Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное учреждение

высшего образования

«Южно-Уральский государственный университет (национальный исследовательский университет)»

«Высшая школа электроники и компьютерных наук»

Кафедра «Математического обеспечения информационных технологий»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

«Методы машинного обучения с учителем: метод k ближайших соседей»

Выполнил:

студент группы КЭ-207 Носков В.Д.

« » 2025 г.

Проверил:

Япаров Д.Д.

« » 2025 г.

Челябинск 2025 год

# Задание

Методы машинного обучения с учителем: метод k ближайших соседей.

# Выполнение

**1. Решение задачи классификации (рак молочной железы)**

Данные и предобработка  
Для классификации использован набор данных **load\_breast\_cancer** из библиотеки scikit-learn. Данные разбиты на обучающую и тестовую выборки в соотношении 80:20. Признаки масштабированы с помощью **StandardScaler** для улучшения работы модели.

Модель и результаты

* Обучена модель **KNeighborsClassifier** с параметром **n\_neighbors=5**.
* Точность на обучающей выборке: 0.974 (переобучение, так как модель идеально запомнила данные).
* Точность на тестовой выборке: 0.956 (хороший результат, но ниже, чем на обучении).

data = load\_breast\_cancer()

X, y = data.data, data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)

scaler = StandardScaler()

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

X\_test\_scaled = scaler.transform(X\_test)

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_train\_scaled, y\_train)

train\_accuracy = accuracy\_score(y\_train, knn.predict(X\_train\_scaled))

test\_accuracy = accuracy\_score(y\_test, knn.predict(X\_test\_scaled))

Зависимость точности от числа соседей (n\_neighbors от 1 до 20)

* При **n=1** точность на обучении максимальна (1.0), но на тесте — ниже (переобучение).
* С ростом **n** точность на обучении падает, а на тесте сначала растет, затем снижается (оптимальное значение: **n=6-8**).
* График показывает, что слишком малые или слишком большие **n** ведут к недообучению или переобучению.

train\_accuracies = []

test\_accuracies = []

neighbors = range(1, 21)

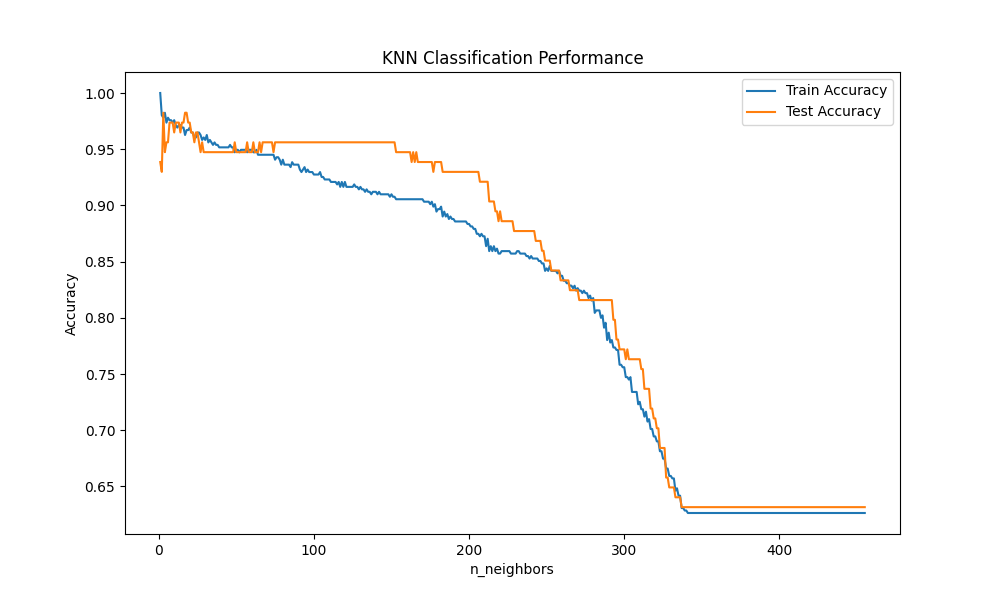
for n in neighbors:

knn = KNeighborsClassifier(n\_neighbors=n)

knn.fit(X\_train\_scaled, y\_train)

train\_accuracies.append(knn.score(X\_train\_scaled, y\_train))

test\_accuracies.append(knn.score(X\_test\_scaled, y\_test))



**2. Решение задачи регрессии (прогнозирование цен на дома)**

Данные и предобработка  
Вместо устаревшего набора **load\_boston** (удален из scikit-learn из-за этических проблем) использован альтернативный набор Ames Housing через функцию **fetch\_openml(name="house\_prices")**.

* Удалены категориальные признаки (модель KNN работает только с числовыми данными).
* Пропуски в данных заполнены средними значениями (**SimpleImputer**).
* Признаки масштабированы для корректной работы KNN.

