**Geekbrains**

**Цифровые профессии. Искусственный интеллект**

**Дипломный проект**

**Использование комбинации разных методов регрессии, разведочного анализа данных и визуального анализа данных для прогнозирования цены продажи жилых домов по их известным характеристикам**

Азимов Виктор Александрович

2024 год

**Содержание**

**Введение………………………………………………………………………..…4**

**Теоретическая часть – термины и понятия,**

**используемые в практическом исследовании………………………………..6**

Регрессия……………………………………………………………………6

*Линейная регрессия………………………………………………………...6*

*Подготовка данных для линейной регрессии…………………..7*

Регуляризация………………………………………………………………8

Регрессия LASSO………………………………………………………….10

Гребневая (ridge) регрессия……………………………………………….11

Регрессия ElasticNet……………………………………………………….13

Метод опорных векторов, или SVM……………………………………...14

XGBoost model…………………………………………………………….16

*Ансамбль……………………………………………………………………17*

*Бустинг……………………………………………………………………..17*

*Градиентный бустинг…………………………………………………..17*

*Дерево решений……………………………………………………………18*

LightGBM………………………………………………………………….19

Разведочный анализ данных……………………………………………...20

*Цели разведочного анализа данных……………………………………21*

*Инструменты и методы разведочного анализа данных…………..22*

Фут………………………………………………………………………....24

Метод кросс-валидации k-Fold…………………………………………..24

MSE………………………………………………………………………..25

RMSE………………………………………………………………………26

Куртозис……………………………………………………………………26

Коэффициент ранговой корреляции Спирмена………………………….27

**Практическая часть……………………………………………………………29**

Обзор……………………………………………………………………….29

Описание файлов с входными данными…………………………………29

Поля данных в train.csv и test.csv…………………………………………30

Разведочный и визуальный анализ данных………………………………34

*Визуальный анализ………………………………………………………..34*

*Анализ пропущенных данных……………………………………….…..40*

*Проверка данных в файле train.csv*

*на нормальность распределения……………………………………….42*

*Анализ влияния качественных признаков*

*на цену продажи…………………………………………………………..45*

*Обработка данных……………………………………………………….46*

Создание, обучение и оценка моделей…………………………………...47

Смешение моделей и предсказание цены………………………………..48

Смешение комбинированной модели

с лучшими ядрами Kaggle………………………………………………...49

**Заключение……………………………………………………………………...51**

**Список использованных источников………………………………………..52**

**Приложения……………………………………………………………………..56**

**Введение**

Данный проект выполнен по специализации «Аналитика, искусственный интеллект». Он представляет собой попытку использования методов машинного обучения, конкретно – комбинации разных видов линейной регрессии, для предсказания цен на жилые дома исходя из совокупности известных характеристик этих домов. Перед созданием моделей машинного обучения проводится разведочный анализ данных, включающий подробный визуальный анализ данных и применение методов математической статистики, а также выполняются необходимые преобразования данных. Тема является актуальной и востребованной – данные по прогнозированию стоимости домов можно продать сайтам-агрегаторам. Также можно создать собственный сайт в Интернете, с помощью которого пользователи смогут прогнозировать стоимость жилья.

**Цель:** изучить особенности разновидностей одного из методов машинного обучения – линейной регрессии – и применить их для решения практической задачи. Попрактиковаться в разведочном и визуальном анализе данных, подготовке данных для машинного обучения.

**Какую проблему решает:** прогноз цен на недвижимость по известным характеристикам этой недвижимости.

**Задачи:**

1. Изучить теорию, касающуюся темы исследования.
2. Применить на практике знания по визуальному анализу данных, провести разведочный анализ данных.
3. Провести необходимые преобразования данных для использования их в машинном обучении.
4. Создать работающие модели на основе нескольких разновидностей одного из методов машинного обучения – линейной регрессии.

**Инструменты:** язык программирования Python, программа Visual Studio Code, библиотеки Python: numpy, pandas, datetime, scipy, sklearn, matplotlib, seaborn, mlxtend, xgboost, lightgbm.

**Состав команды:** Азимов Виктор Александрович, аналитик.

**Теоретическая часть**

**Термины и понятия, используемые в практическом исследовании**

**Регрессия**

([лат.](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B8%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) regressio – обратное движение, отход) в [теории вероятностей](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9) и [математической статистике](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0) – [односторонняя](https://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D1%8F%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) [стохастическая](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%BE%D1%85%D0%B0%D1%81%D1%82%D0%B8%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C) зависимость, устанавливающая соответствие между случайными переменными, то есть [математическое выражение](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), отражающее связь между зависимой переменной у и независимыми переменными х при условии, что это выражение будет иметь [статистическую значимость](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%BD%D0%B0%D1%87%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C). В отличие от чисто [функциональной зависимости](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) y = f(x), когда каждому значению независимой переменной x соответствует одно определенное значение величины y, при регрессионной связи одному и тому же значению x могут соответствовать в зависимости от случая различные значения величины y.

**Линейная регрессия** **(Linear regression)**

Это используемая в статистике математическая модель зависимости одной (объясняемой, зависимой) переменной y от другой или нескольких других переменных (факторов, регрессоров, независимых переменных) x с линейной функцией зависимости. В машинном обучении это метод обучения с учителем, который используется для предсказания непрерывной целевой переменной от одного или нескольких независимых признаков. В основе метода лежит предположение о том, что существует линейная связь между признаками и целевой переменной. Эта связь моделируется с помощью линейной функции. Модель линейной регрессии пытается найти лучшую прямую, которая может описывать зависимость между независимыми признаками и зависимой переменной. Это делается с помощью поиска оптимальных коэффициентов, которые могут быть использованы для описания линейной функции. Задача регрессии в [машинном обучении](https://skillfactory.ru/machine-learning) – это предсказание одного параметра (Y) по известному параметру X, где X – набор параметров, характеризующий наблюдение. Примерами успешного применения этого метода могут служить такие задачи, как: прогнозирование продаж, прогнозирование цены на недвижимость, анализ влияния факторов на уровень заболеваемости и т. д. Однако она неэффективна для решения задач, где не существует линейной связи между признаками и целевой переменной.

Основная идея подбора коэффициентов в линейной регрессии – нахождение лучшей прямой, которая может описать зависимость между зависимой и целевой переменной. Формула линейной регрессии может быть записана следующим образом:

y = β0 + β1×x1 + β2×x2 + … + βn×xn

где y – предсказываемая зависимая переменная, x1, x2, …, xn – независимые переменные, β0, β1, β2, …, βn – коэффициенты регрессии.

Цель – найти оптимальные значения коэффициентов β0, β1, β2, …, βn, которые минимизируют сумму квадратов ошибок. Это может быть сделано с помощью методов оптимизации, к примеру таких, как градиентный спуск или метод наименьших квадратов.

**Подготовка данных для линейной регрессии**

***Линейное предположение.*** Линейная регрессия предполагает, что связь между вашими входными и выходными данными линейна. Она не поддерживает ничего другого. Это может быть очевидно, но об этом полезно помнить, когда у вас много атрибутов. Вам может потребоваться преобразовать данные, чтобы сделать связь линейной (например, логарифмическое преобразование для экспоненциальной связи).

***Удалить шум.*** Линейная регрессия предполагает, что ваши входные и выходные переменные не являются шумными. Рассмотрите возможность использования операций очистки данных, которые позволят вам лучше раскрыть и прояснить сигнал в ваших данных. Это наиболее важно для выходной переменной, и вы хотите удалить выбросы в выходной переменной (y), если это возможно.

***Удалить коллинеарность.*** Линейная регрессия будет переобучать ваши данные, если у вас есть сильно коррелированные входные переменные. Рассмотрите возможность расчета парных корреляций для ваших входных данных и удаления наиболее коррелированных.

***Распределения Гаусса.*** Линейная регрессия будет делать более надежные прогнозы, если ваши входные и выходные переменные имеют распределение Гаусса. Вы можете получить некоторую выгоду, используя преобразования (например, log или BoxCox) для ваших переменных, чтобы сделать их распределение более похожим на распределение Гаусса.

***Изменение масштаба входных данных:*** линейная регрессия часто позволяет делать более надежные прогнозы, если изменить масштаб входных переменных с помощью стандартизации или нормализации.

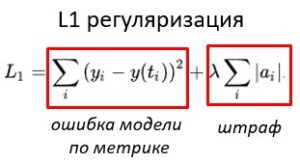
**Регуляризация**

Это набор методов уменьшения переобучения в моделях машинного обучения. Обычно регуляризация меняет незначительное снижение точности обучения на увеличение обобщаемости.

Регуляризация включает в себя ряд методов коррекции [переобучения](https://www-ibm-com.translate.goog/topics/overfitting?_x_tr_sl=en&_x_tr_tl=ru&_x_tr_hl=ru&_x_tr_pto=rq) в моделях машинного обучения. Таким образом, регуляризация – это метод повышения обобщаемости модели, то есть ее способности давать точные прогнозы для новых наборов данных. Регуляризация обеспечивает повышенную обобщаемость ради увеличения ошибки обучения. Другими словами, методы регуляризации обычно приводят к менее точным прогнозам на обучающих данных, но к более точным прогнозам на тестовых данных.

***L1 регуляризация*** также известна как Lasso (Least Absolute Shrinkage and Selection Operator) регуляризация. Она основана на добавлении штрафа, равного абсолютному значению коэффициентов модели.

Формально L1 регуляризация добавляет в функцию потерь дополнительное слагаемое, налагающее штраф за сложность модели, то есть высокие веса:



L1 регуляризация склонна к отбору признаков, так как она может уменьшить веса признаков до нуля. Это позволяет убрать неинформативные признаки из модели, что может уменьшить сложность модели и улучшить ее обобщающую способность.

Помимо L1 регуляризации, существует также ***L2 регуляризация*** (иногда называемая Ridge регуляризацией), которая также применяется в линейной регрессии и многих других моделях.

