3. Методы обучения с учителем

Задачи анализа данных

Выявление и классификация характерных признаков

Классификация (обучение с учителем):
- обучающая выборка – описания множества пар {стимул системы; ее ответная реакция};
- цель – модель прогноза реакции системы на любой входной стимул.

Кластеризация и идентификация классов

Кластеризация (обучение без учителя):
- обучающая выборка – описания множества объектов;
- цель - обнаружить внутренние взаимосвязи между объектами, зависимости, закономерности, существующие между ними, разбить на кластеры с одинаковыми свойствами.

классификация (classification)

прогноз метки класса (class label), которая представляет собой выбор из заранее определенного списка возможных вариантов

- бинарная (binary classification): да/нет, positive / negative
- мультиклассовая

регрессия (regression)

прогноз непрерывного (continious) числа с плавающей точкой (floating-point number)

Mglearn

Учебная библиотека Python

```
pip install mglearn
```

Современные методы анализа данных

Байесовский классификатор

- линейный дискриминант Фишера;
- метод парзеновского окна;
- разделение смеси вероятностных распределений, EM(empirical mode) алгоритмы;
- метод потенциальных функций или метод радиальных базисных функций;
- метод ближайших соседей.

Нейронная сеть

- персептрон;
- многослойный персептрон;
- сети векторного квантования, обучаемые с учителем (Learning Vector Quantization);
- гибридная сеть встречного распространения.

Индукция правил

- решающее дерево;
- решающий список;
- решающий лес;
- тестовый алгоритм;
- алгоритм вычисления оценок.

Линейный разделитель

- линейный дискриминант Фишера;
- однослойный персептрон;
- логистическая регрессия;
- машина опорных векторов.

Алгоритмическая композиция

- взвешенное голосование;
- бустинг;
- бэггинг;
- метод комитетов;
- смесь экспертов.

Выбор модели

- минимизация эмпирического риска;
- структурная минимизация риска;
- минимум длины описания;
- скользящий контроль;
- извлечение признаков
- самоорганизация моделей;
- случайный поиск с адаптацией;
- генетический алгоритм.

Сокращение размерности

- селекция признаков;
- метод главных компонент;
- метод независимых компонент;
- многомерное шкалирование.

3. Методы обучения с учителем

Воспользуемся следующими методами машинного обучения:

-Ближайшего соседа

-Линейные модели

-Наивный байесовский классификатор

-Деревья решений

-Случайные леса

-Градиентный бустинг деревьев решений

-Машины опорных векторов

В ходе решения модельных задач:

-оценим качество использования перечисленных методов в практических задачах;

-выявим достоинства и недостатки перечисленных методов;

-сформулируем рекомендации по их использованию.

3. Методы обучения с учителем

Общие правила и термины методов классификации

-Если классификатор может выдавать точные прогнозы на ранее не встречавшихся данных → он обладает способностью **обобщать (generalize)** результат на тестовые данные.

-Если обучающий и тестовый наборы имеют много общего между собой, то модель должна быть точной и на тестовом наборе.

-Проблемы с обучающими наборами:

- **переобучение (overfitting)** – слишком сложные правила (модель) классификации, недостаток информации для обучения (большое количество признаков при малом количестве примеров) → классификатор слишком точно подстраивается под особенности обучающего набора и хорошо работает на обучающем наборе, но не умеет обобщать результат на новые данные;
- **недообучение (underfitting)** – слишком простые правила (модель) классификации → классификатор не охватил все многообразие и изменчивость данных и плохо работает даже на обучающем наборе.

Компромисс между сложностью модели и правильностью на обучающей и тестовой выборках

3. Методы обучения с учителем

Пример недообучения / переобучения

Спрогнозировать покупку клиентом лодки на основе данных о клиентах:

1я модель классификатора:

«*если*

клиент старше 45 лет и у него менее трех детей,

либо

у него трое детей и он женат,

то он скорее всего купит лодку»

дает идеальное обучение на обучающем наборе, но
необязательно сработает на новом клиенте

[мало данных, правило «либо трое и он женат» сформулировано по одному клиенту]

2я модель классификатора: «люди старше 50 лет хотят купить лодку» не охватывает все многообразие и изменчивость данных
[слишком простое правило]

Больший объем данных дает большее разнообразие → большие наборы данных позволяют строить сложные классификаторы. НО простое дублирование одних и тех же данных ИЛИ сбор очень похожих данных не улучшает качество классификации!!

