3. ПОСТРОЕНИЕ МОДЕЛИ ОЦЕНКИ ОБЪЕКТОВ ЖИЛОЙ МНОГОКВАРТИРНОЙ НЕДВИЖИМОСТИ МЕТОДАМИ МАШИННОГО ОБУЧЕНИЯ

3.1 Выбор модели для последующего обучения

1.1.1 Моделирование числовых величин. Существующие алгоритмы

Многие страны давно ставят задачу получения стоимости недвижимого имущества не только человеческими ресурсами, но и методами машинного обучения. В управлении оценки уже многие годы только и говорят, чтобы оценить недвижимость без рук человека. Возобновление пилотного проекта как нельзя лучше подходит для реализации этих давних задумок. Таким образом, одним из направлений исследований в пилотном проекте 2019, является поиск альтернативных способов для получения стоимости объектов недвижимости на территории всей страны. Также стоит отметить, что данные для построения модели были выбраны только по жилой многоквартирной недвижимости. В этом нет ничего удивительного, так как квартиры представлены на рынке в абсолютном превосходстве над объектами остальных функциональных назначений.

Перед тем, как выбрать модель и начать её обучать, есть несколько важных этапов, без которых построение самой модели попросту не имеет смысла. Первое, что стоит сделать, это выбрать метод машинного обучения. Разделяют обычно 3 (три) основные категории: контролируемое (обучение с учителем), неконтролируемое (обучение без учителя) и подкрепляемое (смешанное) обучение. Обучение с учителем полезно в тех случаях, когда значение, которое необходимо смоделировать (например, стоимость), известно для некоторого набора данных (его называют обучающее множество) и неизвестно для основного набора данных, ведь для них оно должно быть предсказано. Обучение без учителя используется для поиска неявных связей в немаркированном наборе данных. Стандартные задачи для обучения без учителя – кластеризация или выявление групп объектов, поиск ассоциативных правил (анализ потребительской корзины) и т.д. Обучение с подкреплением – это что-то среднее между обучением с учителем и без. При таком методе машинного обучения модель не сразу получает правильный ответ, а стремиться прийти к нему шаг за шагом. Если перевести это на простой язык, то представьте, что вам необходимо научить кошку фокусам. Так как сразу вы этого не сделаете, то будете учить кошку по чуть-чуть и за каждое правильное действие давать молоко и хвалить, а за каждое неверное действие не делать с кошкой ничего (не давать молоко и не наказывать). Таким образом, через некоторое время, ваша кошка будет показывать фокусы.

Анализируя методы машинного обучения, несложно сделать вывод о том, что моделирование стоимости недвижимости – это обучение с учителем. Ведь есть тренировочное множество (сделки), у которых необходимый параметр (стоимость за кв.м) присутствует. Так же, есть набор данных, которым этот параметр необходимо предсказать. Этими объектами являются зарегистрированные объекты в Советском районе г. Минск и Кобринском районе.

После того, как стало известно, каким методом решается поставленная задача, нужно определиться с видом решаемой задачи. Если не сильно углубляться, то среди задач обучения с учителем можно выделить 3 (три) основные вида задач: классификация (двухклассовая и многоклассовая), регрессия и фильтрация выбросов. Классификация используется в том случае, когда нужно спрогнозировать, к какой категории относится объект. Например, по запросу к сайту определить, женщина или мужчина делают запрос. При этом, если категории две, это двухклассовая классификация, если категорий больше двух – многоклассовая классификация. Так же, не стоит путать классификацию с кластеризацией (одна из задач при обучении без учителя). При решении задач классификации количество групп заранее известно, тогда, как при решении задач кластеризации количество групп объектов определяется непосредственно на этапе построения модели. Задачи регрессии возникают тогда, когда необходимо смоделировать некую величину, обычно числовую (курс валют, стоимость и др.). Название «фильтрация выбросов» говорит само за себя. Задачи такого вида решают в тех случаях, когда нужно определить необычные объекты в данных. Это может быть поиск мошенничества, идентификация поломки оборудования и т.п. Что же касается задачи моделирования стоимости квартиры, бесспорно – это регрессия. Теперь, когда появилось полное понимания категории и вида решаемой задачи, можно приступать к поиску модели, которую необходимо обучить.

Лучший способ выбрать подходящую модель – это подсмотреть, как похожую или такую же задачу решали другие специалисты в других странах и какие результаты у них получились.

Поиск информации о том, какую модель выбрать и на каких данных строить, привёл к вполне ожидаемому результату. Разные страны и даже одни и те же города в одной стране использовали абсолютно разные алгоритмы и получали результаты, которые их удовлетворяли. Например, для моделирования стоимости недвижимости в г. Бостоне, штате Айова и для компании «Airbnb» использовались простые и быстрые модели: деревья решений, линейный регрессор и метод k-ближайших соседей соответственно. В тоже время, для определения стоимости недвижимости в г. Бостон, неизвестном городе и г. Москва использовались сложные и медленные алгоритмы: градиентный бустинг и случайных лес. Такие алгоритмы дают высочайшую точность. Основными критериями при выборе модели для обучения в пилотном проекте 2019 были максимально возможная точность. Об интерпретации модели изначально речи не шло, при этом бессмысленно было строить модель, которую потом никаким образом нельзя было бы объяснить. В итоге, было принято решение использовать модель градиентного бустинга на основе деревьев. Подробнее о плюсах и минусах модели, получаемой точности и временных затратах описывается в следующих разделах.

1.1.2 Простейшие модели машинного обучения. Деревья решений

Некоторые понятия и определения

Перед тем, как описывать модели, необходимо дать некоторые базовые понятия, которые используются в машинном обучении и, в частности, при решении задач регрессии. Как известно из предыдущего раздела, моделирование стоимости относится к задачам обучения с учителем. При этом, есть набор данных, для которого моделируемая величина известна. Таким набором данных являются сделки с объектами недвижимости. Опишем этот набор языком машинного обучения.

Матрицей **X** будем называть таблицу объектов-признаков (объектов-факторов), которая имеет размеры **m** × **n** (Рисунок 3.1.1), где

m – количество объектов в обучающем множестве или количество строк в матрице **X**;

n – количество факторов, которыми обладают объекты в обучающем множестве или количество столбцов в матрице **X**.

Матрица **X** представляется в следующем виде:

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$, где $\{x_{i,j}\}$ – это j-ый фактор у i-ого объекта.

Вектором **Y** называется вектор целевой переменной или, другими словами, того фактора, который необходимо смоделировать. Вектор **Y** имеет размерность **m** × **1** (Рисунок 1), где

m – количество объектов в обучающем множестве или количество строк в векторе **Y**.

Вектор **Y** имеет один столбец и представляется следующим образом:

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$, где $\{y_i\}$ – это целевой фактор для i-ого объекта в обучающем множестве.

При решении нашей задачи $\{y_i\}$ – это стоимость i-ого объекта недвижимости в сделках.

nadzem_amount	general_sq	rooms_amount	floor	kitchen_sq	height	new_balkon	new_hot_water	new_kladovaya	new_ventil	mat_sten	price_kvm
15	59.1	2	1	10.20	2.7 - 3	1	1	0	Естественная вентиляция	Кирпич, блоки легкобетонные	1029.85
5	82.5	4	1	8.50	до 2.7	2	1	2	Естественная вентиляция	Сборный железобетон	430.30
5	37.3	1	1	9.89	до 2.7	1	1	1	Естественная вентиляция	Сборный железобетон	471.88
5	54.8	2	5	8.80	до 2.7	1	1	0	Естественная вентиляция	Сборный железобетон	643.87
5	36.8	1	1	9.10	до 2.7	1	1	0	Естественная вентиляция	Сборный железобетон	588.27
2	50.0	2	1	7.40	до 2.7	1	1	0	Естественная вентиляция	Сборный железобетон	489.00
4	40.7	1	2	10.40	2.7 - 3	1	1	0	Естественная вентиляция	Кирпич, блоки легкобетонные	637.34

X **Y**

Рисунок 3.1.1 – Вид таблицы с объектами и факторами с целевой переменной

Линейная модель

Целью линейной регрессии является поиск наилучшей прямой линии (а если признаков больше одного, то наилучшей плоскости), которая соответствует конкретному набору объектов (Рисунок 3.1.2).

