Дополнительные пояснения (кратко)

Класс `LinearRegression` библиотеки Sklearn предназначен для построения моделей линейной регрессии.

Этапы построения модели:

1. Создание экземпляра класса:

```
model_name = LinearRegression()
```

2. Обучение модели с помощью метода `fit`:

```
model_name.fit(X, y)
```

где `X` – значения признаков для объектов,

`y` – значения целевой переменной для объектов.

3. Если необходимо сделать предсказание целевых значений для *новых объектов (X_new – их признаковое описание)*, то используется метод `predict`:

```
model_name.predict(X_new)
```

Новые объекты – это те объекты, которые не участвовали в обучении. Представим, что перед нами стоит следующая задача: есть набор данных, который содержит признаковое описание подержанных автомобилей (марка, год выпуска, пробег и др.) и их цену (целевая переменная). Необходимо построить модель, которая будет по набору признаков определять цену автомобиля. Тот набор данных, который мы используем для обучения модели, называется *обучающим* (тренировочным). В дальнейшем эта модель будет предсказывать цену для других автомобилей, информация о которых не содержалась в обучающем наборе данных. Такие автомобили и называются новыми объектами.

4. Качество полученной модели нужно каким-то образом оценивать, чтобы понять насколько хорошо она решает поставленную задачу. Существуют разные метрики оценки качества в задачах регрессии. Некоторые из них представлены здесь. В классе `LinearRegression` (как и в любом другом классе для построения моделей обучения с учителем) есть метод `score`. Метод `score` позволяет оценить качество модели с помощью коэффициента детерминации. Чем ближе его значение к 1, тем лучше модель.

```
model_name.score(X, y)
```

где `X` – признаки, `y` – истинные значения целевой переменной.

Если мы хотим использовать какую-то другую метрику, то нужно применить соответствующую функцию. К примеру, мы хотим вычислить MSE:

```
from sklearn.metrics import mean_squared_error

y_pred = model_name.predict(X) # предсказываем значения целевой переменной для X

mean_squared_error(y, y_pred) # Вычисляем MSE
```

Обратите внимание, что в функцию `mean_squared_error` мы передаем первым аргументом истинные значения `y`, а вторым предсказанные моделью значения целевой переменной для `X` (*об этом написано в документации, в которую обязательно нужно всегда смотреть!*). Передавать в функцию `X` и `y` не нужно. Это бессмысленно и приведет к ошибке.

`mean_squared_error` – это отдельная функция, т.е. к модели никакого отношения она не имеет (и сама значения целевой переменной `y` для `X` не предскажет, как это делает метод `score`). Мы передаем в нее истинные значения целевой переменной и предсказанные, а она просто подставляет их в формулу (ее можно найти здесь на слайде 4).

Таким образом, мы можем оценить качество модели на **обучающей выборке**, ведь истинные значения целевой переменной для новых данных нам неизвестны. Но *оценка на обучающей выборке не объективна*, ведь модель уже «знает» эти данные, и в большинстве случаев будет на них работать лучше, чем на новых.

А как проверить хорошо ли будет работать модель на новых данных?

Эти подходы используются и для регрессии, и для классификации:

1 подход. Разделить исходную выборку, предназначенную для обучения (т.е. для которой известны `y`) на 2 части – обучающую и тестовую (обычно делят 70/30 или 80/20). Т.о. мы от исходной выборки отрезаем небольшой кусочек для тестирования (с известными `y`), чтобы **сравнить** предсказанные моделью значения целевой переменной с истинными значениями `y`.

Для того, чтобы осуществить такое разбиение используется функция `train_test_split`:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

- `X_train`, `y_train` – для обучения.
- `X_test`, `y_test` – для тестирования.
- `y_pred = model_name.predict(X_test)` – то, что предсказала модель.
- `y_test` – истинные значения `y`.

Для MSE:

```
y_pred = model_name.predict(X_test) # предсказываем значения целевой переменной для X_test

mean_squared_error(y_test, y_pred) # Вычисляем MSE
```

Если нужно использовать метод `score`:

```
model_name.score(X_test, y_test)
```

2 подход. Перекрестная проверка. Метод заключается в разделении исходного набора данных на *k* примерно равных блоков. Затем на *k*–1 блоках производится обучение модели, а оставшийся блок используется для тестирования. Процедура повторяется *k* раз, при этом на каждом проходе для проверки выбирается новый блок, а обучение производится на оставшихся.

```
from sklearn.model_selection import cross_val_score

model_name = LinearRegression()

cross = cross_val_score(model_name, X, y, cv=4)
```

- `X` – признаковое описание объектов (из исходного набора данных).
- `y` – значения целевой переменной для объектов.
- `cv` – количество блоков.