Задание 1.
Исходные данные:
К алгоритму kNN, реализованному на уроке, реализовать добавление весов соседей по любому из показанных на уроке принципов.
Решение:

```
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> import scipy
>>> import sklearn
>>> import matplotlib.pyplot as plt
>>> from matplotlib.colors import ListedColormap
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import classification_report
>>>
>>> from sklearn.neighbors import KNeighborsClassifier
>>> from itertools import product
>>> iris = load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
>>> knn = KNeighborsClassifier(n_neighbors=3, weights="uniform")
>>> knn.fit(X_train, y_train)
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
>>> y_hat = knn.predict(X_test)
>>> print(classification_report(y_test, y_hat))
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       0.78      0.44      0.56        16
           2       0.44      0.78      0.56         9

    accuracy                           0.71        38
   macro avg       0.74      0.74      0.71        38
weighted avg       0.77      0.71      0.71        38

>>> def plot_decision_boundary(model, X, y):
...     color = ["r", "g", "b"]
...     marker = ["o", "v", "x"]
...     class_label = np.unique(y)
...     cmap = ListedColormap(color[: len(class_label)])
...     x1_min, x2_min = np.min(X, axis=0)
...     x1_max, x2_max = np.max(X, axis=0)
...     x1 = np.arange(x1_min - 1, x1_max + 1, 0.02)
...     x2 = np.arange(x2_min - 1, x2_max + 1, 0.02)
...     X1, X2 = np.meshgrid(x1, x2)
...     Z = model.predict(np.c_[X1.ravel(), X2.ravel()])
...     Z = Z.reshape(X1.shape)
```

```
...     plt.contourf(X1, X2, Z, cmap=cmap, alpha=0.5)
...     for i, class_ in enumerate(class_label):
...         plt.scatter(x=X[y == class_, 0], y=X[y == class_, 1],
...                     c=cmap.colors[i], label=class_, marker=marker[i])
...     plt.legend()
...
>>> plt.figure(figsize=(18, 10))
<Figure size 1800x1000 with 0 Axes>
>>> weights = ['uniform', 'distance']
>>> ks = [2, 15]
>>> for i, (w, k) in enumerate(product(weights, ks), start=1):
...     plt.subplot(2, 2, i)
...     plt.title(f"Значение K: {k} вес: {w}")
...     knn = KNeighborsClassifier(n_neighbors=k, weights=w)
...     knn.fit(X, y)
...     plot_decision_boundary(knn, X_train, y_train)
...
<matplotlib.axes._subplots.AxesSubplot object at 0x7fdbc04c4610>
Text(0.5, 1.0, 'Значение K: 2 вес: uniform')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='uniform')
<matplotlib.axes._subplots.AxesSubplot object at 0x7fdbc1fb0640>
Text(0.5, 1.0, 'Значение K: 15 вес: uniform')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=15, p=2,
                     weights='uniform')
<matplotlib.axes._subplots.AxesSubplot object at 0x7fdbc89cecd0>
Text(0.5, 1.0, 'Значение K: 2 вес: distance')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='distance')
<matplotlib.axes._subplots.AxesSubplot object at 0x7fdbbf6d1130>
Text(0.5, 1.0, 'Значение K: 15 вес: distance')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=15, p=2,
                     weights='distance')
>>> plt.show()
>>>
```

