Рубежный контроль 1

по курсу «Методы машинного обучения»

Вопросы:

Оглавление

[1 Линейная регрессия. Назначение, формула, функция потерь, способы обучения (без вывода) 1](#_Toc138189987)

[2 Оценка параметров линейной регрессии посредством градиентного спуска 4](#_Toc138189988)

[3 Ошибка обучения, ошибка тестирования, переобучение. Привести пример в виде графика с кривыми ошибок 5](#_Toc138189989)

[4 Полиномиальная регрессия. Назначение, формула, функция потерь, способы обучения (без вывода) 7](#_Toc138189990)

[5 Логистическая регрессия. Назначение, формула, функция потерь, способы обучения (без вывода) 8](#_Toc138189991)

[6 Регуляризация. Назначение, виды, пример для линейной и логистической регрессии. В чем проявилась регуляризация в дз2 9](#_Toc138189992)

[7 Выбор модели. Отложенная выборка и кросс-валидация 11](#_Toc138189993)

[8 Каким образом определялась наилучшая модель в полиномиальной регрессии в дз2. Привести схему разбиения датасета и построения графиков ошибок 12](#_Toc138189994)

[9 Какие метрики использовались для оценки качества моделей в дз2, на каких этапах они вычислялись и для чего 14](#_Toc138189995)

# 1 Линейная регрессия. Назначение, формула, функция потерь, способы обучения (без вывода)

Линейная регрессия — это один из методов машинного обучения, который используется для моделирования отношения между зависимой переменной (выходными данными) и одной или более независимых переменных (входными данными). Она пытается найти линейную зависимость между входными и выходными данными, чтобы предсказать значения выходных данных для новых входных данных.

**Линейная регрессия** — это метод, который помогает предсказывать значения одной переменной на основе других переменных. Он ищет линейную зависимость между входными и выходными данными.

Формула линейной регрессии:

y = β₀ + β₁x₁ + β₂x₂ + ... + βₚxₚ

Y - зависимая переменная (выходные данные);

X1, X2, ..., Xn - независимые переменные (входные данные);

b0, b1, b2, ..., bn - коэффициенты регрессии, которые нужно найти.

Для общего случая, где h(x) представляет собой гипотезу или модель, включающую различные функции и нелинейные зависимости, формула может выглядеть следующим образом:

h(x) = β₀f₀(x) + β₁f₁(x) + β₂f₂(x) + ... + βₚfₚ(x)

В этой формуле:

* h(x) представляет собой прогнозируемое значение или вывод модели для входного значения x.
* f₀(x), f₁(x), f₂(x), ..., fₚ(x) представляют собой различные функции (линейные или нелинейные) от входного значения x.
* β₀, β₁, β₂, ..., βₚ представляют собой коэффициенты, также известные как веса, которые умножаются на соответствующие функции f₀(x), f₁(x), f₂(x), ..., fₚ(x).

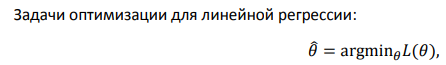
Функция потерь — это функция, которая оценивает, насколько хорошо модель соответствует данным. В линейной регрессии используется функция среднеквадратичной ошибки (Mean Squared Error, MSE), которая измеряет среднее значение квадратов отклонений между предсказанными значениями и фактическими значениями.

**Функция потерь** в линейной регрессии — это способ оценить, насколько хорошо модель соответствует данным. Чем меньше значение функции потерь, тем лучше модель.

Формула функции потерь выглядит следующим образом:

L(θ) = (1/n) \* Σ(yᵢ - xᵢᵀθ)²

* yᵢ — это фактическое значение зависимой переменной для каждого наблюдения i.
* xᵢ — это вектор входных переменных (значений признаков) для каждого наблюдения i. Этот вектор содержит значения различных переменных, которые используются для предсказания зависимой переменной.
* xᵢᵀ — это транспонированная строка вектора xᵢ. Транспонирование позволяет умножить вектор на параметры θ.
* θ — это вектор параметров модели, который умножается на транспонированную строку xᵢᵀ. Эти параметры определяют, как каждая переменная влияет на предсказываемую переменную.