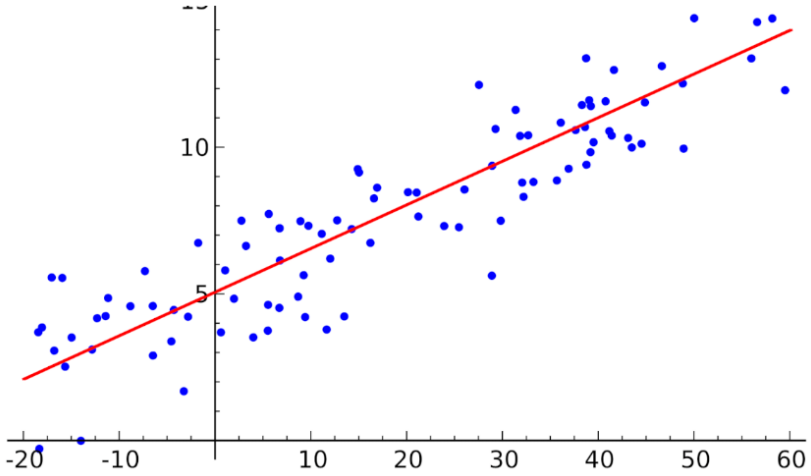

Рисунок 3.1.2 – Линейная модель на плоскости

Общее же уравнение прямой есть $\mathbf{Y} = \mathbf{X} \cdot \boldsymbol{\omega} + \mathbf{e}$, где

X – матрица признаков;

Y – вектор целевой переменной;

$\boldsymbol{\omega}$ – вектор параметров модели размерности $n \times 1$. Эти параметры ещё называют весами модели. Количество элементов в $\boldsymbol{\omega}$ соответствует количеству факторов в модели;

e – вектор переменных размерности $m \times 1$, которые соответствуют ошибке модели.

Если расписать уравнение для каждого конкретного объекта, то получится следующее выражение:

$$y_i = \sum_{j=1}^{n} \omega_j X_{i,j} + e_i \qquad (3.1.1)$$

Другими словами, нужно перемножить веса модели (ω_i) на соответствующие факторы ($X_{i,j}$) и сложить их, прибавить ошибку (e_i) для объекта. Поэтому, суть алгоритма сводится к тому, чтобы, изменяя веса $\boldsymbol{\omega}$, уменьшить ошибку **e** на столько, насколько это возможно. Так же, при построении линейной модели желательно все факторы преобразовать к единой системе координат, то есть, нормировать. Эта процедура делается для того, чтобы, например, не

складывать площадь, возраст и заработную плату. Так как все эти величины имеют разные порядки цифр.

Зачастую, линейную модель усложняют и делают её полиномиальной. Общее уравнение превращается в полиномиальное: $\mathbf{Y = X \cdot \omega_1 + X^2 \omega_2 + \ldots + e}$. К такой модели прибегают, когда зависимость факторов от целевой переменной имеет некую нелинейную зависимость (Рисунок 3.1.3).

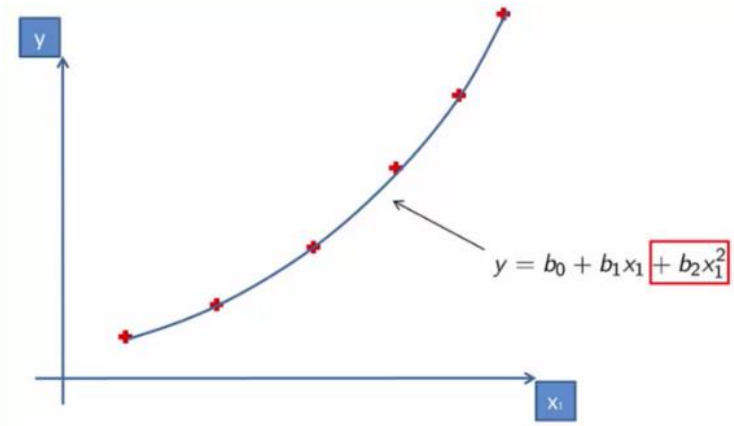

Рисунок 3.1.3 – Пример полиномиальной модели

Плюсом линейной модели является её простота и низкие временные затраты на обучение и работу. Но такая модель хорошо решает только те задачи, где в обучающем множестве факторов не очень много и все они имеют линейную зависимость (в каком-то приближении) с целевой переменной. Однако, несмотря на то, что существует множество моделей, которые намного точнее и эффективнее, линейную модель продолжают использовать во многих задачах. Это связано с тем, что прежде, чем строить более сложную модель, обучают простую модель, чтобы выявить коллинеарные факторы и нижний порог точности модели. Такой подход зачастую экономит много времени для аналитика ещё на ранних этапах.

Метод k-ближайших соседей

Метод kNN (k-nearest neighbor или k-ближайших соседей) – это один из самых простых и понятных алгоритмов, который часто используют для решения задач классификации и регрессии. Наряду с линейной моделью, kNN является отличным примером алгоритма, с которого новички начинают свой путь в мир машинного обучения. Простота модели заключается в её алгоритме. Чтобы решить задачу регрессии, необходимо выполнить несколько несложных операций:

1. Выбрать объект, которому необходимо предсказать целевую переменную (стоимость).
2. Вычислить расстояние до каждого из объектов обучающей выборки.
3. Отобрать k объектов обучающей выборки, расстояние до которых минимально.
4. Целевая переменная для выбранного объекта – это среднее арифметическое среди k ближайших объектов (соседей).

Если же решать задачу классификации, то в пункте 4 вместо нахождения среднего арифметического, нужно найти наиболее часто встречающийся класс среди k ближайших соседей. Это и будет класс выбранного объекта (Рисунок 3.1.4).

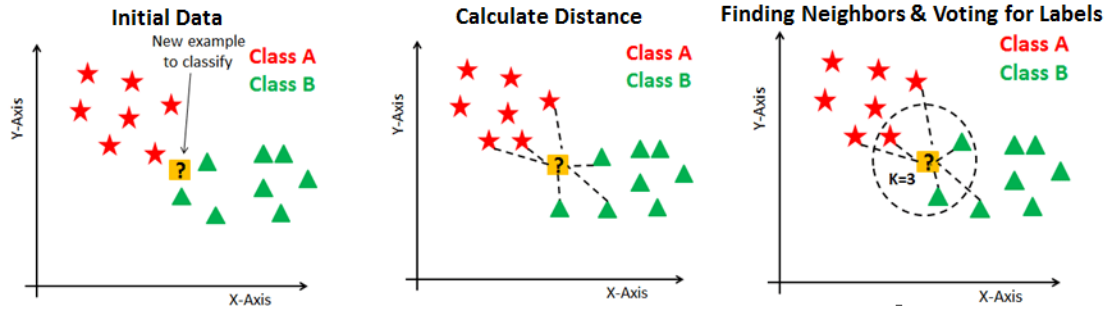

Рисунок 3.1.4 – Пример работы алгоритма kNN для задачи классификации

Не смотря на простоту алгоритма, модель kNN имеет несколько подводных камней, не игнорировать которые попросту нельзя. Во-первых, это выбор функции для нахождения расстояния. Самым простым решением является использовать Евклидово расстояние (птичий полёт), Манхеттенское (расстояние городских кварталов) или расстояние Минковского как общий случай (Формулы 3.1.2-3.1.4). Однако, не во всех задачах целесообразно использовать простейшее решение. Например, почти в каждом обучающем множестве есть категориальные признаки (материал стен, цвет и т.п.). Над такими признаками невозможно проводить какие-либо арифметические операции (сложение, умножение и т.д.). Поэтому, чтобы хоть как-то вычислять расстояние между категориальными признаками, используют расстояние Хемминга (Формула 3.1.5). Смысл его в том, чтобы считать количество различных факторов между двумя объектами.

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$ – формула Евклидового расстояния между двумя векторами X и Y (3.1.2)

$$\sum_{i=1}^{k}|x_i - y_i|$$ – формула Манхеттенского расстояния между двумя векторами X и Y (3.1.3)

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$ – формула расстояния Минковского между двумя векторами X и Y (3.1.4)

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$ – расстояние Хемминга между векторами X и Y (3.1.5)

Если $x_i = y_i$, то $|x_i - y_i| = 0$, если $x_i \neq y_i$, то $|x_i - y_i| = 1$

при этом, x_i и y_i – *i*-ые элементы векторов X и Y соответственно; k – размерность векторов

Второй особенностью модели kNN является обязательное нормирование числовых признаков. Это абсолютно логичное и правильное замечание. Например, если необходимо найти расстояние между объектами, у которых есть факторы заработной платы и возраста, то возраст имеет порядок до сотен, в то время, как зарплата может достигать порядка десятков тысяч и будет вносить намного больший вклад, нежели возраст. Самым простым и часто используемым способом нормирования факторов является отображением на отрезок [0, 1]. Это значит, что минимальное значение фактора становится нулём, а максимальное значение фактора – единицей, все остальные значения фактора принимают значения от 0 до 1. Такой способ преобразования числовых признаков позволяет находить расстояния между объектами, не смотря на разность в разрядах. Например, нормирование вектора с факторами F имеет следующий вид:

$$f_i^{new} = \frac{f_i - \min(F)}{\max(F) - \min(F)}, \quad (3.1.6)$$

где f_i – i-ый фактор вектора F.

Стоит так же отметить тот факт, что часто метод kNN путают с методом k-средних. Метод k-средних – это один из алгоритмов решения задачи кластеризации (разбиения на группы), которые относятся к задачам обучения без учителя, то есть, без наличия целевого признака. В то время, как метод kNN используется для решения задач, в которых есть целевой признак (обучение с учителем).