L2 регуляризация также добавляет к оптимизационной функции модели штрафную функцию:



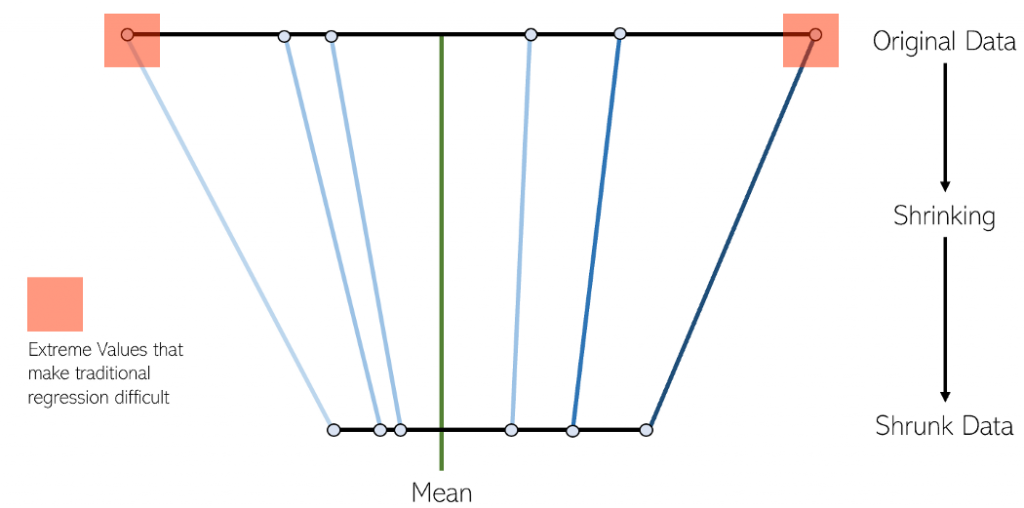
Эта штрафная функция является суммой квадратов весов модели, умноженных на гиперпараметр регуляризации. Это означает, что L2 регуляризация штрафует большие значения весов, заставляя их приближаться к нулю, но, в отличие от L1 регуляризации, не зануляет их полностью. Вместо этого L2 регуляризация штрафует большие значения весов более гладко и непрерывно, что позволяет более уверенно управлять компромиссом между точностью и сложностью модели.

Кроме того, L2 регуляризация может помочь в предотвращении переобучения и улучшении обобщающей способности модели, а также в уменьшении влияния шума в данных на модель.

**Регрессия LASSO**

**Теория.** Метод регрессии лассо (LASSO, Least Absolute Shrinkage and Selection Operator) – это вариация линейной регрессии, специально адаптированная для данных, которые демонстрируют сильную мультиколлинеарность (то есть сильную корреляцию признаков друг с другом).

Она автоматизирует части выбора модели, такие как выбор переменных или исключение параметров. LASSO использует сжатие коэффициентов (shrinkage), то есть процесс, в котором значения данных приближаются к центральной точке (например, среднему значению).

[](https://cloud.vk.com/wp-content/uploads/2020/08/regres3-1024x508.png) *Упрощенная визуализация процесса сжатия*

Процесс сжатия добавляет регрессионным моделям несколько преимуществ:

* Более точные и стабильные оценки истинных параметров.
* Уменьшение ошибок выборки и отсутствия выборки.
* Сглаживание пространственных флуктуаций.

Вместо того чтобы корректировать сложность модели, компенсируя сложность данных, подобно методам регрессии с высокой дисперсией нейронных сетей и дерева решений, лассо пытается уменьшить сложность данных так, чтобы их можно было обрабатывать простыми методами регрессии, искривляя пространство, на котором они лежат. В этом процессе лассо автоматически помогает устранить или исказить сильно коррелированные и избыточные функции в методе с низкой дисперсией.

Регрессия лассо использует регуляризацию L1, то есть взвешивает ошибки по их абсолютному значению. Вместо, например, регуляризации L2, которая взвешивает ошибки по их квадрату, чтобы сильнее наказывать за более значительные ошибки.

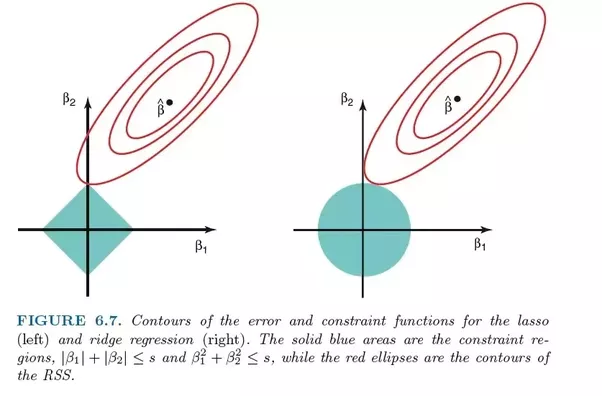
***Реализация.*** В sklearn регрессия лассо поставляется с моделью перекрестной проверки, которая выбирает наиболее эффективные из многих обученных моделей с различными фундаментальными параметрами и путями обучения, что автоматизирует задачу, которую иначе пришлось бы выполнять вручную.

Такая регуляризация часто приводит к более разреженным моделям с меньшим количеством коэффициентов, так как некоторые коэффициенты могут стать нулевыми и, следовательно, будут исключены из модели. Это позволяет ее интерпретировать.

**Гребневая регрессия (ридж-регрессия)**

Гребневая регрессия, или ридж-регрессия, очень похожа на регрессию LASSO в том, что она применяет сжатие. Оба алгоритма хорошо подходят для наборов данных с большим количеством признаков, которые не являются независимыми друг от друга (коллинеарность).

Однако самое большое различие между ними в том, что гребневая регрессия использует регуляризацию L2, то есть ни один из коэффициентов не становится нулевым, как это происходит в регрессии LASSO. Вместо этого коэффициенты всё больше приближаются к нулю, но не имеют большого стимула достичь его из-за природы регуляризации L2.

[](https://cloud.vk.com/wp-content/uploads/2020/08/regres4.png)*Сравнение ошибок в регрессии лассо (слева) и гребневой регрессии (справа). Поскольку гребневая регрессия использует регуляризацию L2, ее площадь напоминает круг, тогда как регуляризация лассо L1 рисует прямые линии.*

В лассо улучшение от ошибки 5 до ошибки 4 взвешивается так же, как улучшение от 4 до 3, а также от 3 до 2, от 2 до 1 и от 1 до 0. Следовательно, больше коэффициентов достигает нуля и устраняется больше признаков.

Однако в гребневой регрессии улучшение от ошибки 5 до ошибки 4 вычисляется как 5² − 4² = 9, тогда как улучшение от 4 до 3 взвешивается только как 7. Постепенно вознаграждение за улучшение уменьшается; следовательно, устраняется меньше признаков.

Гребневая регрессия лучше подходит в ситуации, когда мы хотим сделать приоритетными большое количество переменных, каждая из которых имеет небольшой эффект. Если в модели требуется учитывать несколько переменных, каждая из которых имеет средний или большой эффект, лучшим выбором будет лассо.

**Регрессия ElasticNet**

ElasticNet стремится объединить лучшее из гребневой регрессии и регрессии лассо, комбинируя регуляризацию L1 и L2.

Лассо и гребневая регрессия представляют собой два различных метода регуляризации. В обоих случаях λ – это ключевой фактор, который контролирует размер штрафа:

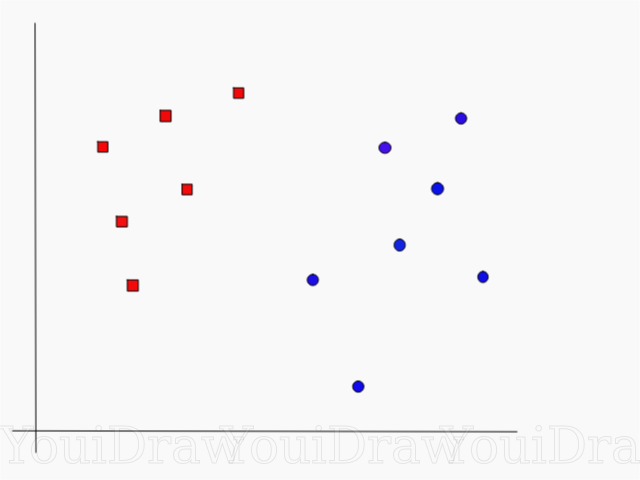
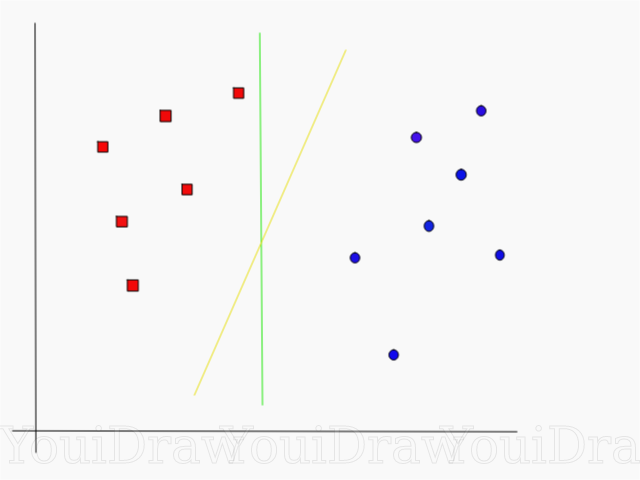
1. Если λ = 0, то задача становится аналогичной простой линейной регрессии, достигая тех же коэффициентов.
2. Если λ = ∞, то коэффициенты будут равны нулю из-за бесконечного веса на квадрате коэффициентов. Всё, что меньше нуля, делает цель бесконечной.
3. Если 0 < λ < ∞, то величина λ определяет вес, придаваемый различным частям объекта.

К параметру λ регрессия ElasticNet добавляет дополнительный параметр *α*, который измеряет, насколько «смешанными» должны быть регуляризации L1 и L2. Когда параметр *α* равен 0, модель является чисто гребневой регрессией, а когда он равен 1 – это чистая регрессия лассо.

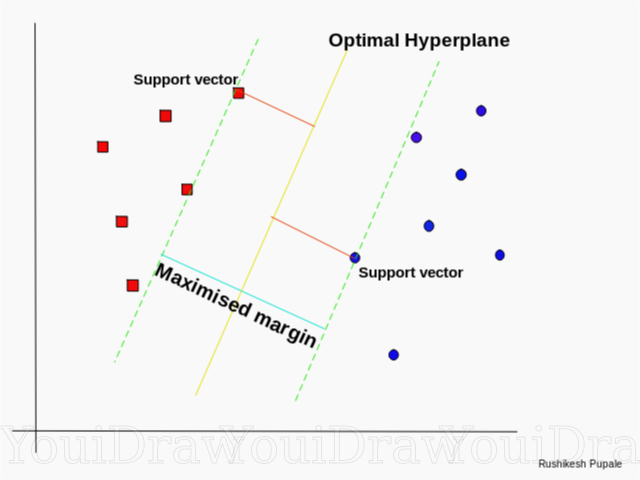
«Коэффициент смешивания» *α* просто определяет, сколько регуляризации L1 и L2 следует учитывать в функции потерь. Все три популярные регрессионные модели – гребневая, лассо и ElasticNet – нацелены на уменьшение размера своих коэффициентов, но каждая действует по-своему.

**Метод опорных векторов, или SVM**

(от англ. Support Vector Machines) – это линейный алгоритм, используемый в задачах классификации и регрессии. Данный алгоритм имеет широкое применение на практике и может решать как линейные так и нелинейные задачи. Суть работы «машин» опорных векторов проста: алгоритм создает линию или гиперплоскость, которая разделяет данные на классы.

