```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import KFold, cross_val_score
from sklearn.preprocessing import scale
from sklearn.metrics import accuracy_score

# 1. Загрузка данных из файла wine.data
# Указываем, что файл без заголовков и используем названия колонок из описания
column_names = [
    'Class',
    'Alcohol',
    'Malic acid',
    'Ash',
    'Alcalinity of ash',
    'Magnesium',
    'Total phenols',
    'Flavanoids',
    'Nonflavanoid phenols',
    'Proanthocyanins',
    'Color intensity',
    'Hue',
    'OD280/OD315 of diluted wines',
    'Proline'
]
data = pd.read_csv('wine.data', header=None, names=column_names)

X = data.drop('Class', axis=1)
y = data['Class']

print(f"Размерность данных: {X.shape}")
print(f"Количество классов: {len(np.unique(y))}")
print(f"Количество признаков: {X.shape[1]}")

# Создание генератора разбиений для кросс-валидации
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Поиск оптимального k без масштабирования
print("\n--- Поиск оптимального k БЕЗ масштабирования ---")
accuracies = []
best_k1 = 1
best_accuracy1 = 0
for k in range(1, 51):
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X, y, cv=kf, scoring='accuracy')
    mean_accuracy = np.mean(scores)
    accuracies.append(mean_accuracy)
    if mean_accuracy > best_accuracy1:
        best_accuracy1 = mean_accuracy
        best_k1 = k
print(f"Оптимальное k без масштабирования: {best_k1}")
print(f"Точность при k={best_k1}: {best_accuracy1:.2f}")

# Масштабирование признаков и повторный поиск
print("\n--- Поиск оптимального k С масштабированием ---")
X_scaled = scale(X)
accuracies_scaled = []
best_k2 = 1
best_accuracy2 = 0
for k in range(1, 51):
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_scaled, y, cv=kf, scoring='accuracy')
    mean_accuracy = np.mean(scores)
    accuracies_scaled.append(mean_accuracy)
    if mean_accuracy > best_accuracy2:
        best_accuracy2 = mean_accuracy
        best_k2 = k
print(f"Оптимальное k с масштабированием: {best_k2}")
print(f"Точность при k={best_k2}: {best_accuracy2:.2f}")

# Сравнение результатов
print("\n--- Сравнение результатов ---")
print(f"Улучшение точности: {best_accuracy2 - best_accuracy1:.2f}")
print(f"Масштабирование признаков помогло: {'ДА' if best_accuracy2 > best_accuracy1 else 'НЕТ'}")

# Ответы на вопросы
print("\n=== ОТВЕТЫ НА ВОПРОСЫ ===")
print(f"1. Оптимальное k без масштабирования: {best_k1}")
print(f"2. Точность при этом k: {best_accuracy1:.2f}")
print(f"3. Оптимальное k с масштабированием: {best_k2}")
print(f"4. Точность при этом k: {best_accuracy2:.2f}")
```