3. Методы обучения с учителем

3.1 Используемые наборы данных

Искусственные наборы forg и wave библиотеки mglearn:

```
#------------набор данных forge----------------------генерируем набор данных
from sklearn.datasets.samples_generator import make_blobs
X, y = mglearn.datasets.make_forge()
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.legend(["Класс 0", "Класс 1"], loc=4)
plt.xlabel("Первый признак")
plt.ylabel("Второй признак")
print("форма массива X: {}".format(X.shape))
```

```
форма массива X: (26, 2)
```

```
#------------набор данных wave----------------------генерируем набор данных
X, y = mglearn.datasets.make_wave(n_samples=40)
plt.plot(X, y, 'o')
plt.ylim(-3, 3)
plt.xlabel("Признак")
plt.ylabel("Целевая переменная (continuous)")
```

```
Text(0, 0.5, 'Целевая переменная (continuous)')
```

3.1 Используемые наборы данных

Экспериментальные наборы типа Bunch: cancer и Boston Housing библиотеки scikit-learn:

```
#объект Bunch - содержит информацию о наборе данных, а также данные
from sklearn.datasets import load_breast_cancer                    # загрузка ф-цией load_breast_cancer
cancer = load_breast_cancer()

from sklearn.datasets import load_boston                           # загрузка ф-цией load_boston
boston = load_boston()
```

Задание1: рассмотреть ключи (keys), формы (shape), признаки (features), описания структур (DESCR) наборов, понять состав наборов, пояснить

```
print("Количество примеров для каждого класса:\n{}".format({n: v for n, v in zip(cancer.target_names, np.bincount(cancer.target))}))
```

Количество примеров для каждого класса:
{'malignant': 212, 'benign': 357}

```
# Набор данных с производными признаками можно загрузить с помощью функции load_extended_boston
X, y = mglearn.datasets.load_extended_boston()
print("форма массива X: {}".format(X.shape))
```

Полученные 104 признака – 13 исходных признаков + 91 производный признак

3.2 Метод k-ближайших соседей: корректный выбор числа n

Задание2: применительно к набору forge использовать метод k-ближайших соседей при n=1, n=3. Сравнить точности. Какая точность получается методом score при n=3? (86%)

Корректный выбор числа n на примере набора cancer

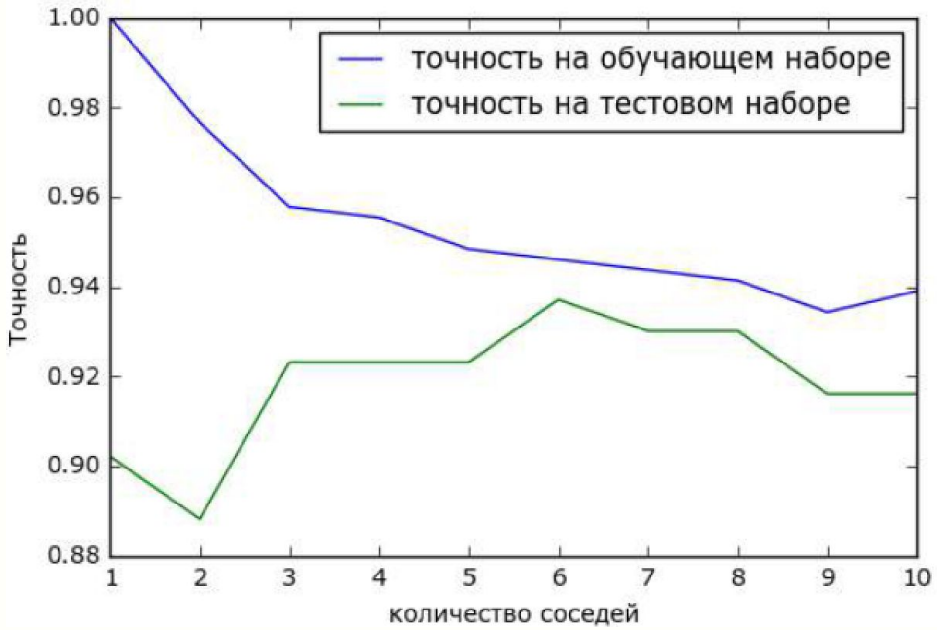
```
# пример корректного выбора числа n - кол-ва соседей
from sklearn.datasets.samples_generator import make_blobs          # надо для работы mglearn

. . . . . .
from sklearn.datasets import load_breast_cancer                    # набор cancer
cancer = load_breast_cancer()
print("Количество примеров для каждого класса:\n{}".format({n: v for n, v in zip(cancer.target_names, np.bincount(cancer.target))}))
```

```
Количество примеров для каждого класса:
{'malignant': 212, 'benign': 357}
```

```
# разбиваем на обуч и тестовый наборы , !!!random_state=66 задать
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, stratify=cancer.target, random_state=66)
training_accuracy = []
test_accuracy = []
neighbors_settings = range(1, 11)          # пробуем n_neighbors от 1 до 10
for n_neighbors in neighbors_settings:
    clf = KNeighborsClassifier(n_neighbors=n_neighbors)     # обучаем классификатор
    clf.fit(X_train, y_train)
    training_accuracy.append(clf.score(X_train, y_train))      # считаем точность обуч. выборки методом score
    test_accuracy.append(clf.score(X_test, y_test))            # записываем точность тестовой выборки
```