Основной задачей алгоритма является найти наиболее правильную линию, или гиперплоскость, разделяющую данные на два класса. SVM – это алгоритм, который получает на входе данные и возвращает такую разделяющую линию. Рассмотрим следующий пример. Допустим, у нас есть набор данных, и мы хотим классифицировать и разделить красные квадраты от синих кругов (допустим, позитивное и отрицательное). Основной целью в данной задаче будет найти «идеальную» линию, которая разделит эти два класса.Найдите идеальную линию, или гиперплоскость, которая разделит набор данных на синий и красный классы. Нет одной, уникальной, линии, которая бы решала такую задачу. Мы можем подобрать бесконечное множество таких линий, которые могут разделить эти два класса. Как же именно SVM находит “идеальную” линию, и что в его понимании “идеальная”? Взгляните на пример ниже и подумайте, какая из двух линий (желтая или зеленая) лучше всего разделяет два класса и подходит под описаниие “идеальной”?Желтая прямая – это та самая линия, которую бы выбрал алгоритм. В данном примере мы можем интуитивно понять, что желтая линия разделяет и, соответственно, классифицирует два класса лучше зеленой. В случае с зеленой линией – она расположена слишком близко к красному классу. Несмотря на то что она верно классифицировала все объекты текущего набора данных, такая линия не будет генерализованной – не будет так же хорошо вести себя с незнакомым набором данных. Задача нахождения генерализованной разделяющей двух классов является одной из основных задач в машинном обучении.

*Как SVM находит лучшую линию?*

Алгоритм SVM устроен таким образом, что он ищет точки на графике, которые расположены непосредственно к линии разделения ближе всего. Эти точки называются опорными векторами. Затем алгоритм вычисляет расстояние между опорными векторами и разделяющей плоскостью. Это расстояние, которое называется зазором. Основная цель алгоритма – максимизировать расстояние зазора. Лучшей гиперплоскостью считается такая гиперплоскость, для которой этот зазор является максимально большим.

**XGBoost model**

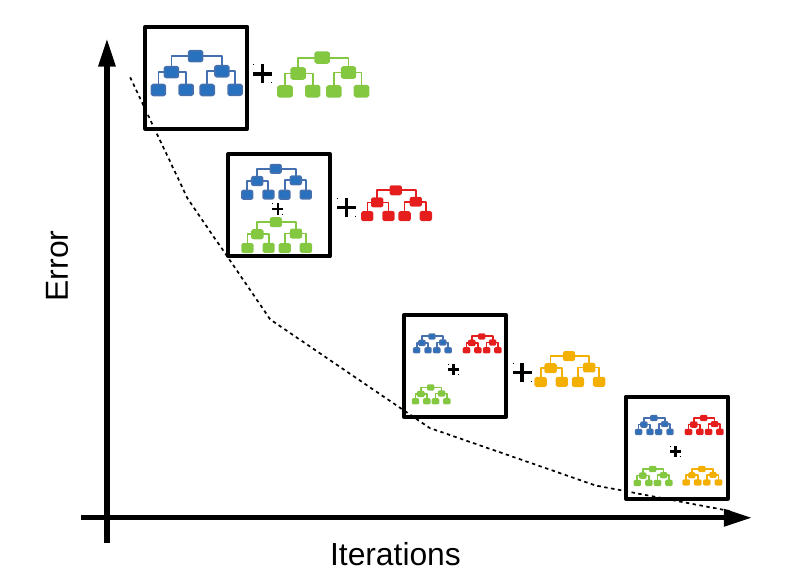
XGBoost (eXtreme Gradient Boosting) – это библиотека с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), используемая в [машинном обучении](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) и предоставляющая функциональность для решения задач, связанных с [регуляризацией](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) градиентного [бустинга](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D1%81%D1%82%D0%B8%D0%BD%D0%B3).

**Ансамбль** – это набор предсказателей, которые вместе дают ответ (например, среднее по всем). Причина, почему мы используем ансамбли, – несколько предсказателей, которые пытаются получить одну и ту же переменную, дадут более точный результат, нежели одиночный предсказатель.

**Бустинг** – это техника построения ансамблей, в которой предсказатели построены не независимо, а последовательно.

Эта техника использует идею о том, что следующая модель будет учиться на ошибках предыдущей. Они имеют неравную вероятность появления в последующих моделях, и чаще появятся те, что дают наибольшую ошибку. Из-за того, что предсказатели обучаются на ошибках, совершённых предыдущими, требуется меньше времени для того, чтобы добраться до реального ответа. Но мы должны выбирать критерий остановки с осторожностью, иначе это может привести к переобучению.

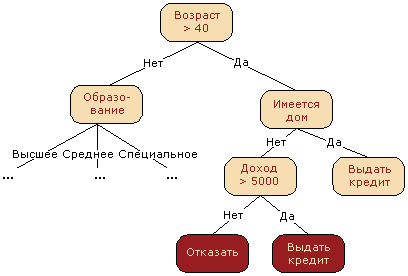
**Градиентный бустинг** – это техника машинного обучения для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений. В основе данного алгоритма лежит итеративное обучение деревьев решений с целью минимизировать функцию потерь. Благодаря особенностям деревьев решений градиентный бустинг способен работать с категориальными признаками, справляться с нелинейностями. Алгоритм усиления градиента:



**Деревом решений** можно назвать наглядную инструкцию, что делать в какой ситуации. Дерево решений состоит из узлов, ветвей и листьев. Узел представляет собой точку, в которой происходит разделение данных на два или более подмножества. Ветви связывают узлы и листья, а лист представляет собой конечный результат или решение, основанное на данных.

Процесс построения дерева решений начинается с корневого узла, который содержит всю исходную информацию. Затем данные разделяются на подмножества в соответствии с определенным критерием, таким как пороговое значение или категория.

Пример дерева решений из банковской сферы:



В этом случае можно сказать, что решается задача бинарной классификации (целевой класс имеет два значения: "Выдать кредит" и "Отказать") по признакам "Возраст", "Наличие дома", "Доход" и "Образование".

Дерево решений как алгоритм машинного обучения – по сути то же самое: объединение логических правил вида "Значение признака a меньше x И Значение признака b меньше y… => Класс 1" в структуру данных «Дерево».

**LightGBM**

Это реализация градиентного бустинга с открытым исходным кодом, разработанная для того, чтобы быть эффективной и даже, возможно, более эффективной, чем другие реализации. Как таковой [LightGBM](https://github.com/Microsoft/LightGBM) – это проект с открытым исходным кодом, библиотека программного обеспечения и алгоритм машинного обучения. Отличия от других методов градиентного бустинга:

1. Повершинное построение деревьев – ускорение за счет структуры.
2. GOSS (Gradient-based One-Side Sampling). Используем меньше наблюдений.
3. EFB (Exclusive Feature Bunding). Используем меньше признаков.

***Повершинное построение деревьев:*** мы строим деревья не слой за слоем, а выбираем лист, в котором наш loss уменьшается сильнее, чем в остальных. Этот лист дальше разбиваем, растим, и получается, что у нас деревья могут быть несимметричные и глубокие. Этот алгоритм значительно быстрее, чем классические алгоритмы построения слой за слоем.

***Градиентная односторонняя выборка (GOSS)*** является модификацией градиентного бустинга, который фокусирует внимание на тех учебных примерах, которые приводят к большему градиенту, в свою очередь, ускоряя обучение и уменьшая вычислительную сложность метода. С помощью GOSS мы исключаем значительную долю экземпляров данных с небольшими градиентами и используем только остальные экземпляры для оценки прироста информации. Мы доказываем, что, поскольку экземпляры данных с большими градиентами играют более важную роль в вычислении информационного выигрыша, GOSS может получить довольно точную оценку информационного выигрыша с гораздо меньшим размером данных.

***Exclusive Feature Bundling (объединение взаимоисключающих признаков), или EFB****,* – это подход объединения разреженных (в основном нулевых) взаимоисключающих признаков, таких как категориальные переменные входных данных, закодированные унитарным кодированием. Таким образом, это тип автоматического подбора признаков. Мы пакетируем взаимоисключающие признаки (то есть они редко принимают ненулевые значения одновременно), чтобы уменьшить количество признаков.

Вместе эти три изменения могут ускорить время обучения алгоритма до 20 раз. Таким образом, LightGBM можно рассматривать как деревья решений с градиентным бустингом (GBDT) с добавлением GOSS и EFB.

**Разведочный анализ данных**

([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *exploratory data analysis, EDA*) – анализ основных свойств данных, нахождение в них общих закономерностей, распределений и аномалий, построение начальных моделей, зачастую с использованием инструментов [визуализации](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

Понятие введено математиком [Джоном Тьюки](https://ru.wikipedia.org/wiki/%D0%A2%D1%8C%D1%8E%D0%BA%D0%B8,_%D0%94%D0%B6%D0%BE%D0%BD), который сформулировал цели такого анализа следующим образом:

* максимальное «проникновение» в данные,
* выявление основных структур,
* выбор наиболее важных переменных,
* обнаружение отклонений и аномалий,
* проверка основных гипотез,
* разработка начальных моделей.

**Цели разведочного анализа данных**

*A. Понимание структуры и характеристик набора данных*

На этом этапе EDA мы пристально изучаем структуру и характеристики данных, чтобы полностью понять, с чем имеем дело. Это включает в себя обзор размера набора данных, типов переменных, наличия пропущенных значений, дубликатов и других важных аспектов. Разведка на этом уровне позволяет нам сформировать четкое представление о данных, что становится фундаментом для дальнейшего анализа.

*B. Выявление аномалий и выбросов*

Иногда данные могут хранить в себе сюрпризы, которые могут существенно повлиять на наши выводы. Используя методы EDA, мы обнаруживаем аномалии и выбросы в наших данных. Эти значения, отклоняющиеся от общего паттерна, могут возникать из-за ошибок ввода, случайных событий или даже указывать на систематические проблемы в сборе данных. Определение и устранение таких аномалий становится критической частью нашего анализа, чтобы гарантировать надежность результатов.

*C. Идентификация связей и корреляций между переменными*

Используя статистические меры, EDA позволяет идентифицировать взаимосвязи между переменными, что помогает понять, как одни факторы влияют на другие. Это может стать отправной точкой для формулирования гипотез и принятия важных бизнес-решений.

*D. Подготовка данных для дальнейших этапов анализа*

На этом этапе EDA мы обрабатываем данные для того, чтобы подготовить их к более сложным аналитическим методам. Мы чистим данные от шума, заполняем пропущенные значения, проводим масштабирование или преобразования переменных, чтобы обеспечить их качественную и интерпретируемую структуру. Только с надлежаще подготовленными данными мы можем достичь точных и надежных результатов на последующих этапах анализа.