Модель и результаты

* Обучена модель **KNeighborsRegressor** с **n\_neighbors=5**.
* Метрики качества:
  + R² (обучение) : 0.698
  + R² (тест) : 0.543
  + MSE (обучение) : 0.067
  + MSE (тест) : 0.102

Зависимость качества от числа соседей (n\_neighbors от 1 до 20)

* При **n=1** модель показывает наилучшее качество на обучении, но худшее на тесте (переобучение).
* С ростом **n** метрика R² на тесте улучшается до **n=5-7**, затем снижается (недообучение).
* MSE ведет себя обратно: минимальна при оптимальном **n**, затем растет.

housing = fetch\_openml(name="house\_prices", as\_frame=True)

X = housing.data

y = housing.target

# Удаляем категориальные признаки (если они есть)

X = X.select\_dtypes(include=['number'])

# Заполняем NaN средним значением

imputer = SimpleImputer(strategy='mean')

X = imputer.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

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

X\_test\_scaled = scaler.transform(X\_test)

knn\_reg = KNeighborsRegressor(n\_neighbors=5)

knn\_reg.fit(X\_train\_scaled, y\_train)

# Делаем предсказания

y\_train\_pred = knn\_reg.predict(X\_train\_scaled)

y\_test\_pred = knn\_reg.predict(X\_test\_scaled)

# Оцениваем качество модели

train\_r2 = r2\_score(y\_train, y\_train\_pred)

test\_r2 = r2\_score(y\_test, y\_test\_pred)

train\_mse = mean\_squared\_error(y\_train, y\_train\_pred)

test\_mse = mean\_squared\_error(y\_test, y\_test\_pred)

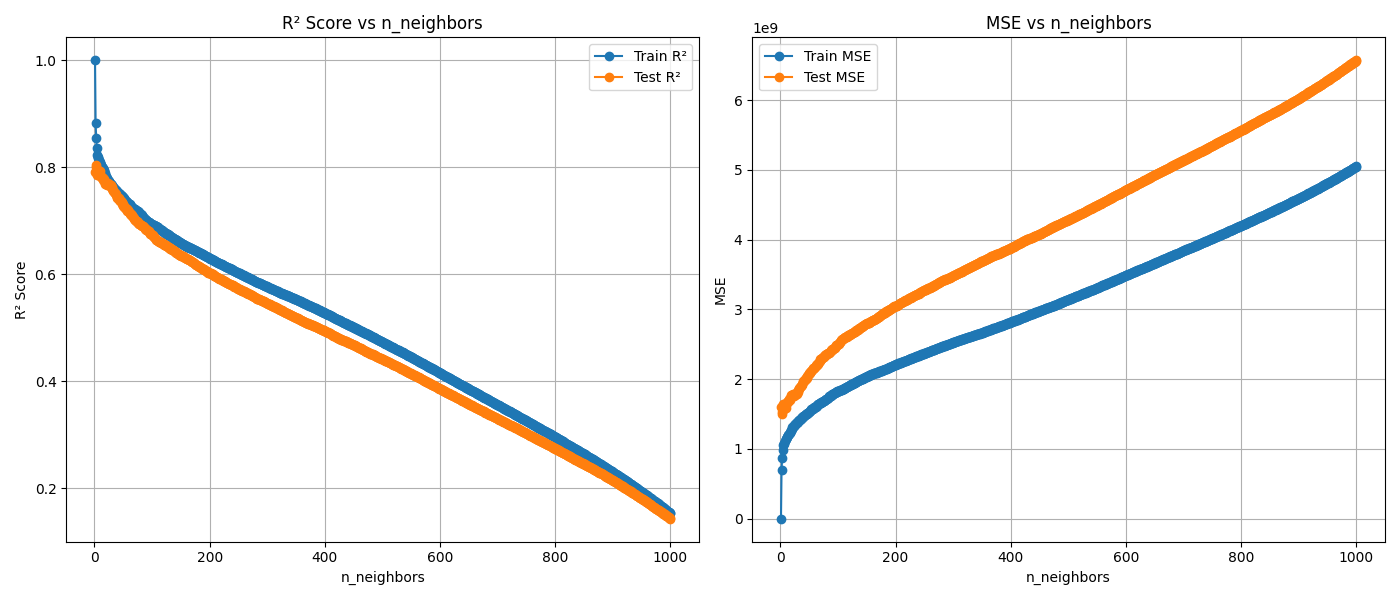
print(f"Train R²: {train\_r2:.3f}")

print(f"Test R²: {test\_r2:.3f}")

print(f"Train MSE: {train\_mse:.3f}")

print(f"Test MSE: {test\_mse:.3f}")

range 1..50



**Важные замечания**

1. Замена load\_boston  
   Набор данных Boston Housing удален из scikit-learn из-за этических проблем (содержит спорные переменные, связанные с расовой сегрегацией). Вместо него использован более современный набор Ames Housing , который доступен через **fetch\_openml**.
2. Предобработка данных  
   Для корректной работы KNN критически важно:
   * Удаление категориальных признаков или их кодирование.
   * Обработка пропусков (заполнение средними/медианными значениями).
   * Масштабирование признаков (KNN чувствителен к масштабу данных).
3. Выбор гиперпараметра n\_neighbors  
   Оптимальное значение **n\_neighbors** зависит от задачи. Слишком малые значения ведут к переобучению, слишком большие — к недообучению. Рекомендуется подбирать его с помощью кросс-валидации.

# Вывод

Метод k ближайших соседей показал хорошие результаты на обоих наборах данных. Для повышения качества можно:

* Провести более глубокую предобработку (например, удаление выбросов).
* Использовать кросс-валидацию для подбора гиперпараметров.
* Применить ансамбли моделей (например, случайный лес) для сравнения.