В данном случае, мы ищем значения параметров 𝜃, при которых функция потерь L(𝜃) достигает своего минимального значения. Функция argmin выбирает аргумент (в данном случае, значения параметров 𝜃), при котором функция L(𝜃) принимает минимальное значение.

𝜃̂ = argmin𝜃L(𝜃) используется для общего представления задачи оптимизации. Она означает, что 𝜃̂ выбирается как аргумент, при котором функция потерь L(𝜃) достигает минимального значения. Здесь функция потерь L(𝜃) может быть любой функцией, включая квадратичную функцию потерь, среднеквадратичную ошибку (MSE) или другую функцию, специфичную для задачи

Способы обучения в линейной регрессии:

Метод наименьших квадратов (Ordinary Least Squares, OLS) — это классический метод, который находит коэффициенты регрессии, минимизируя сумму квадратов отклонений предсказанных значений от фактических значений;

Формула 𝜃̂ = (XᵀX)⁻¹Xᵀy используется в линейной регрессии с использованием метода наименьших квадратов (МНК) для оценки параметров модели. Она позволяет найти оптимальные значения параметров 𝜃, минимизируя сумму квадратов остатков между фактическими значениями y и предсказанными значениями, которые зависят от матрицы X и параметров 𝜃.

Градиентный спуск (Gradient Descent) — это метод, который ищет минимум функции потерь, используя итеративный процесс обновления коэффициентов регрессии. Он может быть использован для решения как задачи линейной, так и нелинейной регрессии.

В начале градиентного спуска необходимо выбрать некоторые начальные значения для коэффициентов регрессии. Затем процесс обновления выполняется в несколько итераций, пока не достигнут достаточно хорошие значения коэффициентов.

Ключевой шаг в градиентном спуске — это вычисление градиента функции потерь по отношению к коэффициентам регрессии. Градиент представляет собой вектор, который указывает направление наиболее быстрого возрастания функции. В случае минимизации функции потерь, нам нужно двигаться в направлении, противоположном градиенту, чтобы приближаться к минимуму.

На каждой итерации градиентного спуска происходит обновление коэффициентов регрессии путем вычитания некоторого значения, называемого скоростью обучения (learning rate), умноженного на градиент. Скорость обучения определяет размер шага, который мы делаем на каждой итерации. Она должна быть выбрана соответствующим образом, чтобы не пропустить минимум функции или сойтись к нему слишком медленно.

Процесс обновления коэффициентов и вычисления градиента повторяется до достижения условия остановки, например, когда достигнуто определенное количество итераций или изменение функции потерь становится достаточно маленьким.

𝜃 := 𝜃 - 𝛼 \* ∇L(𝜃)

Где:

* 𝜃 - вектор параметров модели.
* 𝛼 - скорость обучения (learning rate), это гиперпараметр, определяющий шаг обновления параметров. Он должен быть выбран заранее и обычно настраивается в процессе обучения.
* ∇L(𝜃) - градиент функции потерь L(𝜃) по параметрам 𝜃. Градиент представляет собой вектор, состоящий из частных производных функции потерь по каждому параметру. Градиент представляет собой вектор, состоящий из частных производных функции потерь по каждому параметру модели. Он показывает направление наискорейшего возрастания функции потерь и позволяет определить, как изменение каждого параметра влияет на функцию потерь.

# 2 Оценка параметров линейной регрессии посредством градиентного спуска

Оценка параметров линейной регрессии посредством градиентного спуска — это один из способов обучения линейной регрессии. Он используется для поиска оптимальных значений коэффициентов регрессии, которые минимизируют функцию потерь.