Деревья решений

Ещё одним популярным методом в машинном обучении является дерево решений. В отличии от линейной модели и метода k-ближайших соседей, деревья решений используются в повседневной жизни в самых разных областях человеческой деятельности. Деревом решений ещё можно назвать инструкцию или план действий к какой-либо ситуации. Так же, дерево решений интерпретируют как обобщение опыта экспертов, средство передачи знаний будущим сотрудникам или моделью бизнес-процесса компании. Самый простой и очевидный пример – это выдача кредита банком (Рисунок 3.1.5). До внедрения масштабируемых алгоритмов машинного обучения в банковской сфере, задача кредитного скоринга решалась специалистами, у которых было множество таблиц (инструкций) и решение о выдаче кредита принималось на основе правил, которые можно представить в виде дерева решений.

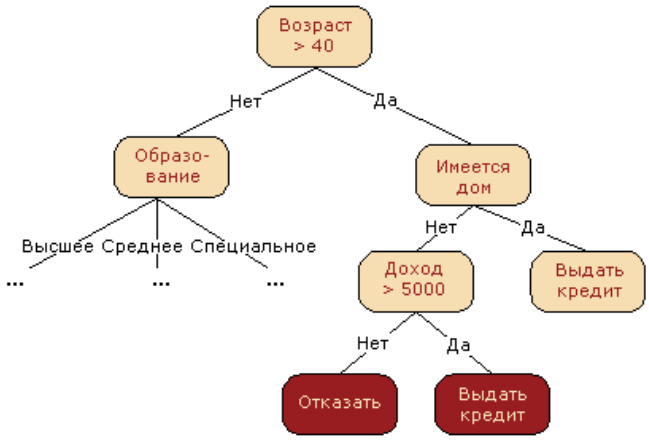

Рисунок 3.1.5– Пример дерева решений при выдаче кредита

Дерево решений как алгоритм машинного обучения – это по сути тоже самое: объединение логических правил вида «если…, то…; если…, то… и т.д.». Само же дерево представляет из себя набор узлов (условий), листьев (результатов) и рёбер (Рисунок 3.1.6). Узел – это условие перехода, определяющее, по какому из рёбер идти, то есть, узлом можно назвать вопрос, на который нужно дать ответ «Да» или «Нет» («возраст >40?», «имеется дом?», «возраст здания >20?» и т.д.). Ребро – это ответ на вопрос из узла («Да» или «Нет»). Другими словами, это переход из узла в другой узел либо в лист. В свою очередь, в листьях стоят значения целевой функции. Если ответом из узла является лист, то эта ветка с вопросами заканчивается, то есть, лист – это ответ на последний вопрос в ветке («Отказать», «Выдать кредит»).

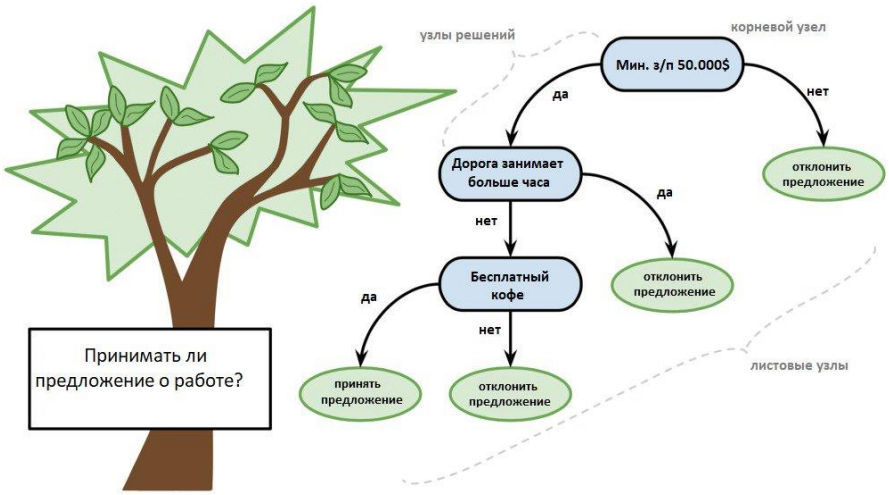

Рисунок 3.1.6– Пример дерева решений при выборе работы

Если при решениях задач классификации, в листовых узлах находятся результирующие классы, то при решении задач регрессии в листах находятся какие-то значения целевой функции (Рисунок 3.1.7). Однако, это не всегда так. Когда объединяют множество деревьев решений при построении более сложной модели, в листовых узлах находятся не значения целевой функции, а некие корректирующие значения, которые применяются последовательно к какому-то начальному приближению целевой переменной. Подробнее о построении таких моделей изложено в разделе «**Объединение простейших моделей для создания более точной модели. Бустинг**».

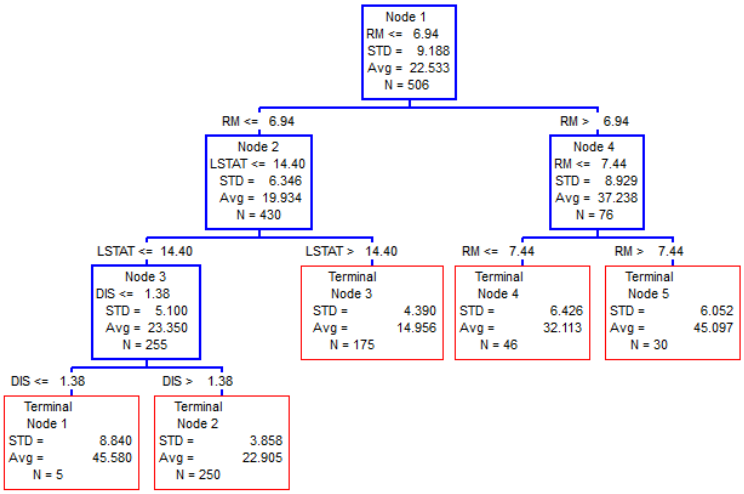

Рисунок 3.1.7 – Пример дерева решения для задачи регрессии

Из всего сказанного остаётся не закрытым вопрос построения самого дерева. На примере с выбором работы решение принималось на основе зарплаты, наличия кофе, времени до работы и др. Но какой же признак выбрать первым? При ответе на этот вопрос можно вспомнить игру «20 вопросов». Суть её заключается в том, что один человек загадывает знаменитость, а второй пытается отгадать, задавая вопросы, на которые можно ответить «Да» или «Нет». Первым всегда необходимо задавать такой вопрос, которые сильнее всего уменьшит количество оставшихся вариантов. Например, вопрос «Это Джонни Депп?» в случае отрицательного ответа оставит около 8 миллиардов для дальнейшего перебора (если не учесть тот факт, что не каждый человек – знаменитость). А вот вопрос «Это мужчина?» отсечёт около половины всех знаменитостей. То есть, признак «Пол» намного лучше делит выборку нежели признак «Это Джонни Депп?». Если описать другими словами, то деление по признаку «Пол» даёт больше информации для дальнейшего анализа. Таким образом, деление выборки интуитивно соответствует понятию прироста информации, основанного на энтропии. Энтропией же, в общем смысле, называют меру хаоса (беспорядка) в множестве. В машинном обучении используют понятие «Энтропия Шеннона». Энтропия Шеннона для системы с K возможными состояниями определяется следующим образом:

$$S = -\sum_{i=1}^{K} p_i \cdot \log_2 p_i , \qquad (3.1.7)$$

где p_i – вероятность нахождения системы в *i*-ом состоянии.

Если опустить предпосылки введения этого определения, то стоит отметить только тот факт, что чем выше энтропия, тем менее упорядочена система и наоборот.

Чтобы проиллюстрировать то, как энтропия определяет хорошие признаки для построения дерева, приведём небольшой пример. Суть примера в том, чтобы предсказать цвет шарика по его координате (Рисунок 3.1.8).

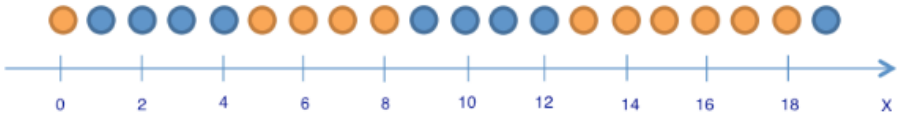

Рисунок 3.1.8 – Пример, в котором необходимо определить цвет шарика по координате

На рисунке изображено 9 синих и 11 жёлтых шариков (всего – 20 шариков). K = 2, так как шарики могут быть только 2 цветов. Если вытащить наугад любой шарик, то он с вероятностью $p_1 = \frac{9}{20}$ будет синим и с вероятностью $p_2 = \frac{11}{20}$ – желтым. Значит, энтропия начального состояния $S_0 = -\frac{9}{20}\log_2\frac{9}{20} - \frac{11}{20}\log_2\frac{11}{20} \approx 1$. Результат абсолютно очевиден, ведь в начальном состоянии о системе ничего не известно. Далее, разделим шарики на две группы – с координатой меньше либо равной 12 и больше 12 (Рисунок 3.1.9).