3.2 Метод k-ближайших соседей: корректный выбор числа n

```
plt.plot(neighbors_settings, training_accuracy, label="правильность на обучающем наборе")
plt.plot(neighbors_settings, test_accuracy, label="правильность на тестовом наборе")
plt.ylabel("Правильность")
plt.xlabel("количество соседей")
plt.legend()
```



3.3 Регрессия k-ближайших соседей

Используем набор wave и обучающую библиотеку mglearn

```
X, y = mglearn.datasets.make_wave(n_samples=40)        # используем набор wave из обуч. библ-ки
# классификатор линейной регрессии k соседей - ф-ция KNeighborsRegressor.
# используется так же, как KNeighborsClassifier
```

Задание3: Разделить исходный массив данных на обучающие и тестовые выборки. Сколько записей в каждом наборе (вывести на печать формы массивов)? Печать первых пяти пар (x,y) обучающего массива. (random_state=0)

Задание4: Используя KNeighborsRegressor так же, как KNeighborsClassifier создать и обучить классификатор при n=3. Оценить точность классификации тестового набора методом score .

Метод score для KNeighborsRegressor выдает значение коэффициента детерминации!!

3.3 Регрессия k-ближайших соседей

Коэффициент детерминации R^2 - это *доля дисперсии* зависимой переменной, объясняемая рассматриваемой моделью зависимости.

Универсальная мера зависимости одной случайной величины от множества других.

В частном случае линейной зависимости является квадратом так называемого множественного коэффициента корреляции между зависимой переменной и объясняющими переменными. В частности, для модели парной линейной регрессии коэффициент детерминации равен квадрату обычного коэффициента корреляции между y и x.

Множественный коэффициент корреляции характеризует тесноту линейной корреляционной связи между одной случайной величиной и некоторым множеством случайных величин.

Является частным случаем канонического коэффициента корреляции. Так, для двух переменных множественный коэффициент корреляции по абсолютной величине совпадает с коэффициентом парной линейной корреляции.

Коэффициент детерминации R^2 используется как показатель качества регрессионной модели классификатора.

Принимает значения от 0 до 1:

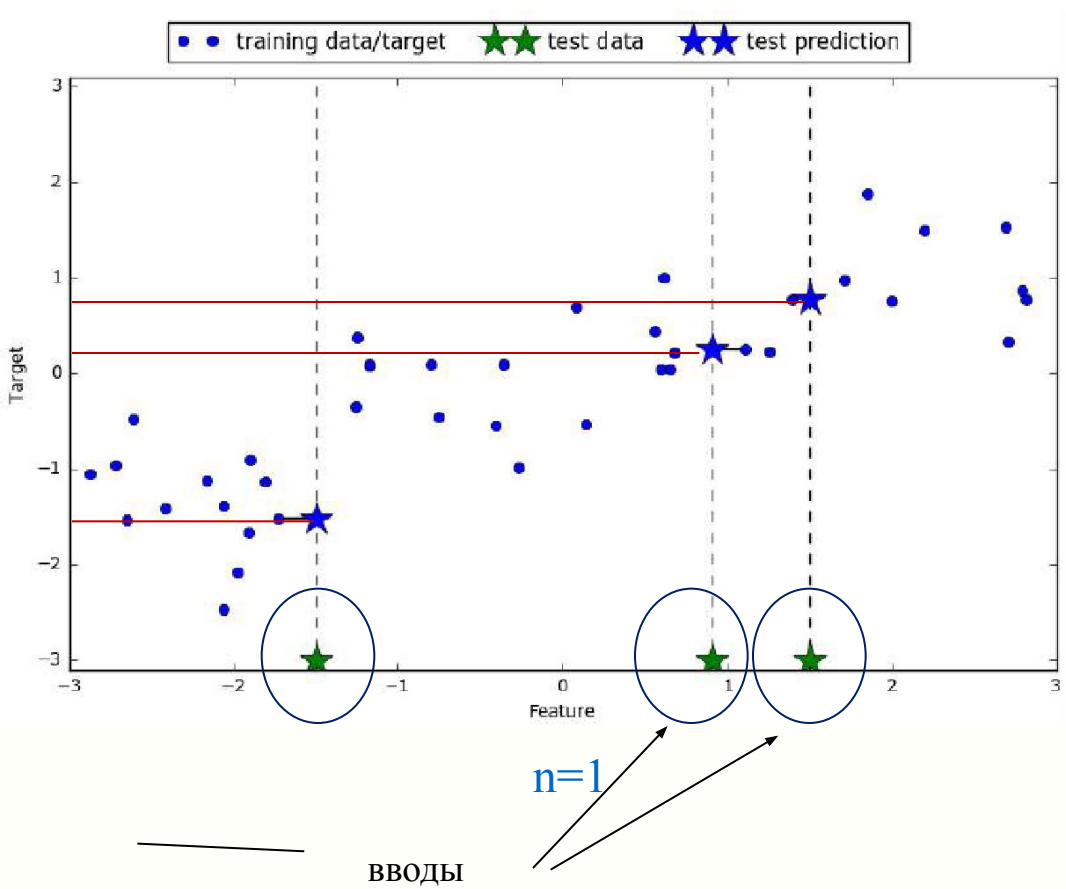
-значение 1 соответствует идеальной прогнозирующей способности классификатора;

