1. Анализ данных средствами Python

Анализ данных с использованием Python включает следующие шаги:

Загрузка данных: Используйте библиотеки Pandas, NumPy или SQL Alchemy для загрузки данных из различных источников, таких как CSV-файлы, базы данных или веб-API.

Очистка данных: очистите данные от дубликатов, пропущенных значений и выбросов с помощью функций и методов Pandas и NumPy.

Визуализация данных: Используйте библиотеки Matplotlib, Seaborn или Plotly для создания графиков, диаграмм и других визуализаций, чтобы наглядно представить данные.

Статистический анализ: Используйте библиотеки SciPy и Statsmodels для выполнения статистических расчетов, гипотезных тестов и построения регрессионных моделей.

Машинное обучение: Применяйте библиотеки scikit-learn, TensorFlow или PyTorch для создания и обучения моделей машинного обучения, включая классификацию, регрессию, кластеризацию и другие задачи.

Интерпретация результатов: Анализируйте результаты, сделайте выводы и примите решения на основе полученных данных, используя свои знания предметной области.

Python предоставляет обширный набор инструментов для анализа данных и может быть использован для выполнения различных задач, связанных с обработкой, очисткой, визуализацией и анализом данных, а также для построения моделей машинного обучения.

1. Визуализация данных средствами Python

Визуализация данных с использованием Python включает следующие шаги:

Загрузка данных: загрузите данные из источника, такого как CSV-файл, база данных или веб-API, с использованием библиотеки Pandas или других подходящих инструментов.

Подготовка данных: Произведите предварительную обработку данных, включая очистку, фильтрацию и преобразование данных, чтобы подготовить их для визуализации.

Выбор типа визуализации: определите, какой тип графика или диаграммы наиболее подходит для отображения ваших данных. В Python доступно множество библиотек для визуализации, таких как Matplotlib, Seaborn, Plotly, и другие, каждая из которых предлагает различные типы графиков и диаграмм.

Создание графика: Используйте выбранную библиотеку для создания графика или диаграммы. Это может быть линейный график, столбчатая диаграмма, круговая диаграмма, диаграмма рассеяния, тепловая карта и многое другое. Настройте параметры, такие как оси, цвета, подписи и заголовки, чтобы получить нужный вид.

Визуальное улучшение: Добавьте дополнительные элементы для улучшения визуализации, такие как легенды, аннотации, сетки и шкалы. Это поможет лучше интерпретировать данные и сделать график более понятным.

Сохранение и представление: сохраните график в файле изображения или встроенном формате, таком как PDF или SVG, для последующего использования или представления. Вы также можете встроить график в веб-страницу или интегрировать его в интерактивный интерфейс с помощью библиотеки Plotly.

Python предлагает широкие возможности для создания качественных и информативных визуализаций данных. Выберите подходящую библиотеку и используйте ее функции и методы для создания графиков, которые наилучшим образом отражают особенности ваших данных.

1. Jupyter notebook

Jupyter Notebook – это интерактивная среда разработки, которая позволяет создавать и выполнять код, а также интегрировать текстовую документацию, визуализации и другие элементы прямо в ноутбуке. Вот краткое описание основных характеристик Jupyter Notebook:

Язык программирования: Jupyter Notebook поддерживает множество языков программирования, включая Python, R, Julia и другие. Однако Python является одним из наиболее популярных языков, используемых в Jupyter Notebook.

Клетки: Jupyter Notebook организован в виде ячеек, которые могут содержать код, текст (в формате Markdown) или результаты выполнения кода. Вы можете выполнять каждую ячейку по отдельности и видеть результаты прямо в ноутбуке.

Интерактивность: Jupyter Notebook позволяет вам взаимодействовать с кодом и получать мгновенные результаты. Вы можете изменять значения переменных, исполнять отдельные участки кода или экспериментировать с параметрами и сразу же видеть результаты.

Визуализации: Jupyter Notebook обеспечивает удобную интеграцию с библиотеками визуализации данных, такими как Matplotlib, Seaborn и Plotly. Вы можете создавать графики, диаграммы и другие визуализации прямо в ноутбуке.

Документация: Вы можете создавать размеченные текстовые ячейки с использованием синтаксиса Markdown для добавления документации, описания процесса анализа данных, выводов и других комментариев, делая ноутбук более понятным и понятным для других пользователей или самого себя.

Экспорт: Jupyter Notebook позволяет экспортировать ноутбуки в различные форматы, такие как HTML, PDF, Markdown или Python скрипт. Это удобно, если вы хотите предоставить ноутбук для просмотра или запуска на другой системе.

Jupyter Notebook предоставляет удобную и гибкую среду для разработки, анализа данных и визуализации, объединяя код, текст и результаты выполнения в одном документе. Это популярный инструмент для работы с данными в Python и других языках программирования.

1. Основные классы и методы библиотеки NumPy

Библиотека NumPy предоставляет функциональность для работы с многомерными массивами и выполняет различные математические операции над ними. Вот краткое описание основных классов и методов NumPy:

ndarray (N-dimensional array): это основной класс NumPy, представляющий многомерный массив. Он обеспечивает эффективное хранение и манипуляции с данными. ndarray поддерживает индексацию, нарезку, математические операции и другие операции над массивами.

Создание массивов:

np.array: Создает массив из обычных Python-списков или других итерируемых объектов.

np.zeros, np.ones, np.empty: Создают массивы с нулевыми, единичными или неинициализированными значениями соответственно.

np.arange: Создает массив с последовательно увеличивающимися значениями.

np.linspace, np.logspace: Создают равномерно распределенные значения или значения в логарифмической шкале.

Математические операции:

ndarray.shape: Возвращает размеры массива в виде кортежа.

ndarray.reshape: Изменяет форму массива, создавая новый массив.

ndarray.ndim: Возвращает количество измерений (осей) массива.

ndarray.dtype: Возвращает тип данных элементов массива.

ndarray.min, ndarray.max, ndarray.mean, ndarray.sum: Выполняют операции поиска минимума, максимума, среднего значения и суммы элементов массива.

ndarray.dot: Выполняет матричное умножение двух массивов.

np.concatenate, np.vstack, np.hstack: Объединяют массивы вдоль разных осей.

Индексация и вырезка:

ndarray[index]: Обращение к элементу массива по индексу или индексам.

ndarray[start:end]: Создание среза массива от индекса start до индекса end.

ndarray[start:end:step]: Создание среза массива с шагом step.

1. Основные классы и методы библиотеки Pandas

DataFrame: Это двумерная структура данных, представляющая собой таблицу с рядами и столбцами, где каждый столбец может содержать данные разных типов.

Основные методы:

DataFrame: Создает объект DataFrame из массива, списка, словаря или другого DataFrame.

df.head, df.tail: Возвращает первые или последние n строк DataFrame.

df.shape: Возвращает размеры DataFrame в виде кортежа (количество строк, количество столбцов).

df.columns: Возвращает названия столбцов DataFrame.

df.index: Возвращает индексы строк DataFrame.

df.describe: Возвращает сводную статистику по числовым столбцам DataFrame.

df.info: Выводит информацию о DataFrame, включая типы данных и общую информацию о памяти.

df.loc, df.iloc: Позволяют обращаться к элементам DataFrame по меткам или целочисленным индексам.

df.groupby: Группирует данные по столбцам и выполняет агрегационные функции.

Чтение и запись данных:

pd.read\_csv, pd.read\_excel: Читают данные из файлов CSV или Excel и создают DataFrame.

df.to\_csv, df.to\_excel: Записывают данные из DataFrame в файлы CSV или Excel.

1. Основные классы и методы библиотеки SymPy

SymPy - это библиотека символьных вычислений на языке Python. Вот некоторые основные классы и методы, предоставляемые этой библиотекой:

Symbol: Класс Symbol представляет символьную переменную. Он используется для создания символьных выражений и символьных функций. Некоторые методы класса Symbol включают subs() (для подстановки значений), diff() (для дифференцирования) и simplify() (для упрощения выражений).

Expr: Класс Expr представляет символьное выражение, состоящее из символов и операций. Он является базовым классом для всех символьных выражений в SymPy. Методы класса Expr включают subs() (для подстановки значений), expand() (для раскрытия выражений), simplify() (для упрощения выражений) и многие другие.

Function: Класс Function представляет символьную функцию. Он используется для создания символьных функций, таких как тригонометрические функции, экспоненциальные функции и логарифмы. Некоторые методы класса Function включают diff() (для дифференцирования), integrate() (для интегрирования) и simplify() (для упрощения выражений).

Eq: Класс Eq представляет уравнение в SymPy. Он используется для создания и работы с уравнениями. Некоторые методы класса Eq включают solve() (для решения уравнений), subs() (для подстановки значений) и simplify() (для упрощения уравнений).

solve(): Функция solve() используется для решения уравнений и систем уравнений. Она принимает уравнения или системы уравнений в виде аргументов и возвращает их решения в символьной форме.

integrate(): Функция integrate() используется для интегрирования символьных выражений. Она принимает выражение и переменную интегрирования в качестве аргументов и возвращает результат интегрирования.

simplify(): Функция simplify() используется для упрощения символьных выражений. Она принимает выражение в качестве аргумента и возвращает его упрощенную форму.

1. Основные классы и методы библиотеки SciPy

SciPy - это библиотека научных вычислений на языке Python. Она предоставляет множество классов и функций для решения различных задач в науке и инженерии. Вот некоторые основные классы и методы, предоставляемые библиотекой SciPy:

scipy.optimize: Модуль optimize предоставляет функции для оптимизации (минимизации или максимизации) функций. Некоторые методы этого модуля включают minimize() (для минимизации функций), root() (для нахождения корней уравнений) и curve\_fit() (для аппроксимации данных).

scipy.integrate: Модуль integrate предоставляет функции для численного интегрирования. Он содержит методы, такие как quad() (для определенного интеграла), dblquad() (для двойного интеграла) и odeint() (для численного решения обыкновенных дифференциальных уравнений).

scipy.stats: Модуль stats предоставляет функции для статистического анализа данных. Он содержит методы, такие как norm() (для работы с нормальным распределением), ttest\_ind() (для t-теста), pearsonr() (для коэффициента корреляции Пирсона) и многое другое.

scipy.linalg: Модуль linalg предоставляет функции для работы с линейной алгеброй. Он содержит методы для работы с матрицами, такие как inv() (для нахождения обратной матрицы), eig() (для нахождения собственных значений и собственных векторов) и solve() (для решения систем линейных уравнений).

scipy.signal: Модуль signal предоставляет функции для обработки сигналов. Он содержит методы для фильтрации, преобразования Фурье, корреляции и других операций над сигналами.

scipy.spatial: Модуль spatial предоставляет функции для работы с пространственными данными. Он содержит методы для работы с деревьями пространственного разбиения, поиска ближайших соседей, вычисления расстояний и других операций над точками и пространственными структурами данных.

scipy.fftpack: Модуль fftpack предоставляет функции для работы с преобразованием Фурье. Он содержит методы для вычисления дискретного преобразования Фурье (DFT) и обратного DFT, а также других преобразований, таких как преобразование Хартли и быстрое преобразование Фурье (FFT).

1. Основные классы и методы библиотеки Scikit-learn

Scikit-learn предоставляет множество классов и методов для различных задач машинного обучения. Вот некоторые основные классы и методы библиотеки Scikit-learn:

sklearn.model\_selection: Модуль model\_selection предоставляет функции для разделения данных на обучающую и тестовую выборки, а также для оценки и выбора моделей. Некоторые методы этого модуля включают train\_test\_split() (для разделения данных), cross\_val\_score() (для оценки моделей с использованием перекрестной проверки) и GridSearchCV() (для поиска оптимальных параметров модели).

sklearn.preprocessing: Модуль preprocessing предоставляет функции для предварительной обработки данных. Он содержит классы и методы для масштабирования, кодирования категориальных признаков, обработки пропущенных значений и других операций. Некоторые классы этого модуля включают StandardScaler (для стандартизации данных), OneHotEncoder (для кодирования категориальных признаков) и Imputer (для обработки пропущенных значений).

sklearn.linear\_model: Модуль linear\_model предоставляет классы и методы для линейных моделей. Он включает классы для линейной регрессии, логистической регрессии, регуляризации и других методов. Некоторые классы этого модуля включают LinearRegression (для линейной регрессии), LogisticRegression (для логистической регрессии) и Lasso (для линейной модели с L1-регуляризацией).

sklearn.tree: Модуль tree предоставляет классы и методы для деревьев решений. Он содержит классы для классификации и регрессии на основе деревьев решений, а также методы для визуализации деревьев. Некоторые классы этого модуля включают DecisionTreeClassifier (для классификации с использованием деревьев решений), DecisionTreeRegressor (для регрессии с использованием деревьев решений) и export\_graphviz (для визуализации деревьев решений).

sklearn.cluster: Модуль cluster предоставляет классы и методы для кластерного анализа. Он включает классы для кластеризации на основе метода k-средних, иерархической кластеризации и других методов. Некоторые классы этого модуля включают KMeans (для кластеризации на основе метода k-средних), AgglomerativeClustering (для иерархической кластеризации) и DBSCAN (для кластеризации на основе плотности).

sklearn.metrics: Модуль metrics предоставляет функции для оценки качества моделей. Он содержит методы для вычисления различных метрик, таких как точность, полнота, F-мера, среднеквадратическая ошибка и других. Некоторые методы этого модуля включают accuracy\_score() (для вычисления точности), precision\_score() (для вычисления полноты), mean\_squared\_error() (для вычисления среднеквадратической ошибки) и многое другое.

sklearn.ensemble: Модуль ensemble предоставляет классы и методы для ансамблевого обучения. Он включает классы для случайного леса, градиентного бустинга и других методов ансамблевого обучения. Некоторые классы этого модуля включают RandomForestClassifier (для классификации на основе случайного леса), GradientBoostingRegressor (для регрессии на основе градиентного бустинга) и VotingClassifier (для объединения нескольких классификаторов).

1. Основные классы и методы библиотеки Matplotlin

Matplotlib - это библиотека визуализации данных на языке Python. Она предоставляет широкий набор классов и методов для создания различных типов графиков и диаграмм. Вот некоторые основные классы и методы библиотеки Matplotlib:

matplotlib.pyplot: Модуль pyplot является основным интерфейсом для создания графиков и диаграмм. Он предоставляет функции для создания графического окна, отображения данных на графике, добавления подписей и многих других операций. Некоторые методы этого модуля включают plot() (для создания линейных графиков), scatter() (для создания точечных графиков), hist() (для создания гистограмм) и xlabel() (для добавления подписи по оси x).

matplotlib.figure: Класс Figure представляет графическую фигуру, на которой отображаются графики и диаграммы. Он предоставляет методы для настройки параметров фигуры, таких как размер, заголовок и подписи осей.

matplotlib.axes: Класс Axes представляет систему координат, на которой отображаются графики и диаграммы. Он предоставляет методы для добавления графиков, настройки осей, добавления сетки и других операций.

matplotlib.pyplot.subplots: Функция subplots() используется для создания группы фигур и осей. Она возвращает кортеж, содержащий фигуру и массив объектов Axes, на которых можно отображать графики.

matplotlib.pyplot.savefig: Функция savefig() используется для сохранения графика в файл. Она позволяет указать формат файла (например, PNG, PDF) и другие параметры сохранения.

matplotlib.colors: Модуль colors предоставляет функции и классы для работы с цветами. Он включает функции для создания цветовых карт, настройки цветовых палитр и других операций.

matplotlib.legend: Класс Legend представляет легенду, отображаемую на графике. Он позволяет добавить описание различных элементов графика, таких как линии, точки или заполнения, и связать их с определенными метками.

1. Основные классы и методы библиотеки Seaborn

Seaborn - это библиотека визуализации данных на языке Python, основанная на Matplotlib. Она предоставляет простой и красивый интерфейс для создания информативных статистических графиков. Вот некоторые основные классы и методы библиотеки Seaborn:

seaborn.barplot: Функция barplot() используется для создания столбчатых графиков. Она позволяет отобразить средние значения или суммы по категориям и визуализировать различия между ними.

seaborn.countplot: Функция countplot() используется для создания гистограммы частоты появления категорий. Она отображает количество наблюдений для каждой категории.

seaborn.boxplot: Функция boxplot() используется для создания ящиков с усами. Она отображает распределение данных и позволяет выявить выбросы и медиану.

seaborn.heatmap: Функция heatmap() используется для создания тепловой карты. Она отображает матрицу данных с использованием цветовой шкалы, позволяя визуализировать корреляцию или относительное значение данных.

seaborn.scatterplot: Функция scatterplot() используется для создания точечных графиков. Она позволяет отобразить связь между двумя переменными и выявить возможные тренды или выбросы.

seaborn.lineplot: Функция lineplot() используется для создания линейных графиков. Она отображает изменение значения переменной в зависимости от другой переменной или времени.

seaborn.pairplot: Функция pairplot() используется для создания матрицы графиков рассеяния. Она позволяет визуально сравнить каждую пару переменных в наборе данных.

seaborn.set: Функция set() используется для настройки стилей графиков Seaborn. Она позволяет изменить цветовую палитру, стиль фона, размеры шрифтов и другие параметры.