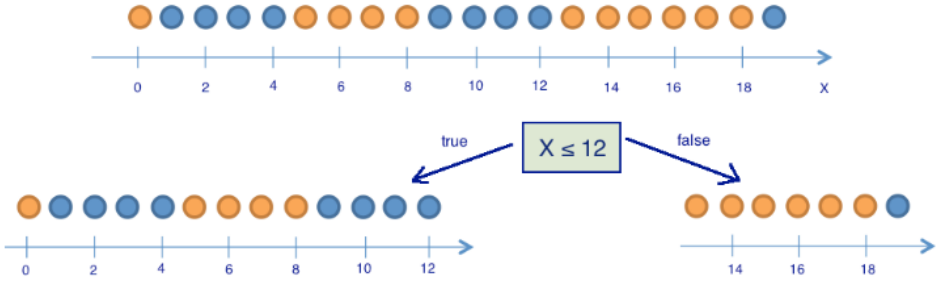

Рисунок 3.1.9 – Система после разделения шариков по координате 12

В левой группе оказалось 13 шариков, из которых 8 синих ($p_1 = \frac{8}{13}$) и 5 жёлтых ($p_2 = \frac{5}{13}$). Энтропия этой группы равна $S_1 = -\frac{8}{13}\log_2\frac{8}{13} - \frac{5}{13}\log_2\frac{5}{13} \approx 0.96$. В правой группе оказалось 7 шаров, из которых 1 синий ($p_1 = \frac{1}{7}$) и 6 желтых ($p_2 = \frac{6}{7}$). Энтропия правой группы равна $S_2 = -\frac{1}{7}\log_2\frac{1}{7} - \frac{6}{7}\log_2\frac{6}{7} \approx 0.6$. Теперь видно, энтропия уменьшилась в обеих группах по сравнению с начальным состоянием. Так как энтропия – это мера хаоса в системе, то уменьшение энтропии называют приростом информации. То есть, чем больше уменьшается энтропия, тем больше мы узнаём про систему. Формально прирост информации при разбиении выборки по признаку *Q* (например, «координата ≤ 12») определяется так:

$$IG(Q) = S_0 - \sum_{i=1}^{q}\frac{N_i}{N}S_i, \qquad (3.1.8)$$

где q – число групп после разбиения;

N_i – число элементов выборки, у которых признак Q имеет i-ое значение;

S_i – энтропия i-ой группы.

Возвращаясь к примеру, после разделения получилось q = 2 группы. Одна – из $N_1 = 13$ элементов, вторая – из $N_2 = 7$ элементов. Прирост информации получился

$$IG(x \le 12) = S_0 - \frac{13}{20}S_1 - \frac{7}{20}S_2 \approx 0.16$$

Таким образом, разделив шарики на две группы по координате 12, получилась более упорядоченная система, чем в начале. Если продолжить деление шариков на группы до тех пор, пока в каждой группе шарики не будут одного цвета (Рисунок 9), получим энтропию каждой такой группы равной 0 ($-1 \cdot \log_2 1 = 0$). Это соответствует тому, что группа шариков одного цвета – упорядочена.

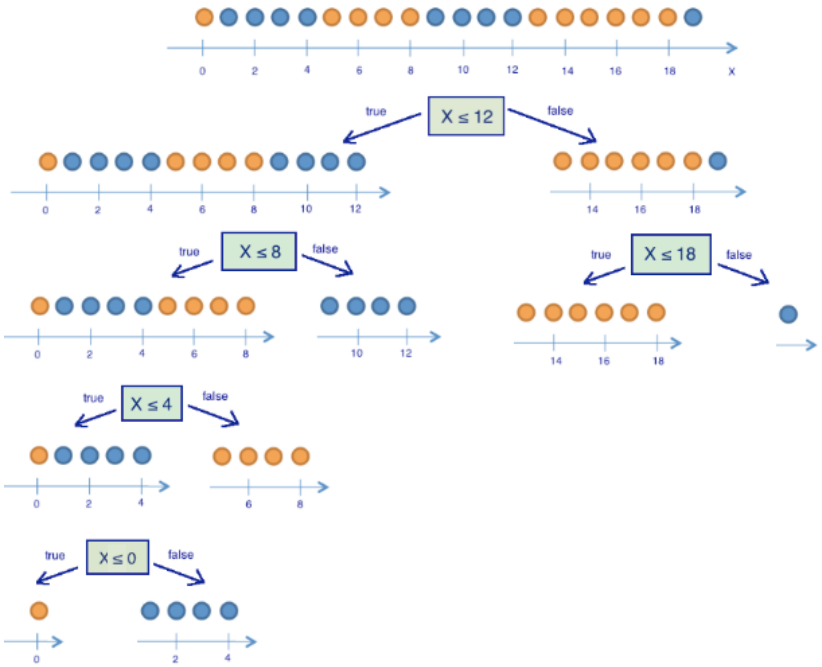

Рисунок 3.1.9 – Система после полного разделения шариков по цвету.

В итоге, получилось дерево решений, предсказывающее цвет шарика по его координате. Стоит отметить, что такое дерево может плохо работать для новых шариков, так как оно идеально подстроилось под обучающую выборку. Такое явление в машинном обучении называется переобучением. Другими словами, модель очень хорошо работает для обучающих данных, но плохо срабатывает на данных, которые не участвовали в обучении. Чтобы избежать переобучения, деревья принято «стричь». Это значит, что деревья нужно строить не до максимальной глубины, а до какой-то заданной либо да такой, что прирост информации будет изменятся незначительно.

Почему бы изначально не разделить шарики по какой-то другой координате? Это абсолютно логичный и правильный вопрос. Дело в том, что деление должно происходить по той координате (либо тому признаку), чтобы прирост информации был максимальным. То есть, необходимо перед каждым разбиением перебрать все координаты (признаки) и выбрать признак с максимальным приростом информации и по нему делить. В примере была выбрана координата 12, так как заранее было известно, что она лучше делит выборку. Таким образом, построение дерева сводится к тому, чтобы задавать вопросы, которые дают наибольший прирост информации.

Огромным преимуществом деревьев решений является то, что они легко интерпретируемы и понятны человеку. Например, на Рисунке 5 видно, что заемщику не выдали кредит, потому что у него нет дома и доход меньше 5 000. Другие более точные модели не могут похвастаться таким свойством, поэтому деревья решений получили огромную популярность, а один из представителей этой группы, метод классификации C4.5, остаётся первым в списке 10 лучших алгоритмов интеллектуального анализа данных

3.1.3 Объединение простейших моделей для создания более точной модели. Бустинг

В предыдущем разделе были рассмотрены алгоритмы, которые могут решать различные задачи классификации и регрессии. Все они имеют некую предельную точность, прыгнуть через которую не представляется возможным. В случае, когда повысить точность модели больше нельзя, но очень хочется, на выручку приходят более продвинутые техники машинного обучения – ансамбли. В общем случае, ансамбль – это некая совокупность, части которой образуют единое целое. Из повседневной жизни можно выделить музыкальные ансамбли, где объединены несколько музыкальных инструментов, архитектурные ансамбли с разными зданиями и т.д.

Хорошим примером ансамблей является теорема «о жюри присяжных» (1784 г.). Которая гласит: если каждый член жюри имеет независимое мнение, и вероятность правильного решения

члена жюри больше 0.5, то вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри и стремится к единице. Обратное то же верно: если вероятность правоты каждого члена жюри меньше 0.5, то вероятность принять правильное решение уменьшается и стремится к нулю с увеличением количества членов жюри. Ещё одним ярким примером ансамблей является так называемая «мудрость толпы». В 1906 года английский психолог Френсис Гальтон посетил рынок, где проводилась лотерея для крестьян. Их собралось около 800 человек, и все пытались угадать вес быка, который стоял перед ними. Сам бык весил 1198 футов. Гальтон записывал ответы всех крестьян и ни один крестьянин не угадал точный вес быка, однако Гальтон вычислил среднее арифметическое всех предсказаний и получил 1197 футов, чему очень удивился.

Описанные идеи применили и в машинном обучении. При этом, ансамблем называют набор предсказателей, которые вместе дают ответ (например, среднее по всем). Причина использования ансамблей проста: несколько моделей, которые пытаются получить одну и ту же переменную, дадут более точный результат, нежели одиночная модель. Обычно классифицируют две техники ансамблирования: бэггинг и бустинг.

