1. **Непараметрическая регрессия**

Словарик

**Решение 1 лабы**

Ищем лучшую комбинацию функции расстояния, окна и ядра для метода ближайших соседей, для этого:

1. Определяем ядровые функции
2. (для удобства) Реализуем класс, в который передаем параметры алгоритма (пропустим, парсер)
3. Описываем дефолтные ф-ии для нашего алгоритма, устанавливаем параметры и записываем в класс парсер. (set\_params, regressorParams и так далее)
4. Считываем и кодируем целевые метки со значением от 0 до n\_classes-1. Далее через StandardScaler нормируем признаки
5. Применяем в ван хот кодирование для кодирования категориальных признаков
6. Рефит используется для гридсерча для поиска лучших параметров. (Не говорить)
7. Скоре – считаем метрики (узнаем качества алгоритма)
8. Функция для графика суммы
9. Функция для графика ширины
10. Запускаем грид серч для суммы, а потом ширины (лучшее для суммы, лучшее для ширины)

**2. Линейная регрессия**

Словарик

**Метод наименьших квадратов** – минимизация суммы квадратов отклонений от искомых переменных.

**SVD** – Используется при работе с матрицами, так как показывает геометрическую структуру матрицы и позволяет наглядно представить имеющиеся данные.

Любая матрица *L x N* представима в виде сингулярного разложения:

Имея сингулярное разложение, можно выписать решение задачи наименьших квадратов в явном виде, не прибегая к трудоёмкому обращению матриц.

https://lh5.googleusercontent.com/9PffYusT_GfL65jaWP25NQhehgSNaJ2WzNueePNKm8VYDIVvESQ204ecowH2wF6NGGWbnGVeVk3XUaoKYGdeE6N6T82YAMc4XaMyNlkPlf9JV6NL41XXK_Fok5ekfrfKe_CBuPic

Свойства - способность показывать ранг матрицы и приближать матрицы данного ранга.

**Градиентный спуск** - метод нахождения локального [экстремума](https://ru-wiki.ru/wiki/%D0%AD%D0%BA%D1%81%D1%82%D1%80%D0%B5%D0%BC%D1%83%D0%BC) [функции](https://ru-wiki.ru/wiki/%D0%A6%D0%B5%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) (чаще минимума) с помощью движения вдоль [градиента](https://ru-wiki.ru/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82). Используется для минимизации функции ошибки и повышении точности модели.

**Генети́ческий алгори́тм** — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе.

**Гиперпараметры модели** — параметры, значения которых задается до начала обучения модели и не изменяется в процессе обучения. У модели может не быть гиперпараметров.

**λ - параметр регуляризации**, имеющий смысл штрафа за сложность.

**Smape** – функция ошибки. Показывает какой процент составляет ошибка.

Решение 2 лабы

Сравните алгоритмы нахождения уравнения прямой

1. Блок – подгружаем пакеты;
2. Функция для загрузки с файла, далее представляем в виде таблицы (для удобства работы с помощью dataframe);
3. Smape – Метрика, функция ошибки. Показывает, какой процент составляет ошибка.
4. Мы нашли самые лучшие параметры (алгоритм) и рассматриваем их на максимальном количестве итераций. Для этого распараллеливаем вычисления для двух графиков.
5. Загружаем данные.

*Ridge – линейная регрессия, с L2 регуляризацией (прибавляешь суммы квадратов весов, а не по модулю).*

PredefinedSplit - Делаем перекрестную проверку во избежание переобучения.

[*Перекрёстная проверка*](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BA%D1%80%D1%91%D1%81%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D0%BA%D0%B0) *(кросс-валидация) - множество данных разбивается на тестовое и проверочное много раз всеми возможными способами. И каждый раз считаем сумму ошибки для каждого разбиения.*

С помощью GridSearchCV оптимизируем параметры (задача машинного обучения по выбору набора оптимальных гиперпараметров для обучающего алгоритма).

process\_val – запускаем лучший алгоритм, после чего при разном количестве итераций ищем ошибку и строим график.

**3. Метод опорных векторов**

Словарик

**Метод опорных векторов** - Задача - найти наиболее правильную линию или гиперплоскость, которая разделит данные на два класса (задача классификации). Алгоритм получает данные на вход и возвращает эту разделяющую линию.

В машинном обучении **ядро** (kernel) — это функция, которая способна вычислять скалярное произведение ϕ(a)T · ϕ(b), базируясь только на исходных векторах a и b, без необходимости в вычислении трансформации ϕ (или даже знании о ней).

Решение 3 лабы

1. Подгружаем пакеты;
2. Считываем из файла;
3. Описываем функцию Scatter для графика;

*скаттер - ( x , y ) создает график рассеивания с кругами в местоположениях, заданных векторами x и y .*

1. С помощью GridSearchCV оптимизируем параметры (задача машинного обучения по выбору набора оптимальных гиперпараметров для обучающего алгоритма). Далее строим график по лучшим параметрам для каждого из ядер для двух наборов данных.