1. Основные классы и методы библиотеки Plotly

Plotly - это библиотека визуализации данных на языке Python, которая позволяет создавать интерактивные и высококачественные графики. Она предоставляет широкий набор классов и методов для создания различных типов графиков и диаграмм. Вот некоторые основные классы и методы библиотеки Plotly:

plotly.graph\_objs.Figure: Класс Figure представляет графическую фигуру, на которой отображаются графики и диаграммы. Он предоставляет методы для настройки параметров фигуры, таких как заголовок, размеры, подписи осей и многое другое.

plotly.graph\_objs.Scatter: Класс Scatter представляет точечный график или линейный график. Он позволяет задать координаты точек или значения по осям и настроить их внешний вид, такой как цвет, размер или тип маркера.

plotly.graph\_objs.Bar: Класс Bar представляет столбчатый график. Он позволяет задать значения столбцов, настроить их внешний вид и добавить дополнительные атрибуты, такие как подписи столбцов или группировка.

plotly.graph\_objs.Pie: Класс Pie представляет круговую диаграмму. Он позволяет задать значения секторов и настроить их внешний вид, такой как цвета, подписи или выделение.

plotly.graph\_objs.Layout: Класс Layout представляет компоновку графика, включая оси, легенду, цветовую палитру и другие атрибуты. Он позволяет настроить внешний вид графика, такой как размеры, отступы, фон и многое другое.

plotly.subplots.make\_subplots: Функция make\_subplots() используется для создания группы подграфиков. Она позволяет разместить несколько графиков на одной фигуре и настроить их расположение.

plotly.offline.plot: Функция plot() используется для сохранения и отображения графика в виде интерактивной веб-страницы. Она позволяет сохранить график в файл или открыть его в браузере.

plotly.express: Модуль express предоставляет простой интерфейс для создания различных типов графиков с минимальным количеством кода. Он содержит функции для создания точечных графиков, столбчатых графиков, линейных графиков, круговых диаграмм и многих других.

1. Основные понятия кластерного анализа

Кластерный анализ - это метод статистического анализа, который позволяет группировать объекты на основе их сходства или близости. Основные понятия, связанные с кластерным анализом, включают:

Кластеры: Кластеры - это группы объектов, которые имеют схожие характеристики или близкое расположение в пространстве признаков.

Расстояние: Расстояние - это мера различия или сходства между объектами. Расстояние может быть определено на основе различных метрик, таких как евклидово расстояние, косинусное расстояние и другие.

Функция близости: Функция близости оценивает степень схожести или различия между парами объектов. Она может быть определена на основе выбранной метрики расстояния.

Методы кластеризации: Методы кластеризации определяют алгоритмы, которые используются для разделения объектов на кластеры. Некоторые популярные методы включают иерархическую кластеризацию, метод k-средних и DBSCAN.

Центроид: Центроид - это точка, которая представляет среднее значение всех объектов в кластере. В методе k-средних центроид используется для определения кластеров.

Индекс качества кластеризации: Индексы качества кластеризации используются для оценки и сравнения результатов кластерного анализа. Они позволяют оценить степень компактности внутрикластерных объектов и разделимость между кластерами.

Валидация кластеризации: Валидация кластеризации - это процесс оценки и подтверждения качества результатов кластерного анализа. Включает в себя методы, такие как индексы силуэта, индексы Дэвиса-Болдина и индексы Рэнда.

1. Постановка задачи кластеризации

Постановка задачи кластеризации - это определение целей и задач, связанных с группировкой объектов на основе их сходства или близости. В краткой форме постановка задачи кластеризации включает следующие элементы:

Набор данных: Имеется набор объектов, для которых необходимо провести кластеризацию. Эти объекты могут быть представлены в виде векторов признаков или другой структуры данных.

Цель: Определение цели или целей кластеризации. Цель может быть различной, например, выявление скрытой структуры данных, группировка похожих объектов, обнаружение аномалий или классификация данных.

Метрика или функция близости: Определение метрики или функции близости для оценки сходства между объектами. Метрика может быть выбрана в зависимости от характеристик данных и поставленной задачи.

Количество кластеров: Определение или предположение о количестве кластеров, которые требуется обнаружить. В некоторых случаях это число известно заранее, а в других случаях требуется провести исследование или использовать алгоритмы для автоматического определения количества кластеров.

Алгоритмы кластеризации: Выбор и применение соответствующих алгоритмов кластеризации для разделения объектов на кластеры. Это может включать иерархическую кластеризацию, метод k-средних, DBSCAN или другие алгоритмы.

Оценка качества: Оценка качества полученных кластеров с использованием соответствующих метрик или индексов качества. Это позволяет оценить эффективность проведенной кластеризации и провести сравнение различных методов или параметров.

1. Меры расстояний

В кластерном анализе используются различные меры расстояний для определения сходства или различия между объектами. Некоторые распространенные меры расстояний включают:

Евклидово расстояние: Евклидово расстояние измеряет прямое расстояние между двумя точками в многомерном пространстве. Оно определяется как квадратный корень из суммы квадратов разностей по каждой координате.

Манхэттенское расстояние: Манхэттенское расстояние измеряет сумму абсолютных разностей между координатами двух точек. Оно представляет собой сумму по модулю разностей по каждой оси.

Косинусное расстояние: Косинусное расстояние измеряет угол между двумя векторами признаков. Оно основывается на косинусе угла между векторами и представляет собой единиц минус косинус этого угла.

Коэффициент Жаккара: Коэффициент Жаккара используется для измерения сходства между двумя множествами. Он вычисляется как отношение размера пересечения множеств к их размеру объединения.

Расстояние Хэмминга: Расстояние Хэмминга измеряет различие между двумя последовательностями одинаковой длины. Оно определяется как количество позиций, в которых символы последовательностей различаются.

Мера сходства корреляции: Мера сходства корреляции используется для измерения сходства между двумя переменными. Она основывается на корреляционных коэффициентах и может быть использована для измерения сходства между непрерывными или дискретными переменными.

1. Типы кластерных структур

В кластерном анализе существуют различные типы кластерных структур, которые могут быть обнаружены в наборе данных. Некоторые из наиболее распространенных типов включают:

Плотные кластеры: Плотные кластеры представляют собой группы объектов, которые находятся близко друг к другу и имеют высокую плотность в пространстве признаков. Внутри плотных кластеров объекты обычно находятся близко друг к другу, а между кластерами - значительные промежутки.

Разреженные кластеры: Разреженные кластеры представляют собой группы объектов, которые находятся далеко друг от друга и имеют низкую плотность в пространстве признаков. Внутри разреженных кластеров объекты могут быть разбросаны, а между кластерами - значительные промежутки.

Иерархическая структура: Иерархическая структура кластеров представляет собой иерархическую организацию объектов, где кластеры могут быть вложенными друг в друга. Это может быть представлено в виде дерева или дендрограммы, которая показывает отношения между кластерами на разных уровнях иерархии.

Цепочки и ленты: Цепочки и ленты представляют собой структуры, в которых объекты расположены вдоль линейных или изогнутых форм. В этом случае кластеры могут быть представлены в виде последовательностей или групп объектов, вытянутых вдоль определенного направления.

Сетевая структура: Сетевая структура кластеров представляет собой группы объектов, связанных друг с другом через сеть или граф. Кластеры в сетевых структурах могут быть определены как плотные области связей или группы сильно связанных узлов.

Шум или выбросы: Шум или выбросы представляют объекты, которые не принадлежат явным кластерам или находятся в отдалении от других объектов. Они не подчиняются общей структуре кластеров и могут быть нежелательными или интересными аномалиями в данных.

1. Классификация алгоритмов кластеризации

Алгоритмы кластеризации могут быть классифицированы по различным критериям. Один из способов классификации основан на их характеристиках и подходах к кластеризации. Вот некоторые основные классы алгоритмов кластеризации:

Иерархические алгоритмы: Иерархические алгоритмы строят иерархическую структуру кластеров, представленную в виде дерева или дендрограммы. Они могут быть агломеративными, начиная с каждого объекта в отдельном кластере и объединяя их по мере продвижения вверх по дереву, или дивизивными, начиная с одного крупного кластера и разделяя его на более мелкие кластеры.

Алгоритмы на основе прототипов: Эти алгоритмы ищут прототипы, представляющие центры кластеров или типичные представители каждого кластера. Примерами таких алгоритмов являются метод k-средних (k-means) и алгоритмы средней связи.

Плотностные алгоритмы: Плотностные алгоритмы кластеризации опираются на понятие плотности объектов в пространстве признаков. Они идентифицируют области с высокой плотностью как кластеры и разделяют их с помощью областей с низкой плотностью. Примеры плотностных алгоритмов включают DBSCAN (Density-Based Spatial Clustering of Applications with Noise) и OPTICS (Ordering Points To Identify the Clustering Structure).

Вероятностные алгоритмы: Вероятностные алгоритмы кластеризации основаны на моделировании вероятностной структуры данных и использовании статистических методов для определения кластеров. Примерами таких алгоритмов являются алгоритм Expectation-Maximization (EM) и Gaussian Mixture Models (GMM).

1. Обучение без учителя

Обучение без учителя в кластерном анализе - это методология анализа данных, при которой модель обучается находить скрытые структуры и закономерности в данных без предварительно заданных меток или учителя. Она позволяет искать естественные группировки или кластеры объектов на основе их сходства.

В процессе обучения без учителя в кластерном анализе модель автоматически определяет кластеры, основываясь на сходстве объектов в пространстве признаков. Алгоритмы кластеризации выполняют разделение набора данных на группы таким образом, чтобы объекты внутри каждого кластера были максимально похожи друг на друга, а объекты из разных кластеров были максимально различны.

Обучение без учителя в кластерном анализе имеет множество применений. Например, оно может использоваться для сегментации клиентов в маркетинге, исследования генетических данных, анализа социальных сетей, обработки изображений и многих других областей. Оно позволяет обнаруживать скрытые паттерны и структуры, что помогает в понимании данных и принятии информированных решений.

Процесс обучения без учителя в кластерном анализе требует выбора подходящего алгоритма кластеризации, определения меры сходства или расстояния между объектами, а также интерпретации и анализа полученных кластеров.

1. Первичный анализ данных

Первичный анализ данных в кластерном анализе - это процесс предварительного исследования и подготовки данных перед применением алгоритмов кластеризации. Целью первичного анализа данных является понимание характеристик и свойств набора данных, выявление выбросов, пропущенных значений и других проблем, а также подготовка данных для эффективного применения алгоритмов кластеризации.

В процессе первичного анализа данных в кластерном анализе можно выполнить следующие шаги:

Изучение и описание данных: Просмотрите данные, ознакомьтесь с их структурой, типами переменных и общим содержанием. Опишите признаки и их характеристики, чтобы получить представление о данных.

Обработка пропущенных значений: Идентифицируйте пропущенные значения в данных и принимайте решение о том, как с ними поступить. Вы можете заполнить пропущенные значения средними, медианами или наиболее часто встречающимися значениями, либо удалить соответствующие объекты или признаки.

Обработка выбросов: Выполните анализ выбросов и принимайте решение о их обработке. Вы можете удалить выбросы, заменить их на более типичные значения или использовать методы обработки выбросов, такие как обрезание или масштабирование.

Масштабирование признаков: Если признаки имеют различные шкалы или единицы измерения, рассмотрите возможность масштабирования признаков, чтобы они имели одинаковый диапазон значений. Это может помочь предотвратить преобладание признаков с более широким диапазоном над другими при кластеризации.

Визуализация данных: Используйте графики и визуализацию для исследования данных и поиска возможных кластерных структур. Рассмотрите диаграммы рассеяния, гистограммы, тепловые карты и другие графические методы для понимания распределения и взаимосвязей признаков.

Выбор подходящего алгоритма и метрики: Исходя из характера данных и поставленных задач, выберите подходящий алгоритм кластеризации и метрику расстояния или сходства для оценки качества кластеризации.

1. Критерии информативности признаков

Критерии информативности признаков в кластерном анализе помогают определить, насколько хорошо каждый признак различает объекты внутри кластеров или между кластерами. Они помогают выбрать наиболее значимые признаки для кластеризации и исключить менее информативные. Вот некоторые критерии информативности признаков:

Дисперсия: Критерий дисперсии оценивает, насколько переменность значений признака различается между объектами внутри кластера и между разными кластерами. Более высокая дисперсия указывает на более информативный признак.

Коэффициент корреляции: Корреляция между признаками и кластерными метками может быть использована для оценки их информативности. Признаки с более высокой корреляцией с кластерными метками считаются более информативными.

Однородность внутри кластеров: Критерий однородности оценивает, насколько хорошо признак разделяет объекты внутри каждого кластера. Если значения признака однороды внутри кластера и сильно отличаются между кластерами, то признак считается информативным.

Информативность признака при разделении: Этот критерий оценивает, насколько хорошо признак разделяет объекты при разделении на кластеры. Он может быть измерен, например, путем оценки прироста информации или прироста дисперсии при использовании признака для разделения.

Важность признака в моделях машинного обучения: Если вы используете алгоритмы машинного обучения для кластеризации, вы можете использовать важность признаков, оцененную моделью, как критерий информативности. Более важные признаки, определенные моделью, считаются более информативными.

1. Отбор информативных признаков

Отбор информативных признаков в кластерном анализе - это процесс выбора наиболее значимых и релевантных признаков из исходного набора данных для улучшения качества кластеризации. Отбор информативных признаков позволяет устранить шумовые и малозначимые признаки, сосредоточиваясь на наиболее информативных аспектах данных.

Вот некоторые методы отбора информативных признаков в кластерном анализе:

Унивариантный отбор: При этом методе признаки оцениваются независимо друг от друга на основе их статистических характеристик, таких как вариация или дисперсия. Затем выбираются признаки с наибольшими значениями оценки.

Методы основанные на моделях: Здесь используются алгоритмы машинного обучения, которые оценивают важность каждого признака на основе его вклада в модель. Некоторые из таких методов включают случайный лес (Random Forest), градиентный бустинг (Gradient Boosting) и логистическую регрессию (Logistic Regression).

Методы основанные на корреляции: Оценка корреляции между признаками и целевой переменной или между признаками самими по себе может помочь идентифицировать информативные признаки. Признаки с высокой корреляцией могут быть выбраны для дальнейшего анализа, в то время как слабо коррелирующие признаки могут быть исключены.

Методы основанные на вложении данных: Эти методы, такие как метод главных компонент (Principal Component Analysis, PCA) и метод независимых компонент (Independent Component Analysis, ICA), позволяют сжать информацию в исходном пространстве признаков до более компактного пространства, сохраняя при этом наибольшую долю дисперсии или независимости. Таким образом, можно выбрать наиболее информативные компоненты вместо исходных признаков.

1. Графовые методы кластеризации

Графовые методы кластеризации - это подходы к кластеризации, основанные на анализе структуры графа данных. В этих методах данные представляются в виде графа, где объекты данных представляют узлы, а связи или ребра между узлами отражают степень сходства или связи между объектами.

Вот некоторые графовые методы кластеризации:

Метод кратчайших путей (Shortest Path): В этом методе кластеризации вычисляются кратчайшие пути между узлами графа на основе меры сходства или расстояния между объектами. Объекты, расположенные ближе друг к другу на графе, считаются принадлежащими одному кластеру.

Спектральная кластеризация (Spectral Clustering): Этот метод использует спектральное представление графа данных для выделения кластеров. Вначале строится матрица сходства или расстояния между объектами данных, затем вычисляются собственные векторы и собственные значения этой матрицы. Далее, используя эти спектральные компоненты, производится кластеризация данных.

Метод Markov Clustering (MCL): MCL является итеративным методом, который моделирует процесс распространения информации по графу. Он основан на идее, что объекты в одном кластере сильно взаимосвязаны, и между кластерами связи слабы или отсутствуют. MCL использует матрицу сходства или расстояния между объектами и применяет операции инфляции и стохастической матричной экспоненты для обнаружения кластеров.

Методы на основе связности (Connectedness-based methods): В этих методах анализируется структура связности графа данных. Объекты с более плотными связями внутри кластеров и более разреженными связями между кластерами считаются принадлежащими одному кластеру. Примером такого метода является метод агломеративной кластеризации, где объекты последовательно объединяются в кластеры на основе своих связей.