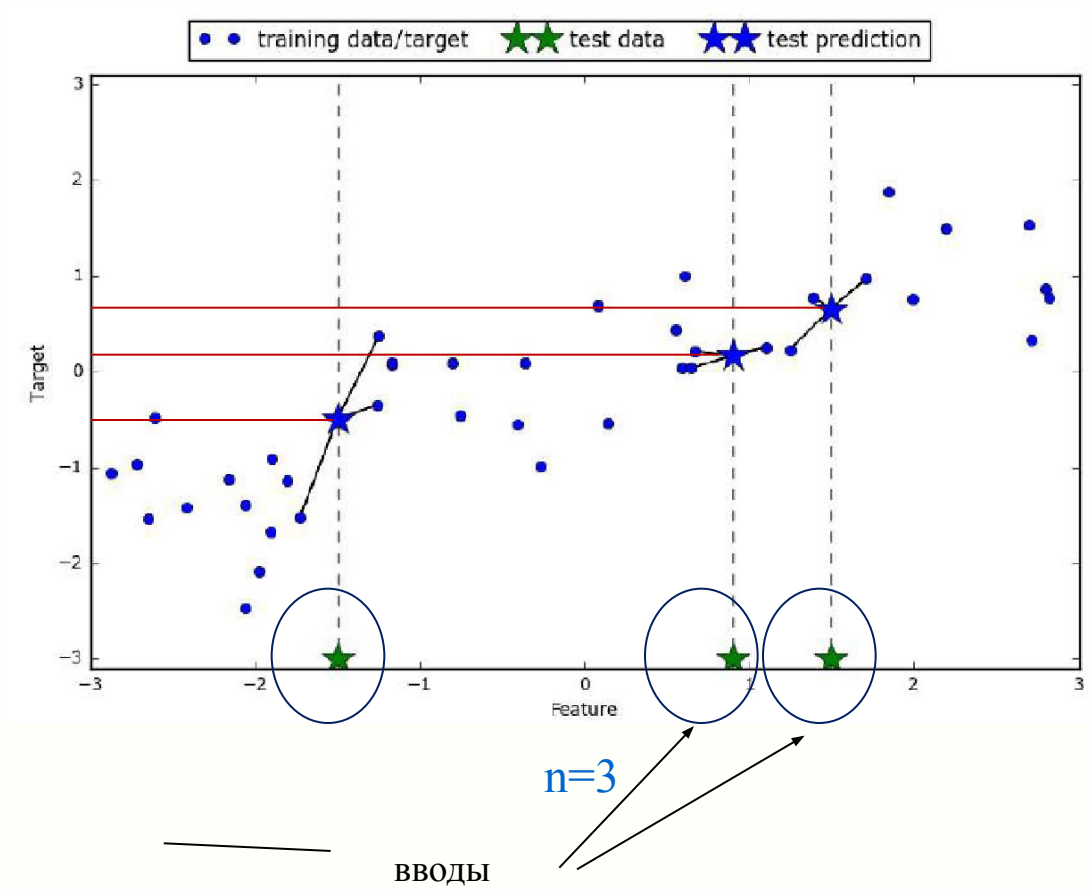
-значение 0 – классификатор предсказывает константу = (среднее значение ответов в обучающем наборе).

3.3 Регрессия k-ближайших соседей

Пояснения точности при разных n с использованием библиотеки mglearn

# пояснение количества соседей с пом. обучающей библиотеки mglearn
mglearn.plots.plot_knn_regression(n_neighbors=1)
mglearn.plots.plot_knn_regression(n_neighbors=3)