способ найти наилучшие значения коэффициентов, которые связывают входные данные с выходными данными

Градиентный спуск — это оптимизационный алгоритм, который ищет минимум функции путем итеративного изменения параметров с использованием производных функции. Он основан на идее, что если мы движемся в направлении антиградиента функции, то мы можем достичь ее минимума. В контексте линейной регрессии градиентный спуск используется для поиска оптимальных значений коэффициентов регрессии, которые минимизируют функцию потерь.

алгоритм, который помогает нам найти эти значения, двигаясь в направлении, которое уменьшает ошибку предсказаний.

Для использования градиентного спуска для оценки параметров линейной регрессии необходимо выполнить следующие шаги:

1. Инициализировать начальные значения коэффициентов регрессии.
2. Определить функцию потерь, которая будет минимизироваться.
3. Вычислить градиент функции потерь по каждому параметру.
4. Обновить значения параметров, используя градиент и скорость обучения (learning rate).
5. Повторять шаги 3-4 до тех пор, пока не будет достигнуто условие остановки (например, достижение определенного числа итераций или достижение определенного порога точности).

Одним из преимуществ градиентного спуска является его способность работать с большими объемами данных, а также с функциями потерь, которые не являются выпуклыми. Однако градиентный спуск может привести к проблеме "застревания" в локальных минимумах, если функция потерь не является выпуклой. Также градиентный спуск требует настройки гиперпараметров, таких как скорость обучения, что может быть сложным процессом.

В целом, градиентный спуск является мощным инструментом для оценки параметров линейной регрессии, который может использоваться в различных сферах, таких как финансы, маркетинг, медицина и многие другие.

В начале мы выбираем некоторые начальные значения для коэффициентов регрессии. Затем мы вычисляем ошибку предсказания, которая показывает, насколько хорошо модель соответствует данным. С помощью производных (градиентов) функции ошибки по каждому коэффициенту регрессии, мы определяем, как изменить значения коэффициентов, чтобы уменьшить ошибку.

Мы повторяем этот процесс множество раз, обновляя значения коэффициентов с каждой итерацией. Мы перемещаемся в направлении антиградиента функции ошибки с определенным шагом (скоростью обучения). Это позволяет нам приблизиться к оптимальным значениям коэффициентов, которые минимизируют ошибку предсказания.

В итоге, градиентный спуск помогает найти оптимальные значения коэффициентов линейной регрессии, что позволяет модели делать точные предсказания на основе входных данных.

# 3 Ошибка обучения, ошибка тестирования, переобучение. Привести пример в виде графика с кривыми ошибок

Ошибка обучения и ошибка тестирования — это две основные метрики, используемые для оценки качества модели машинного обучения. Ошибка обучения оценивает точность модели на данных, которые использовались для ее обучения, а ошибка тестирования - на новых данных, которые модель не видела в процессе обучения.

**Ошибка обучения** показывает, насколько хорошо модель предсказывает данные, на которых она была обучена. **Ошибка тестирования** показывает, насколько хорошо модель предсказывает новые данные, которые она не видела во время обучения.

Переобучение — это явление, когда модель слишком хорошо подгоняется к данным обучения, но не обобщается на новые данные. Это происходит, когда модель слишком сложная и может поймать шум в данных, что приводит к плохой обобщающей способности. Переобучение можно обнаружить, сравнивая ошибку обучения и ошибку тестирования. Если ошибка обучения значительно ниже ошибки тестирования, то модель может быть переобучена.

Переобучение происходит, когда модель "запоминает" данные обучения слишком точно, но не может хорошо предсказывать новые данные. Если ошибка обучения намного меньше ошибки тестирования, это может быть признаком переобучения модели.

Пример графика с кривыми ошибок:



На этом графике показаны кривые ошибок для модели линейной регрессии на обучающем и тестовом наборах данных. Здесь можно увидеть, что на начальном этапе ошибки обучения и тестирования снижаются одновременно. Однако после определенного количества итераций ошибка тестирования начинает увеличиваться, тогда как ошибка обучения продолжает уменьшаться. Это является признаком переобучения, когда модель становится слишком сложной и начинает подгоняться к шуму в данных.