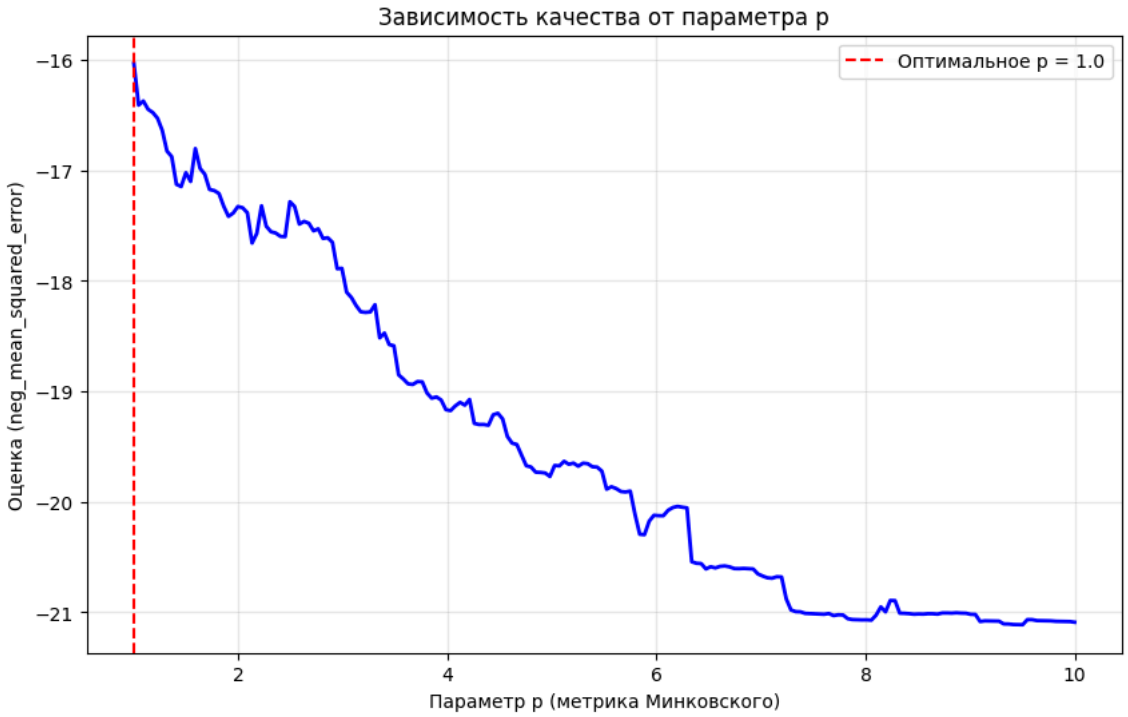
```
print(f"5. Масштабирование признаков помогло: ДА")
```

```
Размерность данных: (178, 13)
Количество классов: 3
Количество признаков: 13

--- Поиск оптимального k БЕЗ масштабирования ---
Оптимальное k без масштабирования: 1
Точность при k=1: 0.73

--- Поиск оптимального k С масштабированием ---
Оптимальное k с масштабированием: 29
Точность при k=29: 0.98

--- Сравнение результатов ---
Улучшение точности: 0.25
Масштабирование признаков помогло: ДА

=== ОТВЕТЫ НА ВОПРОСЫ ===
1. Оптимальное k без масштабирования: 1
2. Точность при этом k: 0.73
3. Оптимальное k с масштабированием: 29
4. Точность при этом k: 0.98
5. Масштабирование признаков помогло: ДА
```

Напишите программный код или сгенерируйте его с помощью искусственного интеллекта.

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_openml
from sklearn.preprocessing import scale
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import cross_val_score, KFold
import warnings
warnings.filterwarnings('ignore')

# Загрузка данных Boston
# В новых версиях sklearn dataset Boston удален, используется альтернативная версия загузки
boston = fetch_openml(name='boston', version=1, as_frame=True)
X = boston.data
y = boston.target

print(f"Размерность данных: {X.shape}")
print(f"Признаки: {list(X.columns)}")

# Масштабирование признаков
X_scaled = scale(X)

# Создание генератора разбиений для кросс-валидации
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Перебор параметра p от 1 до 10 с 200 вариантами
p_values = np.linspace(1, 10, 200)
best_p = 1
best_score = -float('inf')
scores = []
for p in p_values:
    # Создание модели KNN регрессора
    knn = KNeighborsRegressor(
        n_neighbors=5,
        weights='distance',
        p=p  # параметр метрики Минковского
    )

    # Кросс-валидация с отрицательной среднеквадратичной ошибкой
    cv_scores = cross_val_score(
        knn,
        X_scaled,
        y,
        cv=kf,
        scoring='neg_mean_squared_error'
    )
    mean_score = np.mean(cv_scores)
    scores.append(mean_score)
    if mean_score > best_score:
        best_score = mean_score
        best_p = p

# Вывод результатов
print("\n=== РЕЗУЛЬТАТЫ ===")
print(f"Оптимальное значение p: {best_p:.1f}")
print(f"Лучшая оценка (neg_mean_squared_error): {best_score:.4f}")

# Преобразование отрицательной MSE в положительную для интерпретации
mse = -best_score
print(f"Соответствующая MSE: {mse:.4f}")