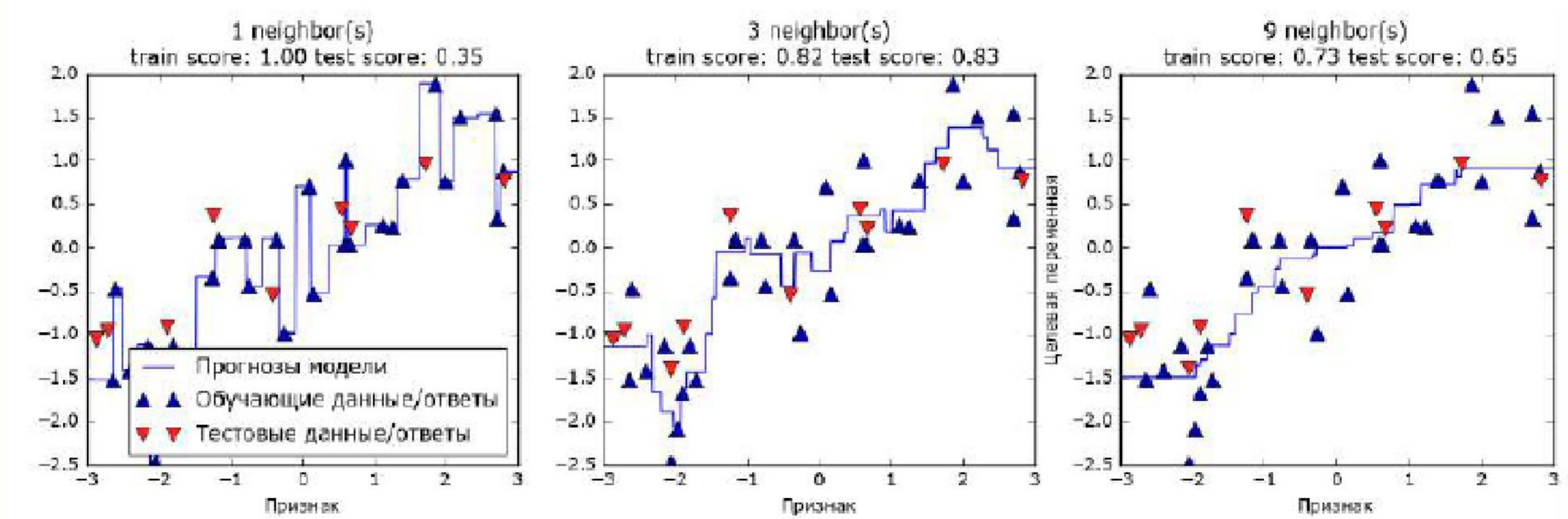


3.3 Регрессия k-ближайших соседей

Анализ точности при разных n на основе расчета коэффициента детерминации

```
# используем набор wave из обуч. библ-ки# исследование качества классификации при n=1,3,9
fig, axes = plt.subplots(1, 3, figsize=(15, 4))
line = np.linspace(-3, 3, 1000).reshape(-1, 1)              # создаем искусственный массив [-3,3] из 1000 точек

# классификация n= 1, 3, и 9 соседей
for n_neighbors, ax in zip([1, 3, 9], axes):
    reg = KNeighborsRegressor(n_neighbors=n_neighbors)
    reg.fit(X_train, y_train)
    ax.plot(line, reg.predict(line))                                  # рисуем линию прогноза
    ax.plot(X_train, y_train, '^', c=mglearn.cm2(0), markersize=8)
    ax.plot(X_test, y_test, 'v', c=mglearn.cm2(1), markersize=8)
    ax.set_title(
        "{} neighbor(s)\n train score: {:.2f} test score: {:.2f}".format(
            n_neighbors, reg.score(X_train, y_train),
            reg.score(X_test, y_test)))
    ax.set_xlabel("Признак")
    ax.set_ylabel("Целевая переменная")
```

3.3 Регрессия k-ближайших соседей



- при n=1 каждая точка обучающего набора имеет очевидное влияние на классификацию ⇒ неустойчивая работа классификатора, прогноз фактически повторяет обучающий набор;
- увеличение числа соседей n приводит к получению более сглаженных прогнозов, но при этом снижается правильность подгонки к обучающим данным

3.4 Выводы по алгоритмам k-ближайших соседей

Наиболее важные параметры классификатора KNeighbors:

-количество соседей n,

-метрика расстояния между точками данных (выбор меры расстояния выходит за рамки курса).

Достоинства:

-простота, наглядность, легкость интерпретации;

-во многих случаях дает приемлемое качество без необходимости использования большого количества настроек;

-является хорошим базовым алгоритмом, который нужно попробовать в первую очередь, прежде чем рассматривать более сложные методы.

Недостатки:

-необходима процедура предварительной обработки данных preprocessing (!);

-сложности в использовании при больших обучающих наборах (когда или количество признаков большое - сотни и более, или большое количество наблюдений samples):

-резкое увеличение вычислительных затрат,

- низкая точность;

-низкая точность при разреженных наборах данных sparse datasets.

ВЫВОД: Несмотря на достоинства, на практике используется не часто из-за низкой скорости вычислений и неспособности обрабатывать большое количество признаков.

3.5 Линейные модели регрессии в задачах классификации непрерывных величин

Это регрессионные модели, в которых прогнозом является разделяющая прямая линия для одного признака, (разделяющая плоскость – для двух признаков, гиперплоскость для большего количества признаков).

Уравнения классификатора для расчета оценки (классификации):

$$y_{est} = wx + b$$ *x* - один признак;

$$y_{est} = w_1 x_1 + w_2 x_2 + b$$ - два признака;

……

$$y_{est} = w_1 x_1 + w_2 x_2 + \boxtimes + w_n x_n + b$$ - *n* признаков;

Существует различные виды линейных моделей для регрессии. Различие между этими моделями заключается в способе оценивания параметров модели по обучающим данным и контроле сложности модели.