1. Алгоритм выделения связных компонент

Алгоритм выделения связных компонент в кластерном анализе используется для идентификации и группировки объектов данных, которые связаны между собой посредством некоторой формы связности. Вот краткое описание алгоритма выделения связных компонент:

Построение графа данных: Сначала данные представляются в виде графа, где каждый объект данных представляет узел, а связи или ребра отражают степень связности или сходства между объектами. Существуют различные способы определения связей, такие как расстояние или сходство между объектами.

Обход графа: Затем производится обход графа с целью выделения связных компонент. Это может быть выполнено с использованием различных алгоритмов обхода, таких как обход в глубину (Depth-First Search, DFS) или обход в ширину (Breadth-First Search, BFS).

Пометка компонент: Во время обхода каждый объект данных помечается соответствующей связной компонентой. Объекты, которые находятся в одной связной компоненте, считаются принадлежащими одному кластеру или группе.

Итеративный процесс: Если граф содержит несколько связных компонент, алгоритм может быть повторен несколько раз для выделения всех компонент. Каждая итерация начинается с выбора неисследованного объекта данных и продолжается до тех пор, пока не будет достигнута новая связная компонента.

Алгоритм выделения связных компонент является простым и эффективным методом для группировки связанных объектов данных в кластеры. Он находит применение в различных областях, включая обработку изображений, сетевой анализ, анализ текстов и другие.

1. Алгоритм ФОРЭЛ

Алгоритм ФОРЭЛ (Force Directed Relational Algorithm, FORREL) является одним из методов кластерного анализа, используемых для выявления структуры и группировки объектов данных на основе сходства и различия между ними. Основная идея алгоритма ФОРЭЛ заключается в применении силовой модели, где объекты представляются в виде узлов, а связи между объектами - ребрами графа.

Вот краткое описание алгоритма ФОРЭЛ:

Инициализация: Начальные координаты узлов определяются случайным образом или с помощью других методов инициализации. Каждому узлу также назначается начальное значение силы.

Расчет сил: Для каждого узла рассчитывается сила, которая воздействует на него. Силы могут быть определены на основе различных метрик, таких как сходство или расстояние между объектами. Например, объекты, более похожие друг на друга, будут притягиваться, а различные объекты будут отталкиваться.

Обновление координат: Координаты узлов обновляются на основе силы, действующей на них. Обычно используется модель физической системы, где узлы двигаются в пространстве, подвергаясь силам воздействия.

Итерации: Шаги 2 и 3 повторяются до тех пор, пока система не достигнет стабильного состояния или до выполнения определенного критерия остановки, например, достижения максимального числа итераций.

Кластеризация: После достижения стабильного состояния системы, объекты могут быть группированы в кластеры на основе их близости в пространстве. Например, можно использовать методы кластеризации, такие как метод k-средних или иерархическая кластеризация, чтобы присвоить объекты кластерам.

Алгоритм ФОРЭЛ позволяет визуализировать структуру данных и выявлять группы объектов, основываясь на их сходстве и различии. Он находит применение в различных областях, таких как социальные сети, анализ текстов, биоинформатика и другие.

1. Функционалы качества кластеризации

Функционалы качества кластеризации в кластерном анализе используются для оценки результатов кластеризации и определения, насколько хорошо объекты данных были сгруппированы в кластеры. Вот краткое описание некоторых популярных функционалов качества кластеризации:

Внутрикластерное расстояние (Intra-cluster distance): Этот функционал качества измеряет среднее расстояние между объектами внутри одного кластера. Чем меньше значение внутрикластерного расстояния, тем компактнее и более однородным считается кластер.

Межкластерное расстояние (Inter-cluster distance): Этот функционал качества измеряет среднее расстояние между объектами из разных кластеров. Чем больше значение межкластерного расстояния, тем лучше разделены кластеры друг от друга.

Коэффициент силуэта (Silhouette coefficient): Этот функционал качества комбинирует внутрикластерное и межкластерное расстояния для оценки качества кластеризации. Он вычисляет средний коэффициент силуэта для каждого объекта, который показывает, насколько хорошо он принадлежит своему кластеру по сравнению с другими кластерами. Значение коэффициента силуэта может быть от -1 до 1, где ближе к 1 указывает на хорошую кластеризацию, а ближе к -1 указывает на плохую кластеризацию.

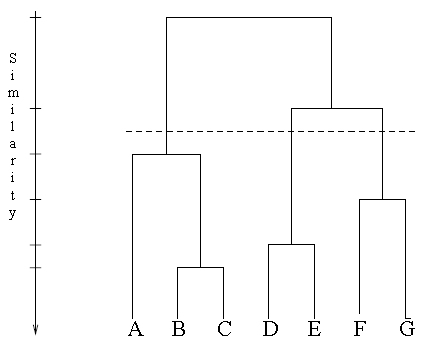
Индекс Данна (Davies-Bouldin index): Этот функционал качества вычисляет межкластерные расстояния и внутрикластерные расстояния для всех пар кластеров и использует их для определения разделения и компактности кластеров. Чем меньше значение индекса Данна, тем лучше кластеризация.

Показатель Rand (Rand index): Этот функционал качества сравнивает схожесть кластеризации с истинными метками классов (если они доступны). Он вычисляет соответствие между парами объектов в кластерах и истинными метками классов, а затем вычисляет показатель Rand, который может быть от 0 до 1, где 1 указывает на полное соответствие.

Функционалы качества кластеризации помогают оценить результаты кластеризации и выбрать наилучший набор параметров или алгоритм для данного набора данных. Они помогают кластеризационным алгоритмам быть более объективными и эффективными в выделении скрытых структур в данных.

25. Иерархическая кластеризация (таксономия)  
 Иерархическая кластеризация — множество алгоритмов кластеризации, направленных на создание иерархии разбиений исходного множества объектов.

Иерархические алгоритмы кластеризации часто называют алгоритмами таксономии. Для визуального представления результатов кластеризации используется дендрограмма — дерево, построенное по матрице мер близости между кластерами. В узлах дерева находятся подмножества объектов из обучающей выборки. При этом на каждом ярусе дерева множество объектов из всех узлов составляет исходное множество объектов. Объединение узлов между ярусами соответствует слиянию двух кластеров. При этом длина ребра соответствует расстоянию между кластерами.



26 Агломеративная иерархическая кластеризация

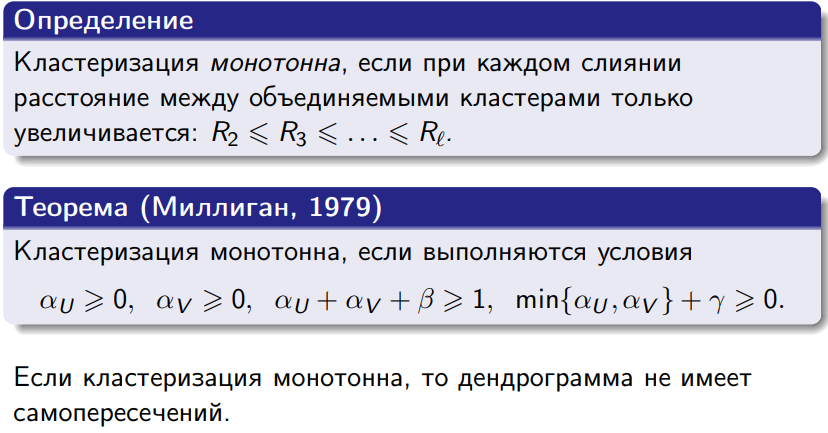
1. Процесс агломеративной кластеризации начинается с определения меры расстояния или близости между парами объектов. Обычно используются метрические методы, такие как евклидово расстояние или корреляционное расстояние. Затем каждый объект рассматривается как отдельный кластер.
2. На каждом шаге агломеративной кластеризации два ближайших кластера объединяются в один новый кластер. Это продолжается до тех пор, пока все объекты не объединятся в один кластер или пока не будет достигнут заданный критерий остановки, например, определенное количество кластеров.
3. В ходе процесса строится дендрограмма, представляющая собой дерево, где листья соответствуют отдельным объектам, а внутренние узлы - объединенным кластерам (см. скрин выше). Расстояние между объединяемыми кластерами определяется на основе выбранной метрики расстояния.

27 Дендрограмма и свойство монотонности

Дендрограмма является графическим представлением результатов агломеративной иерархической кластеризации. Она представляет собой дерево, где листья соответствуют отдельным объектам, а внутренние узлы - объединенным кластерам.

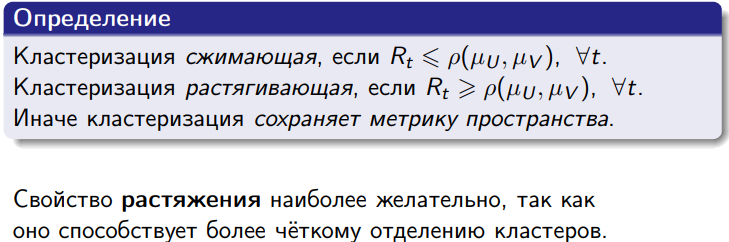
Свойство монотонности дендрограммы означает, что изменение расстояния или близости между кластерами вниз по дереву всегда увеличивается или остается неизменным. Другими словами, чем ближе кластеры в дереве, тем меньше расстояние или больше близость между ними.

Монотонность позволяет определить, насколько удалены или близки друг к другу кластеры. Например, если дендрограмма является монотонной, то кластеры, находящиеся ближе к корню дерева, обычно являются более похожими или близкими друг к другу по сравнению с кластерами, находящимися на более низких уровнях дерева.



28 Свойства сжатия, растяжения и редуктивности

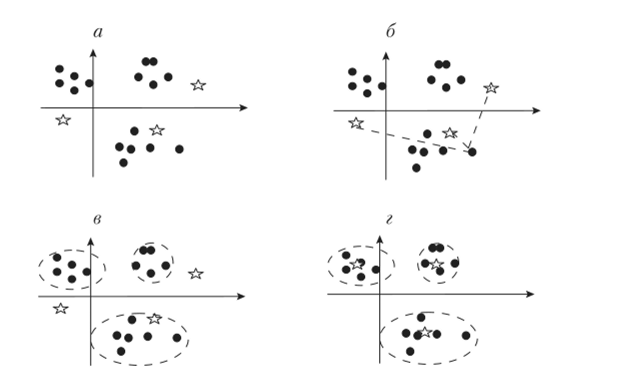
1. Свойство сжатия (compression): Кластеры считаются сжимаемыми, если при объединении двух кластеров расстояние между ними уменьшается или остается неизменным. Свойство сжатия указывает на то, что объекты внутри кластера более похожи друг на друга.
2. Свойство растяжения (stretching): Кластеры считаются растяжимыми, если при объединении двух кластеров расстояние между ними увеличивается. То есть, объединенный кластер более разреженный или растянутый по сравнению с исходными кластерами. Свойство растяжения указывает на то, что объекты внутри кластеров имеют различные характеристики или структуру.
3. Свойство редуктивности (reducibility): Кластеры считаются редуцируемыми, если при разделении одного кластера на два близких кластера расстояние между ними уменьшается или остается неизменным. То есть, разделенные кластеры более компактные или плотные по сравнению с исходным кластером. Свойство редуктивности указывает на то, что объекты внутри кластера схожи друг с другом и можно дальше уточнять структуру кластеров.



29 Статистические методы кластеризации

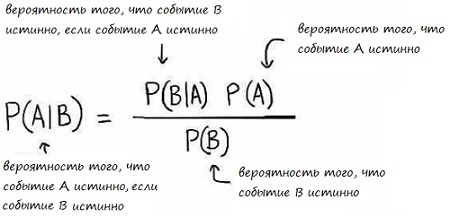
Метод K-средних (K-means) является одним из наиболее распространенных и простых методов кластеризации. Он используется для разделения набора данных на заранее заданное количество кластеров (K).

1. Инициализация: Задается количество кластеров K, которое требуется получить. Также необходимо выбрать начальные центроиды для каждого кластера. Центроиды обычно инициализируются случайным образом, например, выбирая случайные точки из набора данных.
2. Присвоение объектов к кластерам: Для каждого объекта в наборе данных вычисляется расстояние до каждого центроида кластера. Объект присваивается к кластеру с наименьшим расстоянием до его центроида.
3. Пересчет центроидов: После присвоения объектов к кластерам пересчитываются центроиды для каждого кластера. Центроид обновляется как среднее значение всех объектов, принадлежащих к данному кластеру. Это делается для каждого кластера.
4. Итерационный процесс: Шаги 2 и 3 повторяются до достижения сходимости. В каждой итерации объекты пересчитываются, присваиваются к новым кластерам на основе обновленных центроидов, а затем центроиды пересчитываются на основе новых присвоений. Этот процесс продолжается до тех пор, пока кластеры и центроиды не стабилизируются.



EM-алгоритм (Expectation-Maximization) является итеративным методом для оценки параметров статистической модели.

1. Инициализация: Начальные значения параметров модели (например, параметры смеси распределений) выбираются случайным образом или на основе предварительной информации.
2. E-шаг (Expectation step): В этом шаге вычисляются ожидания скрытых переменных для каждого объекта в наборе данных. Ожидания представляют вероятности принадлежности объектов к каждому из кластеров или компонент смеси. Они вычисляются с использованием текущих параметров модели и применением формулы условной вероятности.

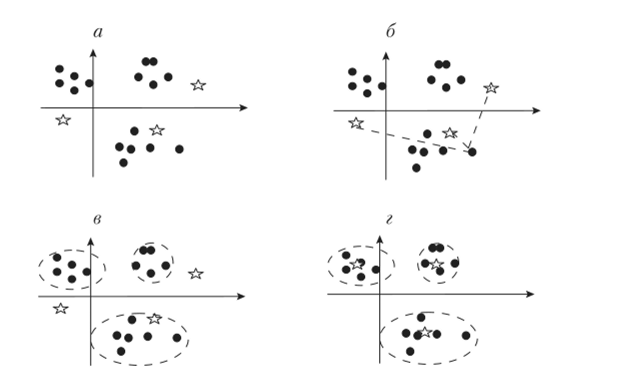


1. M-шаг (Maximization step): В этом шаге обновляются параметры модели, чтобы максимизировать ожидание правдоподобия (expectation likelihood), вычисленное на предыдущем шаге.
2. Итерационный процесс: Шаги E и M повторяются до сходимости. На каждой итерации E-шага вычисляются новые ожидания, а затем на M-шаге обновляются параметры модели. Этот процесс продолжается до достижения определенного критерия остановки, такого как максимальное количество итераций или незначительное изменение параметров модели.

30 Метод k-средних

Метод K-средних (K-means) является одним из наиболее распространенных и простых методов кластеризации. Он используется для разделения набора данных на заранее заданное количество кластеров (K).

1. Инициализация: Задается количество кластеров K, которое требуется получить. Также необходимо выбрать начальные центроиды для каждого кластера. Центроиды обычно инициализируются случайным образом, например, выбирая случайные точки из набора данных.
2. Присвоение объектов к кластерам: Для каждого объекта в наборе данных вычисляется расстояние до каждого центроида кластера. Объект присваивается к кластеру с наименьшим расстоянием до его центроида.
3. Пересчет центроидов: После присвоения объектов к кластерам пересчитываются центроиды для каждого кластера. Центроид обновляется как среднее значение всех объектов, принадлежащих к данному кластеру. Это делается для каждого кластера.
4. Итерационный процесс: Шаги 2 и 3 повторяются до достижения сходимости. В каждой итерации объекты пересчитываются, присваиваются к новым кластерам на основе обновленных центроидов, а затем центроиды пересчитываются на основе новых присвоений. Этот процесс продолжается до тех пор, пока кластеры и центроиды не стабилизируются.



31 Метод главных компонент

<https://vk.com/@itresume-metod-glavnyh-komponent-poshagovaya-instrukciya>

Идея PCA (Метод главных компонент) проста - уменьшить количество переменных в наборе данных, сохранив при этом как можно больше информации.

1. ШАГ 1: СТАНДАРТИЗАЦИЯ

Цель первого шага - стандартизировать диапазоны исходных переменных, чтобы избавиться от большого разброса значений.

Причина, по которой так важно выполнить стандартизацию до применения метода: метод главных компонент очень чувствителен к значениям исходных переменных. То есть, если есть большие различия между значениями переменных, то переменные с большими диапазонами будут преобладать (например, x1, которая находится в диапазоне от 0 до 100, будет преобладать над x2, которая находится в диапазоне от 0 до 1). Это приведет к необъективным результатам. А преобразование данных к одному масштабу может предотвратить эту проблему.