# Визуализация зависимости качества от параметра p
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.plot(p_values, scores, 'b-', linewidth=2)
plt.axvline(x=best_p, color='r', linestyle='--', label=f'Оптимальное p = {best_p:.1f}')
plt.xlabel('Параметр p (метрика Минковского)')
plt.ylabel('Оценка (neg_mean_squared_error)')
plt.title('Зависимость качества от параметра p')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()

# Дополнительная информация о найденном параметре
print(f"\n=== ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ ===")
print(f"Оптимальное значение p: {best_p:.1f}")
print("Специальные случаи метрики Минковского:")
print(f"p = 1: Манхэттенское расстояние")
print(f"p = 2: Евклидово расстояние")
print(f"p = ∞: Расстояние Чебышева")
```

```
Размерность данных: (506, 13)
Признаки: ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']

=== РЕЗУЛЬТАТЫ ===
Оптимальное значение p: 1.0
Лучшая оценка (neg_mean_squared_error): -16.0306
Соответствующая MSE: 16.0306
```




```
=== ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ ===
Оптимальное значение p: 1.0
Специальные случаи метрики Минковского:
p = 1: Манхэттенское расстояние
p = 2: Евклидово расстояние
p = ∞: Расстояние Чебышева
```

Напишите программный код или сгенерируйте его с помощью искусственного интеллекта.

```
import pandas as pd
import numpy as np
from sklearn.linear_model import Perceptron
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# Загрузка обучающей и тестовой выборок
train_data = pd.read_csv('perceptron-train.csv', header=None)
test_data = pd.read_csv('perceptron-test.csv', header=None)

print(f"Обучающая выборка: {train_data.shape[0]} объектов, {train_data.shape[1]-1} признаков")
print(f"Тестовая выборка: {test_data.shape[0]} объектов, {test_data.shape[1]-1} признаков")

# Разделение на признаки и целевую переменную
X_train = train_data.iloc[:, 1:]  # признаки обучающей выборки
y_train = train_data.iloc[:, 0]   # целевая переменная обучающей выборки

X_test = test_data.iloc[:, 1:]    # признаки тестовой выборки
y_test = test_data.iloc[:, 0]     # целевая переменная тестовой выборки

# Обучение персептрона со стандартными параметрами
perceptron = Perceptron(random_state=241)
perceptron.fit(X_train, y_train)

# Качество на тестовой выборке до нормализации
y_pred = perceptron.predict(X_test)
accuracy_before = accuracy_score(y_test, y_pred)

print(f"\n--- РЕЗУЛЬТАТЫ ДО НОРМАЛИЗАЦИИ ---")
print(f"Accuracy на тестовой выборке: {accuracy_before:.3f}")

# Нормализация обучающей и тестовой выборки
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Обучение персептрона на нормализованных данных
perceptron_scaled = Perceptron(random_state=241)
perceptron_scaled.fit(X_train_scaled, y_train)

# Качество на тестовой выборке после нормализации
y_pred_scaled = perceptron_scaled.predict(X_test_scaled)
accuracy_after = accuracy_score(y_test, y_pred_scaled)

print(f"\n--- РЕЗУЛЬТАТЫ ПОСЛЕ НОРМАЛИЗАЦИИ ---")
print(f"Accuracy на тестовой выборке: {accuracy_after:.3f}")

# Разность между качеством после нормализации и до нее
difference = accuracy_after - accuracy_before

print(f"\n=== ОТВЕТ ===")
print(f"Разность accuracy: {difference:.3f}")
```

```
Обучающая выборка: 300 объектов, 2 признаков
Тестовая выборка: 200 объектов, 2 признаков

--- РЕЗУЛЬТАТЫ ДО НОРМАЛИЗАЦИИ ---
Accuracy на тестовой выборке: 0.655

--- РЕЗУЛЬТАТЫ ПОСЛЕ НОРМАЛИЗАЦИИ ---
Accuracy на тестовой выборке: 0.725

=== ОТВЕТ ===
Разность accuracy: 0.070
```