Задание 2.
Исходные данные:
Написать функцию подсчета метрики качества кластеризации как среднее квадратичное внутрикластерное расстояние и построить график ее зависимости от k (взять от 1 до 10) для выборки данных из данного урока.
Решение:

```
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> import scipy
```

```
>>> import sklearn
>>> import matplotlib.pyplot as plt
>>> from matplotlib.colors import ListedColormap
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import classification_report
>>> from sklearn.neighbors import KNeighborsClassifier
>>> from itertools import product
>>> iris = load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
>>> knn = KNeighborsClassifier(n_neighbors=1, weights="uniform")
>>> knn.fit(X_train, y_train)
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
>>> y_hat = knn.predict(X_test)
>>> classification_report(y_test, y_hat)
'              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00        13\n           1
0.69      0.56      0.62        16\n           2       0.42      0.56      0.48         9\n\n    accuracy
0.71        38\n   macro avg       0.70      0.71      0.70        38\nweighted avg       0.73      0.71      0.72
38\n'
>>> def plot_decision_boundary(model, X, y):
...     color = ["r", "g", "b"]
...     marker = ["o", "v", "x"]
...     class_label = np.unique(y)
...     cmap = ListedColormap(color[: len(class_label)])
...     x1_min, x2_min = np.min(X, axis=0)
...     x1_max, x2_max = np.max(X, axis=0)
...     x1 = np.arange(x1_min - 1, x1_max + 1, 0.02)
...     x2 = np.arange(x2_min - 1, x2_max + 1, 0.02)
...     X1, X2 = np.meshgrid(x1, x2)
...     Z = model.predict(np.c_[X1.ravel(), X2.ravel()])
...     Z = Z.reshape(X1.shape)
...     plt.contourf(X1, X2, Z, cmap=cmap, alpha=0.5)
...     for i, class_ in enumerate(class_label):
...         plt.scatter(x=X[y == class_, 0], y=X[y == class_, 1],
...                 c=cmap.colors[i], label=class_, marker=marker[i])
...     plt.legend()
...
>>> plt.figure(figsize=(18, 10))
<Figure size 1800x1000 with 0 Axes>
>>> weights = ['uniform', 'distance']
>>> ks = [2, 15]
>>> for i, (w, k) in enumerate(product(weights, ks), start=1):
...     plt.subplot(2, 2, i)
...     plt.title(f"Значение K: {k} вес: {w}")
...     knn = KNeighborsClassifier(n_neighbors=k, weights=w)
...     knn.fit(X, y)
...     plot_decision_boundary(knn, X_train, y_train)
...
<matplotlib.axes._subplots.AxesSubplot object at 0x7f86271bdc40>
```

```
Text(0.5, 1.0, 'Значение K: 2 вес: uniform')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='uniform')
<matplotlib.axes._subplots.AxesSubplot object at 0x7f8628b1cc40>
Text(0.5, 1.0, 'Значение K: 15 вес: uniform')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=15, p=2,
                     weights='uniform')
<matplotlib.axes._subplots.AxesSubplot object at 0x7f862643e1c0>
Text(0.5, 1.0, 'Значение K: 2 вес: distance')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='distance')
<matplotlib.axes._subplots.AxesSubplot object at 0x7f86271561c0>
Text(0.5, 1.0, 'Значение K: 15 вес: distance')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=15, p=2,
                     weights='distance')
>>> plt.show()
>>> import numpy as np
>>> import pandas as pd
>>> import scipy
>>> import sklearn
>>> import sklearn
>>> import matplotlib.pyplot as plt
>>> from matplotlib.colors import ListedColormap
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import classification_report
>>> from sklearn.neighbors import KNeighborsClassifier
>>> from itertools import product
>>> iris = load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
>>> knn = KNeighborsClassifier(n_neighbors=2, weights="uniform")
>>> knn.fit(X_train, y_train)
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='uniform')
>>> y_hat = knn.predict(X_test)
>>> classification_report(y_test, y_hat)
'              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00        13\n           1
0.71      0.75      0.73        16\n           2       0.50      0.44      0.47         9\n\n    accuracy
0.76        38\n   macro avg       0.74      0.73      0.73        38\nweighted avg       0.76      0.76      0.76
38\n'
>>> def plot_decision_boundary(model, X, y):
...     color = ["r", "g", "b"]
...     marker = ["o", "v", "x"]
...     class_label = np.unique(y)
...     cmap = ListedColormap(color[: len(class_label)])
...     x1_min, x2_min = np.min(X, axis=0)
```

```
...     x1_max, x2_max = np.max(X, axis=0)
...     x1 = np.arange(x1_min - 1, x1_max + 1, 0.02)
...     x2 = np.arange(x2_min - 1, x2_max + 1, 0.02)
...     X1, X2 = np.meshgrid(x1, x2)
...     Z = model.predict(np.c_[X1.ravel(), X2.ravel()])
...     Z = Z.reshape(X1.shape)
...     plt.contourf(X1, X2, Z, cmap=cmap, alpha=0.5)
...     for i, class_ in enumerate(class_label):
...         plt.scatter(x=X[y == class_, 0], y=X[y == class_, 1],
...                 c=cmap.colors[i], label=class_, marker=marker[i])
...     plt.legend()
...
>>> plt.figure(figsize=(18, 10))
<Figure size 1800x1000 with 0 Axes>
>>> weights = ['uniform', 'distance']
>>> ks = [2, 15]
>>> for i, (w, k) in enumerate(product(weights, ks), start=1):
...     plt.subplot(2, 2, i)
...     plt.title(f"Значение K: {k} вес: {w}")
...     knn = KNeighborsClassifier(n_neighbors=k, weights=w)
...     knn.fit(X, y)
...     plot_decision_boundary(knn, X_train, y_train)
...
<matplotlib.axes._subplots.AxesSubplot object at 0x7f8626fd6580>
Text(0.5, 1.0, 'Значение K: 2 вес: uniform')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='uniform')
<matplotlib.axes._subplots.AxesSubplot object at 0x7f862779d970>
Text(0.5, 1.0, 'Значение K: 15 вес: uniform')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=15, p=2,
                     weights='uniform')
<matplotlib.axes._subplots.AxesSubplot object at 0x7f8626f19fd0>
Text(0.5, 1.0, 'Значение K: 2 вес: distance')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='distance')
<matplotlib.axes._subplots.AxesSubplot object at 0x7f8626ed9f70>
Text(0.5, 1.0, 'Значение K: 15 вес: distance')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=15, p=2,
                     weights='distance')
>>> plt.show()
>>>
>>> import numpy as np
>>> import pandas as pd

>>> import scipy
>>> import sklearn
>>> import matplotlib.pyplot as plt
>>> from matplotlib.colors import ListedColormap
```