Бэггинг – один из первых и простых видов ансамблей, при котором строят независимые модели и комбинируют их, используя некоторую модель усреднения (среднее арифметическое, взвешенное среднее или голосование большинства). Особенностью такого подхода является то, что при построении моделей не используют обучающую выборку целиком. Для каждой модели отбирают случайную подвыборку из обучающего множества. Таким образом, все модели немного отличаются друг от друга. Это значит, что модели не корреллируют между собой и, как следствие, увеличивают точность итоговой модели. Ярким примером бэггинга служит модель случайного леса (random forest). Исходя из определения бэггинга и названия модели, суть случайного леса проста до невозможности: берётся множество решающих деревьев, строятся модели на случайных подвыборках из обучающего множества, а ответы усредняются для задач регресии или выбираются по большинству для задач классификации.

Бустинг – это техника построения ансамблей, которая кардинально отличается от бэггинга. При бэггинге модели строятся независимо и параллельно, в то время как при бустинге модели строятся последовательно и одна за одной. В бустинге используется идея о том, что следующая модель должна учится на ошибках предыдущей. Следуя этой идее, общей модели требуется меньше времени для того, чтобы добраться до реального ответа. Однако, стоит выбирать критерий остановки обучения с осторожностью, иначе это может привести к переобучению. Сам же алгоритм легко визуально интерпретировать. Для примера, рассмотрим задачу классификации на плоскости шариков разного цвета (Рисунок 3.1.10).

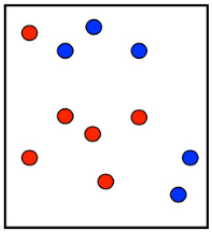

Рисунок 3.1.10 – Задача классификации шариков разного цвета на плоскости

Проведём линию (Рисунок 3.1.11), которой разделим шарики по правилу: слева – красные, справа – синие. Видим, что 3 синих шара проклассифицировались неправильно. Визуально увеличиваем им размер и при построении следующей разделяющей прямой учтём, что их нужно постараться классифицировать правильно.

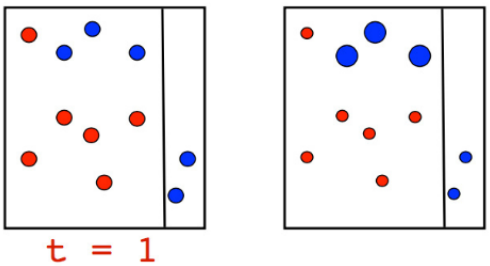

Рисунок 3.1.11 – Классы после 1-ой итерации

Проводим 2-ую разделяющую линию (Рисунок 3.1.12). При этом, определяем, что слева – красные шары, справа – синие шары. Видим, что те 3 шарика, которые на 1-ом этапе проклассифицировались неверно, в этот раз имеют правильный цвет, однако появилось 4 шарика красного цвета, которые определились как синие. Как и на предыдущей итерации, увеличим им размер и на следующей итерации постараемся предсказать им нужный цвет.

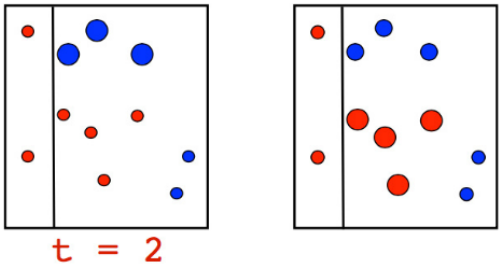

Рисунок 3.1.12– Классы после 2-ой итерации

Проведём третью линию (рисунок 3.1.13), на этот раз горизонтальную, которая предсказывает, что сверху – синие шарики, а снизу – красные. Видим, что с теми шариками, которые на прошлой итерации получили неправильный класс, всё в порядке, но появился 1 красный и 2 синих шарика, которые неправильно проклассифицировались. Однако, если посмотреть 1-ую модель, она классифицирует эти точки абсолютно правильно. Из этого можно сделать вывод, что за три последовательно построенные модели, мы полностью проклассифицировали шарики.

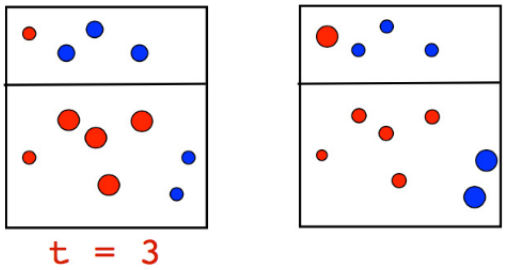

Рисунок 3.1.13– Классы после 3-ей итерации

Далее, необходимо сложить три построенные модели вместе и получить одну общую модель, которая предсказывает цвет шарика на плоскости по координате (Рисунок 3.1.14).

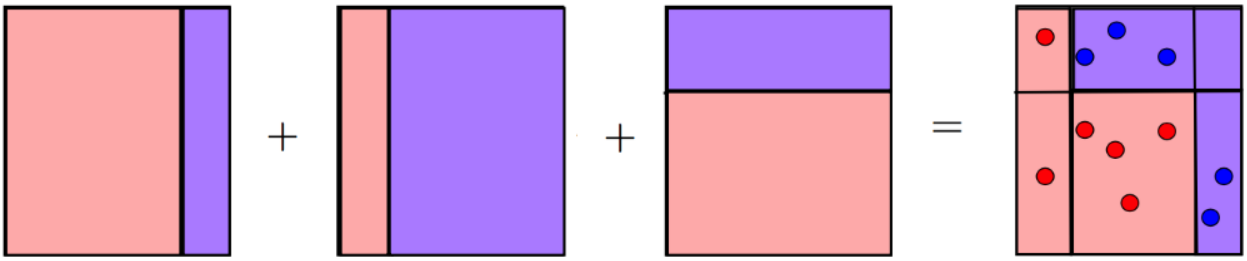

Рисунок 3.1.14 – Итоговая модель классификации шариков на плоскости

Одним из примеров бустинга является градиентный бустинг. Именно этот алгоритм использовался для построения модели, которая предсказывает стоимость объекта недвижимости. Градиентный бустинг – это техника машинного обучения, которая часто используется при решении задач классификации и регрессии. При этом, построение модели происходит в форме ансамбля слабых предсказывающих моделей, обычно – деревьев решений.

Цель любого алгоритма обучения с учителем – найти функцию потерь и минимизировать её. Другими словами, найти функцию, которая показывает, на сколько предсказанное значение отличается от целевой переменной (найти ошибку) и попытаться её уменьшить. В задачах

регрессии наиболее типичными являются среднеквадартичная (MSE, Формула 3.1.9) и среднеабсолютная (MAE, Формула 10) функции потерь.

$$MSE = \frac{1}{k}\sum_{i=1}^{k}(y_i - y_i^p)^2$$ – среднеквадратичная ошибка для векторов Y и Y^p (3.1.9)

$$MAE = \frac{1}{k}\sum_{i=1}^{k}|y_i - y_i^p|$$ – среднеабсолютная ошибка для векторов Y и Y^p (3.1.10),

где Y –вектор целевой переменной (цены в сделках);

Y^p–вектор с предсказанной целевой переменной (смоделированная стоиомсть);

y_i и y_i^p–i-ые значения векторов Y и Y^p соответственно;

k – размерность векторов Y и Y^p.

В машинном обучении для поиска минимума функции используется интуитивно простой и эффективный способ – градиентный спуск. Если не вдаваться в математическую теорию и описать этот алгоритм простыми словами, то можно представить снежную горку, на вершине которой стоит человек (начальное приближение) и ему нужно попасть вниз горки (минимум функции). Человек сделает всё очень просто: сядет на клеёнку и съедет вниз (Рисунок 3.1.15). Эта простая интерпретация позволяет понять смысл алгоритма без ознакомления с производными, гладкими функциями и прочими вещами из курса математического анализа.

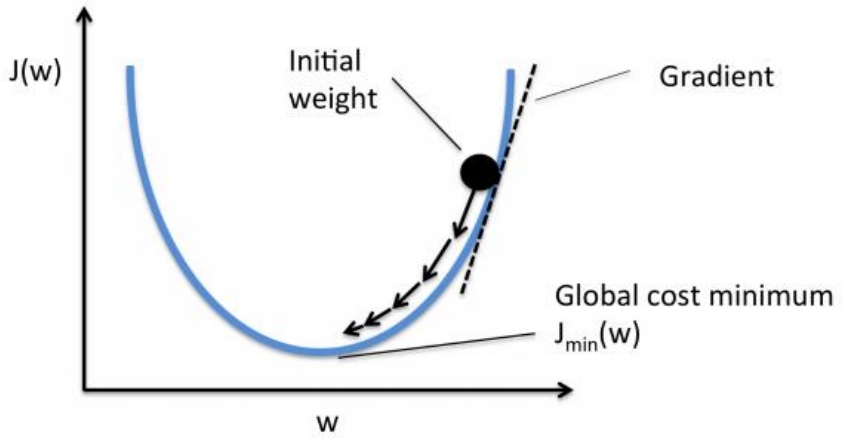

Рисунок 3.1.15 – Пример работы градиентного спуска для поиска минимума квадратичной функции

Градиентный спуск – это последнее понятие (которому не было дано определение) в методе под названием «Градиентный бустинг на деревьях». Действительно, каждое слово в названии метода несёт под собой некую логическую информацию и значения всех этих слов были описаны в разделах этой главы. Теперь, когда с категорией и видом задачи, а также с моделью, которую необходимо построить, всё предельно ясно, можно переходить непосредственно к обучению модели и анализу результатов, которые показывает полученная модель.