В целом, для создания хорошей модели машинного обучения необходимо уделять внимание как ошибке обучения, так и ошибке тестирования, и соблюдать баланс между простотой модели и ее способностью обобщаться на новые данные.

|  |
| --- |
| Представьте, что вы учитесь играть в баскетбол и у вас есть тренер, который оценивает вашу точность бросков. Ваша цель - научиться бросать мяч так, чтобы попадать в корзину.  Ошибкой обучения будет то, как часто вы промахиваетесь или не попадаете в корзину во время тренировок с вашим тренером. Она показывает, насколько хорошо вы справляетесь с бросками во время тренировок.  Ошибкой тестирования будет то, как часто вы промахиваетесь или не попадаете в корзину, когда вы играете на настоящей игре или тестируетесь вне тренировок. Она показывает, насколько хорошо вы можете применить свои навыки в реальных условиях.  Теперь представьте, что вы потратили много времени на тренировки с вашим тренером и стали очень точными в бросках. В этом случае, ваша ошибка обучения будет очень низкой, потому что вы отлично справляетесь с бросками во время тренировок.  Однако, когда вы выходите на настоящую игру, вы обнаруживаете, что ваши броски не такие точные, как вы думали. Ваша ошибка тестирования может быть выше, чем ошибка обучения, потому что в реальной игре есть больше факторов, которые могут повлиять на ваши броски (например, противники, физическая усталость и т.д.).  Это называется переобучением. Ваша модель (в данном случае, ваши навыки в баскетболе) слишком хорошо подстроена под тренировки и не может хорошо обобщаться на новые ситуации (в реальной игре). Это происходит, когда вы слишком точно запоминаете тренировочные данные вместо того, чтобы обнаруживать общие закономерности, которые могут быть применимы в разных ситуациях. |

# 4 Полиномиальная регрессия. Назначение, формула, функция потерь, способы обучения (без вывода)

Полиномиальная регрессия — это модификация линейной регрессии, которая позволяет моделировать нелинейные зависимости между переменными.

Она используется, когда данные не могут быть описаны простой прямой линией.

Назначение полиномиальной регрессии заключается в том, чтобы аппроксимировать данные не прямой линией, а кривой n-ой степени (где n — это порядок полинома). Это может быть полезно, когда данные имеют нелинейные зависимости, которые не могут быть описаны линейной функцией.

Для примера, представьте, что у вас есть данные о температуре воздуха и количестве льда, и вы хотите узнать, как температура воздуха влияет на количество образующегося льда. Линейная регрессия не смогла бы уловить нелинейные связи, поэтому мы можем использовать полиномиальную регрессию.

Формула полиномиальной регрессии имеет вид:

y = b0 + b1x + b2x^2 + ... + bn\*x^n

где y - зависимая переменная, x - независимая переменная, b0, b1, ..., bn - коэффициенты, определяющие форму полинома.

Здесь y представляет зависимую переменную (например, количество льда), а x - независимую переменную (например, температура воздуха). Коэффициенты b0, b1, b2 и т. д. помогают нам настроить форму полинома, чтобы он подходил к нашим данным.

Функция потерь в полиномиальной регрессии обычно выбирается такой же, как и в линейной регрессии, например, среднеквадратическая ошибка (MSE).

часто используется функция потерь, называемая среднеквадратической ошибкой (MSE).

Существует несколько способов обучения модели полиномиальной регрессии, включая метод наименьших квадратов (OLS), градиентный спуск и регуляризацию, такую как Lasso и Ridge регрессия. Эти методы являются аналогами методов, применяемых в линейной регрессии.

Одним из важных аспектов полиномиальной регрессии является выбор порядка полинома. Слишком низкий порядок полинома может не смоделировать сложные зависимости в данных, а слишком высокий порядок может привести к переобучению модели. Поэтому выбор порядка полинома должен осуществляться с осторожностью, чтобы достичь наилучшей обобщающей способности модели.