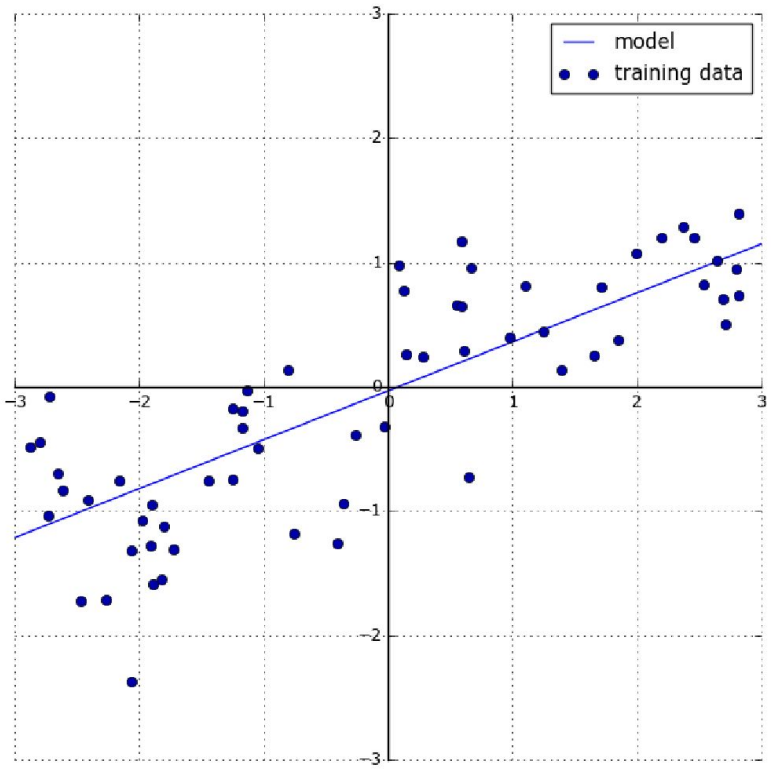
Рассмотрим наиболее используемые линейные модели классификаторов.

3.5.1 Линейная модель МНК (ordinary least squares, OLS)

Модели МНК - самые простые. Для набора wave воспользоваться уч. библиотекой и построить график регрессионной модели

```
# уч. библ.
mglearn.plots.plot_linear_regression_wave()
```

w[0]: 0.393906 b: -0.031804



Задание5: Проверить правильность параметров модели w, b

Задание6: Сравнить с графиками классификации KNeighborsRegressor

Использование линии для классификации кажется очень строгим - мелкие детали данных не учитываются, используется сильное (и нереальное) предположение, что целевая переменная y является линейной комбинацией признаков.

НО для наборов данных с большим количеством признаков линейные модели могут быть достаточно точными. Например, когда количество признаков превышает количество точек данных для обучения.

3.5.1 Линейная модель МНК (ordinary least squares, OLS)

Задание6: Разделить исходный массив данных wave (n_samples=60) на обучающие и тестовые выборки (random_state=42).
Сколько записей в каждом наборе (вывести на печать формы массивов)?
Печать первых пяти пар (x,y) обучающего массива.
Используя модель LinearRegression из модуля sklearn.linear_model создать и обучить линейный классификатор (пусть name = lr).

Параметры «наклона» w (веса, или коэффициенты coefficients), хранятся в атрибуте coef_, сдвиг b (offset, или константа intercept), хранится в атрибуте intercept_:

```
print("lr.coef_: {}".format(lr.coef_))
print("lr.intercept_: {}".format(lr.intercept_))   # сравнить рез-ты с примером учебным (выше)
```

```
lr.coef_: [0.39390555]
lr.intercept_: -0.031804343026759746
```

атрибут intercept_ - отдельное число с плавающей точкой;
атрибут coef_ - массив NumPy, в котором каждому элементу соответствует входной признак.
Т.к. в наборе данных wave используется только один входной признак, lr.coef_ содержит только один элемент.

Подчеркивание _ ⇒ «атрибут». Чтобы не путать с пользовательскими параметрами.

3.5.1 Линейная модель МНК (ordinary least squares, OLS)

Задание7: Оценить точность классификатора на обучающем и на тестовом наборах методом score. (Выводится значение коэффициента детерминации)

Не очень высокая точность, но результаты на тестовом и обучающем наборах схожи ⇒ м.б. недообучение

Для высокоразмерных наборов данных (наборов данных с большим количеством признаков) линейные модели становятся сложными и существует высокая вероятность переобучения. Для набора Boston Housing:

Задание8: Загрузить набор данных с производными признаками с помощью функции load_extended_boston из учебной библиотеки mglearn.
Какая размерность данных?
Сколько производных признаков?
Разбить на обучающий и тестовый наборы. (random_state=0)
Какие размеры массивов?
Используя модель LinearRegression из модуля sklearn.linear_model создать и обучить линейный классификатор.
Оценить точность классификатора на обучающем и на тестовом наборах методом score. (Выводится значение коэффициента детерминации)

Правильность на обучающем наборе: 0.95
Правильность на тестовом наборе: 0.61

Несоответствие между правильностью на обучающем наборе и правильностью на тестовом наборе является признаком **переобучения** ⇒ использовать другой классификатор для этого набора.