3.2 Построение модели оценки

3.2.1 Данные, использованные для оценки. Предварительный анализ данных

Любой анализ данных или построение модели в машинном обучении начинается с того, что изучается входная информация. То есть, та информация, на основе которой необходимо построить модели и та информация, для которой необходимо найти целевую переменную. При решении нашей задачи целевой переменной является стоимость квадратного метра в

изолированном помещении жилой многоквартирной недвижимости. Как говорилось ранее, обучающим множеством являются сделки с квартирами, а все зарегистрированные объекты в г. Минск и Кобринском районе, которые относятся к жилой многоквартирной недвижимости, являются множеством, которому необходимо смоделировать целевую переменную.

Перед тем, как углубиться в анализ исходных выборок, стоит остановится на том, что может содержать в себе любая выборка. В разделе «**Простейшие модели машинного обучения. Деревья решений**» вводилось понятие обучающего множества, которое представляет собой матрицу (таблицу), где по строкам записаны объекты, а по столбам записаны какие-то признаки (факторы). Признаки описывают объекты. То есть, каждый объект имеет некоторый набор признаков. Эти самые признаки бывают разных типов: числовые, бинарные и категориальные. Числовые признаки – это те признаки, которые описывают количественные свойства объекта (рост, вес, площадь, количество этажей и т.п.). Основным отличием числовых признаков является то, что над ними можно производить различного рода математические операции. Бинарные признаки описывают сам факт существования некого признака у объекта (наличие воды, возможность свободной планировки и т.д.). Таким образом, значения бинарного признака могут быть «Да» (1) или «Нет» (0). Категориальные же признаки чаще всего описывают качественные свойства объекта (материал стен, тип конструкций, город, имя и т.д.). Отличительным свойством категориальных признаков является то, что над ними невозможно производить никаких математических операций и операций сравнения, кроме подсчёта количества.

Первое, что необходимо сделать, когда только получили выборку, это описать существующие признаки и отбросить изначально те, которые не несут в себе никакой полезной информации. Таким описанием может быть таблица, в которой есть название признака, его расшифровка и тип (Таблица 3.2.1).

Таблица 3.2.1 Описание признаков в исходных данных

Название (в Python)	**Описание**	**Тип**
num_pp	Номер по порядку	Числовой
inv_num_ip	Инвентарный номер изолированного помещения	Категориальный
date_sdelki	Дата сделки	Числовой
build_year_ks	Год постройки капитального строения	Числовой
nadzem_amount	Количество надземных этажей	Числовой
general_sq	Общая площадь помещения, кв.м	Числовой
floor	Этаж расположения	Числовой
rooms_amount	Количество жилых помещений (комнат)	Числовой
kitchen_sq	Площадь кухни, кв.м	Числовой
balkon	Наличие балкона, лоджии	Категориальный
kladovaya	Наличие кладовой, гардеробной или других помещений для хранения	Категориальный
height	Высота потолка	Категориальный
steklo	Стеклопакеты в окнонных проемах	Бинарный
svob_plan	Возможность свободной планировки	Бинарный
hot_water	Горячее водоснабжение	Категориальный
sanuzli	Санузлы	Категориальный
otoplenie	Отопление	Категориальный
ventil	Кондиционирование и вентиляция	Категориальный
general_state	Общее состояние здания/помещения	Категориальный
land_design	Элементы ландшафтного дизайна	Бинарный
sport_place	Спортивная площадка	Бинарный
ogorod_terr	Огороженная территория	Бинарный
parkovka_amount	Количество парковочных мест для автомобилей	Числовой

Таблица 3.2.1 Описание признаков в исходных данных

Название (в Python)	Описание	Тип
service	Собственная служба эксплуатации, управляющая компания	Бинарный
video	Консъерж/внутреннее видеонаблюдение	Бинарный
underground_parkovka_amont	Количество мест на крытом паркинге, шт.)	Числовой
mat_sten	Наименование материала стен	Категориальный
konstr_type	Виды конструкций	Категориальный
territory	Территория (Минск, Кобрин, СНП)	Категориальный
dostup_center	Доступность центра	Числовой
cp_dist	Расстояние от СНП до центра притяжения	Числовой
price_kvm	Цена, USD/кв.м	Числовой
price_all	Цена за весь объект, USD	Числовой
marker	Маркер релевантности	Бинарный

Из Таблицы 3.2.1 видно, что входная выборка имеет 34 фактора, однако, для последующего анализа нужно убрать следующие факторы: номер по порядку, инвентарные номер, цена за весь объект. Маркер релевантности тоже нужно убрать, но это признак понадобится потом, когда будем отбирать данные для построения модели.

Далее, оставшиеся признаки нужно проверить на наличие пустых, ошибочных и аномальных значений. Если с пустыми и ошибочными значениями всё предельно ясно, то аномальными являются значения, которые не характерны для конкретного фактора (например, год постройки – 1376). Аномальные значения зачастую являются ошибочными, но способ поиска таких факторов отличается. Признаки, в которых пустых значений было более 55%, удалялись из выборки. Пустые значения оставшихся факторов заполнялись в соответствии с правилами, которые специалисты по оценке выработали для каждого фактора. Данные правила представлены в Приложении к Отчету. Аномальные и ошибочные значения, которые удалось найти, изменялись на значения из ЕГРНИ или массового формирования либо как пустое, если не удавалось найти правильное значение. Так же, некоторые факторы заменялись на производных от них. Например, вместо даты сделки, года постройки здания и территории, добавились возраст сделки (в днях), возраст здания (в годах) и ранг города. Некоторые факторы из категориальных превратились в бинарные (горячая вода и отопление).

Таким образом, после форматирования всех признаков, в выборке стало 20 факторов: возраст здания, возраст сделки, количество надземных этажей, общая площадь, количество комнат, этаж расположения, площадь кухни, высота потолка, наличие балкона, горячая вода, наличие кладовой, вентиляция, материал стен, свободная планировка, расстояние до центра притяжения, видеонаблюдение, тип конструкций, доступность центра, ранг и стоимость за кв.м.

До построения модели был проведён разведочный анализ данных. Его суть в том, чтобы попытаться найти коррелирующие факторы. Для удобного анализа в библиотеке ***seaborn*** языка программирования ***Python*** есть функция, которая строит матрицу корреляций между числовыми факторами (Рисунок 3.2.1-3.2.2).

	building_age	sdelka_age	nadzem_amount	general_sq	floor	rooms_amount	kitchen_sq	new_dostup_center	cp_dist	price_kvm
building_age	1.000000	0.015542	-0.590169	-0.443508	-0.354024	-0.116836	-0.533387	-0.136852	0.008138	0.168116
sdelka_age	0.015542	1.000000	-0.006206	-0.028031	0.005054	-0.060094	0.036550	-0.015129	0.004513	0.248433
nadzem_amount	-0.590169	-0.006206	1.000000	0.352197	0.627610	0.069499	0.437816	0.225626	-0.123964	0.299413
general_sq	-0.443508	-0.028031	0.352197	1.000000	0.251491	0.791019	0.535892	0.088185	-0.006575	0.002941
floor	-0.354024	0.005054	0.627610	0.251491	1.000000	0.067818	0.282483	0.127706	-0.083759	0.213933
rooms_amount	-0.116836	-0.060094	0.069499	0.791019	0.067818	1.000000	0.210323	0.035434	0.008322	-0.050563
kitchen_sq	-0.533387	0.036550	0.437816	0.535892	0.282483	0.210323	1.000000	0.080410	-0.007769	0.028879
new_dostup_center	-0.136852	-0.015129	0.225626	0.088185	0.127706	0.035434	0.080410	1.000000	0.345115	0.095286
cp_dist	0.008138	0.004513	-0.123964	-0.006575	-0.083759	0.008322	-0.007769	0.345115	1.000000	-0.236097
price_kvm	0.168116	0.248433	0.299413	0.002941	0.213933	-0.050563	0.028879	0.095286	-0.236097	1.000000
rang	0.170072	-0.011220	0.390593	0.085865	0.261491	0.043797	0.032407	0.294815	-0.200549	0.812450