Важно помнить, что выбор правильного порядка полинома очень важен. Если мы выберем слишком низкий порядок полинома, то наша модель может быть недостаточно гибкой и не сможет улавливать сложные зависимости в данных. С другой стороны, если мы выберем слишком высокий порядок полинома, модель может переобучиться и плохо работать на новых данных.

# 5 Логистическая регрессия. Назначение, формула, функция потерь, способы обучения (без вывода)

Логистическая регрессия — это метод машинного обучения, который используется для решения задач бинарной классификации, то есть для разделения данных на два класса.

**Логистическая регрессия** — это метод, который помогает нам делать прогнозы в двух вариантах: "да" или "нет". Например, мы можем использовать ее для прогнозирования, будет ли завтра солнечно или пасмурно.

Назначение логистической регрессии заключается в прогнозировании вероятности принадлежности объекта к одному из двух классов на основе его признаков.

Мы смотрим на различные факторы (например, температуру, влажность и ветер) и хотим предсказать вероятность того, что завтра будет солнечно (класс 1) или пасмурно (класс 0).

Формула логистической регрессии выглядит следующим образом:

P(y=1|x) = 1 / (1 + exp(-(b0 + b1x1 + b2x2 + ... + bn\*xn)))

где P(y=1|x) - вероятность принадлежности объекта к классу 1, x1, x2, ..., xn - значения признаков объекта, b0, b1, ..., bn - коэффициенты модели.

Здесь P(y=1|x) представляет собой вероятность того, что завтра будет солнечно (класс 1), x1, x2, ..., xn - значения факторов (например, температуры, влажности и ветра), b0, b1, ..., bn - коэффициенты модели.

Функция потерь в логистической регрессии - логистическая функция потерь (log loss), которая выражается следующим образом:

J(b) = -1/m \* Σ[yi\*log(P(yi=1|xi)) + (1-yi)\*log(P(yi=0|xi))]

где m - количество объектов в выборке, yi - истинный класс объекта i, xi - значения признаков объекта i, P(yi=1|xi) - вероятность принадлежности объекта i к классу 1 по модели.

Существуют различные способы обучения логистической регрессии, включая градиентный спуск, стохастический градиентный спуск и метод Ньютона-Рафсона. Градиентный спуск — это метод минимизации функции потерь, который обновляет значения коэффициентов модели в направлении антиградиента функции потерь.

Одним из важных аспектов логистической регрессии является регуляризация модели для борьбы с переобучением. Регуляризация может быть L1-регуляризацией (Lasso) или L2-регуляризацией (Ridge), которые добавляют штрафы за большие значения коэффициентов модели.

Важно также помнить о переобучении модели. Для борьбы с ним мы можем использовать регуляризацию. Это дополнительное правило, которое помогает создать более устойчивую модель, предотвратить переобучение и сделать более точные прогнозы.

# 6 Регуляризация. Назначение, виды, пример для линейной и логистической регрессии. В чем проявилась регуляризация в дз2

Регуляризация — это метод в машинном обучении, который используется для предотвращения переобучения модели и уменьшения ее сложности. Она заключается в добавлении дополнительного члена в функцию потерь, который штрафует модель за большие значения ее параметров.

**Регуляризация** в машинном обучении помогает предотвратить переобучение модели и сделать ее более простой. Это делается путем добавления штрафа за большие значения параметров модели в функцию потерь.

Существует два основных типа регуляризации: L1-регуляризация (также называемая Lasso) и L2-регуляризация (также называемая Ridge).

L1-регуляризация добавляет штрафное слагаемое, равное сумме абсолютных значений коэффициентов модели. Она обычно приводит к разреженным моделям, то есть к моделям, в которых некоторые признаки имеют нулевые веса.

Она заставляет модель выбирать только наиболее важные признаки и устанавливать нулевые значения для менее важных признаков.