2)ШАГ 2: РАСЧЕТ МАТРИЦЫ КОВАРИАЦИИ

Цель второго шага - увидеть, есть ли какая-либо связь между переменными во входных данных. Иногда переменные сильно коррелированы (имеют зависимость). А это значит, что данные содержат избыточную информацию (предположительно это так, но бывают разные случаи - мало ли какая природа у коррелированных величин).

Итак, чтобы идентифицировать взаимосвязи, нужно вычислить ковариационную матрицу. Ковариационная матрица представляет собой симметричную матрицу размера p × p (где p - количество измерений), элементами которой являются ковариации всех возможных пар исходных переменных. Например, для 3-мерного набора данных с 3 переменными x, y и z ковариационная матрица представляет собой матрицу 3 × 3, состоящую из следующих элементов:

Что матрица ковариации говорит о связях между переменными?

На самом деле значение имеет знак ковариации:

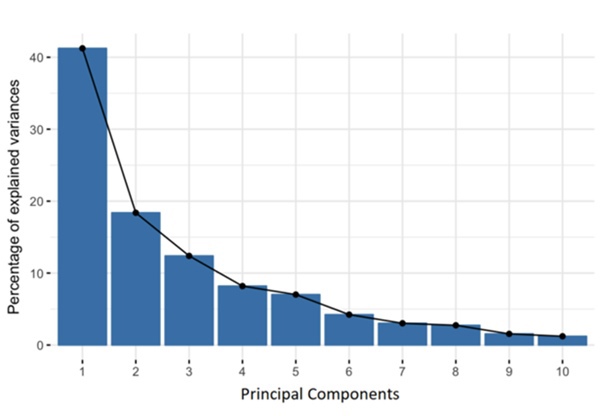
* если положительный, то две переменные увеличиваются или уменьшаются вместе (коррелированы)
* если отрицательно, то одно увеличивается, когда другое уменьшается (обратно коррелированы)

3)ШАГ 3. ВЫЧИСЛЕНИЕ СОБСТВЕННЫХ ВЕКТОРОВ И СОБСТВЕННЫХ ЗНАЧЕНИЙ КОВАРИАЦИОННОЙ МАТРИЦЫ

Собственные векторы и собственные значения - это понятия линейной алгебры. Они нам нужны, чтобы определить главные компоненты. Прежде чем перейти к объяснению этих концепций, давайте сначала поймем, что мы подразумеваем под главными компонентами.

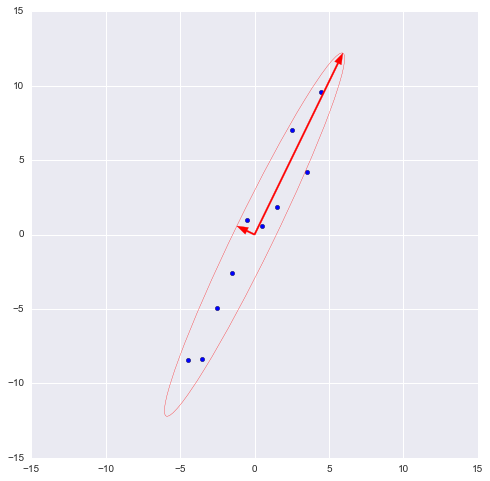
Главные компоненты - это новые переменные, с помощью которых мы и будем описывать наши объекты.

Идея состоит в том, что 10-мерные данные дают 10 главных компонент, но PCA пытается поместить максимум возможной информации в первый компонент, затем максимум оставшейся информации во второй и так далее, пока не получится следующая диаграмма (похоже на плотность геометрического распределения):



Здесь важно понимать, что полученные главные компоненты менее интерпретируемы (по сравнению с исходными данными), а их значения не описывают реальное положение дел. Т.е. мы как бы описываем наш исходный объект в других терминах - переходим к другим переменным. Если охват груди был 90 сантиметров, а после преобразования стал 0.3 - это не значит, что грудь уменьшилась. Просто как бы поменялась “система измерения”. Если вернемся в исходную систему, то размер станет прежним.

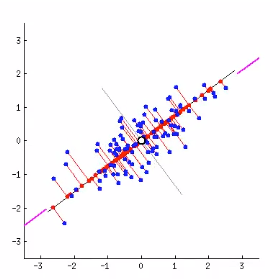
В библиотеке numpy реализована функция numpy.linalg.eig(X), где X – квадратная матрица. Она возвращает 2 массива – массив айгензначений и массив айгенвекторов (векторы-столбцы). И векторы нормированы — их длина равна 1. Как раз то, что надо. Эти 2 вектора задают новый базис для выборки, такой что его оси совпадают с полуосями аппроксимирующего эллипса нашей выборки.



На этом графике мы апроксимировали нашу выборку эллипсом с радиусами в 2 сигмы (т.е. он должен содержать в себе 95% всех наблюдений – что в принципе мы здесь и наблюдаем). Я инвертировал больший вектор (функция eig(X) направляла его в обратную сторону) – нам важно направление, а не ориентация вектора.

Как PCA конструирует главные компоненты

Поскольку количество главных компонент равно числу изначальных переменных, они строятся таким образом, что первый главный компонент учитывает максимально возможную дисперсию в наборе данных. Например, предположим, что диаграмма рассеяния нашего набора данных выглядит так, как показано ниже (синие точки). Можем ли мы угадать первый главный компонент? Да, это фиолетовая линия, вдоль которой синие точки сильней всего раскиданы. Или, говоря математически, это линия максимизирует дисперсию.





Второй главный компонент рассчитывается таким же образом, при условии, что он не коррелирован (то есть перпендикулярен) первому главному компоненту и учитывает следующую по величине дисперсию. Это продолжается до тех пор, пока не будет вычислено общее количество p главных компонент, равное исходному количеству переменных.

4)ШАГ 4: МАТРИЦА ГЛАВНЫХ КОМПОНЕНТ

На предыдущем шаге мы нашли главные компоненты в порядке значимости. На этом этапе мы решаем: оставить все эти компоненты или отбросить те, которые имеют меньшую значимость и сформировать матрицу векторов признаков.

Итак, составляем матрицу главных компонент - матрицу, по столбцам которой расположены собственные векторы, которые мы решили оставить. Это первый шаг на пути к снижению размерности, потому что, если мы решим оставить только n собственных векторов (компонентов) из p, то окончательный набор данных будет иметь размерность n (раньше мячик на ковре описывался 3 переменными, а будет 1; портной сделал 20 замеров, а использовал при пошиве только 2).

Пример:

Продолжая пример из предыдущего шага, мы можем сформировать вектор признаков с двумя собственными векторами v1 и v2:

Метод главных компонент. Пошаговая инструкция., изображение №6

Метод главных компонент. Пошаговая инструкция., изображение №6

Или отбросить собственный вектор v2, который имеет меньшую значимость (всего 4% против 96%), и сформировать вектор признаков только с v1:

Метод главных компонент. Пошаговая инструкция., изображение №7

Отказ от собственного вектора v2 уменьшит размерность на 1 и, следовательно, вызовет потерю информации в окончательном наборе данных. Но учитывая, что v2 несет только 4% информации, потеря не будет существенной, и у нас останется 96% информации, которая “переносится” в направлении v1. Мы достигнем своей цели - информации потеряли не много, а размерность уменьшили.

Итак, как мы видим в примере, Вам решать, сохранять все компоненты или отбрасывать менее важные. Если Вы не стремитесь уменьшить размерность, не нужно исключать менее значимые компоненты. Хотя тогда непонятно, зачем вообще применять метод главных компонент :)

5)ПОСЛЕДНИЙ ШАГ: ПРЕОБРАЗОВАНИЕ ДАННЫХ ПО ОСЯМ ОСНОВНЫХ КОМПОНЕНТ

На предыдущих шагах, помимо стандартизации, мы не вносили никаких изменений в данные. Входной набор данных всегда оставался в терминах исходных переменных, а мы мудрили с матрицей ковариации и айгенпарами.

На этом последнем шаге цель состоит в том, чтобы использовать полученные главные компоненты, для замены данных с исходных переменных на новые. Это можно сделать, умножив транспонированный исходный набор данных на транспонированную матрицу главных компонент.

Метод главных компонент. Пошаговая инструкция., изображение №8

Метод главных компонент. Пошаговая инструкция., изображение №8

После такого преобразования Вы получите матрицу k\*n, где k - число измерений (например, портной замерил k человек или на ковре лежит k мячиков), а n - количество главных компонент (теперь портной использует только 2 замера, а мячик описывается одной переменной).

32 Практическое применение кластеризации

1. Маркетинг: Кластеризация может использоваться для разделения клиентов по их поведению, предпочтениям или демографическим характеристикам. Это позволяет компаниям создавать более точные маркетинговые стратегии, персонализированные предложения и улучшать удовлетворенность клиентов.
2. Медицина и биология: В медицине кластеризация может помочь в идентификации подгрупп пациентов с похожими характеристиками или рисковыми факторами
3. Финансы и банковское дело: Кластеризация может быть применена для анализа финансовых данных, выявления групп сходных инвестиционных профилей или рисков, а также для обнаружения аномалий и мошенничества в финансовых транзакциях.
4. Географический анализ: Кластеризация может использоваться для группировки географических областей с похожими характеристиками, такими как плотность населения, социоэкономический статус или экологические параметры. Это может помочь в планировании развития городов, определении приоритетных районов для инвестиций или анализе влияния географических факторов на социальные явления.

33 Группировка и распознавание объектов

Группировка объектов: Кластеризация может использоваться для группировки объектов на основе их сходства или признаков. Например, в компьютерном зрении кластеризация может быть использована для группировки изображений на основе их содержания, цветовых характеристик или текстурных признаков. Это позволяет создать коллекции изображений схожих объектов или сцен.

Классификация объектов: Кластеризация может использоваться в качестве этапа предварительной классификации объектов. На основе кластеризации объекты делятся на группы, а затем каждая группа классифицируется отдельно с использованием специализированных алгоритмов классификации. Например, в медицине кластеризация может быть использована для грубой классификации образцов тканей или генов, а затем для более детального анализа каждой группы используются специализированные классификаторы.

34 Сегментация изображений

Сегментация — это процесс разделения цифрового изображения на несколько сегментов (множеств пикселей).

Очень эффективным с точки зрения вычислительных ресурсов является использование для сегментации методов кластерного анализа. Суть кластеризации состоит в том, что все исходные объекты (в данном случае пиксели ) разбиваются на несколько не пересекающихся групп таким образом, чтобы объекты, попавшие в одну группу, имели сходные характеристики, в то время как у объектов из разных групп эти характеристики должны значительно отличаться. Полученные группы называются кластерами. Исходными значениями в простейшем способе для кластеризации являются координаты пикселя (x, y), в более сложных случаях, например для полутоновых изображений, используется трехмерный вектор (x, y, I(x, y) ), где I(x, y) — градации серого

Метод сегментации, основанный на кластеризации

k-средних — это итеративный метод, который используется, чтобы разделить изображение на K кластеров. Базовый алгоритм приведён ниже:

1. Выбрать K центров кластеров, случайно или на основании некоторой эвристики;
2. Поместить каждый пиксель изображения в кластер, центр которого ближе всего к этому пикселю;
3. Заново вычислить центры кластеров, усредняя все пиксели в кластере;
4. Повторять шаги 2 и 3 до сходимости (например, когда пиксели будут оставаться в том же кластере).

Здесь в качестве расстояния обычно берётся сумма квадратов или абсолютных значений разностей между пикселем и центром кластера. Разность обычно основана на цвете, яркости, текстуре и местоположении пикселя, или на взвешенной сумме этих факторов. K может быть выбрано вручную, случайно или эвристически.

35 Машинное обучение

Машинное обучение - это область искусственного интеллекта, которая изучает методы и алгоритмы, позволяющие компьютерам автоматически обучаться и делать предсказания или принимать решения на основе данных, без явного программирования.

Существуют различные подходы и методы в машинном обучении, включая:

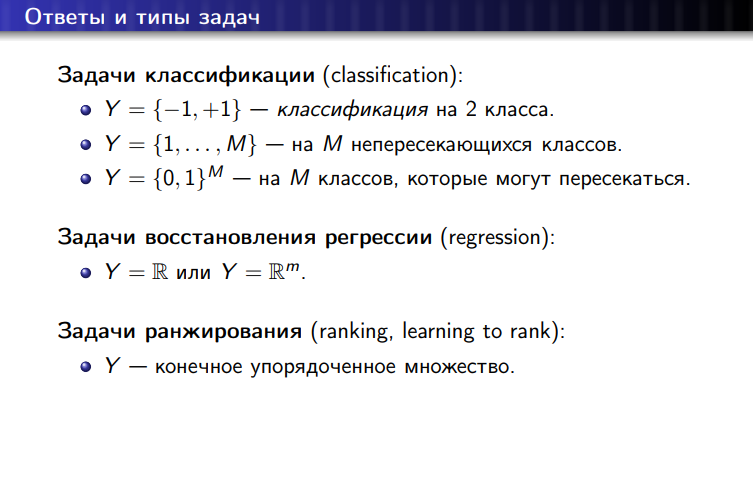
1. Обучение с учителем (Supervised learning): В этом типе задач у обучающих данных имеются явно указанные метки или выходные значения, которые модель должна научиться предсказывать. Например, задачи классификации, регрессии и прогнозирования относятся к обучению с учителем.
2. Обучение без учителя (Unsupervised learning): В обучении без учителя обучающие данные не имеют явных меток, и модель должна самостоятельно находить закономерности или группировать данные без какой-либо предварительной информации. Примеры таких задач включают кластеризацию, снижение размерности и ассоциативное правило обнаружение.

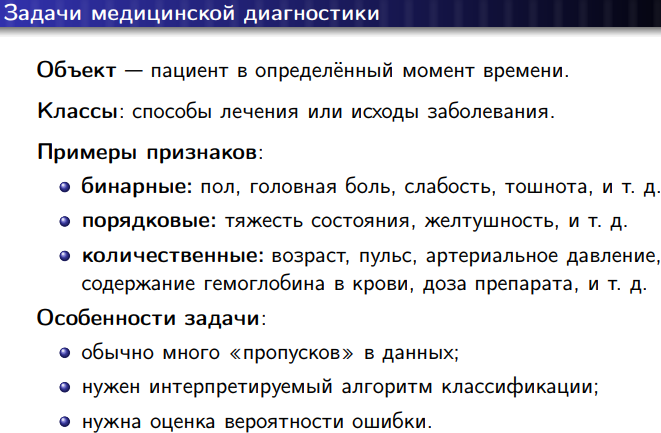
Задача машинного обучения состоит в том, чтобы на основе обучающих данных построить модель, которая способна делать предсказания или принимать решения для новых данных, которые не были использованы в процессе обучения.

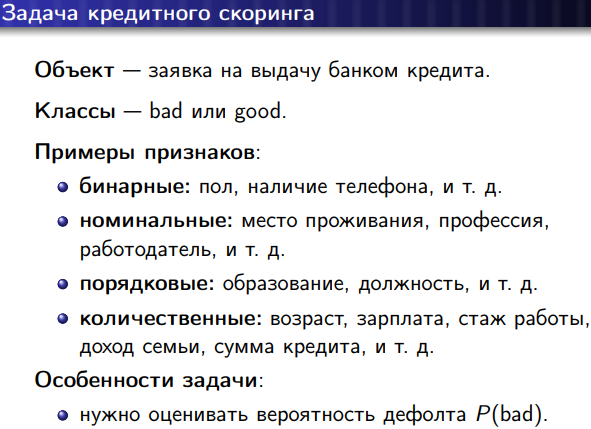
36 Основные понятия и определения машинного обучения

1. Объект: В контексте машинного обучения объект представляет собой элемент данных, на котором модель будет обучаться или делать предсказания. Объект может быть представлен различными форматами, такими как текстовые документы, изображения, звуковые записи или числовые значения.
2. Ответ: Ответ (также называемый меткой или целевой переменной) - это значение или классификация, которую модель пытается предсказать для каждого объекта. Например, в задаче классификации объекты могут быть разделены на определенные категории, а в задаче регрессии модель предсказывает непрерывные значения.
3. Признак: Признаки (также известные как атрибуты или переменные) представляют собой характеристики или измерения, которые описывают объекты. Каждый объект обладает набором признаков, которые могут быть числовыми, категориальными или бинарными значениями. Например, признаками для изображения могут быть пиксели или цвета, а для текстового документа - частота слов или длина предложений.
4. Алгоритм: Алгоритм представляет собой процедуру или набор инструкций, которые модель использует для обучения на данных или делания предсказаний. Алгоритмы машинного обучения могут быть различными, включая линейную регрессию, решающие деревья, метод опорных векторов, нейронные сети и многие другие.
5. Модель алгоритмов: Модель алгоритмов представляет собой конкретную реализацию выбранного алгоритма машинного обучения. Модель является результатом обучения на обучающих данных и содержит параметры, которые определяют веса или связи между признаками и ответами. Модель используется для предсказания ответов для новых данных.
6. Метод обучения: Метод обучения определяет, как модель будет обучаться на обучающих данных. В зависимости от задачи и доступных данных, методы обучения могут включать обучение с учителем (с метками ответов), обучение без учителя (без меток ответов)
7. Валидация: Валидация - это процесс оценки производительности модели на независимом валидационном наборе данных. Он помогает оценить, насколько хорошо модель обобщает и предсказывает ответы на новых данных. Валидационный набор данных используется для настройки гиперпараметров модели и принятия решений о ее улучшении.
8. Переобучение: Переобучение происходит, когда модель слишком точно подстраивается под обучающие данные и теряет способность обобщать на новые данные. В результате модель становится слишком сложной и слишком хорошо запоминает шум или случайности в данных, что может приводить к плохим предсказаниям на новых данных.