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import classification_report
>>> from sklearn.neighbors import KNeighborsClassifier
>>> from itertools import product
>>> iris = load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
>>> knn = KNeighborsClassifier(n_neighbors=3, weights="uniform")
>>> knn.fit(X_train, y_train)
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
>>> y_hat = knn.predict(X_test)
>>> classification_report(y_test, y_hat)
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       0.78      0.44      0.56        16
           2       0.44      0.78      0.56         9

    accuracy                           0.71        38
   macro avg       0.74      0.74      0.71        38
weighted avg       0.77      0.71      0.71        38

>>> def plot_decision_boundary(model, X, y):
...     color = ["r", "g", "b"]
...     marker = ["o", "v", "x"]
...     class_label = np.unique(y)
...     cmap = ListedColormap(color[: len(class_label)])
...     x1_min, x2_min = np.min(X, axis=0)
...     x1_max, x2_max = np.max(X, axis=0)
...     x1 = np.arange(x1_min - 1, x1_max + 1, 0.02)
...     x2 = np.arange(x2_min - 1, x2_max + 1, 0.02)
...     X1, X2 = np.meshgrid(x1, x2)
...     Z = model.predict(np.c_[X1.ravel(), X2.ravel()])
...     Z = Z.reshape(X1.shape)
...     plt.contourf(X1, X2, Z, cmap=cmap, alpha=0.5)
...     for i, class_ in enumerate(class_label):
...         plt.scatter(x=X[y == class_, 0], y=X[y == class_, 1],
...                     c=cmap.colors[i], label=class_, marker=marker[i])
...     plt.legend()
...
>>> plt.figure(figsize=(18, 10))
Figure(1800x1000)
>>> weights = ['uniform', 'distance']
>>> ks = [2, 15]
>>> for i, (w, k) in enumerate(product(weights, ks), start=1):
...     plt.subplot(2, 2, i)
...     plt.title(f"Значение K: {k} вес: {w}")
...     knn = KNeighborsClassifier(n_neighbors=k, weights=w)
...     knn.fit(X, y)
```

```
...     plot_decision_boundary(knn, X_train, y_train)
...
<matplotlib.axes._subplots.AxesSubplot object at 0x7f8627235dc0>
Text(0.5, 1.0, 'Значение K: 2 вес: uniform')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='uniform')
<matplotlib.axes._subplots.AxesSubplot object at 0x7f863aee4cd0>
Text(0.5, 1.0, 'Значение K: 15 вес: uniform')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=15, p=2,
                     weights='uniform')
<matplotlib.axes._subplots.AxesSubplot object at 0x7f8628d25640>
Text(0.5, 1.0, 'Значение K: 2 вес: distance')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='distance')
<matplotlib.axes._subplots.AxesSubplot object at 0x7f86271f8100>
Text(0.5, 1.0, 'Значение K: 15 вес: distance')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=15, p=2,
                     weights='distance')
>>> plt.show()
```