Рисунок 3.2.1 – Матрица корреляций в числовом виде

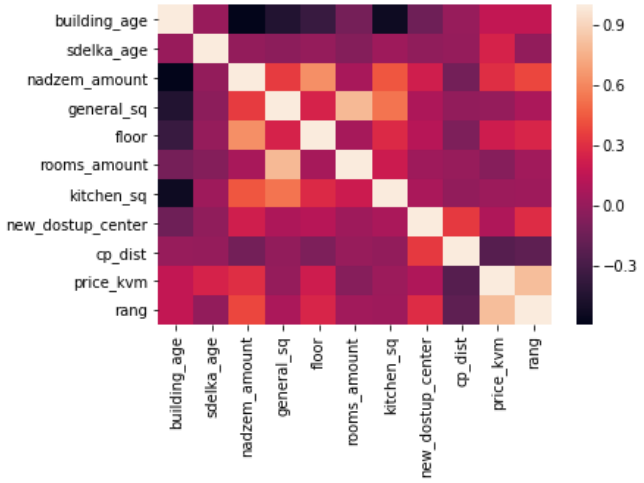

Рисунок 3.2.2 – Матрица корреляций в графическом виде

Сразу бросается в глаза тот факт, что ранг сильно коррелирует с целевой переменной (~0.81). Ничего страшного в этом нет, так как для построения модели используются данные только по двум городам (г. Минск и г. Кобрин) и цены в сделках в этих городах очень сильно отличаются, поэтому линейная разделимость между рангом и целевой переменной вполне справедлива. Можно сделать предположение о том, если в модель добавить ещё несколько городов, то влияние ранга на целевую переменную существенно снизится. Однако, можно выдвинуть ещё одно предположение: заменить фактор ранг на какой-то другой либо несколько факторов, которые будут описывать город.

Также можно выделить факт о том, что количество комнат коррелирует с общей площадью помещения (~0.79). Это стоит запомнить и после создания модели, убрать каждый из этих факторов и посмотреть, что произойдёт с моделью. На стадии разведочного анализа ни одного фактора удалено не было.

3.2.2 Процесс построения модели оценки

Выборка с релевантными сделками, на которых должна строится модель состояла из 6 583 объектов. Из них 5 328 объектов в советском районе г. Минска, 1 168 – в г. Кобрин и 87 – в сельский населенных пунктах Кобринского района.

Для того, чтобы подать данные в модель, необходимо сделать некоторые манипуляции с факторами категориального типа. Их необходимо кодировать в бинарные. Это делается следующим образом: для каждого категориального фактора надо найти уникальные значения, затем добавить все найденные значения как новые признаки. Далее, для каждого объекта и каждого категориального признака необходимо поставить 1 (единицу) тому признаку, которому соответствует значение категориального фактора, остальным признаком присваивается значение 0 (ноль). Например, на Рисунке 3.2.3 наглядно показано, как кодируется фактор высота потолка. В первом объекте (по порядку) высота потолка имеет значение «2.7 - 3». Значит, напротив фактора «height__ 2.7 - 3» стоит единица, у всех остальных – ноль.

	height	height__до 2.7	height__2.7 - 3	height__3.0 - 4.0
0	2.7 - 3	0	1	0
1	до 2.7	1	0	0
2	до 2.7	1	0	0
3	до 2.7	1	0	0
4	до 2.7	1	0	0
5	до 2.7	1	0	0
6	2.7 - 3	0	1	0
7	до 2.7	1	0	0
8	до 2.7	1	0	0
9	до 2.7	1	0	0

Рисунок 3.2.3 – Кодирование категориальных признаков в бинарные

Кодирование категориальных признаков порождает новые факторы, поэтому таблица расширяется. После кодировки вместо 20 факторов, стало 36.

При построении любой модели есть правило, которое игнорировать попросту нельзя, каким бы небольшим не было обучающее множество. Этим этапом является разделение обучающего множества на тренировочное и тестовое множества. Это обязательное действие, которое позволяет отложить некоторые данные, которые не будут участвовать в обучении модели (тестовое множество). Делается это для того, чтобы модель не переобучилась. Переобучение – это процесс, когда модель показывает хорошие результаты на тренировочных данных, но плохие – на каких-то новых данных (тестовых). Данные можно делить в различном отношении (60% на 40%, 75% на 25%, 82.328% на 17.672% и т.д.). Обучающее множество решено было делить в отношении 85% на 15% (Рисунок 3.2.4). Деление происходит случайным образом. При этом, в обучающем и тестовом множествах объектов из различных пилотных территорий получается в относительных величинах равное количество – по ~85% и ~15% данных г. Минска, г. Кобрина и СНП в тренировочном и тестовом множествах.

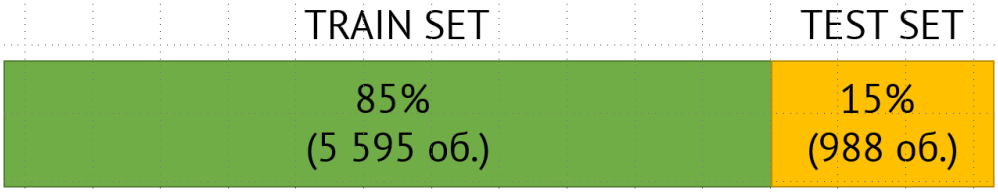

Рисунок 3.2.4 – Размер тренировочного и тестового множеств

Теперь данные окончательно подготовлены для обучения модели. Как отмечалось ранее, для построения была выбрана модель градиентного бустинга на деревьях. Суть построения модели – это выбрать оптимальные гиперпараметры модели. Гиперпараметрами модели называются те параметры, изменяя которые, меняется и логика предсказывания. У градиентного бустинга есть несколько основных гиперпараметров:

1. Количество предсказателей (n_estimators) – количество моделей (деревьев), которые предсказывают целевую переменную. Например, в нашей задаче количество деревьев ищется в диапазоне от 20 до 600 с шагом 20.

2. Максимальная глубина дерева (max_depth) – максимальная глубина, на которую может опустится каждое дерево при своём построении, то есть, этот параметр ограничивает количество узлов в дереве.

3. Минимальное количество объектов для разделения (min_samples_split) – минимальное количество объектов, которое должно находится в узле, чтобы тот смог разделиться.

4. Минимальное количество объектов в листе (min_samples_leaf) – минимальное число объектов, которое должно быть в листовом узле.

5. Количество признаков для поиска лучшего разделения (max_features) – этот параметр указывает на максимальное количество факторов (n_features), которые будут учтены при разделении узла. Может принимать значения «auto» (max_features= n_features), «sqrt» (max_features= sqrt(n_features)) и «log2 (max_features= log2(n_features))».

6. Функция потерь (loss) – параметр, который указывает, какую функцию потерь необходимо минимизировать. Ранее было рассказано про среднеквадратичную и среднеабсолютную функции потерь. Однако, кроме этих двух, добавлялись ещё функции «huber» (сочетание квадратичной и абсолютной) и «quantile».

Сам подбор гиперпараметров производился по кросс-валидации (перекрёстная проверка). Это значит, что для каждого параметра задается множество значений, которые он принимает (количество моделей от 20 до 600, глубина дерева от 2 до 11 и т.д.). Далее, строятся модели со всевозможными комбинациями гиперапаметров (для нашей модели получилось 108 000 комбинаций) и выбирается модель, гиперпараметры которой показали наименьшую ошибку. Для справки: 108 000 обучений модели на тестовой выборке (5 595 строк и 36 столбцов) для поиска оптимальных гиперпараметров занимает порядка **полутора** минут.

В начале работ по построению модели было известно, что нужно будет проверять много различных гипотез. Следовательно, модель нужно будет перестраивать много раз. Исходя из этого, была написана функция и вынесена в отдельный модуль, которая получает на вход тренировочное множество (признаки и целевая переменная) и выдает модель с оптимальными гиперпараметрами. Таким образом, этот модуль можно подключать к любой программе на Python и использовать эту функцию.

Первая построенная модель показала очень хорошие результаты в городах и очень плохие – в СНП. Как упоминалось ранее, точность и другие показатели модели вычислялись на отложенной (тестовой) выборке. Средняя абсолютная ошибка модели – 11.58% (для Минска – 8.74%, для Кобрина – 13.26%), однако, для СНП ошибка составила 135.54%. Очевидно, что данных по СНП в относительных величинах очень мало по сравнению с городами. Таким образом, было решено отказаться от сделок в СНП и строить модели только по городам. После удаления данных в СНП и перестроения модели, уменьшилась общая абсолютная ошибка по модели (10.38%), но незначительно увеличились ошибки по городам (на ~1%). Распределение ошибок по модели отображено на Рисунке 3.2.5.