37 Примеры прикладных задач машинного обучения









38 Основные модели машинного обучения

1. Линейная регрессия: Модель, которая стремится установить линейную зависимость между входными признаками и выходным значением. Она используется для задач регрессии, где предсказывается непрерывная переменная.
2. Логистическая регрессия: Эта модель используется для задач классификации и предсказывает вероятность принадлежности объекта к определенному классу. Она основана на логистической функции и может быть использована для бинарной классификации или многоклассовой классификации.
3. Дерево решений: Это модели, которые строятся в виде дерева решений, где каждый узел представляет тест на признаке, а каждая ветвь - возможное значение этого признака.
4. Случайный лес: Случайный лес является ансамблем решающих деревьев, где несколько деревьев комбинируются для улучшения предсказательной способности модели. Каждое дерево обучается на случайном подмножестве данных и признаков, и итоговое предсказание получается путем агрегации предсказаний всех деревьев. Принцип работы случайного леса состоит из следующих шагов:
   * + Построение деревьев: Случайный лес строит множество решающих деревьев, каждое из которых обучается на различном подмножестве обучающих данных. Это называется "бутстрап-выборка" или "случайная выборка с возвращением". Кроме того, для каждого узла дерева случайный лес выбирает только подмножество признаков, на которых будет осуществляться разделение.
     + Прогнозирование: Когда все деревья построены, случайный лес делает прогнозы путем агрегации (среднего или голосования) прогнозов всех деревьев. Для задач классификации прогнозом может быть наиболее часто встречающийся класс, а для задач регрессии - среднее значение предсказаний.

39 Ансамблевые модели машинного обучения

Ансамблевые модели машинного обучения представляют собой комбинацию нескольких базовых моделей для решения задачи прогнозирования. Они объединяют предсказания отдельных моделей, чтобы получить более точные и стабильные результаты. Принцип работы ансамблевых моделей основан на концепции "мудрости толпы", где множество слабых моделей объединяется вместе для создания сильной модели.

1. Случайный лес: Случайный лес является ансамблем решающих деревьев, где несколько деревьев комбинируются для улучшения предсказательной способности модели. Каждое дерево обучается на случайном подмножестве данных и признаков, и итоговое предсказание получается путем агрегации предсказаний всех деревьев. Принцип работы случайного леса состоит из следующих шагов:
   * 1. Построение деревьев: Случайный лес строит множество решающих деревьев, каждое из которых обучается на различном подмножестве обучающих данных. Это называется "бутстрап-выборка" или "случайная выборка с возвращением". Кроме того, для каждого узла дерева случайный лес выбирает только подмножество признаков, на которых будет осуществляться разделение.
     2. Прогнозирование: Когда все деревья построены, случайный лес делает прогнозы путем агрегации (среднего или голосования) прогнозов всех деревьев. Для задач классификации прогнозом может быть наиболее часто встречающийся класс, а для задач регрессии - среднее значение предсказаний.
2. Бустинг AdaBoost (Adaptive Boosting): Алгоритм AdaBoost также использует итеративный процесс для создания ансамбля моделей. Он назначает больший вес объектам, которые были неправильно классифицированы предыдущими моделями, и настраивает новую модель, чтобы она справлялась с этими сложными случаями. В итоге, модели объединяются с учетом весов объектов.

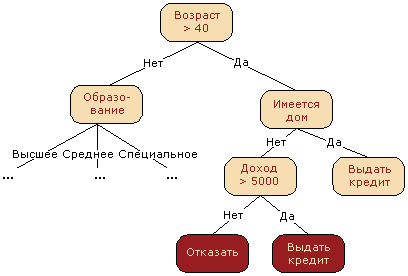
40 Деревья решений и случайный лес

Дерево решений - это графическая структура, используемая в машинном обучении для принятия решений или выполнения прогнозов. Оно представляет собой древовидную структуру, состоящую из узлов и ребер, где каждый узел представляет тест на определенном признаке, а каждое ребро указывает на возможное значение этого признака. Корневой узел представляет начальное разделение, а листья (терминальные узлы) представляют конечные прогнозы или классы.

Процесс построения дерева решений основан на алгоритме рекурсивного разделения данных на основе признаков. Вот основные шаги:

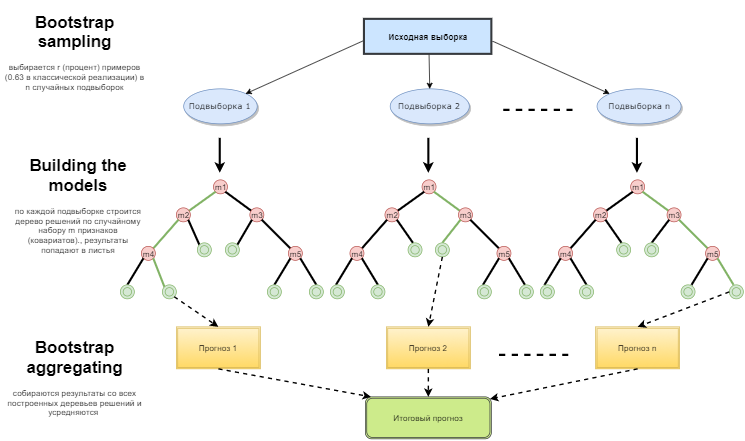
1. Выбор признака: Из множества доступных признаков выбирается наиболее информативный признак для разделения данных на две или более группы. Это может быть выполнено, например, с использованием меры информативности, такой как энтропия или прирост информации.
2. Разделение данных: Данные разделяются на основе значения выбранного признака. Каждая ветвь дерева представляет одно из возможных значений этого признака.
3. Рекурсивное разделение: Шаги 1 и 2 повторяются для каждого подмножества данных, полученного после разделения. Этот процесс выполняется до достижения критерия останова, например, достижения определенной глубины дерева или недостаточного числа объектов в узле.

Прогнозирование: Когда дерево решений полностью построено, прогнозы делаются путем классификации (в случае задачи классификации) или вычисления среднего значения (в случае задачи регрессии) в соответствующем листе дерева.



Случайный лес: Случайный лес является ансамблем решающих деревьев, где несколько деревьев комбинируются для улучшения предсказательной способности модели. Каждое дерево обучается на случайном подмножестве данных и признаков, и итоговое предсказание получается путем агрегации предсказаний всех деревьев. Принцип работы случайного леса состоит из следующих шагов:

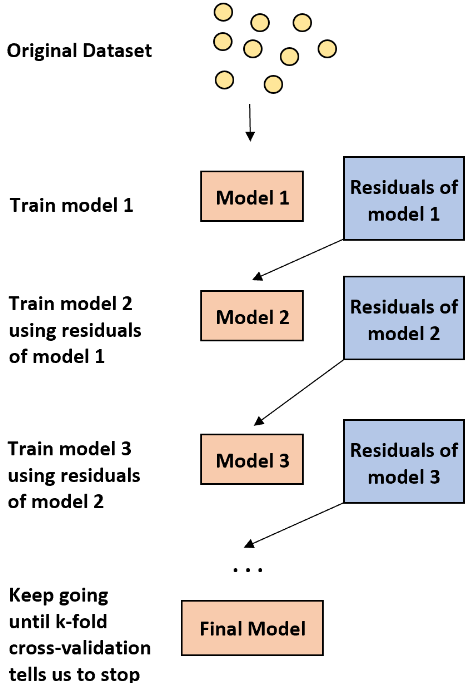
* + 1. Построение деревьев: Случайный лес строит множество решающих деревьев, каждое из которых обучается на различном подмножестве обучающих данных. Это называется "бутстрап-выборка" или "случайная выборка с возвращением". Кроме того, для каждого узла дерева случайный лес выбирает только подмножество признаков, на которых будет осуществляться разделение.
    2. Прогнозирование: Когда все деревья построены, случайный лес делает прогнозы путем агрегации (среднего или голосования) прогнозов всех деревьев. Для задач классификации прогнозом может быть наиболее часто встречающийся класс, а для задач регрессии - среднее значение предсказаний.



41 Бустинг

Бустинг (Boosting) - это алгоритм машинного обучения, который используется для создания ансамбля моделей, объединяя их последовательно с целью улучшения предсказательной способности. Основная идея бустинга заключается в создании модели, которая является комбинацией нескольких слабых моделей (называемых базовыми моделями), которые в совокупности способны решать сложные задачи.

Процесс бустинга обычно выполняется в несколько итераций, и каждая итерация настраивает новую модель с учетом ошибок, допущенных предыдущими моделями. Ошибки модели из предыдущей итерации уделяются большее внимание, таким образом, новая модель фокусируется на тех образцах, которые были сложны для предыдущих моделей. Это делает бустинг способным к обучению на "сложных" данных и повышает его способность к обобщению.



42 Построение решающих правил

Решающее правило представляет собой логическое выражение, которое определяет, какой класс или категория присваивается объекту на основе его признаков.

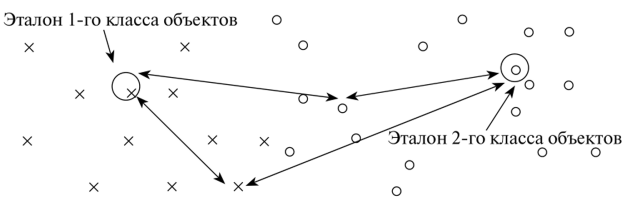
Вот несколько методов построения решающих правил:

1. Деревья решений: Деревья решений являются мощным инструментом для построения решающих правил. Каждый узел дерева представляет тест на определенном признаке, а каждое ребро указывает на возможное значение этого признака. Решающие правила получаются путем просмотра пути от корневого узла до листового узла в дереве.
2. Метод ближайших соседей: В методе ближайших соседей решающие правила строятся на основе близости между объектами. Классификация нового объекта основывается на классах его ближайших соседей.

43 Методы построения эталонов

Эталон представляет собой представителя класса или кластера и используется для сравнения с другими объектами с целью определения их принадлежности к определенному классу или кластеру. Вот несколько методов построения эталонов:

K-Means: Метод K-средних может быть использован для построения эталонов в задаче кластеризации. Алгоритм начинает с случайного выбора K эталонов и затем итеративно обновляет их позиции, минимизируя сумму квадратов расстояний между объектами и их ближайшими эталонами. В результате получаются K эталонов, которые являются центральными точками для каждого кластера.



Распознавания осуществляются следующим образом: на вход системы поступает объект (набор признаков x\*), принадлежность которого к тому или иному образу системы неизвестно. От этого объекта измеряются расстояния до эталона всех образов, и считается принадлежащим тому образу расстояние, до которого минимально.

44 Метод ближайшего соседа

Метод он основан на принципе ближайшего соседа: объекту присваивается тот класс или значение, которое имеет ближайший к нему объект из обучающей выборки.

В методе ближайшего соседа предполагается, что близкие объекты в пространстве признаков обладают схожими свойствами и, следовательно, принадлежат к одному классу или имеют похожие значения целевой переменной.

Пример: Если системе предъявлен нераспознаваемый объект Х\*, то она относит этот объект тому образу, чей «представитель» оказался ближе всех к Х\*. Это правило называется «правилом ближайшего соседа». Правило состоит в том, что строится гиперсфера объемом V с центром Х\* (или окружность). Распознавание осуществляется по большому числу «представителей» какого-либо образа оказавшегося внутри гиперсферы.

Метод ближайшего соседа имеет недостаток, так как необходимо хранить всю обучающую выборку. Другим недостатком является большая величина ошибки, при x\* близкой к границе X, то есть на границе сферы.

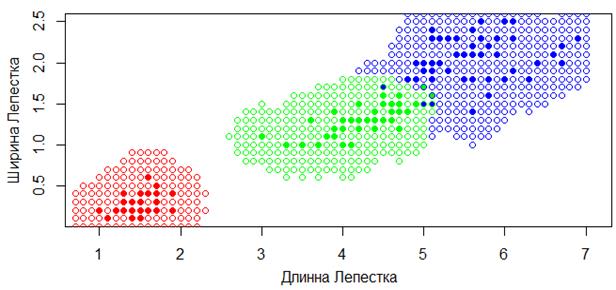
45 Метод Парзеновского окна

Cпособ задать веса соседям -- определить wi как функцию от расстояния с(u, xi,u), а не от ранга соседа i. Введём функцию ядра K(z), невозрастающую на [0,?), и рассмотрим алгоритм

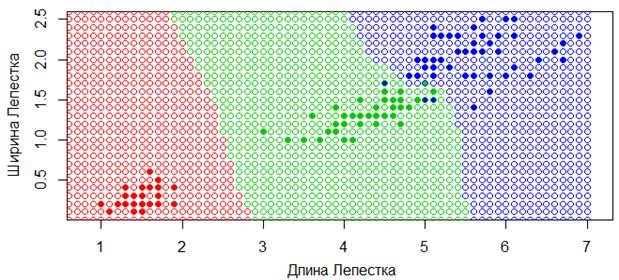
https://studbooks.net/imag_/43/228705/image005.png

Параметр h называется шириной окна и играет примерно ту же роль, что и число соседей k. «Окно» -- это сферическая окрестность объекта u радиуса h, при попадании в которую обучающего объекта xi объект u «притягивается» к классу yi. Мы пришли к этому алгоритму чисто эвристическим путём, однако он имеет более строгое обоснование в байесовской теории классификации, и тесно связан с непараметрическим оцениванием плотности распределения по Парзену-Розенблатту.

Фактически, алгоритм работает по похожему принципу, что и метод k ближайших соседей, только рассматривает всех соседей на расстоянии h и с учетом расстояния до них. Например, вот так выглядит карта классификации для квадратического парзеновского окна с параметром h=0,4 в пространстве двух признаков для цветов ирисов:



В сравнении та же карта классификации для метода k ближайших соседей при k=6, выглядит следующим образом:



Видите, как заметно отличаются результаты моделей при этих двух подходах. И, наверное, более логичные результаты, мы получаем именно при использовании парзеновских окон, так как области принятия решений становятся ограниченными, как и должно быть в большинстве задач классификации.

46 Обобщённый метрический классификатор

Обобщенный метрический классификатор (Generalized Metric Classifier) - это алгоритм машинного обучения, который использует метрические методы для классификации данных. Он основан на идее, что объекты из одного класса имеют более близкие между собой расстояния, чем объекты из разных классов.

Обобщенный метрический классификатор работает следующим образом:

1. По заданному набору обучающих данных классификатор строит метрику (функцию расстояния) между объектами.
2. Для классификации новых данных классификатор вычисляет расстояние между новым объектом и объектами обучающего набора.
3. На основе вычисленных расстояний классификатор присваивает новому объекту метку класса, основываясь на близости к объектам обучающего набора.

Основной пример обобщенного метрического классификатора - метод k-ближайших соседей. В этом методе для классификации нового объекта вычисляется расстояние до k ближайших соседей в обучающем наборе

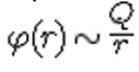
47 Понятие отступа объекта

Вариант 1: Отступ (англ. margin) — характеристика, оценивающая, насколько объект "погружён" в свой класс, насколько типичным представителем класса он является. Чем меньше значение отступа Mi, тем ближе объект x⃗ i подходит к границе классов и тем выше становится вероятность ошибки. Отступ Mi отрицателен тогда и только тогда, когда алгоритм a(x) допускает ошибку на объекте x⃗ i.

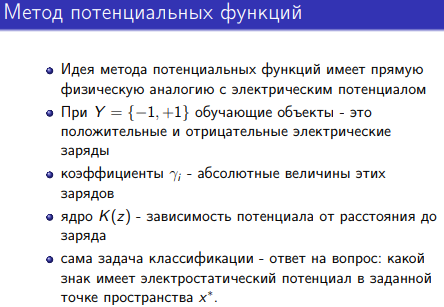
Вариант 2: Отступ (для классификатора) — характеристика, оценивающая то, насколько объект "погружён" в свой класс, насколько эталонным представителем он является. Чем меньше значение отступа, тем ближе объект находится к границе класса, соответственно тем выше вероятность ошибочного прогноза.