**Инструменты и методы разведочного анализа данных (EDA)**

***A. Визуализация данных***

Визуализация данных позволяет нам увидеть и понять паттерны, тренды и взаимосвязи в данных через графику и диаграммы.

1. Гистограммы и диаграммы рассеяния.
2. Ящик с усами (box plot).
3. Тепловые карты (heatmap).

***B. Сводные статистики и меры центральной тенденции***

Сводные статистики и меры центральной тенденции позволяют нам получить обобщенное представление о распределении данных и основных характеристиках. Это ключевые числовые метрики, которые помогают нам понять типичные и наиболее значимые значения в наборе данных.

* **Среднее (Mean)**: Это сумма всех значений, разделенная на количество значений. Оно представляет общую «среднюю» величину данных.
* **Медиана (Median)**: Это среднее значение двух средних значений, если количество значений четное, или среднее значение самого центрального числа, если количество значений нечетное. Медиана предоставляет более устойчивую меру центральной тенденции в присутствии выбросов.
* **Мода (Mode)**: Это значение, которое встречается наиболее часто в наборе данных. Мода может быть полезна для определения наиболее типичного значения.

***C. Корреляционный анализ***

Корреляционный анализ помогает нам понять, какие переменные взаимосвязаны между собой и насколько сильна эта связь. Коэффициент корреляции измеряет степень линейной зависимости между двумя переменными.

* **Положительная корреляция**: Если одна переменная увеличивается, другая также увеличивается. Коэффициент корреляции находится в диапазоне от 0 до 1.
* **Отрицательная корреляция**: Если одна переменная увеличивается, другая уменьшается. Коэффициент корреляции находится в диапазоне от 0 до -1.
* **Нулевая корреляция**: Отсутствие линейной зависимости между переменными. Коэффициент корреляции близок к 0.

***D. Анализ выбросов и аномалий***

Анализ выбросов и аномалий – это процесс выявления и исследования значений данных, которые существенно отличаются от остальных наблюдений. Выбросы и аномалии могут возникнуть из-за ошибок в данных, случайных событий или указывать на особенности исследуемого явления.

Основные шаги анализа выбросов и аномалий:

* **Визуализация данных**: используйте графики, такие как ящик с усами (box plot) или диаграммы рассеяния, чтобы визуально выявить потенциальные выбросы.
* **Статистический анализ**: используйте статистические методы, чтобы определить, какие значения считаются выбросами на основе критериев, таких как интерквартильный размах или Z-оценка.
* **Принятие решения**: решите, какие действия необходимо предпринять с выбросами, например удалить их, заменить на другие значения или оставить без изменений.

***E. Преобразование данных (например, нормализация или стандартизация)***

Преобразование данных – это процесс изменения шкалы или распределения переменных, чтобы сделать их более подходящими для анализа или моделирования. Это важный этап EDA, который помогает сгладить различия между переменными и создать более устойчивые и интерпретируемые данные.

* **Нормализация (Normalization)**: Этот метод масштабирует значения переменных так, чтобы они находились в диапазоне от 0 до 1. Это особенно полезно, когда у нас есть переменные с разными единицами измерения и масштабами.
* **Стандартизация (Standardization)**: Этот метод преобразует значения переменных так, чтобы их среднее было равно 0, а стандартное отклонение – 1. Он делает распределение более «стандартным» и симметричным.

Преобразование данных может быть особенно полезным, когда у нас есть переменные с разными диапазонами значений, что может затруднять интерпретацию результатов.

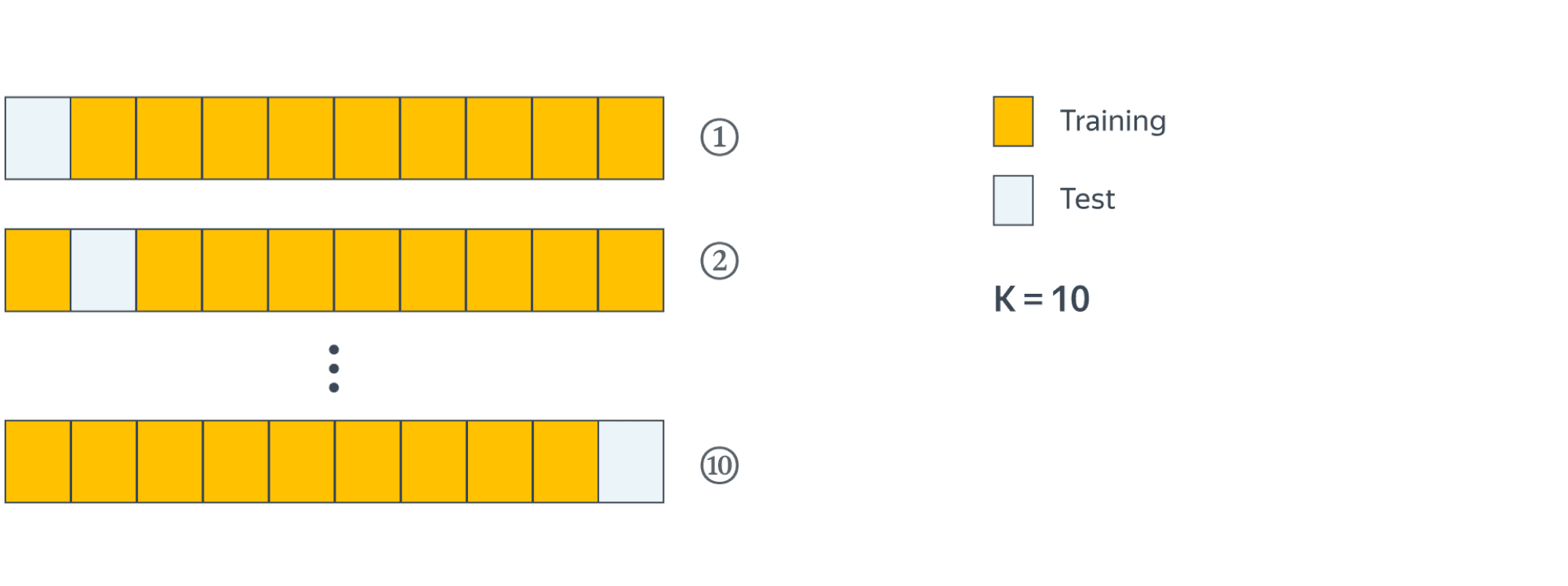
**Фут**

Это единица измерения длины в английской системе мер, равен 0,3048 м.

**Метод кросс-валидации k-Fold**

Перекрестная проверка (кросс-валидация, или скользящий контроль) – это статистический метод, используемый для оценки модели машинного обучения на независимых данных. Один из методов кросс-валидации – k-Fold. Он представляет из себя следующий алгоритм:

1. Фиксируется некоторое целое число k (обычно от 5 до 10), меньшее числа семплов в датасете.
2. Датасет разбивается на k одинаковых частей (в последней части может быть меньше семплов, чем в остальных). Эти части называются *фолдами*.
3. Далее происходит k итераций, во время каждой из которых один фолд выступает в роли тестового множества, а объединение остальных – в роли тренировочного. Модель учится на k−1 фолде и тестируется на оставшемся.
4. Финальный скор модели получается либо усреднением k получившихся тестовых результатов, либо измеряется на отложенном тестовом множестве, не участвовавшем в кросс-валидации.



**MSE**

Это метрика, которая сообщает нам среднеквадратичную разницу между прогнозируемыми значениями и фактическими значениями в наборе данных. Чем ниже MSE, тем лучше модель соответствует набору данных.

СКО = Σ(ŷ i – y i ) 2 / n

где:

* Σ – это символ, который означает «сумма»
* ŷ i – прогнозируемое значение для i-го наблюдения
* y i – наблюдаемое значение для i-го наблюдения
* n - размер выборки

**RMSE**

Это метрика, которая сообщает нам квадратный корень из средней квадратичной разницы между прогнозируемыми значениями и фактическими значениями в наборе данных. Чем ниже RMSE, тем лучше модель соответствует набору данных.

Он рассчитывается как:

СКО = √ Σ(ŷ i – y i ) 2 / n

где:

* Σ – это символ, который означает «сумма»
* ŷ i - прогнозируемое значение для i -го наблюдения
* y i - наблюдаемое значение для i -го наблюдения
* n - размер выборки

Обратите внимание, что формулы почти идентичны. На самом деле среднеквадратическая ошибка – это просто квадратный корень из среднеквадратичной ошибки.

**Куртозис**

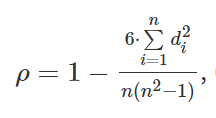
Куртозис – это статистическая мера, используемая для описания характеристик набора данных. Он предоставляет информацию о форме и распределении данных. Когда данные наносятся на график, обычно в виде колоколообразной кривой, эксцесс измеряет степень отклонения данных от нормального распределения. Он фокусируется на хвостах распределения, показывая, сколько данных находится в хвостах по сравнению с центром.

Куртозис измеряет суммарный вес хвостов по отношению к центру кривой распределения. В нормально распределенном наборе данных колоколообразная кривая симметрична, а точки данных сосредоточены вокруг среднего значения. Однако при высоком эксцессе хвосты распределения простираются дальше, чем при нормальном распределении. Это указывает на то, что в наборе данных больше экстремальных точек. Важно отметить, что эксцесс часто путают с показателем пикообразности, который означает, насколько резко данные достигают пика вокруг среднего значения. Куртозис, с другой стороны, фокусируется на хвостах распределения, описывая их форму и вес.

**Коэффициент ранговой корреляции Спирмена**

Ранговая [корреляция](https://wiki.loginom.ru/articles/correlation.html) представляет собой непараметрический (т. е. не зависящий от предположений о типе распределения исходных данных и его параметров) метод [корреляционного анализа](https://wiki.loginom.ru/articles/correlation-analysis.html), предназначенный для описания зависимости между [признаками](https://wiki.loginom.ru/articles/attribute.html), представленными в [порядковой (ранговой) шкале](https://wiki.loginom.ru/articles/scale-type.html). Ранги – это порядковые номера единиц наблюдения, упорядоченных по значениям исследуемого признака. Если ранжировать набор данных по двум признакам, зависимость между которыми требуется определить, то полное совпадение рангов означает максимально высокую прямую зависимость, а полная противоположность рангов – максимально высокую обратную зависимость. Ранжировать оба признака необходимо в одном и том же порядке: либо от меньших значений признака к большим, либо наоборот. Как и любой коэффициент [ранговой корреляции](https://wiki.loginom.ru/articles/rank-correlation.html), коэффициент Спирмена используется для обнаружения и описания статистической зависимости между признаками, а также проверки гипотез о наличии этой зависимости. Расчет коэффициента ранговой корреляции Спирмена включает следующие этапы:

1. Присвоить по каждому из признаков, зависимость между которыми требуется определить, порядковый номер (ранг) для каждого наблюдения по возрастанию или убыванию.
2. Определить разности рангов каждой пары сопоставляемых значений (d).
3. Возвести в квадрат каждую разность и суммировать полученные результаты.
4. Вычислить коэффициент ранговой корреляции по формуле:



где di – разность между рангами di=R(Xi)−R(Yi), n – число наблюдений, по которым вычисляется коэффициент.

