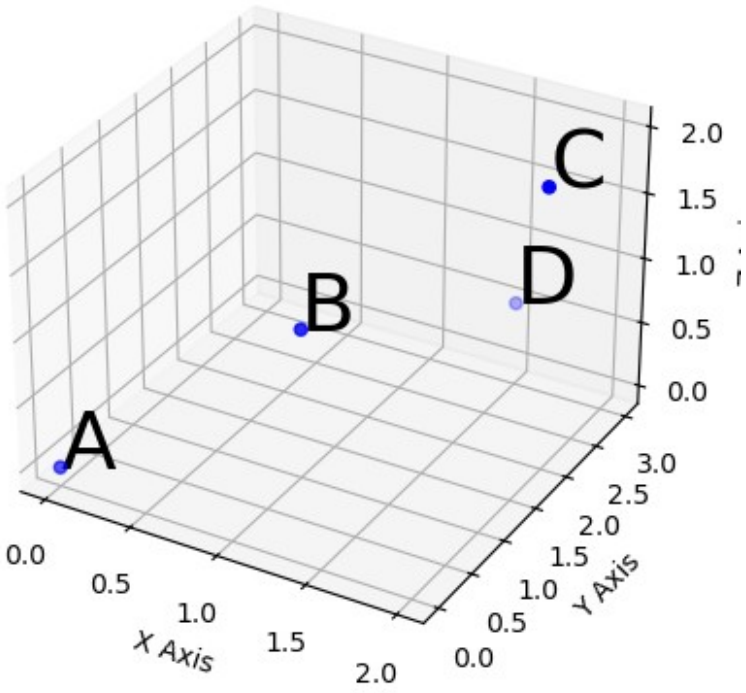
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np


def get_distance(type: str, points: np.array) -> np.array:
    distance = np.zeros((4, 4))
    for i in range(4):
        for j in range(i + 1, 4):
            match type:
                case 'norm':
                    distance[i, j] = np.linalg.norm(points[i] -
points[j])
                case "norm^2":
                    distance[i, j] = np.linalg.norm(
                        (points[i] - points[j]) ** 2)
                case "norm_cheb":
                    distance[i, j] = np.linalg.norm(points[i] -
points[j],
                                                    ord=np.inf)
                case "norm_chem":
                    distance[i, j] = np.linalg.norm(points[i] -
points[j],
                                                    ord=1)
            distance[j, i] = distance[i, j]
    return distance


points = np.array([[0, 0, 0],  # Point A
                   [1, 1, 1],  # Point B
                   [2, 2, 2],  # Point C
                   [1.5, 3, 0.5]])

distances_norm = get_distance("norm", points)
distances_norm_2 = get_distance("norm^2", points)
distances_norm_cheb = get_distance("norm_cheb", points)
distances_norm_chem = get_distance("norm_chem", points)

fig = plt.figure()

ax1 = fig.add_subplot(1,1,1, projection='3d')

ax1.scatter(points[:, 0], points[:, 1], points[:, 2], color='blue')

# Annotate the points for clarity
labels = ['A', 'B', 'C', 'D']
for i, txt in enumerate(labels):
    ax1.text(points[i, 0], points[i, 1], points[i, 2], txt, size=30,
zorder=1)
```

```
# Label the plot
ax1.set_xlabel('X Axis')
ax1.set_ylabel('Y Axis')
ax1.set_zlabel('Z Axis')

plt.show()

# Print the calculated pairwise distances
print(f"Евклидово расстояние между точками:", distances_norm, sep="\
n")
print(f"Квадрат Евклидового расстояния между точками:",
distances_norm_2,
      sep="\n")
print(f"Расстояние Чебышева между точками:", distances_norm_cheb,
sep="\n")
print(f"Расстояние Хемминга между точками:", distances_norm_chem,
sep="\n")
```