48 Метод потенциальных функций

Общая идея метода иллюстрируется на примере электростатического взаимодействия элементарных частиц. Известно, что потенциал («мера воздействия») электрического поля элементарной заряженной частицы в некоторой точке пространства пропорционален отношению заряда частицы (Q) к расстоянию до частицы (r):

.

Метод потенциальных функций реализует полную аналогию указанного выше примера. При классификации объект проверяется на близость к объектам из обучающей выборки. Считается, что объекты из обучающей выборки «заряжены» своим классом, а мера «важности» каждого из них при классификации зависит от его «заряда» и расстояния до классифицируемого объекта.



49 Отбор эталонных объектов

Отбор эталонных объектов (Prototype Selection) - это процесс выбора небольшого подмножества представительных объектов (эталонов) из обучающего набора данных с целью улучшения эффективности и качества обучения алгоритмов машинного обучения.

Эти соображения приводят к идее исключить из выборки шумовые и неинформативные объекты, оставив только минимальное достаточное количество эталонов. Этим достигается несколько целей одновременно — повышается качество и устойчивость классификации, сокращается объём хранимых данных и уменьшается время классификации, затрачиваемое на поиск ближайших эталонов.

50 Непрерывные аппроксимации пороговой функции потерь

Пороговая функция потерь, также известная как функция Хевисайда, является дискретной функцией, которая принимает значение 0 при x < 0 и значение 1 при x ≥ 0. В некоторых задачах машинного обучения и оптимизации может быть полезно использовать непрерывные аппроксимации пороговой функции потерь, которые имеют гладкую форму и дифференцируемы на всей числовой оси.

Ниже приведены некоторые распространенные непрерывные аппроксимации пороговой функции потерь:

Функция сигмоиды:

Одной из наиболее популярных аппроксимаций пороговой функции потерь является сигмоидная функция, такая как логистическая функция или гиперболический тангенс. Эти функции имеют форму "S" и ограничены значениями между 0 и 1.

Функция гладкого максимума:

Функция гладкого максимума, также известная как функция максимума Коши, является другой аппроксимацией пороговой функции потерь. Она определена как сумма логарифмов экспонент, что позволяет ей быть гладкой и дифференцируемой.

Функция ReLU:

Функция ReLU (Rectified Linear Unit) определена как max(0, x) и также может быть использована в качестве аппроксимации пороговой функции потерь. Она является простой и вычислительно эффективной, а также имеет нулевую производную для отрицательных значений и производную 1 для положительных значений.

Это лишь некоторые примеры непрерывных аппроксимаций пороговой функции потерь. В зависимости от задачи и требований, можно использовать и другие функции, которые соответствуют нужным свойствам и форме потерь.

51. Понятие вероятности и случайного события

Вероятность — это численная мера, которая отражает степень возможности возникновения определенного события. Она обычно выражается числом от 0 до 1, где 0 означает полную невозможность события, а 1 — абсолютную уверенность в его наступлении. Промежуточные значения указывают на вероятность события, находящуюся между полной невозможностью и абсолютной уверенностью.

Случайное событие — это исход, который может произойти или не произойти в результате некоторого случайного эксперимента или процесса. Например, бросок монеты может привести к появлению "орла" или "решки", и эти исходы являются случайными событиями. События могут быть простыми, когда им соответствует только один исход, или составными, когда они состоят из нескольких простых событий.

52. Понятие математического ожидания

Математическое ожидание – это понятие в теории вероятностей и статистике, которое представляет собой среднее значение случайной величины.

Для дискретной случайной величины, математическое ожидание вычисляется как сумма произведений значений случайной величины на их вероятности. Формально, если X – дискретная случайная величина, принимающая значения x1, x2, ..., xn с вероятностями p1, p2, ..., pn соответственно, то математическое ожидание E(X) вычисляется по формуле:

M(X) = x1 \* p1 + x2 \* p2 + ... + xn \* pn

Для непрерывной случайной величины, математическое ожидание представляет собой интеграл от произведения значения случайной величины на её плотность распределения. Формально, если X – непрерывная случайная величина с плотностью распределения f(x), то математическое ожидание E(X) вычисляется по формуле:

M(X) = ∫x \* f(x) dx,

где интегрирование производится по всем значениям x, для которых плотность распределения определена.

Математическое ожидание позволяет оценить среднее значение случайной величины.

53. Дисперсия

Дисперсия – это мера разброса случайной величины относительно её математического ожидания. Она показывает, насколько значения случайной величины различаются от её среднего значения.

Для дискретной случайной величины X с математическим ожиданием M(X), дисперсия D(X) вычисляется по формуле:

D(X)=M(X^2)−(M(X))^2,

Для непрерывной случайной величины X с плотностью распределения f(x), дисперсия D(X) вычисляется по формуле:

D(X) = ∫ [(x - M(X))^2 \* f(x)] dx,

где интегрирование производится по всем значениям x, для которых плотность распределения определена.

Дисперсия может быть положительной или нулевой. Если дисперсия равна нулю, это означает, что все значения случайной величины равны её математическому ожиданию, и нет разброса. Чем больше дисперсия, тем больше разброс значений случайной величины относительно её среднего значения.

Дисперсия позволяет оценить степень разброса и предсказуемости случайной величины. Кроме того, дисперсия является основой для вычисления других мер разброса, таких как стандартное отклонение, которое является квадратным корнем из дисперсии.

54. Уравнение регрессии

Уравнение регрессии – это математическое выражение, которое описывает связь между зависимой переменной и одной или несколькими независимыми переменными в статистическом модели. Уравнение регрессии используется для предсказания значений зависимой переменной на основе значений независимых переменных.

Наиболее распространенная форма уравнения регрессии называется линейной регрессией. Для простоты рассмотрим уравнение линейной регрессии с одной независимой переменной. Предположим, у нас есть зависимая переменная Y, и независимая переменная X. Уравнение линейной регрессии может быть записано в следующей форме:

Y = β0 + β1\*X + ε,

где Y - зависимая переменная, X - независимая переменная, β0 и β1 - коэффициенты регрессии, ε - остаточная ошибка.

Коэффициент β0 представляет собой сдвиг (пересечение) линии регрессии с осью Y, а коэффициент β1 - наклон (угол) линии регрессии. Они определяют величину и направление связи между переменными. Остаточная ошибка ε представляет разницу между фактическими значениями зависимой переменной и предсказанными значениями с помощью уравнения регрессии. Цель состоит в том, чтобы минимизировать сумму квадратов остатков и достичь наилучшего соответствия между моделью и данными.

Уравнение регрессии может быть более сложным, если в модели присутствуют несколько независимых переменных. В этом случае коэффициенты регрессии будут иметь соответствующие значения для каждой независимой переменной, и уравнение будет содержать дополнительные слагаемые.

Оценка коэффициентов регрессии и построение уравнения регрессии основываются на методе наименьших квадратов, который позволяет найти наилучшую линию, которая минимизирует сумму квадратов остатков.

55. Метод наименьших квадратов

Метод наименьших квадратов (МНК) – это статистический метод, используемый для оценки коэффициентов регрессии и построения уравнения регрессии. Он основывается на минимизации суммы квадратов остатков между фактическими значениями зависимой переменной и предсказанными значениями, полученными с помощью уравнения регрессии.

Процедура МНК включает следующие шаги:

1. Формулировка модели: определите зависимую переменную и независимые переменные, которые будут использоваться для построения уравнения регрессии.
2. Оценка коэффициентов: Используя набор данных, найдите оценки коэффициентов регрессии, которые минимизируют сумму квадратов остатков. Обычно для этого используется метод нахождения производных и решение системы уравнений.
3. Построение уравнения регрессии: С использованием оценок коэффициентов регрессии составьте уравнение, которое связывает зависимую переменную с независимыми переменными.
4. Оценка модели: оцените качество и значимость модели регрессии с помощью различных статистических метрик, таких как коэффициент детерминации (R-квадрат), F-статистика и стандартные ошибки коэффициентов.

Метод наименьших квадратов широко используется в различных областях, включая экономику, финансы, социологию, биологию и другие, для анализа данных и построения математических моделей, описывающих связи между переменными. Он позволяет получить оптимальные оценки коэффициентов регрессии и прогнозировать значения зависимой переменной на основе значений независимых переменных.

56. Законы распределения случайной величины

В теории вероятностей и статистике существует множество законов распределения случайной величины, которые описывают вероятностные свойства случайных явлений. Некоторые из наиболее распространенных законов распределения включают в себя:

1. Нормальное распределение: Нормальное распределение является одним из наиболее известных и широко используемых распределений. Оно характеризуется симметричной колоколообразной формой и определяется двумя параметрами: математическим ожиданием (средним) и стандартным отклонением. Многие естественные процессы и случайные явления приближаются нормальным распределением.
2. Равномерное распределение: Равномерное распределение предполагает, что вероятность случайной величины принять любое значение на заданном интервале одинакова. Оно характеризуется двумя параметрами: минимальным и максимальным значением.
3. Биномиальное распределение: Биномиальное распределение применяется для моделирования случайных событий, которые имеют два возможных исхода (успех и неудача) с заданной вероятностью. Оно характеризуется двумя параметрами: количеством испытаний и вероятностью успеха в каждом испытании.
4. Пуассоновское распределение: Пуассоновское распределение используется для моделирования числа редких случайных событий, которые происходят в фиксированном интервале времени или пространства. Оно характеризуется одним параметром, который является средней интенсивностью событий.

Кроме этих законов, существуют и другие распределения, такие как экспоненциальное, гамма, бета, Хи-квадрат и др., каждое из которых применимо в определенных контекстах и имеет свои уникальные свойства.

57. Методы построения функции распределения

Существует несколько методов построения функции распределения для случайной величины. Вот некоторые из наиболее распространенных методов:

1. Аналитический метод: Этот метод применяется для аналитически определенных распределений, таких как нормальное, равномерное, экспоненциальное и другие. При использовании аналитического метода функция распределения определяется с помощью математических формул, которые описывают вероятностные свойства случайной величины.
2. Метод эмпирической функции распределения: Этот метод используется, когда у нас есть набор фактических наблюдений или данных. Для построения эмпирической функции распределения мы сортируем наблюдения в порядке возрастания и строим функцию, которая увеличивается на 1/n в каждой точке, где n - общее количество наблюдений. Это позволяет нам оценить функцию распределения на основе имеющихся данных.
3. Использование статистического программного обеспечения: Статистические пакеты и программное обеспечение, такие как Python (с библиотеками numpy, scipy) и другие, предоставляют функции и инструменты для построения функций распределения. Они обычно содержат готовые функции для различных распределений, которые можно использовать для анализа данных и построения функций распределения.

Выбор метода построения функции распределения зависит от доступности данных, природы случайной величины и требований конкретной задачи.

На языке Python существует несколько библиотек, которые предоставляют функции и методы для работы с функциями распределения.

Библиотека NumPy:

NumPy предоставляет функции для работы с числовыми массивами и выполнения математических операций. Вот несколько примеров использования NumPy для построения функции распределения:

import numpy as np

# Построение функции распределения для нормального распределения

mu = 0 # среднее значение

sigma = 1 # стандартное отклонение

x = np.linspace(-5, 5, 100) # создание массива значений

cdf = np.norm.cdf(x, mu, sigma) # функция распределения

# Построение графика функции распределения

import matplotlib.pyplot as plt

plt.plot(x, cdf)

plt.xlabel('x')

plt.ylabel('CDF')

plt.title('Нормальное распределение')

plt.grid(True)

plt.show()

Библиотека SciPy:

SciPy расширяет функциональность NumPy, предоставляя более широкий набор функций и методов для научных вычислений. Вот пример использования SciPy для построения функции распределения:

import numpy as np

from scipy.stats import norm

# Построение функции распределения для нормального распределения

mu = 0 # среднее значение

sigma = 1 # стандартное отклонение

x = np.linspace(-5, 5, 100) # создание массива значений

cdf = norm.cdf(x, mu, sigma) # функция распределения

# Построение графика функции распределения

import matplotlib.pyplot as plt

plt.plot(x, cdf)

plt.xlabel('x')

plt.ylabel('CDF')

plt.title('Нормальное распределение')

plt.grid(True)

plt.show()

Обратите внимание, что эти примеры демонстрируют построение функции распределения только для нормального распределения. В библиотеках NumPy и SciPy также доступны функции для работы с другими распределениями, такими как равномерное, экспоненциальное, биномиальное и др. Для каждого распределения существуют соответствующие функции распределения, которые можно использовать для построения.

58. Плотность распределения вероятности

Плотность распределения вероятности (PDF) – это функция, которая описывает вероятностные свойства непрерывной случайной величины. Плотность распределения вероятности определяет вероятность того, что случайная величина примет определенное значение или попадет в определенный интервал.

Формально, плотность распределения вероятности определяется как производная функции распределения (CDF) по переменной случайной величины. Обозначается она как f(x), где x - значение случайной величины. Плотность распределения вероятности имеет следующие свойства:

1. Неотрицательность: Значения плотности распределения вероятности должны быть неотрицательными.
2. Интеграл равен 1: Интеграл от плотности распределения вероятности по всем возможным значениям случайной величины равен 1. ∫ f(x) dx = 1.
3. Вероятность через плотность: Вероятность того, что случайная величина примет значение в определенном интервале, P(a ≤ X ≤ b) вычисляется как ∫ f(x) dx от a до b.

Для построения графика плотности распределения вероятности можно использовать различные библиотеки Python, такие как Matplotlib или Seaborn. Вот пример построения графика плотности распределения нормального распределения с использованием библиотеки Seaborn:

import seaborn as sns

# Генерация выборки из нормального распределения

mu = 0 # среднее значение

sigma = 1 # стандартное отклонение

sample = np.random.normal(mu, sigma, size=1000)

# Построение графика плотности распределения

sns.histplot(sample, kde=True)

plt.xlabel('x')

plt.ylabel('Плотность распределения')

plt.title('Нормальное распределение')

plt.grid(True)

plt.show()

59. Метод ближайших соседей

Метод ближайших соседей (k-nearest neighbors, k-NN) – это простой алгоритм классификации и регрессии, который основывается на принципе "ближайших соседей". Он используется для определения класса или значения целевой переменной нового наблюдения на основе его ближайших соседей в пространстве признаков.

Принцип работы метода ближайших соседей:

1. Задается значение параметра k, который определяет количество ближайших соседей, которые будут использоваться для классификации или регрессии нового наблюдения.
2. Вычисляется расстояние между новым наблюдением и всеми остальными наблюдениями в обучающем наборе данных. Расстояние может быть вычислено с использованием различных метрик, таких как евклидово расстояние или манхэттенское расстояние.
3. Находятся k наблюдений с наименьшим расстоянием до нового наблюдения, и их классы или значения целевой переменной используются для принятия решения о классификации или регрессии нового наблюдения.

В случае классификации, класс нового наблюдения определяется на основе большинства классов его ближайших соседей.

В случае регрессии, значение целевой переменной нового наблюдения вычисляется как среднее или медианное значение целевых переменных его ближайших соседей.

1. Процесс повторяется для каждого нового наблюдения, которое требует классификации или регрессии.

Метод ближайших соседей является простым и интуитивно понятным алгоритмом, но он имеет некоторые ограничения, такие как чувствительность к шуму и выбросам, зависимость от выбора значения параметра k и неэффективность в работе с большими наборами данных.

60. Статистический классификатор

Статистический классификатор – это алгоритм, использующий статистические методы для классификации объектов на основе их признаков. Он основывается на статистическом анализе данных и моделирует зависимости между признаками объектов и их классами.

Основные принципы работы статистического классификатора:

1. Обучение (построение модели): на этапе обучения классификатор использует обучающий набор данных, состоящий из примеров объектов и известных классов или меток. Он анализирует статистические характеристики признаков объектов и строит статистическую модель, которая описывает связь между признаками и классами. Примеры статистических классификаторов включают наивный Байесовский классификатор, логистическую регрессию, дискриминантный анализ и другие.
2. Классификация (применение модели): после завершения обучения классификатор применяет построенную статистическую модель для классификации новых, неизвестных объектов. Для каждого нового объекта классификатор анализирует его признаки и использует статистическую модель для прогнозирования его класса. В зависимости от типа классификатора, прогноз может быть представлен как вероятность принадлежности к каждому классу или конкретная метка класса.

Выбор конкретного статистического классификатора зависит от характеристик данных, задачи классификации и предпочтений исследователя или практика.

61. Байесовский классификатор на основе наблюдаемых признаков

Байесовский классификатор на основе наблюдаемых признаков, также известный как наивный Байесовский классификатор, является статистическим классификатором, основанным на принципе байесовской статистики и предположении о независимости признаков.