**Практическая часть**

**Обзор**

В файле train.csv 79 независимых переменных, описывающих (почти) каждый аспект жилого дома в городе Эймсе, штат Айова. Набор данных о жилье в Эймсе был составлен Dean De Cock для использования в образовании в области науки о данных. Наша задача – предсказать цену продажи для каждого дома с использованием разных методов регрессии. Результаты оцениваются по среднеквадратичной ошибке (RMSE) между логарифмом прогнозируемого значения и логарифмом наблюдаемой цены продажи. (Использование логарифмов означает, что ошибки в прогнозировании дорогих и дешевых домов в равной степени повлияют на результат).

**Описание файлов с входными данными**

**train.csv** - обучающий набор;

**test.csv** - тестовый набор;

**data\_description.txt** – полное описание каждого столбца, которое было первоначально подготовлено Dean De Cock, но незначительно отредактировано, чтобы соответствовать именам столбцов здесь;

**sample\_submission.csv** – эталонное представление на основе линейной регрессии по году и месяцу продажи, площади участка и количеству спален.

**lasso\_sol22\_Median.csv** – результат обработки данных из файлов train и test с помощью регрессии lasso программиста Boris Klyus.

**House\_Prices\_submit.csv** – результат обработки данных из файлов train и test с помощью комбинации методов lasso, ridge, elastic net, GBM и LGB с последующим полным перебором (brutal force) программиста Aleksandrs Gehsbargs.

**hybrid\_solution.csv** – результат обработки данных из файлов train и test методом опорных векторов программиста Owen Ouyang.

**Поля данных в train.csv и test.csv**

Вот краткая версия того, что находится в файлах описания данных.

* **SalePrice** – цена продажи недвижимости в долларах. Это целевая переменная, которую мы пытаемся предсказать.
* **MSSubClass**: класс здания.
* **MSZoning**: общая классификация зонирования.
* **LotFrontage**: линейные футы улицы, соединенной с собственностью.
* **LotArea**: размер участка в квадратных футах.
* **Street**: тип подъезда к дороге.
* **Alley**: тип доступа в переулок.
* **LotShape**: общая форма недвижимости.
* **LandContour**: ровность участка.
* **Utilities**: типы доступных коммунальных услуг.
* **LotConfig**: конфигурация лота.
* **LandSlope**: уклон участка.
* **Neighborhood**: физическое местоположение в черте города Эймса.
* **Condition1**: близость к главной дороге или железной дороге.
* **Condition2**: близость к главной дороге или железной дороге (при наличии второй).
* **BldgType**: тип жилого помещения.
* **HouseStyle**: стиль жилища.
* **OverallQual**: общее качество материалов и отделки.
* **OverallCond**: общая оценка состояния.
* **YearBuilt**: первоначальная дата постройки.
* **YearRemodAdd**: дата реконструкции.
* **RoofStyle**: тип крыши.
* **RoofMatl**: кровельный материал.
* **Exterior1st**: наружное покрытие дома.
* **Exterior2nd**: наружное покрытие дома (если используется более одного материала).
* **MasVnrType**: тип облицовки каменной кладкой.
* **MasVnrArea**: площадь облицовки каменной кладки в квадратных футах.
* **ExterQual**: качество наружных материалов.
* **ExterCond**: текущее состояние материала снаружи.
* **Foundation**: тип фундамента.
* **BsmtQual**: высота подвала.
* **BsmtCond**: общее состояние подвала.
* **BsmtExposure**: стены подвала на уровне пешеходной дорожки или сада.
* **BsmtFinType1**: качество отделки цокольного этажа.
* **BsmtFinSF1**: тип 1 готовые квадратные футы.
* **BsmtFinType2**: качество второй готовой зоны (при наличии).
* **BsmtFinSF2**: тип 2 готовые квадратные футы.
* **BsmtUnfSF**: незаконченные квадратные метры площади подвала.
* **TotalBsmtSF**: общая площадь подвала в квадратных футах.
* **Heating**: тип отопления.
* **HeatingQC**: качество и состояние отопления.
* **CentralAir**: центральное кондиционирование воздуха.
* **Electrical**: электрическая система.
* **1stFlrSF**: площадь первого этажа в квадратных футах.
* **2ndFlrSF**: площадь второго этажа в квадратных футах.
* **LowQualFinSF**: низкое качество отделки квадратных метров (все этажи).
* **GrLivArea**: жилая площадь над землей (квадратные футы).
* **BsmtFullBath**: полностью оборудованные ванные комнаты на цокольном этаже.
* **BsmtHalfBath**: наполовину оборудованные ванные комнаты на цокольном этаже.
* **FullBath**: полностью оборудованные ванные комнаты выше уровня земли.
* **HalfBath**: наполовину оборудованные ванные комнаты выше уровня земли.
* **Bedroom**: количество спален над цокольным этажом.
* **Kitchen**: количество кухонь.
* **KitchenQual**: качество кухонь.
* **TotRmsAbvGrd**: общее количество комнат над уровнем земли (без учета ванных комнат).
* **Functional**: оценка функциональности дома.
* **Fireplaces**: количество каминов.
* **FireplaceQu**: качество каминов.
* **GarageType**: расположение гаража.
* **GarageYrBlt**: год постройки гаража.
* **GarageFinish**: внутренняя отделка гаража.
* **GarageCars**: размер гаража в количестве автомобилей, которые он вмещает.
* **GarageArea**: площадь гаража в квадратных футах.
* **GarageQual**: качество гаража.
* **GarageCond**: состояние гаража.
* **PavedDrive**: асфальтированная подъездная дорога.
* **WoodDeckSF**: площадь деревянной террасы в квадратных футах.
* **OpenPorchSF**: площадь открытой веранды в квадратных футах.
* **EnclosedPorch**: площадь крытой веранды в квадратных футах.
* **3SsnPorch**: площадь веранды на три сезона в квадратных футах.
* **ScreenPorch**: площадь веранды с экраном в квадратных футах.
* **PoolArea**: площадь бассейна в квадратных футах.
* **PoolQC**: качество бассейна.
* **Fence**: качество ограждения.
* **MiscFeature**: прочие свойства, не включенные в другие категории.
* **MiscVal**: стоимость других свойств в долларах США.
* **MoSold**: месяц продажи.
* **YrSold**: год продажи.
* **SaleType**: тип продажи.
* **SaleCondition**: условия продажи.

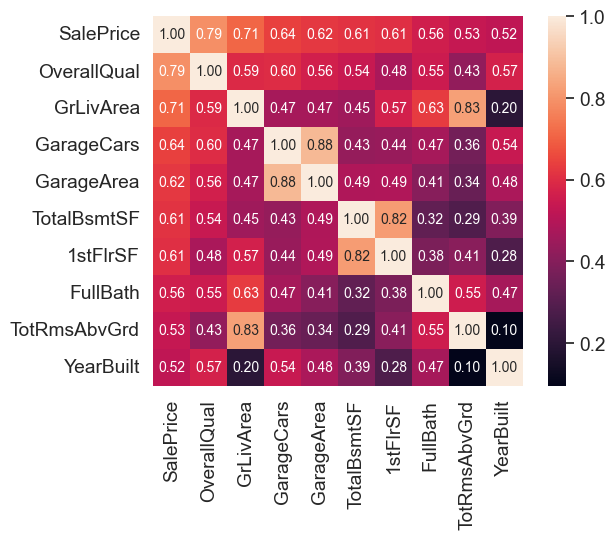
**Код содержится в приложенном файле «Дипломный проект - код.ipynb».**

**Разведочный и визуальный анализ данных**

Код написан на языке программирования **Python**. Первый шаг – загрузка библиотек: **nympy** – служит для математических вычислений; **pandas** – для работы с массивами данных; **scipy** – для глубоких и сложных научных вычислений, анализа данных и построения графиков; **sklearn** содержит функции и алгоритмы для машинного обучения; **matplotlib и seaborn** – для визуализации данных; **mlxtend** – для машинного обучения; **xgboost** – для решения задач, связанных c градиентным бустингом; **lightgbm** –для запуска одноименного метода машинного обучения. Далее – чтение данных из файлов train.csv и test.csv. Программа показывает, что в таблицах приводится значение 79 признаков около 1460 объектов недвижимости. Как показывает краткая описательная статистика цены продажи в файле train, минимальная цена продажи дома – 34 900 долл., средняя – около 181 000 долл., максимальная – 755 000 долл. Далее делим признаки в файле train на количественные и качественные.

***Визуальный анализ***

Строим матрицу корреляций, которая показывает, какие количественные признаки сильнее всего влияют на цену продажи:

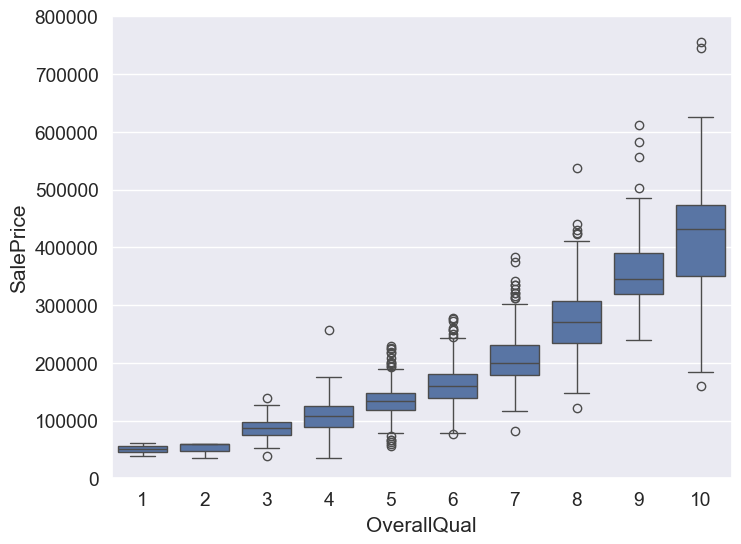


Видно, что заметно коррелируют с ценой (коэффициент корреляции от 0.79 до 0.52, в порядке убывания):

1. Общее качество материалов и отделки.
2. Жилая площадь над землей (квадратные футы).
3. Размер гаража в количестве автомобилей, которые он вмещает.
4. Площадь гаража в квадратных футах.
5. Общая площадь подвала в квадратных футах.
6. Площадь первого этажа в квадратных футах.
7. Количество полностью оборудованных ванных комнат выше уровня земли.
8. Общее количество комнат над уровнем земли (без учета ванных комнат).
9. Год постройки.