3.5.2 Линейная модель «гребневая регрессия» (Ridge)

Гребневая регрессия = линейная модель регрессии, формула та же.

Отличия гребневой регрессии от МНК-регрессии:

коэффициенты w_1, w_1, … выбираются согласно МНК, но далее корректируются так, чтобы каждый w_i был близок к 0 (i=1,2,…).

Это означает, что каждый признак должен иметь как можно меньшее влияние на результат, но, в то же время, этот признак должен по-прежнему отражать особенности класса.

Такая коррекция является примером регуляризации (regularization) L_2. (L_2-норма – евклидово расстояние)

Здесь регуляризация используется для предотвращения переобучения:

переобучение в большинстве случаев проявляется в том, что в получающихся многочленах слишком большие коэффициенты w_i.

Соответственно, в целевую функцию добавляют «штраф», который «наказывает» модель за слишком большие коэффициенты.

Целевая функция «МНК + регуляризация L_2»:

$$J = \sum_i \left(y_i - y_{esti}\right)^2 + \alpha \sum_i w_i^2$$

3.5.2 Линейная модель «гребневая регрессия» (Ridge)

Задание9: для набора данных Boston с производными признаками, используя модель Ridge из модуля sklearn.linear_model создать и обучить линейный классификатор. (random_state=0)
Оценить точность классификатора на обучающем и на тестовом наборах методом score. (Выводится значение коэффициента детерминации)
Сравнить результаты с точностью модели МНК-регрессии LinearRegression

На обучающем наборе модель Ridge дала меньшую правильность, чем модель LinearRegression.
На тестовом наборе – наоборот, выше.

ВЫВОДS:

-Высокая точность линейной МНК-регрессии на обучающем наборе объясняется ее переобучением.

-Переобучение Ridge меньше. Это можно объяснить введением дополнительных ограничений L_2.

-Точность на тестовом наборе выше ⇒ модель Ridge имеет выше, чем модель МНК, *обобщающую способность*.

Для решения задачи надо выбирать модель классификатора Ridge (в сравнении с LinearRegression).

3.5.2 Линейная модель «гребневая регрессия» (Ridge)

Варьируя коэффициент регуляризации α (параметр alpha), можно регулировать «штраф» ⇒ изменять качество на обучающем наборе:

-при α=0 ⇒ МНК;

-можно задавать α>>1.

В решаемом примере: значение параметра по умолчанию alpha=1.0.

Резюме:

-оптимальное значение alpha зависит от конкретного используемого набора данных;

-увеличение alpha снижает качество работы модели на обучающем наборе, но может улучшить обобщающую способность.

```
# альфа = 10
ridge01 = Ridge(alpha=0.1).fit(X_train, y_train)
print("Правильность на обучающем наборе должна вырасти (была 0.89): {:.2f}".format(ridge01.score(X_train, y_train)))
print("Правильность на тестовом наборе должна остаться (была 0.75): {:.2f}".format(ridge01.score(X_test, y_test)))
```

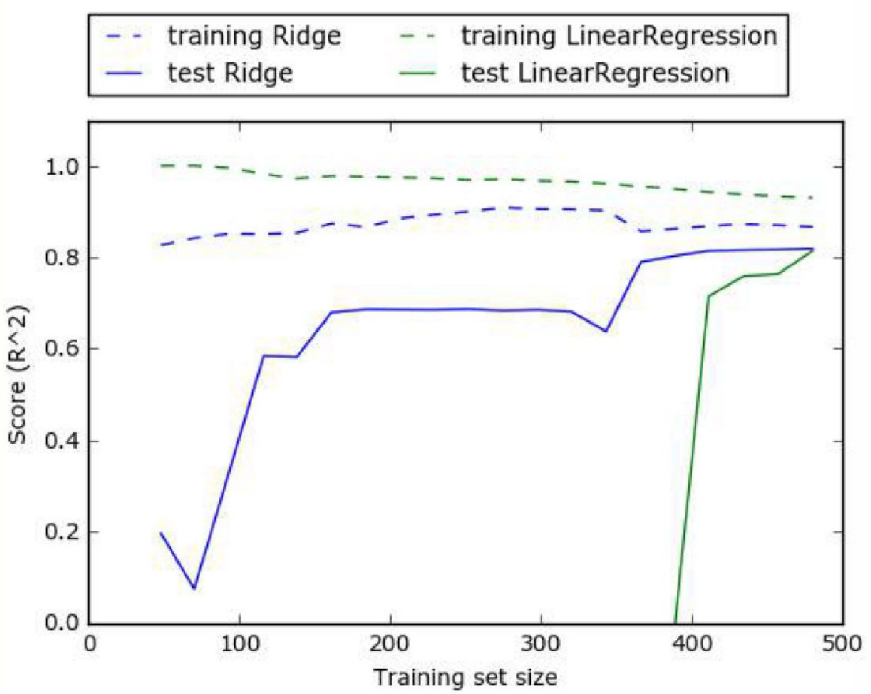
```
Правильность на обучающем наборе должна вырасти (была 0.89): 0.93
Правильность на тестовом наборе должна остаться (была 0.75): 0.77
```

3.5.2 Линейная модель «гребневая регрессия» (Ridge)

Анализ влияния объема выборки данных Boston на точности моделей LinearRegression и Ridge(alpha=1)

Воспользовались обучающей библиотекой mglearn:

```
# Анализ влияния объема выборки на модели LinearRegression и Ridge(alpha=1)
mglearn.plots.plot_ridge_n_samples()
```



На обучающем наборе точности всегда выше в сравнении с тестовым набором.