Принцип работы наивного Байесовского классификатора на основе наблюдаемых признаков:

1. Обучение (построение модели): На этапе обучения классификатор анализирует обучающий набор данных, состоящий из примеров объектов и их классов. Он вычисляет статистические характеристики признаков для каждого класса и строит модель, которая описывает вероятности классов и условные вероятности признаков для каждого класса. Для категориальных признаков, например, можно вычислить вероятности появления каждого значения признака для каждого класса. Для непрерывных признаков, можно использовать вероятностные распределения, такие как нормальное распределение, для описания признаков каждого класса.
2. Классификация (применение модели): После завершения обучения классификатор применяет построенную модель для классификации новых, неизвестных объектов. Для каждого нового объекта классификатор анализирует его наблюдаемые признаки и использует модель, чтобы вычислить вероятности принадлежности к каждому классу. Он применяет байесовскую формулу для вычисления апостериорной вероятности класса на основе априорной вероятности класса и условных вероятностей признаков для каждого класса. Затем классификатор выбирает класс с наивысшей апостериорной вероятностью как предсказанный класс для нового объекта.

Наивный Байесовский классификатор предполагает, что все признаки объекта независимы друг от друга, что является упрощением, но обеспечивает высокую производительность и эффективность алгоритма. Он широко используется в задачах классификации текстов, фильтрации спама, анализе настроений и других областях, где признаки независимыми между собой или слабо коррелируют.

62. Валидационные наборы, кросс-валидация

Валидационные наборы:

Валидационные наборы используются для оценки производительности модели на данных, которые не были использованы при её обучении. При использовании валидационного набора, обучающий набор данных разделяется на две части: обучающую выборку и валидационную выборку. Обучающая выборка используется для тренировки модели, а валидационная выборка - для оценки её производительности. Оценка может осуществляться путем вычисления различных метрик, таких как точность, полнота, F1-мера и других, в зависимости от задачи.

Однако, использование одного фиксированного валидационного набора может привести к несмещенной оценке только в случае, если объем данных достаточно большой и хорошо представляет всю генеральную совокупность. В случае ограниченного объема данных, это может привести к нестабильным или неправильным оценкам модели.

Кросс-валидация:

Кросс-валидация является более надежным методом оценки модели, особенно при наличии ограниченного объема данных. Он предлагает разделение данных на несколько "складок" или "фолдов". Процесс кросс-валидации состоит из следующих шагов:

1. Разделение данных: Исходный набор данных разделяется на K непересекающихся фолдов (обычно K=5 или K=10).
2. Обучение и оценка: Процесс обучения и оценки модели выполняется K раз. На каждой итерации один фолд используется в качестве валидационного набора, а остальные фолды используются в качестве обучающего набора. Модель обучается на обучающем наборе и оценивается на валидационном наборе. Результаты оценки (например, метрики производительности) сохраняются для каждой итерации.
3. Усреднение результатов: После завершения K итераций, результаты оценок модели усредняются для получения обобщающей оценки производительности модели.

Кросс-валидация позволяет получить более надежную оценку производительности модели, так как каждая часть данных используется как обучающая и валидационная выборки. Это также позволяет оценить стабильность модели и идентифицировать проблемы, такие как переобучение или недообучение. Наиболее распространенными методами кросс-валидации являются "k-fold cross-validation" и "stratified k-fold cross-validation".

63. Робастное оценивание плотности

Робастное оценивание плотности (robust density estimation) - это метод оценки плотности вероятности распределения данных, который устойчив к наличию выбросов или аномальных значений в данных. В отличие от классических методов оценки плотности, которые чувствительны к наличию выбросов и могут искажать оценку плотности, робастные методы стараются минимизировать влияние выбросов на оценку.

Робастное оценивание плотности имеет важное применение в статистике и машинном обучении, особенно в случаях, когда данные могут содержать выбросы или аномалии, которые могут искажать статистические оценки. Оно позволяет получить более устойчивые и надежные оценки плотности, которые лучше отражают основную структуру данных.

64. Цензурирование выборки (отсев объектов-выбросов)

Цензурирование выборки - это метод обработки данных, при котором некоторые значения или объекты в выборке, считаемые выбросами или экстремальными значениями, удаляются или заменяются для улучшения анализа данных или моделирования. Цензурирование может быть полным или частичным.

1. Полное цензурирование: При полном цензурировании все значения или объекты, считаемые выбросами, удаляются из выборки. Это означает, что данные, которые нарушают определенные условия или считаются экстремальными, исключаются из анализа. Например, если у вас есть набор данных о доходах людей, и в нем есть некоторые аномальные значения, значительно отличающиеся от остальных, вы можете полностью удалить эти значения, чтобы избежать их влияния на результаты анализа.
2. Частичное цензурирование: При частичном цензурировании выбросы заменяются или преобразуются в значения, которые считаются менее экстремальными. Это может включать замену выбросов медианными значениями, средними значениями или другими статистическими показателями, которые представляют более типичное поведение данных. Частичное цензурирование позволяет сохранить некоторую информацию об экстремальных значениях, но смягчает их влияние на анализ.

Цензурирование выборки может быть полезным при обработке данных, содержащих выбросы или аномалии, которые могут исказить статистические оценки или модели. Однако цензурирование следует применять с осторожностью и основываться на обоснованных предположениях о данных. Важно учитывать, что цензурирование может привести к потере информации и искажению результатов анализа, поэтому необходимо тщательно рассмотреть применимость этого метода в контексте конкретной задачи и типа данных.

65. Нахождение оценок максимального правдоподобия параметров вероятностных моделей

Оценка максимального правдоподобия (MLE) - это метод нахождения наиболее вероятных значений параметров вероятностной модели, основываясь на наблюдаемых данных. Идея заключается в выборе параметров, которые максимизируют вероятность получить наблюдаемые данные.

Предположим, у нас есть вероятностная модель с набором параметров θ и наблюдаемыми данными X. Функция правдоподобия L(θ|X) показывает вероятность получить наблюдаемые данные X при условии заданных параметров θ. Оценка максимального правдоподобия состоит в нахождении значений параметров θ, при которых функция правдоподобия L(θ|X) достигает своего максимума.

Математически, MLE можно представить как:

θ̂ = argmax L(θ|X)

где θ̂ - оценка параметров, argmax - операция, возвращающая аргумент, при котором функция достигает своего максимума.

Часто проще максимизировать логарифмическую функцию правдоподобия, поскольку она упрощает вычисления и не изменяет положения экстремума. Таким образом, оценка максимального правдоподобия может быть записана как:

θ̂ = argmax log L(θ|X)

Для нахождения оценок максимального правдоподобия можно использовать различные методы оптимизации, такие как градиентный спуск или метод Ньютона-Рафсона. Эти методы позволяют находить значения параметров, при которых функция правдоподобия достигает своего максимума.

Оценка максимального правдоподобия широко используется в статистике и машинном обучении для оценки параметров вероятностных моделей на основе наблюдаемых данных. Это позволяет наиболее адаптировать модель к данным и использовать ее для прогнозирования, классификации и других задач анализа данных.

66. EM-алгоритм: основная идея, понятие скрытых переменных, Е-шаг, М-шаг

EM-алгоритм (Expectation-Maximization algorithm) – это итеративный метод для нахождения оценок максимального правдоподобия в моделях с пропущенными или скрытыми переменными. Основная идея алгоритма состоит в чередовании двух шагов - Е-шага (Expectation step) и М-шага (Maximization step), чтобы приближенно максимизировать правдоподобие модели.

Основная идея:

1. Инициализируются начальные значения параметров модели.
2. Циклически повторяются следующие шаги, пока не будет достигнуто условие сходимости:

a) E-шаг: Вычисляются ожидания скрытых переменных (expectations of latent variables) при заданных текущих значениях параметров модели.

b) М-шаг: Находятся новые оценки параметров, которые максимизируют правдоподобие модели, используя ожидания скрытых переменных, полученные на Е-шаге.

Понятие скрытых переменных:

Скрытые переменные - это переменные, которые не наблюдаются непосредственно, но влияют на генерацию данных или на параметры модели. В EM-алгоритме, скрытые переменные используются для описания неполной информации о данных или для учета сложных зависимостей между переменными. EM-алгоритм позволяет оценивать параметры модели, учитывая наличие скрытых переменных.

E-шаг (Expectation step):

На этом шаге, при заданных текущих значениях параметров модели, вычисляются ожидания скрытых переменных, которые максимизируют правдоподобие модели. Ожидания могут быть вычислены с использованием метода Монте-Карло, метода Максимума апостериорной вероятности (MAP) или других подходов, зависящих от модели.

М-шаг (Maximization step):

На этом шаге, с использованием ожиданий скрытых переменных, полученных на Е-шаге, находятся новые оценки параметров, которые максимизируют правдоподобие модели. Это может быть сделано с использованием метода нахождения оценок максимального правдоподобия, как описано ранее. Новые оценки параметров затем становятся текущими значениями параметров для следующей итерации.

EM-алгоритм обеспечивает итеративную оптимизацию оценок параметров, учитывая наличие скрытых переменных, и позволяет находить значения параметров, которые максимизируют правдоподобие модели. Этот метод широко применяется в статистике и машинном обучении для оценки параметров сложных моделей с неполными данными.

67. EM-алгоритм и кластеризация

Основная идея применения EM-алгоритма к задаче кластеризации состоит в представлении каждого кластера в виде распределения вероятности. Алгоритм позволяет найти наилучшие параметры этих распределений, учитывая скрытую принадлежность каждого объекта к кластеру.

Процесс кластеризации с использованием EM-алгоритма выглядит следующим образом:

1. Инициализация: задаются начальные значения параметров распределений для каждого кластера.
2. E-шаг (Expectation step): для каждого объекта вычисляются вероятности его принадлежности к каждому кластеру на основе текущих параметров распределений. Здесь используется формула условной вероятности или формула Байеса.
3. М-шаг (Maximization step): на основе оценок принадлежности объектов к кластерам (полученных на E-шаге), пересчитываются параметры распределений кластеров таким образом, чтобы максимизировать правдоподобие модели.
4. Шаги E-шага и М-шага повторяются до достижения сходимости, то есть пока изменение параметров модели не станет незначительным или до достижения заданного критерия остановки.

После завершения EM-алгоритма, полученные параметры распределений кластеров могут быть использованы для классификации новых объектов или для анализа скрытой структуры данных.

EM-алгоритм в задаче кластеризации позволяет моделировать данные с учетом скрытых принадлежностей кластерам, что делает его особенно полезным в ситуациях, когда точные принадлежности объектов к кластерам неизвестны или неоднозначны. Он позволяет найти наилучшие параметры модели, учитывая неопределенность или неизвестность принадлежности объектов к кластерам.

68. Нормальный закон распределения

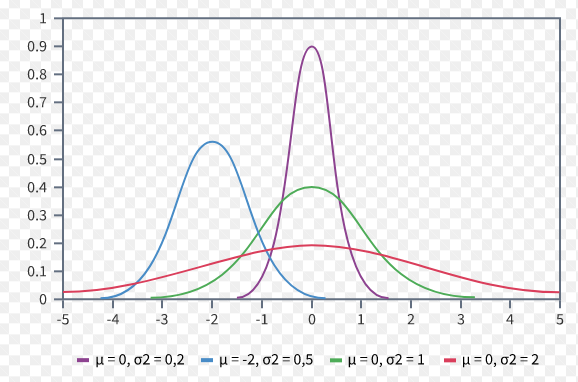
Нормальное распределение, также известное как распределение Гаусса или закон Гаусса, является одним из наиболее распространенных и важных вероятностных распределений. Оно часто используется в статистике и вероятностных моделях для моделирования случайных величин, которые имеют непрерывные значения.

Функция плотности вероятности (PDF) нормального распределения задается следующей формулой:



где x - значение случайной величины, μ - среднее значение (математическое ожидание) распределения, σ - стандартное отклонение распределения, π - число π (пи), exp - экспонента.

Графически, нормальное распределение имеет форму симметричного колокола с пиком в точке μ и шириной, определяемой стандартным отклонением σ. Среднее значение μ задает центр распределения, а стандартное отклонение σ определяет его разброс. Чем больше значение σ, тем шире распределение.



Нормальное распределение обладает следующими свойствами:

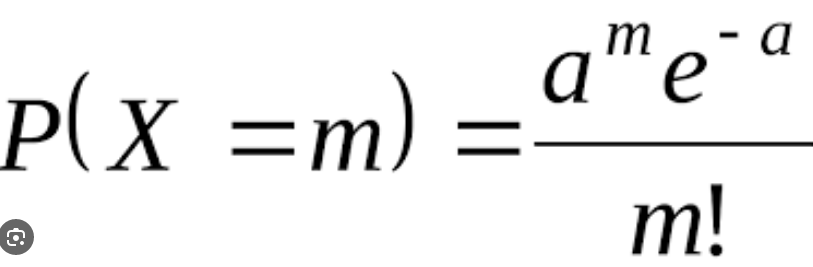
1. Симметричность: распределение симметрично относительно среднего значения μ.
2. Однородность: все его формы являются однородными и могут быть преобразованы друг в друга с помощью аффинных преобразований.
3. Центральная предельная теорема: сумма большого числа независимых случайных величин, имеющих любое распределение, приближается к нормальному распределению.

Нормальное распределение широко используется в статистике для моделирования реальных данных, таких как измерения, ошибки, физические величины и т.д. Многие статистические методы и тесты основаны на предположении о нормальности данных или на применении аппроксимации нормальным распределением.

69. Закон распределения Пуассона

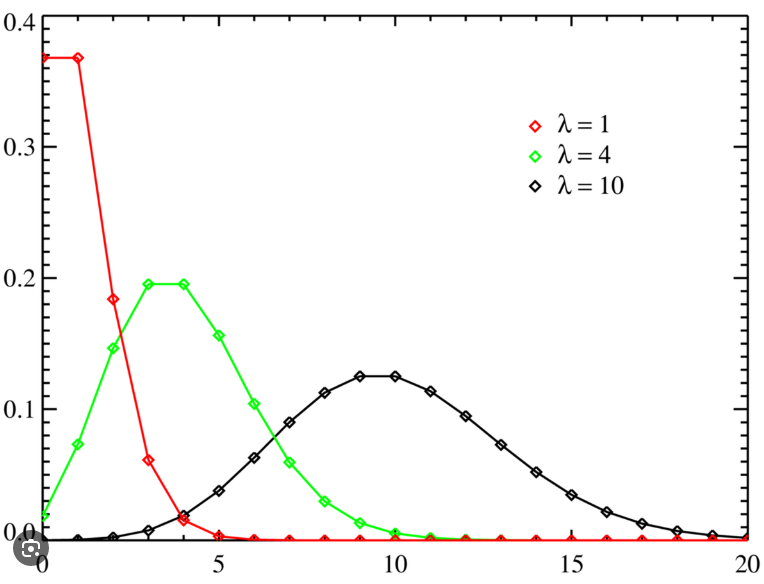
Закон распределения Пуассона является дискретным вероятностным распределением, которое моделирует количество событий, произошедших в определенном интервале времени или пространстве, при условии, что события происходят с некоторой фиксированной средней интенсивностью и независимо друг от друга.

Функция вероятности распределения Пуассона задается следующей формулой:



где X - случайная величина, m - количество событий, a - среднее число событий, e - основание натурального логарифма.

Графически, распределение Пуассона имеет форму скошенной вправо кривой, которая начинается от нуля и не имеет верхней границы. Параметр λ определяет среднее число событий, которое ожидается произойти в данном интервале. Чем больше значение λ, тем более плотное и симметричное распределение.



Основные свойства распределения Пуассона:

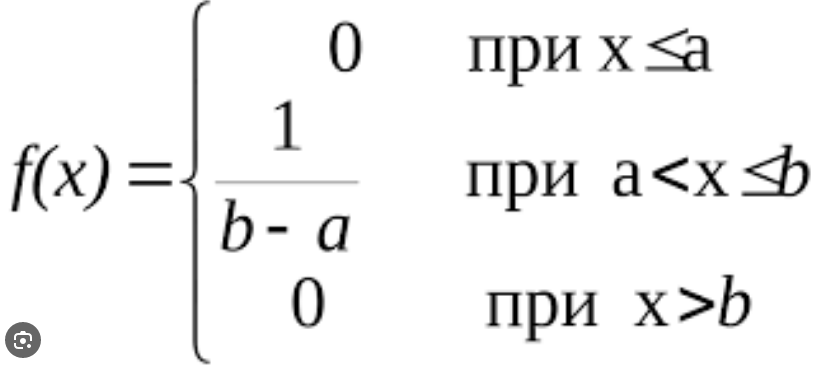
1. Среднее и дисперсия: Среднее значение распределения равно λ, а дисперсия равна λ.
2. Вероятность нулевого события: P(X=0) = e^(-λ), что означает вероятность отсутствия событий в данном интервале.
3. Независимость событий: События, моделируемые распределением Пуассона, предполагаются независимыми друг от друга.