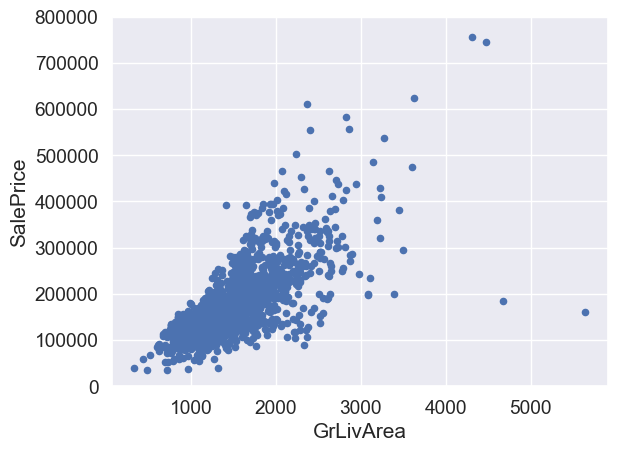
Количество автомобилей, помещающихся в гараже, зависит от площади гаража. GarageCars и GarageArea – как братья-близнецы. «Общая площадь подвала в квадратных футах» и «Площадь первого этажа в квадратных футах» – также практически одно и то же.

Теперь строим график типа box plot, показывающий взаимосвязь общего качества материалов и отделки с ценой продажи:



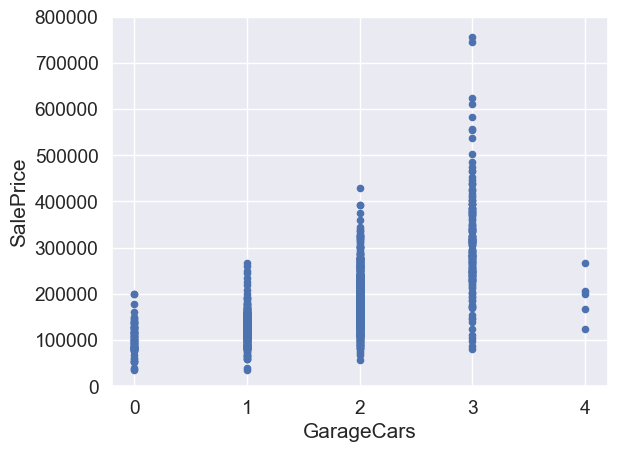
Видно прямую зависимость: средняя стоимость дома тем выше, чем выше качество материалов и отделки (медианы повышаются ступенчато), причем в интервале средней цены до 200 000 долларов находятся дома с качеством от 1 до 7, а недвижимость с качеством от 7 до 10 имеет среднюю цену от 200 000 до 450 000 долл., то есть дома с отличными и премиальными материалами и отделкой с повышением качества прибавляют в цене гораздо сильнее, чем дома со средним качеством.

Теперь строим точечный график взаимосвязи жилой площади над землей и ценой продажи дома:



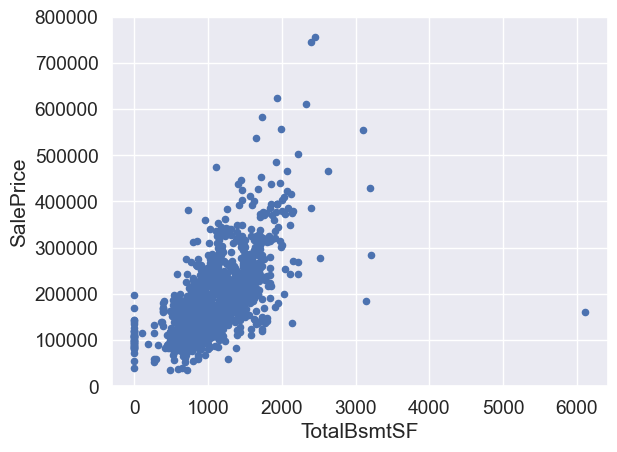
Видна четкая прямая зависимость: чем больше жилая площадь, тем выше цена дома. Два значения с большой величиной жилой площади и низкой ценой кажутся странными, и они нетипичны. Мы можем предположить, почему это происходит. Возможно, они относятся к сельскохозяйственному району, и это может объяснить низкую цену. Я не уверен в этом, но я совершенно уверен, что эти два пункта не являются типичными для типичного случая. Поэтому мы определим их как выбросы.

Далее строим точечный график взаимосвязи размера гаража в количестве автомобилей, которые он вмещает, и ценой продажи дома:



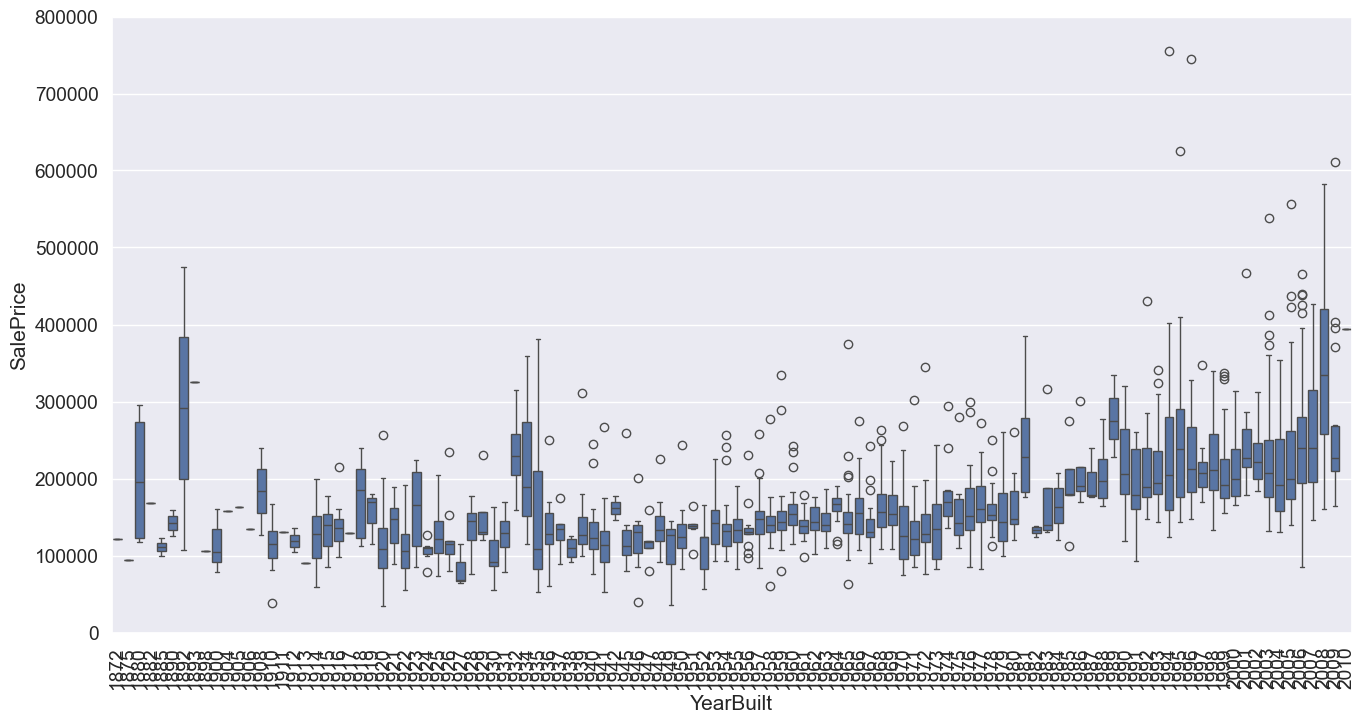
Сначала на графике всё ожидаемо – чем больше гараж, тем больше цена. Но с домами, которые имеют гараж на 4 машины, что-то необычное – их цена не выше 280 000 долларов (таких домов мало). Очевидно, это вызвано другими характеристиками этих домов.

График взаимосвязи общей площади подвала в квадратных футах и ценой продажи дома:



Видно, что с ростом общей площади подвала цена дома растет по экспоненте.

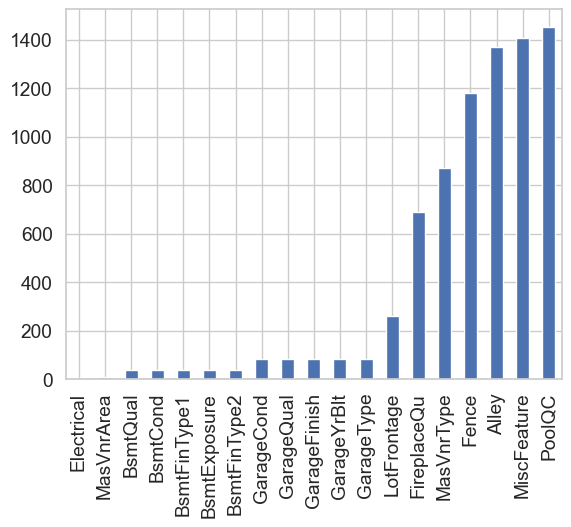
Посмотрим на взаимосвязь года постройки дома и цены продажи:



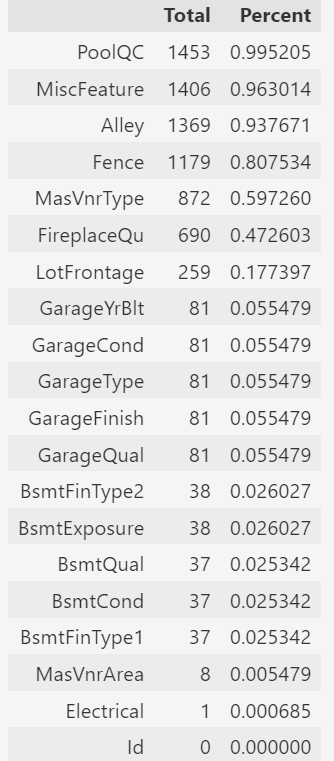
Видно, что самые старые дома в таблице данных – 1872 года постройки, самые новые – после 2010-го. Хотя это и не очень выраженная тенденция, я бы сказал, что покупатели чаще тратят больше денег на новые дома, чем на старые. Однако здесь мы не знаем, указана ли цена продажи в постоянных ценах. Постоянные цены пытаются устранить влияние инфляции. Если «Цена продажи» не указана в постоянных ценах, то она должна быть такой, чтобы цены были сопоставимы на протяжении многих лет.