*GridSearchCV от трех параметров (алгоритм, гипер\_параметры, перекрестная проверка по 10 блокам)*

*Best\_score – значение на лучшем алгоритме.*

1. **Наивный байесовский классификатор**

Словарик

**Байесовский классификатор**  — широкий класс [алгоритмов](http://www.machinelearning.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) [классификации](http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F), основанный на принципе максимума апостериорной вероятности. Для классифицируемого объекта вычисляются функции правдоподобия каждого из классов, по ним вычисляются апостериорные вероятности классов. Объект относится к тому классу, для которого апостериорная вероятность максимальна.

**Байесовская вероятность** — это интерпретация понятия вероятности, используемая в байесовской теории. Вероятность определяется как степень уверенности в истинности суждения.

Решение 4 лабы

*Контролирую априорное распределение*

1. функция, которая описывает загрузку данных
2. ~~CountVectorizer преобразовывает входной текст в матрицу, значениями которой, являются количества вхождения данного ключа (слова) в текст. В отличие от FeatureHasher имеет больше настраиваемых параметров, но работает медленнее.
3. Потом алгоритм для обучения модели (наивный байсовский кассификатор)
4. Строим рок кривую по полученным результатам (*рок кривая - график, позволяющий оценить качество*[*бинарной классификации*](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BD%D0%B0%D1%80%D0%BD%D0%B0%D1%8F_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F)*, отображает соотношение между долей объектов от общего количества носителей признака, верно классифицированных как несущие признак)*

Комментарий: пытаемся добиться наилучшего результата, меняя распределение, чтобы не одно реальное сообщение не было классифицировано как спам (но мы не добились этого, не идеальный результат)…

1. **Бустинг**

Словарик

**Адаптивный бустинг - Концентрируемся на ошибках**. Сначала всем объектам присваиваем одинаковые веса. Затем выполняем алгоритм и смотрим, на каких объектах мы ошиблись. Обновляем веса: увеличиваем веса у тех объектов, в классификации которых мы ошиблись. Повторяем процесс, пока функция ошибки не минимизируется, либо пока не достигается максимальное количество предикторов (алгоритмов).

 Алгоритм подбирает для обучения алгоритм, который лучше работает на тех объектах, на которых мы больше ошибаемся.

**Функция потерь** — функция, которая характеризует потери при неправильном принятии решений на основе наблюдаемых данных.

Алгоритм AdaBoost использует в качестве верхней гладкой оценки пороговой функции потерь экспоненциальную функцию потерь. На каждом шаге очередной базовый классификатор с его весом подбираются так, чтобы достичь наибольшего уменьшения следующего функционала ошибки на обучении, который, очевидно, является верхней оценкой доли ошибок на обучающей выборке.

Решение 5 лабы

1. Подгружаем пакеты;
2. Функция make\_meshgrid (создаем сетку точек для графика)

*make\_meshgrid - создает список массивов координатных сеток*N*-мерного координатного пространства для указанных одномерных массивов координатных векторов. Координатное пространство - это пространство*N*-мерных точек-координат, причем каждой точке в таком пространстве соответствует комбинация одного значения из каждого координатного массива.*

*x: данные для создания сетки сетки оси x*

*y: данные для создания сетки сетки по оси Y*

*h: размер шага для сетки, необязательно*

Далее функция plot\_contours (строит График границ решения для классификатора)

*axe: объект осей matplotlib*

*clf: классификатор*

*xx: сетка ndarray*

*yy: сетка ndarray*

*params: словарь параметров для передачи в контур, необязательно*

*Z – разделяет плоскость;*

1. chips, geyser – два набора данных;

Считали данные из датасета, далее разделили на тестовую и обучающую выборку.

Далее создаю подграфики (plt.subplots), чтобы это все визуализировать.

(for i, ax in enumerate) – для каждого графика свой алгоритм, у которого свой n\_estimators (количество образцов алгоритма). Далее рисуем графики.

**7. Кластеризация**

**Словарик**

**К-мианс** - Алгоритм создает к групп из набора объектов, чтобы члены группы были наиболее однородными. Используется **кластерный анализ** - алгоритм для формирования групп таким образом, чтобы их члены были максимально похожи друг на друга. (кластер - синоним группы)

**Решение 7 лабы**

Алгоритм к – мианс

1. Функция загрузки данных
2. get\_Y\_pred – запускает алгоритм и возвращает предсказанные классы
3. загружаем данные, обучаем алгоритм
4. с помощью методов главных компонентов оставляем только два компонента и отрисовыаем их на плоскости
5. Печатаем матрицу ошибок кластеризации (тру позитив…)
6. С помощью метода главных компонентов переводим многомерное пространство в двумерное и рисуем на плоскости.
7. 1 график – реальный 2 – предсказанный