L2-регуляризация добавляет штрафное слагаемое, равное сумме квадратов коэффициентов модели. Она обычно приводит к моделям, в которых все признаки имеют ненулевые веса.

на заставляет модель уменьшать значения всех параметров, но не приводит к выбору нулевых значений. Вместо этого, она уменьшает значения параметров, чтобы они были ближе к нулю.

Примеры регуляризации в линейной и логистической регрессии:

1. Для линейной регрессии с L1-регуляризацией, функция потерь выглядит следующим образом:
   1. J(b) = MSE + λ \* ||b||1
   2. где MSE - среднеквадратичная ошибка, ||b||1 - L1-норма вектора коэффициентов b, λ - гиперпараметр, который контролирует влияние регуляризации.
   3. Параметр λ контролирует влияние регуляризации: более большое значение λ приведет к большему сжатию коэффициентов к нулю.
2. Для логистической регрессии с L2-регуляризацией, функция потерь выглядит следующим образом:
   1. J(b) = -1/m \* Σ[yi\*log(P(yi=1|xi)) + (1-yi)\*log(P(yi=0|xi))] + λ/2m \* ||b||2^2
   2. где ||b||2 - L2-норма вектора коэффициентов b.

Проявление регуляризации в ДЗ2:

В представленном коде регуляризация проявилась в следующих моментах:

В функции run\_cross\_val при создании модели логистической регрессии с помощью LogisticRegression были заданы параметры регуляризации:

penalty='l2' указывает использование L2-регуляризации (гребневой регрессии).

C=1e5 задает обратную силу регуляризации, где более высокое значение C соответствует более слабой регуляризации.

В цикле for degree in degrees:

* Создается конвейер модели с использованием Pipeline, в котором добавляется модель логистической регрессии с полиномиальными признаками и регуляризацией.
* Модель логистической регрессии задается с параметрами регуляризации, где penalty='l2' указывает на использование L2-регуляризации, а C=1e5 задает обратную силу регуляризаци

Далее, создаются две модели model\_holdout и model\_cv с оптимальной степенью полинома:

* model\_holdout использует отложенную выборку и имеет оптимальную степень полинома optimal\_degree\_holdout.
* model\_cv использует кросс-валидацию и имеет оптимальную степень полинома optimal\_degree\_cv.

Обе модели также включают регуляризацию с параметрами penalty='l2' и C=1e5.

Затем, модели обучаются на обучающих данных и оцениваются их доля правильных классификаций на обучающем и тестовом подмножествах.

# 7 Выбор модели. Отложенная выборка и кросс-валидация

Выбор модели — это процесс выбора наилучшей модели машинного обучения для конкретной задачи. Когда мы создаем модель машинного обучения, мы обычно подстраиваем ее гиперпараметры, чтобы получить лучшие результаты на обучающем наборе данных. Однако это может привести к переобучению модели, когда она настроена на обучающий набор данных и не обобщается на новые данные.

Чтобы избежать переобучения, необходимо оценивать модели на тестовых данных. Однако мы не можем использовать тестовые данные во время обучения модели, поэтому мы должны использовать другие методы, такие как отложенная выборка и кросс-валидация.

Отложенная выборка - это метод, при котором мы разделяем наши данные на обучающий и тестовый наборы данных. Мы обучаем модель на обучающем наборе данных и затем проверяем ее на тестовом наборе данных. Этот метод прост в использовании, но имеет недостаток: если размер тестового набора данных маленький, то оценка модели может быть не очень точной.

**Отложенная** **выборка** - это простой метод, при котором мы разделяем данные на обучающий и тестовый наборы. Мы обучаем модель на обучающем наборе и оцениваем ее на тестовом наборе. Однако, если размер тестового набора маленький, оценка модели может быть неточной.

Кросс-валидация — это метод, при котором мы разделяем наши данные на несколько фолдов и выполняем несколько итераций обучения и оценки модели. На каждой итерации один из фолдов используется в качестве тестового набора данных, а оставшиеся фолды используются для обучения модели. Этот метод более точен, чем отложенная выборка, потому что он оценивает модель на всех доступных данных, но требует больше времени на обучение и оценку модели.