**Анализ пропущенных данных**

Смотрим количество пропущенных значений в характеристиках недвижимости:



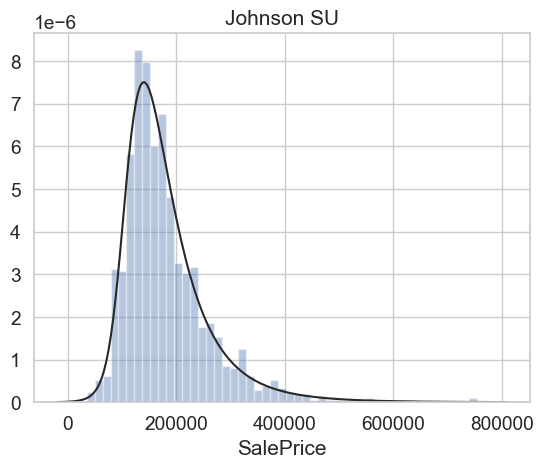
В 19 атрибутах отсутствуют значения, в пяти из них отсутствует более 50% всех данных. Конкретно:

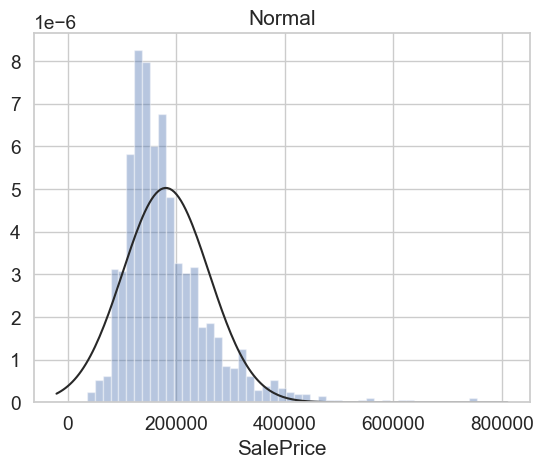


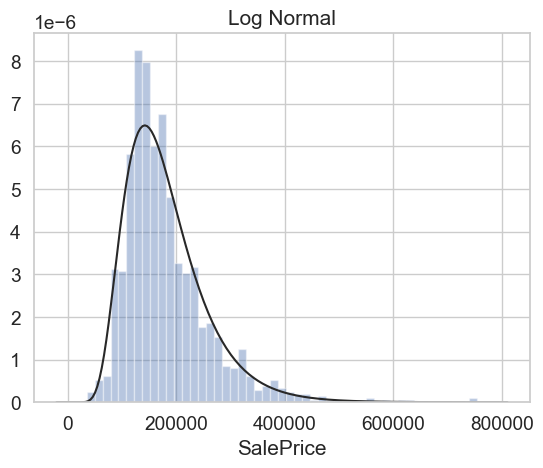
Больше 90% данных отсутствуют в признаках «Качество бассейна», «Прочие свойства, не включенные в другие категории», «Тип доступа в переулок». От 47 до 80% данных отсутствуют в признаках «Качество каминов», «Тип облицовки каменной кладкой», «Качество ограждения». В большинстве случаев NA означает отсутствие объекта, описываемого атрибутом, например отсутствие бассейна, забора. Ни одна из этих переменных не кажется мне очень важной, поскольку большинство из них не являются аспектами, о которых мы думаем при покупке дома (может быть, именно по этой причине отсутствуют данные?).

**Проверка данных в файле train.csv на нормальность распределения**

Проверяем цену продажи на нормальность распределения с помощью графиков:







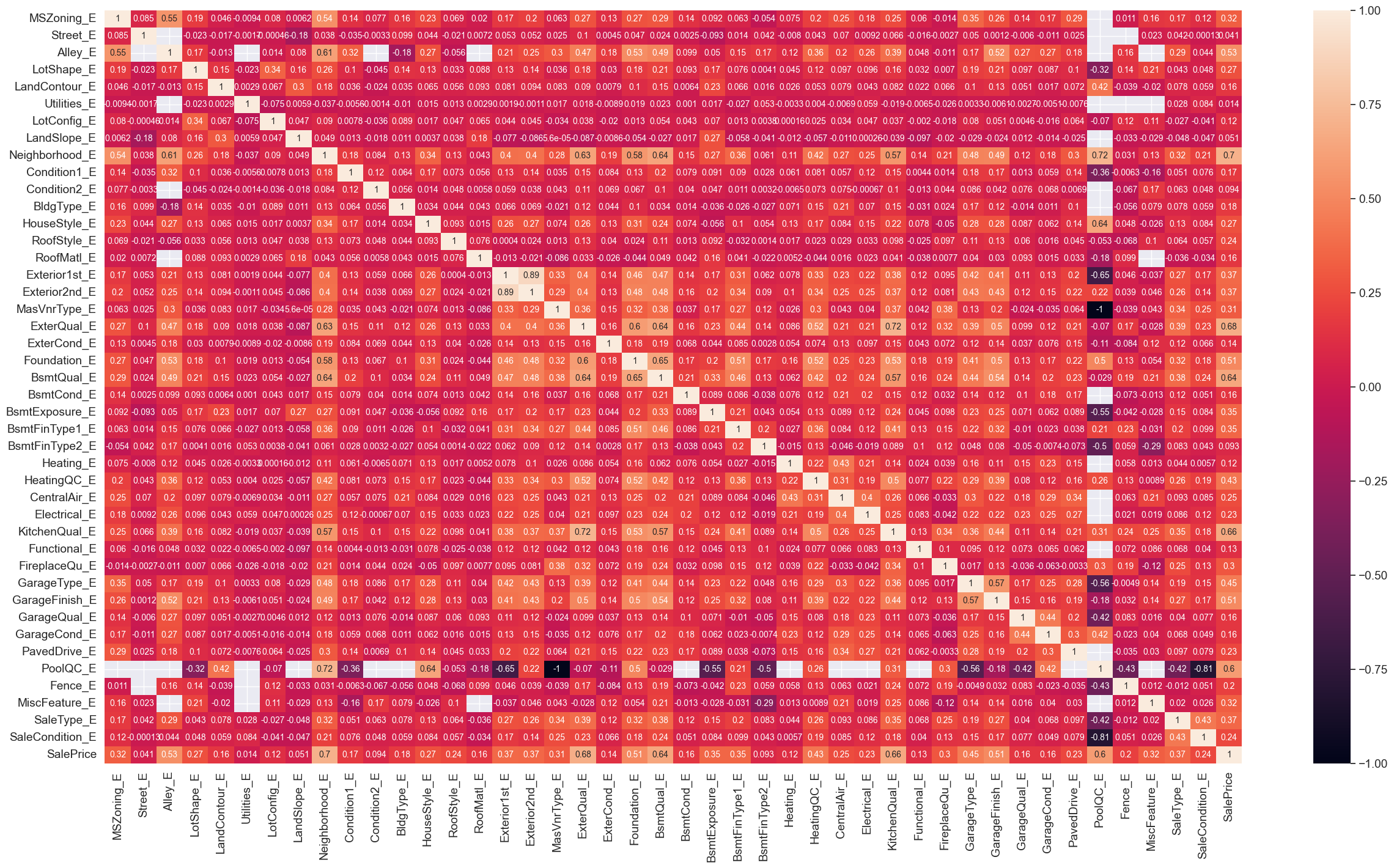
Очевидно, что цена продажи не соответствует нормальному распределению, она показывает «пиковую», положительную асимметрию и не соответствует диагональной линии. Поэтому перед выполнением регрессии ее необходимо преобразовать. Хотя логарифмическое преобразование работает довольно хорошо, лучше всего подходит неограниченное распределение Джонсона.

Тест Шапиро-Уилка на нормальность распределения количественных признаков недвижимости показывает, что ни одна из количественных переменных не имеет нормального распределения. Поэтому они также должны быть преобразованы.

В этом случае лучше работать с корреляцией Спирмена, поскольку она улавливает связи между переменными, даже если они нелинейны. «Общее качество материалов и отделки» – главный количественный критерий при установлении цены на дом.

Чтобы использовать категориальные признаки в алгоритмах машинного обучения, нужно их преобразовать в числовые значения. Это делается путем кодирования категориальных признаков. Создаем функцию кодирования категориальных признаков def encode(). Далее создаем функцию поиска парной корреляции столбцов по методу ранговой корреляции Спирмена def spearman(). Далее строим матрицы корреляций: количественных признаков с количественными, кодированных качественных с кодированными качественными, количественных с кодированными качественными.

***Проанализируем влияние качественных признаков на цену продажи:***



Видно, что заметно коррелируют с ценой (коэффициент корреляции от 0.7 до 0.5, в порядке убывания):

1. Физическое местоположение в черте города Эймса.
2. Качество наружных материалов.
3. Качество кухонь.
4. Высота подвала.
5. Качество бассейна.
6. Тип доступа в переулок.
7. Тип фундамента.
8. Внутренняя отделка гаража.

То есть «Физическое местоположение в черте города Эймса» – главный качественный критерий при установлении цены на дом.

**Обработка данных**

В наборах данных train и test удаляем столбец Id, так как он не нужен для процесса прогнозирования. Далее в датафрейме train удаляем выбросы критерия «Жилая площадь над землей (квадратные футы)».

Далее в файле train выполняем логарифмическое преобразование цен продажи. Для этогоиспользуем функцию numpy log1p, которая применяет log(1+x) ко всем элементам столбца. Логарифмическое преобразование помогает обрабатывать искаженные данные, и после преобразования распределение становится более приближенным к нормальному. Оно также снижает влияние выбросов за счет нормализации разницы величин, и модель становится более надежной.

Теперь объединяем наборы данных train и test в один набор данных, удаляя при этом столбец «Цена продажи». Получаем набор данных features и дальше работаем с ним.

Отсутствующие значения – одна из наиболее распространенных проблем, с которыми можно столкнуться при попытке подготовить данные. Этот фактор очень сильно влияет на производительность моделей машинного обучения.

Самое простое решение для пропущенных значений – отбросить строки или весь столбец. Однако заполнение пропущенных значений – более предпочтительный вариант, чем отбрасывание, потому что оно сохраняет размер данных. Следующими шагами заполняем пропущенные значения в наборе данных features.

Далее была сделана попытка преобразования Бокса-Кокса – это широко используемый метод преобразования ненормально распределенного набора данных в более [нормально распределенный](https://www.codecamp.ru/blog/the-normal-distribution/). Однако она привела к ошибке **BracketError**: The algorithm terminated without finding a valid bracket. Consider trying different initial points. Попытки исправить ошибку путем изменения кода и найте решение в интернете результатов не дали, поэтому код преобразования Бокса-Кокса закомментировал.

Теперь удаляем 3 столбца: PoolQC – поскольку в нем пропущено 99.5% данных; Utilities – корреляция с ценой 0.014, то есть отсутствует; Street – корреляция с ценой 0.041, то есть отсутствует.

Далее создаем новые категории и в каждую из них объединяем по несколько сходных признаков.

