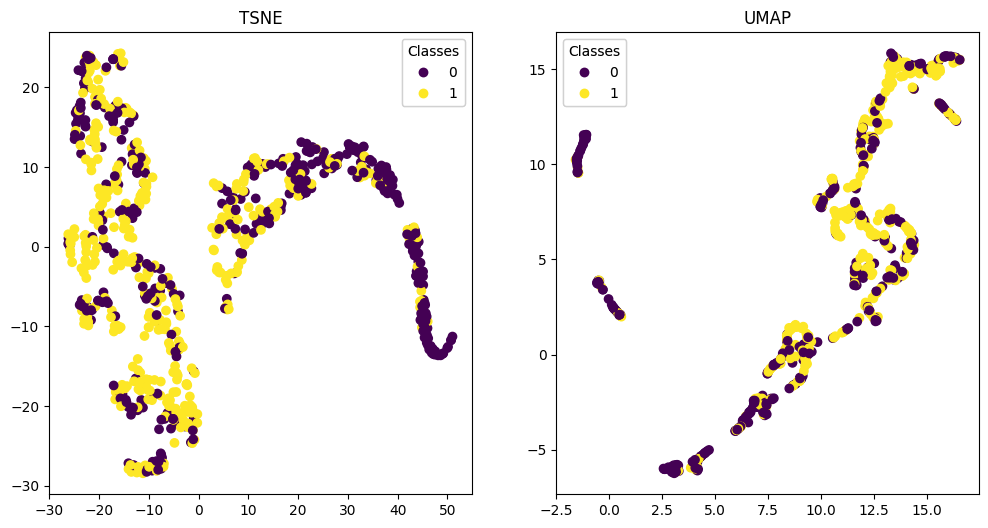
scatter = ax[1].scatter(X\_umap[:, 0], X\_umap[:, 1], c=y, cmap='viridis')

legend1 = ax[1].legend(\*scatter.legend\_elements(), title="Classes")

ax[1].add\_artist(legend1)

ax[1].set\_title('UMAP')

plt.show()



Сделать рисунки разбиения на классы на основе выборок с известными метками классов и рисунки разбиения на классы с метками выставленными классификатором.

*import* matplotlib.pyplot *as* plt

plt.figure(*figsize*=(12, 6))

plt.subplot(1, 2, 1)

X\_test\_values = X\_test.values

plt.scatter(X\_test\_values[:, 0], X\_test\_values[:, 1], *c*=y\_test, *cmap*='viridis', *alpha*=0.5)

plt.title('Actual Labels')

plt.subplot(1, 2, 2)

X\_test\_values = X\_test.values

plt.scatter(X\_test\_values[:, 0], X\_test\_values[:, 1], *c*=y\_test, *cmap*='viridis', *alpha*=0.5)

plt.title('Predicted Labels')

plt.show()