Распределение Пуассона широко используется для моделирования случайных событий, таких как поступление заявок в систему обслуживания, число телефонных вызовов в определенный промежуток времени, количество ошибок в тексте и т.д. Также оно является аппроксимацией биномиального распределения при большом числе испытаний и малой вероятности успеха.

70. Равномерный закон распределения

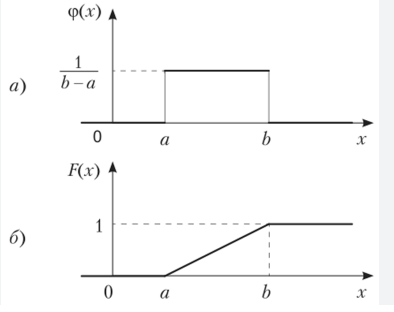
Равномерное распределение, также известное как равномерный закон распределения, является непрерывным вероятностным распределением, при котором вероятность попадания случайной величины в любой интервал равна длине этого интервала, пропорциональной общей длине интервала, в котором определена случайная величина.

Функция плотности вероятности (PDF) равномерного распределения определена на некотором интервале [a, b] и имеет следующий вид:



где x - значение случайной величины, a - нижняя граница интервала, b - верхняя граница интервала.

Графически, равномерное распределение представляет собой прямую линию с постоянной высотой на интервале [a, b]. Все значения внутри этого интервала имеют одинаковую вероятность. Вероятность попадания случайной величины в любой подинтервал равна длине этого подинтервала, деленной на общую длину интервала [a, b].



Основные свойства равномерного распределения:

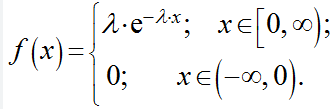
1. Симметричность: распределение симметрично относительно середины интервала (a + b) / 2.
2. Непрерывность: функция плотности вероятности непрерывна на интервале [a, b] и равна нулю за его пределами.
3. Математическое ожидание: M(X) = (a + b) / 2 - среднее значение равномерного распределения.
4. Дисперсия: D(X) = (b - a)^2 / 12 - мера разброса значений равномерного распределения.

Равномерное распределение широко используется в статистике и моделировании для моделирования случайных величин, которые имеют равномерное распределение в заданном интервале. Примеры таких случайных величин могут включать выбор случайного числа в заданном диапазоне или моделирование случайного времени ожидания события в равномерном интервале.

71. Показательный закон распределения

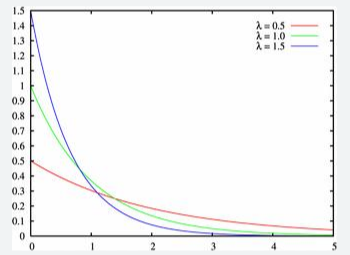
Показательное распределение, также известное как экспоненциальное распределение, является непрерывным вероятностным распределением, которое моделирует время между последовательными независимыми событиями, происходящими с постоянной интенсивностью.

Функция плотности вероятности (PDF) показательного распределения задается следующей формулой:



где x - значение случайной величины (время), λ - параметр интенсивности (обратное среднее время между событиями), e - основание натурального логарифма.

Графически, показательное распределение имеет форму убывающей экспоненты, начиная с нуля и не имеет верхней границы. Чем меньше значение параметра интенсивности λ, тем медленнее убывает функция плотности вероятности.



Основные свойства показательного распределения:

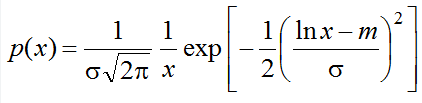
1. Отсутствие памяти: Показательное распределение обладает свойством отсутствия памяти. Это означает, что вероятность ожидания события не зависит от времени, уже проведенного без событий. Например, вероятность ожидания события в следующую минуту будет такой же, как и вероятность ожидания события в первую минуту.
2. Математическое ожидание и дисперсия: Математическое ожидание M(X) = 1 / λ, а дисперсия D(X) = 1 / λ^2. Чем больше значение параметра интенсивности λ, тем меньше математическое ожидание и дисперсия.

Показательное распределение часто используется для моделирования случайных величин, таких как время между приходом заявок в систему обслуживания, время между отказами оборудования, время ожидания случайного события и т.д. Также оно является основным распределением в теории надежности и теории очередей.

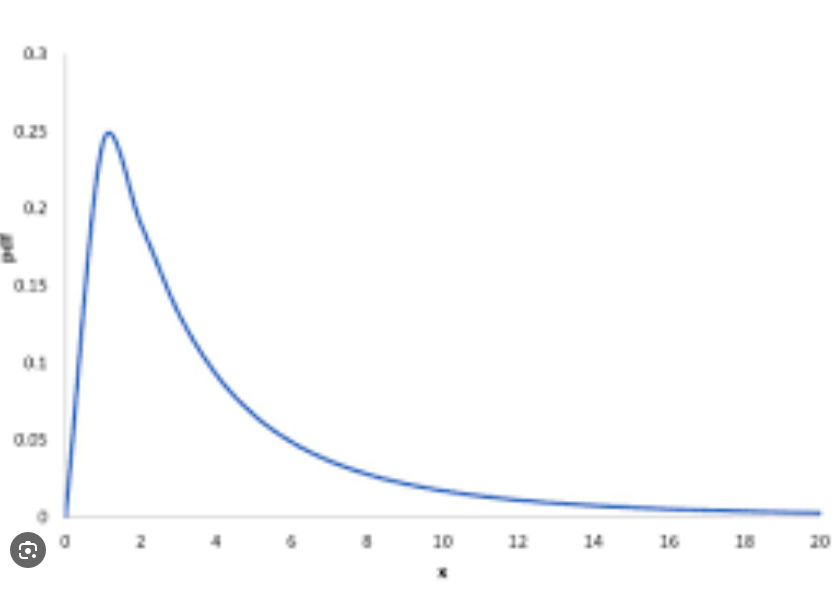
72. Логарифмически-нормальное распределение

Логарифмически-нормальное распределение – это непрерывное вероятностное распределение, в котором логарифм случайной величины имеет нормальное распределение. То есть, если X следует логарифмически-нормальному распределению, то ln(X) следует нормальному распределению.

Функция плотности вероятности (PDF) логарифмически-нормального распределения задается следующей формулой:



где x - значение случайной величины, μ - среднее значение логарифма случайной величины, σ - стандартное отклонение логарифма случайной величины, π - число пи, e - основание натурального логарифма.



Логарифмически-нормальное распределение обладает следующими свойствами:

1. Ограниченность: Случайная величина X может принимать значения только в положительном диапазоне (0, +∞).
2. Параметры: Распределение определяется параметрами μ и σ, которые определяют среднее и разброс логарифма случайной величины.
3. Симметрия: если μ = 0, то распределение симметрично относительно x = 1.
4. Связь с нормальным распределением: если ln(X) следует нормальному распределению с параметрами μ и σ, то X следует логарифмически-нормальному распределению с теми же параметрами.

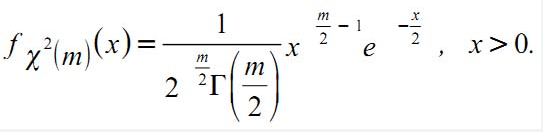
Логарифмически-нормальное распределение используется в статистике и моделировании для описания случайных величин, которые имеют положительное и скошенное распределение. Примеры включают доходы населения, размеры популяций, финансовые данные и другие случайные величины, которые могут быть описаны через логарифмические преобразования.

73. Хи квадрат распределение

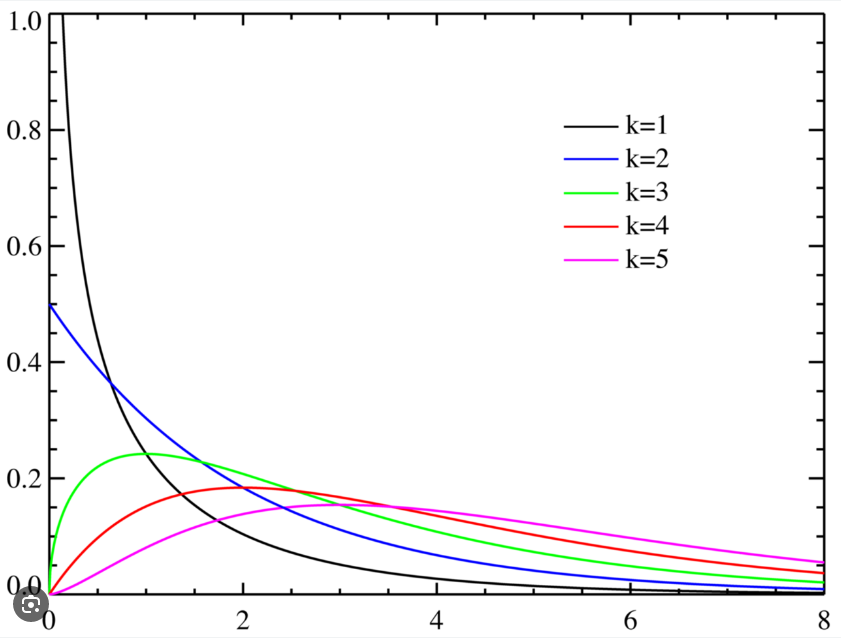
Хи-квадрат (χ²) распределение является одним из важных распределений в статистике. Оно используется для моделирования суммы квадратов независимых стандартных нормальных случайных величин.

Хи-квадрат распределение определяется одним параметром, называемым числом степеней свободы (df). Число степеней свободы указывает на количество независимых стандартных нормальных случайных величин, которые суммируются для получения случайной величины, имеющей хи-квадрат распределение.

Функция плотности вероятности (PDF) хи-квадрат распределения определена для x ≥ 0 и задается следующей формулой:



где x - значение случайной величины, df - число степеней свободы, Γ - функция гамма.



Свойства хи-квадрат распределения:

1. Число степеней свободы: чем больше число степеней свободы, тем более симметричным и сгруппированным около среднего значения становится распределение.
2. Среднее значение и дисперсия: Среднее значение распределения равно числу степеней свободы (M(X) = df), а дисперсия равна удвоенному числу степеней свободы (D(X) = 2df).

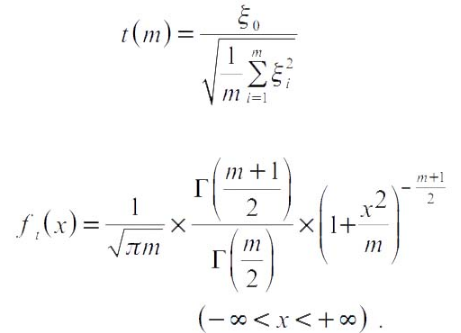
Хи-квадрат распределение широко используется в статистике для проверки гипотез, построения доверительных интервалов, оценки дисперсии и моделирования случайных величин, для которых суммируются квадраты независимых стандартных нормальных случайных величин.

74. Распределение Стьюдента (t - распределение)

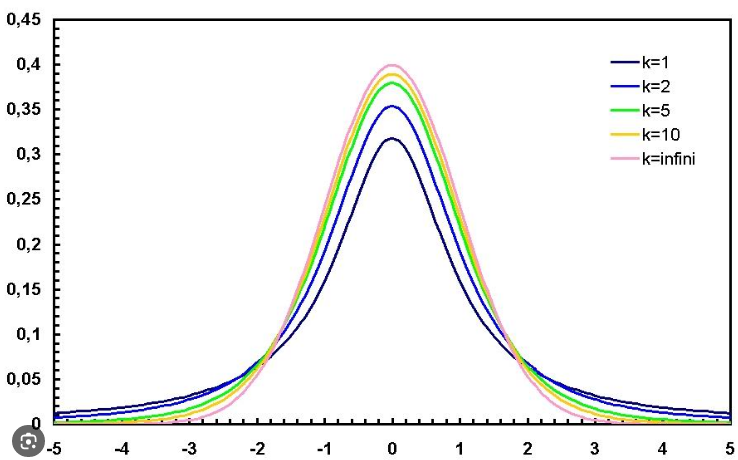
Распределение Стьюдента, также известное как t-распределение, является одним из основных распределений, используемых в статистике. Оно широко применяется для оценки параметров и проверки гипотез, особенно в случаях, когда размер выборки мал и генеральная совокупность имеет нормальное распределение.

Распределение Стьюдента определяется одним параметром, называемым числом степеней свободы (df). Число степеней свободы зависит от размера выборки и определяет форму распределения.

Функция плотности вероятности (PDF) t-распределения задается следующей формулой:



где x - значение случайной величины, m - число степеней свободы, Г - функция бета.



Свойства t-распределения:

1. Симметрия: Распределение Стьюдента симметрично относительно нуля, то есть f(x) = f(-x).
2. Хвосты распределения: при увеличении числа степеней свободы, t-распределение все больше приближается к стандартному нормальному распределению, то есть распределение со средним значением 0 и стандартным отклонением 1.
3. Тяжелые хвосты: при меньшем числе степеней свободы, t-распределение имеет более тяжелые хвосты по сравнению с нормальным распределением, что означает, что экстремальные значения более вероятны.
4. Среднее значение и дисперсия: Среднее значение t-распределения равно 0 при m > 1, а дисперсия равна m / (m - 2) при m > 2.

Распределение Стьюдента используется в различных статистических тестах, таких как t-тесты, для сравнения средних значений двух выборок или для оценки доверительных интервалов при малых выборках, когда генеральное распределение неизвестно или не является нормальным.

75. Распределение Фишера-Снедекора

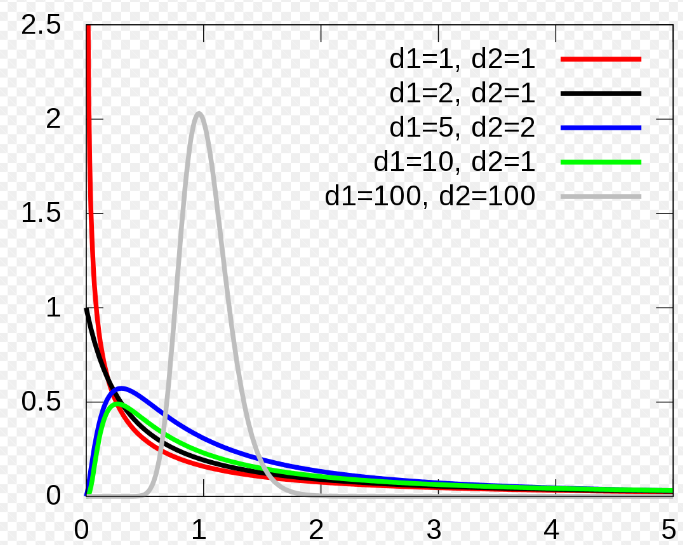
Распределение Фишера-Снедекора, также известное как F-распределение, является вероятностным распределением, которое используется в статистике для сравнения дисперсий двух нормально распределенных выборок.

F-распределение определяется двумя параметрами, которые называются числом степеней свободы: df₁ (для числителя) и df₂ (для знаменателя). Число степеней свободы определяет форму и свойства распределения.

Функция плотности вероятности (PDF) F-распределения задается следующей формулой:

f(x) = (1 / (B(df₁/2, df₂/2) \* (df₁/df₂)^(df₁/2) \* x^((df₁/2)-1) \* (1 + (df₁/df₂)\*x)^(-(df₁+df₂)/2)

где x - значение случайной величины, df₁ - число степеней свободы для числителя, df₂ - число степеней свободы для знаменателя, B - функция бета.



Свойства F-распределения:

1. Ограниченность: F-распределение определено только для x ≥ 0.
2. Симметрия: Распределение не является симметричным и имеет положительное скошенное распределение.
3. Параметры: F-распределение определяется числами степеней свободы df₁ и df₂. Оба параметра должны быть положительными целыми числами.
4. Среднее значение и дисперсия: Среднее значение распределения зависит от числа степеней свободы и выражается как E(X) = df₂ / (df₂ - 2), при df₂ > 2. Дисперсия равна (2 \* df₂^2 \* (df₁ + df₂ - 2)) / (df₁ \* (df₂ - 2)^2 \* (df₂ - 4)), при df₂ > 4.

F-распределение используется в статистических тестах, таких как анализ дисперсии (ANOVA) и линейная регрессия, для проверки гипотез о равенстве дисперсий двух или более групп или моделей. Также оно может быть применено в других статистических задачах, где требуется сравнение дисперсий или оценка доверительных интервалов.