Теперь на основе данных набора features создаем новые категории – наличия бассейна, второго этажа, гаража, подвала, камина (отталкиваясь от исходных категорий площади бассейна, второго этажа, гаража, подвала; количества каминов).

Далее с помощью [функции pandas.get\_dummies()](https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html?ref=codecamp.ru) преобразуем категориальные переменные в фиктивные (числовые) переменные. Удаляем выбросы.

**Создание, обучение и оценка моделей**

Теперь создаем функцию оценки наших моделей методом среднеквадратической ошибки RMSE на основе кросс-валидации k-Fold. Чем ниже RMSE, тем лучше модель соответствует набору данных.

Задаем настройки моделей (их значения подобраны методом проб и ошибок).

Далее создаем последовательность шагов обработки данных (конвейер машинного обучения) с помощью функции make\_pipeline с масштабированием данных RobustScaler для разных методов регрессии – ridge, lasso, elasticnet, svr, gradient boosting, LightGBM, xgboost.

Выполняем RMSE-скоринг моделей регрессии. Результаты:

LASSO: 0.1037

elastic net: 0.1035

SVR: 0.1710

lightgbm: 0.1057

gbr: 0.1076

xgboost: 0.1047

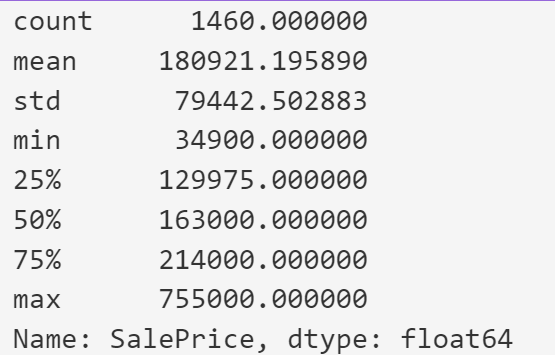
Видим, что самые точные методы – elastic net и LASSO, средняя точность – у xgboost и lightgbm, и наименьшая точность (с большим отрывом) – у SVR.

Далее с помощью функции fit() выполняем фактическое обучение моделей на входных данных.

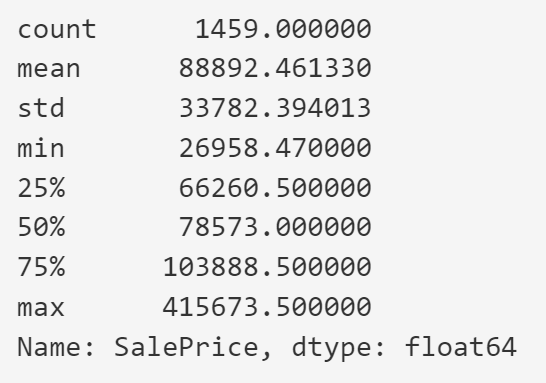
**Смешение моделей и предсказание цены**

Теперь создаем функцию смешивания разных методов регрессии (вес каждой модели в смешении можно варьировать). При весах, содержащихся в коде, RMSE получается приблизительно 0,05. То есть смешение методов регрессии имеет предсказательную способность более чем в 2 раза точнее, чем самый точный одиночный метод.

Теперь с помощью полученной комбинированной модели предсказываем цену – на основе исходных данных из файлов train.csv, test.csv, sample\_submission.csv. Записываем результаты предсказания в файл incorrect\_prediction.csv. Почему я его так назвал? Сравним статистику цены продажи в исходном файле train.csv:



и статистику предсказанной цены в файле incorrect\_prediction.csv:



Расхождение очень сильное. Как исправить ситуацию?

**Смешение комбинированной модели с лучшими ядрами Kaggle**

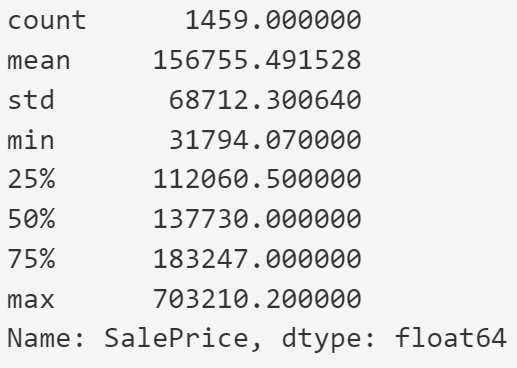
Берем с Kaggle лучшие результаты обработки тех же данных:

**lasso\_sol22\_Median.csv** – результат обработки данных из файлов train.csv и test.csv с помощью регрессии lasso программиста Boris Klyus.

**House\_Prices\_submit.csv** – результат обработки данных из файлов train.csv и test.csv с помощью комбинации методов lasso, ridge, elastic net, GBM и LGB с последующим полным перебором (brutal force) программиста Aleksandrs Gehsbargs.

**hybrid\_solution.csv** – результат обработки данных из файлов train.csv и test.csv методом опорных векторов программиста Owen Ouyang.

Делаем комбинацию из трех этих результатов с результатом нашей модели, веса задаем равные – по 25%. Результат работы этой комбинации записываем в файл submission.csv. Смотрим описательную статистику предсказанной цены в этом файле:



Вот эти значения уже гораздо ближе к исходным. Использование результатов лучших кодов других программистов в качестве входных данныхзначительно повышает точность прогноза.

**Заключение**

Был проведен подробный разбор теоретических понятий, затронутых в исследовании; проделан визуальный и разведочный анализ исходных данных. Сделана подготовка данных и их обработка. На этих данных были обучены модели, основанные на разных видах регрессии. Установлено, что комбинация разных методов регрессии предсказывает в 2 раза точнее, чем самый точный одиночный метод. Однако предсказанные цены всё равно получились далекими от исходных, и, чтобы повысить точность прогноза, пришлось добавить к полученной смешанной модели результаты работы других программистов с Kaggle.

**Список использованных источников**

<https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview>

<https://www.kaggle.com/code/jesucristo/1-house-prices-solution-top-1/notebook>

<https://www.kaggle.com/code/itslek/blend-stack-lr-gb-0-10649-house-prices-v57/script>

https://www.kaggle.com/code/pmarcelino/comprehensive-data-exploration-with-python

<https://www.kaggle.com/code/zavodrobotov/lasso-model-for-regression-problem>

<https://www.kaggle.com/code/agehsbarg/top-10-0-10943-stacking-mice-and-brutal-force>

<https://www.kaggle.com/code/couyang/hybrid-svm-benchmark-approach-0-11180-lb-top-2>

<https://machinelearningmastery.com/linear-regression-for-machine-learning/>

<https://blog.skillfactory.ru/glossary/linejnaya-regressiya/>

<https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F>

<https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)>

<https://cloud.vk.com/blog/-5-algoritmov-regressii-v-mashinnom-obuchenii#:~:text=%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F%20LASSO&text=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%20%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%B8%20%D0%BB%D0%B0%D1%81%D1%81%D0%BE%20(LASSO%2C%20Least,%D0%BA%D0%BE%D1%80%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D1%8E%20%D0%BF%D1%80%D0%B8%D0%B7%D0%BD%D0%B0%D0%BA%D0%BE%D0%B2%20%D0%B4%D1%80%D1%83%D0%B3%20%D1%81%20%D0%B4%D1%80%D1%83%D0%B3%D0%BE%D0%BC)>.

<https://neurohive.io/ru/osnovy-data-science/gradientyj-busting/>

<https://spark-school.ru/blog/gradient-boosting-ml/>

<https://habr.com/ru/companies/ods/articles/322534/>

<https://ru.wikipedia.org/wiki/XGBoost>

<https://www-ibm-com.translate.goog/topics/regularization?_x_tr_sl=en&_x_tr_tl=ru&_x_tr_hl=ru&_x_tr_pto=rq#:~:text=Regularization%20is%20a%20set%20of,overfitting%20in%20machine%20learning%20models>.

<https://sky.pro/media/chto-takoe-derevya-reshenij-i-kak-ih-ispolzovat-v-analitike-dannyh/#:~:text=%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%20%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D0%B8%D1%82%20%D0%B8%D0%B7%20%D1%83%D0%B7%D0%BB%D0%BE%D0%B2,%D0%B8%D0%BB%D0%B8%20%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5%2C%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5%20%D0%BD%D0%B0%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85>.

<https://neptune.ai/blog/xgboost-vs-lightgbm>

<https://habr.com/ru/companies/otus/articles/752434/>

<https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%B2%D0%B5%D0%B4%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85>

<https://www.codecamp.ru/blog/mse-vs-rmse/>

<https://python-school.ru/blog/osnovy-ml/linearregression-ols/>

<https://python-school.ru/blog/osnovy-ml/regularization-l1-l2/>

<https://habr.com/ru/articles/428503/>

<https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D1%82>

<https://www.codecamp.ru/blog/cross-validation-k-fold/>

<https://education.yandex.ru/handbook/ml/article/kross-validaciya>

<https://nesrakonk.ru/kurtosis/>

<https://www.codecamp.ru/blog/shapiro-wilk-test-python/>

<https://habr.com/ru/companies/karuna/articles/769366/>

<https://wiki.loginom.ru/articles/rank-correlation.html>

<https://wiki.loginom.ru/articles/rank-correlation-spearman.html#koeffitsient-rangovoy-korrelyatsii-spirmena-spearmans-rank-correlation-coefficient>

<https://www.codecamp.ru/blog/pandas-get-dummies/>

<https://medium.com/@TheDataScience-ProF/simplifying-machine-learning-workflows-with-scikit-learns-make-pipeline-19cb80ee3796>

<https://habr.com/ru/companies/tochka/articles/751012/>

<https://habr.com/ru/companies/skillfactory/articles/530594/>

<https://www-geeksforgeeks-org.translate.goog/lightgbm-light-gradient-boosting-machine/?_x_tr_sl=en&_x_tr_tl=ru&_x_tr_hl=ru&_x_tr_pto=rq#:~:text=The%20advantages%20of%20the%20LightGBM,datasets%20and%20time%2Dsensitive%20applications>

<https://4brain.ru/aibasics/data.php#:~:text=%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5%20%D0%BF%D1%80%D0%B8%D0%B7%D0%BD%D0%B0%D0%BA%D0%B8%20%D0%BC%D0%BE%D0%B3%D1%83%D1%82%20%D0%B1%D1%8B%D1%82%D1%8C%20%D0%BF%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D1%8B,%D0%B4>

<https://proglib.io/p/postroenie-i-otbor-priznakov-chast-1-feature-engineering-2021-09-15>

<https://www.codecamp.ru/blog/box-cox-transformation-python/>

**Приложения**

Дипломный проект - код.ipynb

*Исходные файлы:* train.csv, test.csv, sample\_submission.csv;

lasso\_sol22\_Median.csv, hybrid\_solution.csv, House\_Prices\_submit.csv.

*Результаты работы кода:* incorrect\_prediction.csv, submission.csv.