На тестовом наборе точность Ridge ниже точности МНК.

На обучающем наборе точность Ridge выше, особенно для небольших наборов данных.

При объеме данных менее 400 наблюдений для этого количества признкаов (104) МНК-модель не способна обучится.

При больших объемах – почти одинаковые точности у моделей.

Следовательно, при достаточном объеме обучающих данных регуляризация становится менее важной и при удовлетворительном объеме данных гребневая и линейная регрессии демонстрируют одинаковое качество работы.

3.5.3 Линейная модель «лассо» (Lasso)

Лассо регрессия ≈ гребневая модель регрессии:

коэффициенты w_1, w_1, … выбираются согласно МНК и корректируются так, чтобы каждый w_i был близок к 0 (i=1,2,…).

Но корректируются посредством регуляризации L_1. (L_1-норма – манхэттенское расстояние)

Целевая функция «МНК + регуляризация L_1»:

$$J = \sum_i \left(y_i - y_{est_i}\right)^2 + \alpha \sum_i |w_i|$$

Результат L_1 регуляризации:

некоторые коэффициенты становятся =0 и полностью исключаются из модели.

Это можно рассматривать как один из видов автоматического отбора признаков. Получение нулевых значений для некоторых коэффициентов часто упрощает интерпретацию модели и может выявить наиболее важные признаки модели.

3.5.3 Линейная модель «лассо» (Lasso)

Задание10: для набора данных Boston с производными признаками, используя модель Lasso из модуля sklearn.linear_model создать и обучить линейный классификатор. (random_state=0)
Оценить точность классификатора (по умолчанию α=1) на обучающем и на тестовом наборах методом score. (Выводится значение коэффициента детерминации)
Сравнить результаты с точностью модели гребневой регрессии

Вывести количество использованных признаков:

```
print("Количество использованных признаков: {}".format(np.sum(lasso.coef_ != 0)))
```

```
Количество использованных признаков: 4
```

Низкая точность как на обучающем, так и на тестовом наборе указывает на недообучение – из 105 признаков используются только 4.

Чтобы снизить недообучение, надо уменьшить alpha. При этом нужно увеличить значение max_iter (максимальное количество итераций)

3.5.3 Линейная модель «лассо» (Lasso)

```
# уменьшаем альфа, одновремено увеличиваем max_iter (иначе выдаст предупреждение)
lasso001 = Lasso(alpha=0.01, max_iter=100000).fit(X_train, y_train)
print("Правильность на обучающем наборе: {:.2f}".format(lasso001.score(X_train, y_train)))
print("Правильность на тестовом наборе: {:.2f}".format(lasso001.score(X_test, y_test)))
print("Количество использованных признаков: {}".format(np.sum(lasso001.coef_ != 0)))
```

```
Правильность на обучающем наборе: 0.90
Правильность на тестовом наборе: 0.77
Количество использованных признаков: 33
```

Если установить слишком низкое значение alpha ⇒ = МНК, нет регуляризации, переобучение:

```
# уменьшаем альфа, одновремено увеличиваем max_iter (иначе выдаст предупреждение)
lasso001 = Lasso(alpha=0.01, max_iter=100000).fit(X_train, y_train)
print("Правильность на обучающем наборе: {:.2f}".format(lasso001.score(X_train, y_train)))
print("Правильность на тестовом наборе: {:.2f}".format(lasso001.score(X_test, y_test)))
print("Количество использованных признаков: {}".format(np.sum(lasso001.coef_ != 0)))
```

```
Правильность на обучающем наборе: 0.90
Правильность на тестовом наборе: 0.77
Количество использованных признаков: 33
```

3.5.4 Выводы по линейным моделям регрессии в задачах классификации непрерывных величин

- Когда стоит выбор между Ridge и Lasso, предпочтение, как правило, отдается Ridge.
- НО, если количество признаков большое и есть основания считать, что не все из них информативны, Lasso может быть лучше.
- В библиотеке scikit-learn имеется класс ElasticNet, который сочетает в себе штрафы Ridge и Lasso.
- Часто используют комбинацию Ridge и Lasso. При этом качество достигается за счет настройки параметров α: отдельно для L_1 регуляризации, отдельно – для L_2 регуляризации.

3.6 Линейные модели регрессии в задачах дискретной классификации