*Для отрисовки многомерных данных используйте преобразование (например PCA), но проводим в многомерном пространстве.*

*Последний график – две метрики качества (внешняя и внутренняя) График зависимости значений метрик относительно числа кластеров в нашем алгоритме*

*Эктернал – внешняя*

*В качестве внешней метрики акюраси, а в качестве внутренней silhouette\_score*

**1 лабораторная работа**

Задание

Выберите любой понравившийся набор данных для задачи классификации на сайте  [openml.org](https://www.openml.org/search?type=data). Выбранный набор данных должен содержать не менее 100 объектов и не менее трёх классов. Старайтесь не брать [уже выбранные датасеты](https://docs.google.com/spreadsheets/d/157cD6mDTQCln3o2RybZxZ07QMXE2QFzxfqn_1GqtHMk/). Отметьте выбранный датасет в [этой форме](https://forms.gle/v2ZPkJzbwBNgdYy79). Не забудьте векторизовать признаки вашего набора данных (перейти от категорий к числам), заполнить пропуски (если есть) и нормализовать.

# Сведение к задаче регрессии

Перейдите от задачи классификации к задаче регрессии двумя разными способами:

Наивный способ. Каждое значение класса представляется одним числом. Во время предсказания полученный ответ округляется до ближайшего целого числа.

Используя [OneHot преобразование](https://ru.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B8%D1%82%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4). Вместо одного целевого признака в набор данных добавляется столько новых числовых переменных, сколько в нём содержится классов.

Настройка гиперпараметров

Для каждого из преобразований найдите лучшую комбинацию функции расстояния, окна и ядра для метода ближайших соседей. Для лучшего преобразования и найденной комбинации постройте графики зависимости F-меры от числа ближайших соседей или ширины окна. Используйте Leave-One-Out перекрёстную проверку для подсчёта F-меры.

**2 лабораторная работа**

Задание

Сравните алгоритмы нахождения уравнения прямой:

* Метод наименьших квадратов (псевдообратная матрица / SVD)
* Градиентный спуск
* Генетический алгоритм, либо любой другой алгоритм оптимизации “чёрного ящика” (без использования информации о производной функции)

Для каждого алгоритма найдите наилучшие гиперпараметры (в том числе параметры регуляризации).

Каждый алгоритм следует запускать с различными лимитами времени или итераций. Для каждого алгоритма постройте график зависимости функции ошибки на тренировочном и тестовом множестве от выбранного значения лимита.

В качестве функции ошибки используйте NRMSE, либо SMAPE.

**3 лабораторная работа**

Задание

Используйте несколько ядер для метода опорных векторов. Для каждого набора данных и каждого ядра найдите лучшие параметры ядра, используя перекрёстную проверку.

После нахождения оптимальных параметров для каждого набора данных и каждого ядра нарисуйте, как классификатор классифицирует всё пространство. В данном случае классификатор следует строить на всём наборе данных, так как “тестовым набором” будет являться всё пространство.

**4 лабораторная работа**

Задание

Рассмотрим задачу классификации сообщения на спам и на реальные письма.

Придумайте вероятностную модель превращающую письмо в разряженный вектор признаков. Модель должна поддерживать n-граммы и учитывать как заголовок, так и содержание письма. Постройте ROC кривую для выбранной модели. Посчитайте точность используя перекрестную проверку.

Контролируя веса классов λspam или λlegit , либо априорное распределение добейтесь того, чтобы ни одно реальное сообщение не было классифицировано как спам. Постройте график зависимости точности от выбранного параметра λ, где λ изменяется от значения по умолчанию (λspam = λlegit), до найденного значения в предыдущем пункте.

**5 лабораторная работа**

Задание

Изучите алгоритм адаптивного бустинга для задачи классификации с экспоненциальной функцией потерь (AdaBoost). Базовый алгоритм можно использовать любой. **Изобразите, как алгоритм классифицирует всё пространство после каждого шага бустинга**. Постройте график зависимости качества от номера шага. Функцию качества можно использовать любую, но вычислять её стоит на отдельном тестовом множестве.

**7 лабораторная работа**

**Задание**

Реализуйте любой алгоритм кластеризации на выбор и две метрики качества кластеризации: одну внешнюю и одну внутреннюю.

Возьмите любой набор данных для задачи классификации, желательно взять  набор данных из лабораторной работы про KNN. Не забудьте векторизовать и нормализовать набор данных. Нарисуйте два графика: набор данных с реальными метками и с метками полученными в результате кластеризации. Постарайтесь выбрать такие гипер параметры  алгоритма кластеризации, чтобы результат кластеризации был как можно более похож на реальные метки. Для отрисовки многомерных данных используйте преобразование (например PCA), но кластеризацию по прежнему проводите в многомерном пространстве.

Постройте график зависимости выбранных метрик качества кластеризации от числа кластеров.