**Кросс-валидация** - это метод, при котором мы разделяем данные на несколько фолдов и выполняем несколько итераций обучения и оценки модели. На каждой итерации один фолд используется для тестирования, а остальные фолды - для обучения. Этот метод более точен, так как оценивает модель на всех доступных данных, но требует больше времени.

Оба метода, отложенная выборка и кросс-валидация, позволяют нам оценивать производительность модели на новых данных и выбирать наилучшую модель для конкретной задачи машинного обучения.

Фолд — это подмножество данных, используемое в кросс-валидации для обучения и оценки модели. Когда мы применяем кросс-валидацию, мы разделяем наш набор данных на k фолдов, где k — это количество фолдов. Обычно используются значения k от 5 до 10.

На каждой итерации кросс-валидации один из фолдов используется в качестве тестового набора данных, а оставшиеся фолды используются для обучения модели. Таким образом, каждый фолд используется как тестовый набор данных ровно один раз. После завершения всех итераций мы получаем k оценок производительности модели, которые могут быть использованы для оценки ее средней производительности.

Фолды должны быть выбраны случайным образом, чтобы убедиться, что обучающий набор данных не содержит какие-либо особенности, которые могут повлиять на производительность модели. Кроме того, каждый фолд должен быть достаточно большим, чтобы обеспечить достаточную точность оценки производительности модели.

# 8 Каким образом определялась наилучшая модель в полиномиальной регрессии в дз2. Привести схему разбиения датасета и построения графиков ошибок

В домашнем задании наилучшая модель в полиномиальной регрессии была определена с использованием метода отложенной выборки. Процесс определения наилучшей модели включал следующие шаги:

Создание функции select\_poly\_degree, которая принимает обучающие и проверочные данные, а также максимальную степень полинома. Внутри функции происходит итерация по различным степеням полинома и для каждой степени:

a. Создание объекта PolynomialFeatures с текущей степенью полинома для преобразования признаков.

b. Применение преобразования к обучающим и проверочным данным с помощью fit и transform соответственно.

c. Создание модели Pipeline с использованием StandardScaler() для стандартизации данных и моделью LinearRegression() для обучения.

d. Вычисление среднеквадратической ошибки и коэффициента детерминации на обучающем и проверочном подмножествах с помощью функции calc\_scores.

e. Сохранение значений среднеквадратической ошибки и коэффициента детерминации для каждой степени полинома.

Выбор оптимальной степени полинома на основе среднеквадратической ошибки на проверочном подмножестве. Используется функция np.argmin для поиска индекса с минимальным значением среднеквадратической ошибки, а затем добавляется 1, так как индексы начинаются с 0.

Построение графиков зависимости среднеквадратической ошибки и коэффициента детерминации от степени полинома для обучающего и проверочного подмножеств. На графиках также отображается вертикальная линия, соответствующая оптимальной степени полинома.

Обучение модели линейной регрессии на обучающих данных с использованием полиномиальных признаков и оптимальной степени полинома. В этом шаге используется Pipeline с StandardScaler() для стандартизации данных и моделью LinearRegression() для обучения.

Применение обученной модели к тестовым данным для выполнения предсказаний.

Построение графиков: диаграммы разброса исходных данных с наилучшей функцией регрессии и графика истинных и предсказанных значений тестовых данных.

Таким образом, определение наилучшей модели в полиномиальной регрессии включает итерацию по различным степеням полинома, вычисление метрик качества на обучающем и проверочном подмножествах, выбор оптимальной степени полинома на основе проверочной ошибки и построение графиков для визуализации результатов.