```
[0.  1.  2.  1.5]
```




```
Евклидово расстояние между точками:
[[0.         1.73205081 3.46410162 3.39116499]
 [1.73205081 0.         1.73205081 2.12132034]
 [3.46410162 1.73205081 0.         1.87082869]
 [3.39116499 2.12132034 1.87082869 0.        ]]
Квадрат Евклидового расстояния между точками:
```

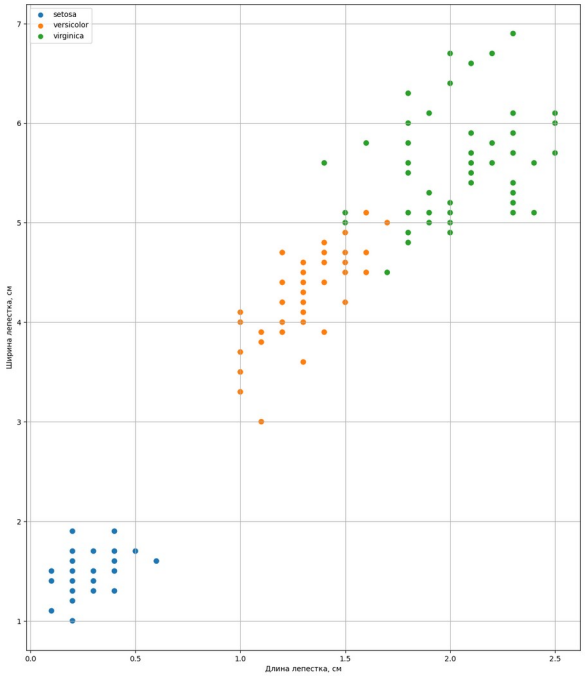
```
[[0.         1.73205081 6.92820323 9.2803556 ]
 [1.73205081 0.         1.73205081 4.0155946 ]
 [6.92820323 1.73205081 0.         2.47487373]
 [9.2803556  4.0155946  2.47487373 0.        ]]
Расстояние Чебышева между точками:
[[0.  1.  2.  3. ]
 [1.  0.  1.  2. ]
 [2.  1.  0.  1.5]
 [3.  2.  1.5 0. ]]
Расстояние Хемминга между точками:
[[0. 3. 6. 5.]
 [3. 0. 3. 3.]
 [6. 3. 0. 3.]
 [5. 3. 3. 0.]]
```

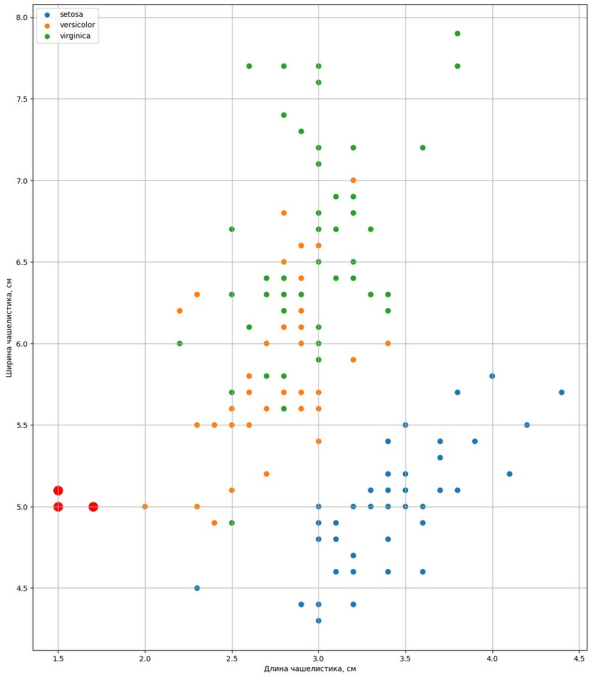
```python
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

iris = sns.load_dataset("iris")

x_train, x_test, y_train, y_test = train_test_split(
    iris.iloc[:, :-1],
    iris.iloc[:, -1],
    test_size=0.15
)

print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
print(x_train.head())
print(y_train.head())


# Обучим метод 1, 5, 10 ближайших соседей
model_1 = KNeighborsClassifier(n_neighbors=1)
model_5 = KNeighborsClassifier(n_neighbors=5)
model_10 = KNeighborsClassifier(n_neighbors=10)
model_1.fit(x_train, y_train)
model_5.fit(x_train, y_train)
model_10.fit(x_train, y_train)

# Получим предсказания модели
y_pred1 = model_1.predict(x_test)
y_pred5 = model_5.predict(x_test)
y_pred10 = model_10.predict(x_test)
print("Предсказания модели 1 ближайшего соседа:", y_pred1, sep="\n")
print("Предсказания модели 5 ближайших соседей:", y_pred5, sep="\n")
print("Предсказания модели 10 ближайших соседей:", y_pred10, sep="\n")
```