Схема разбиения датасета и построения графиков ошибок в полиномиальной регрессии может выглядеть следующим образом:

* Разбиение датасета на обучающее, проверочное и тестовое подмножества:
* Исходный датасет делится на три части: обучающее подмножество, проверочное подмножество и тестовое подмножество.
* Обычно используется соотношение примерно 60-70% для обучения, 15-20% для проверки и 15-20% для тестирования. Размеры подмножеств могут варьироваться в зависимости от конкретной задачи.

Итерация по различным степеням полинома:

* Для каждой степени полинома от 1 до максимальной заданной степени происходит следующее:
* Создание объекта PolynomialFeatures с текущей степенью полинома для преобразования признаков.
* Применение преобразования к обучающим, проверочным и тестовым данным с помощью fit\_transform и transform соответственно.
* Создание модели Pipeline с использованием StandardScaler() для стандартизации данных и моделью LinearRegression() для обучения.
* Обучение модели на обучающих данных с использованием преобразованных признаков.
* Вычисление среднеквадратической ошибки на обучающем и проверочном подмножествах с помощью функции оценки ошибки, например, среднеквадратичной ошибки (MSE) или коэффициента детерминации (R^2).
* Сохранение значений ошибок для текущей степени полинома.

Построение графиков ошибок:

* Построение графика зависимости ошибки (MSE или R^2) от степени полинома для обучающего и проверочного подмножеств.
* На графике также может быть отображена вертикальная линия или маркер, указывающий на оптимальную степень полинома с наименьшей ошибкой на проверочном подмножестве.
* График помогает визуально определить оптимальную степень полинома, при которой модель достигает наилучшего качества на проверочных данных.
* Выбор оптимальной модели:
* Измерение ошибки на тестовом подмножестве для модели с оптимальной степенью полинома.
* Сравнение ошибки на тестовом подмножестве с ошибкой на обучающем и проверочном подмножествах для оценки обобщающей способности модели.
* Выбор модели с наилучшим качеством на проверочном подмножестве или тестовом подмножестве для последующего использования на новых данных.

Это общая схема разбиения датасета и построения графиков ошибок в полиномиальной регрессии. В реальных задачах могут быть некоторые отличия в зависимости от конкретной реализации и используемых библиотек или алгоритмов.

# 9 Какие метрики использовались для оценки качества моделей в дз2, на каких этапах они вычислялись и для чего

В домашнем задании использовались две метрики для оценки качества моделей в полиномиальной регрессии: среднеквадратическая ошибка (Mean Squared Error, MSE) и коэффициент детерминации (R^2).

Среднеквадратическая ошибка (MSE):

* MSE вычислялась на этапе обучения модели и на этапе тестирования.
* На этапе обучения, MSE использовалась для оценки качества модели на обучающих данных. Она вычислялась путем сравнения прогнозируемых значений модели с фактическими значениями целевой переменной на обучающих данных.
* На этапе тестирования, MSE использовалась для оценки качества модели на тестовых данных. Она вычислялась путем сравнения прогнозируемых значений модели с фактическими значениями целевой переменной на тестовых данных.
* MSE является показателем средней квадратичной ошибки между прогнозируемыми и фактическими значениями. Чем меньше значение MSE, тем лучше качество модели.

Коэффициент детерминации (R^2):

* Коэффициент детерминации R^2 вычислялся на этапе тестирования.
* Он использовался для оценки объясняющей способности модели, то есть способности модели объяснить изменчивость целевой переменной.
* Коэффициент детерминации R^2 принимает значения от 0 до 1, где 1 означает идеальное соответствие модели данным. Близкие к 1 значения R^2 указывают на хорошую прогностическую способность модели.
* R^2 вычисляется сравнением суммы квадратов отклонений прогнозируемых значений от среднего значения целевой переменной со суммой квадратов отклонений фактических значений от среднего значения целевой переменной.

Метрики MSE и R^2 использовались для оценки качества моделей на обучающих и тестовых данных. Они позволяют оценить точность модели, сравнивая прогнозируемые значения с фактическими значениями. Меньшее значение MSE и близкое к 1 значение R^2 указывают на лучшее качество модели.