```
# Покажем на графике, что отражает полученное число
# Красным цветом обозначены точки, для которых классификация сработала
неправильно

# Объявляем фигуру из двух графиков и ее размера
plt.figure(figsize=(30, 16))

# Левый график
plt.subplot(121)
sns.scatterplot(x='petal_width', y='petal_length', data=iris,
hue='species', s=70)
plt.xlabel("Длина лепестка, см")
plt.ylabel("Ширина лепестка, см")
plt.legend(loc=2)
plt.grid()

# Правый график
plt.subplot(122)
sns.scatterplot(data=iris, x='sepal_width', y='sepal_length',
hue='species', s=70)
plt.xlabel("Длина чашелистика, см")
plt.ylabel("Ширина чашелистика, см")
plt.legend(loc=2)
plt.grid()


for i in range(len(y_test)):
    # Если предсказание неправильное
    if np.array(y_test)[i] != y_pred1[i]:
        # то подсвечиваем точку красным
        plt.scatter(x_test.iloc[i, 3], x_test.iloc[i, 2], color='red',
s=150)

for i in range(len(y_test)):
    # Если предсказание неправильное
    if np.array(y_test)[i] != y_pred5[i]:
        # то подсвечиваем точку красным
        plt.scatter(x_test.iloc[i, 3], x_test.iloc[i, 2], color='red',
s=150)

for i in range(len(y_test)):
    # Если предсказание неправильное
    if np.array(y_test)[i] != y_pred10[i]:
        # то подсвечиваем точку красным
        plt.scatter(x_test.iloc[i, 3], x_test.iloc[i, 2], color='red',
s=150)
```

```
# Качество модели (доля правильно классифицированных точек)
print(f"accuracy: {accuracy_score(y_test, y_pred1):.3}")
```

```
(127, 4) (23, 4) (127,) (23,)
     sepal_length  sepal_width  petal_length  petal_width
70            5.9          3.2           4.8          1.8
35            5.0          3.2           1.2          0.2
148           6.2          3.4           5.4          2.3
88            5.6          3.0           4.1          1.3
51            6.4          3.2           4.5          1.5
70     versicolor
35         setosa
148     virginica
88     versicolor
51     versicolor
Name: species, dtype: object
Предсказания модели 1 ближайшего соседа:
['versicolor' 'virginica' 'virginica' 'setosa' 'setosa' 'setosa'
 'versicolor' 'versicolor' 'setosa' 'versicolor' 'versicolor'
'versicolor'
 'setosa' 'versicolor' 'virginica' 'versicolor' 'virginica'
'virginica'
 'versicolor' 'setosa' 'setosa' 'versicolor' 'setosa']
Предсказания модели 5 ближайших соседей:
['versicolor' 'virginica' 'virginica' 'setosa' 'setosa' 'setosa'
 'virginica' 'versicolor' 'setosa' 'virginica' 'versicolor'
'versicolor'
 'setosa' 'virginica' 'virginica' 'versicolor' 'virginica' 'virginica'
 'versicolor' 'setosa' 'setosa' 'versicolor' 'setosa']
Предсказания модели 10 ближайших соседей:
['versicolor' 'virginica' 'virginica' 'setosa' 'setosa' 'setosa'
 'versicolor' 'versicolor' 'setosa' 'virginica' 'versicolor'
'versicolor'
 'setosa' 'virginica' 'virginica' 'versicolor' 'virginica' 'virginica'
 'versicolor' 'setosa' 'setosa' 'versicolor' 'setosa']
accuracy: 0.913
```






```
from sklearn.feature_extraction import DictVectorizer


a = [
    {"характер": 1, "глаза": 2},
    {"волосы": 1, "цвет кожи": 1},
    {"рост": 170, "вес": 60, "зубы": 32}
]

dict_vectorizer = DictVectorizer(sparse=False)

features = dict_vectorizer.fit_transform(a)
print("Матрица набора признаков человека: ", features, sep="\n")
```

```
Матрица набора признаков человека:
[[  0.   0.   2.   0.   0.   1.   0.]
 [  0.   1.   0.   0.   0.   0.   1.]
 [ 60.   0.   0.  32. 170.   0.   0.]]
```