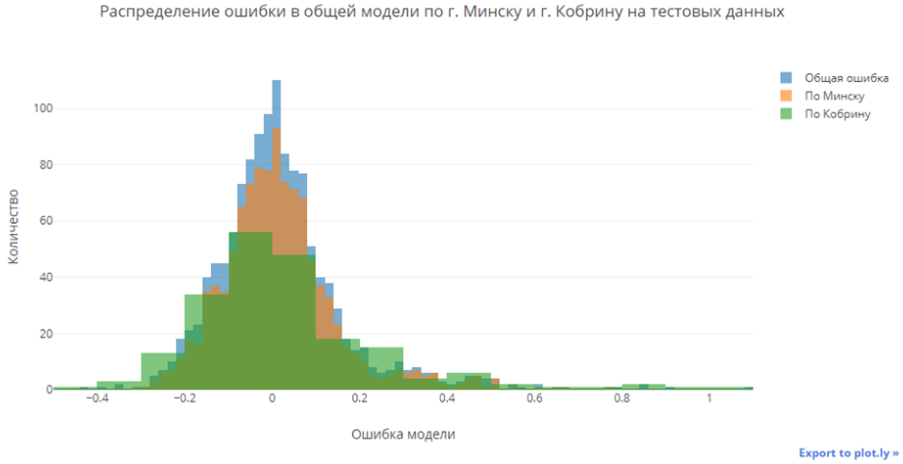

Рисунок 3.2.5 – Распределение ошибок по модели после удаления СНП

После проведения разведочного анализа, отмечалось, что стоит проверить признаки общей площади и количества комнат. Поочерёдно удалив каждый из этих признаков из обучающего множества и перестроив модель, получили, что общая ошибка по модели и ошибка по Минску возрастали на ~0.5%, в Кобрине возрастала на ~0.05%. Исходя из этого, было решено оставить оба признака в модели. Также поступило предположение от экспертов построить отдельно модели для Минска и Кобрина без учёта ранга города. Средняя ошибка по модели для Минска составила 9.51%, для Кобрина – 13.67%, что соответствует уменьшению на ~ 0.1% и

~0.8% по сравнению с моделью, которая была построена на обоих городах. Стоит также отметить: если масштабировать получение стоимости на всю территорию республики и учесть тот факт, что ошибка уменьшается от 0.1 до 0.8%, но при этом будет построено 202 модели вместо одной, сам собой приходит вывод: оставляем одну модель и пытаемся её улучшить.

Сам из себя процесс получения модели представляет собой итерации. То есть, делается некое предположение, строится модель и анализируется результат. И так раз за разом. Таким образом, после первых итераций стало понятно, что города между собой разделяются хорошо, да и с характеристиками ничего плохого нету, но нет фактора(ов), которые бы указывали на расположение объектов внутри каждого города. При этом фактор доступности центра города полностью не описывает влияние расположения объекта на его стоимость. Поэтому было решено добавить ещё несколько факторов: категориальный «Оценочная зона» и бинарные «Доступность санитарно-зашитной зоны», «Доступность рекреационной зоны». Сразу отметим несколько проблем, которые могут возникнуть с фактором «Оценочная зона»:

1. Так как фактор категориальный, то обучающее множество растёт в ширину (из 36 факторов стало 61). А это только добавились зоны по Советскому району г. Минска и г. Кобрину. А если учесть, что нужно будет добавить зоны по всем городам, то назревает «катастрофа».

2. Ещё одна проблема заключается в том, что если в обучающей выборке какой-то зоны не будет (нет сделок в некоторой зоне), а в выборке, где нужно смоделировать стоимость, эта зона будет, то фактор «Оценочная зона» просто-напросто не учтётся.

Поэтому, для проверки гипотезы фактор оставляем, но на перспективу необходимо заменить этот фактор на один или несколько числовых факторов, которые смогут описать местоположение объектов в одном городе.

После добавления новых трёх факторов и перестроив модели заново, гипотеза подтвердилась. Общая ошибка по модели, для Минска и для Кобрина уменьшились на ~2%, ~1.5% и ~3% и составили 8.6%, 7.92% и 11.77% соответственно. При этом, количество объектов тестовой выборки, ошибки по которым попадают в допустимый диапазон (±15% – для Минска и ±20% – для Кобрина), составляют: 91.41% – для Минска и 86.63% – для Кобрина. Распределение ошибок на пилотных территориях можно увидеть на Рисунке 3.2.6.

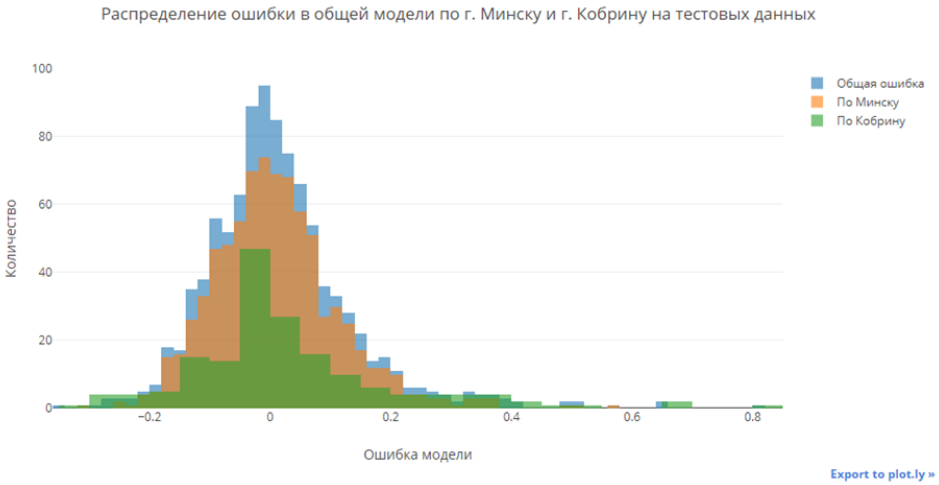

Рисунок 3.2.6 – Распределение ошибок по модели после добавления оценочных зон, СЗЗ и рекреации

Решено было оставить получившуюся модель и приступить к оценке зарегистрированных объектов, интерпретации работы модели. Ещё одним этапом, который стоит сделать – это анализ факторов, для которых строится модель. То есть, оценить вклад каждого фактора в модель и возможность удаления некоторых из них без существенной потери качества модели.

3.2.3 Интерпретация работы модели

В разделе, где описывались деревья решений было сказано, что на листовых узлах (это те, на которых заканчивается дерево) находятся некоторые значения целевой переменной (стоимости за кв.м). Однако там же было сделано замечание о том, что при построении некоторых алгоритмов, которые используют ансамбли (вспомним, что градиентный бустинг к ним относится), в листовых узлах могут находится не сами значения целевой переменной, а некоторые корректирующие значения. Эти значения, начиная с первого дерева, прибавляются к некоторому нулевому приближению (предположительно, нулевым приближением является среднее по целевой переменной). Стоит заметить, что прибавление – это не ошибка, так как если корректирующее значение со знаком минус, то прибавляя это значение, отнимается его модуль (плюс на минус). После получения и корректирования коэффициентом из последнего дерева, получается итоговая стоимость для объекта. Получение стоимости для объекта в Минске и Кобрине представлено на Рисунке 3.2.7.

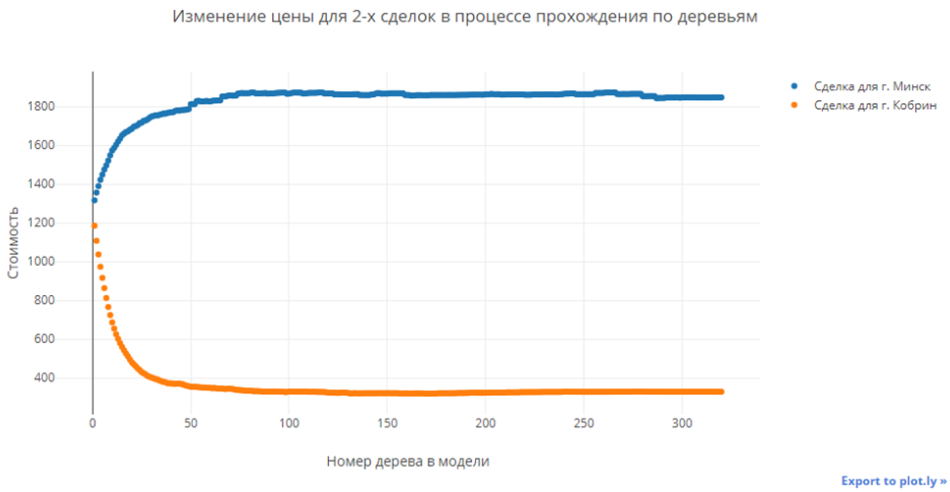

Рисунок 3.2.7– Изменение целевой переменной после прохождения каждой модели (дерева)

Помимо способа получения стоимости, важно понимать, как считаются корректирующие коэффициенты. Про то, как строится дерево решений, можно прочитать в разделе «**Деревья решений**». Однако, перед тем, как анализировать корректирующие коэффициенты, нужно каким-то образом отобразить каждую модель (дерево). Путём долгого поиска в интернет-источниках, были найдены некоторые фрагменты кода, из которых удалось собрать процедуру. Эта процедура позволяет из общей модели может выгрузить в pdf дерево под любым номером (Рисунок 3.2.8).

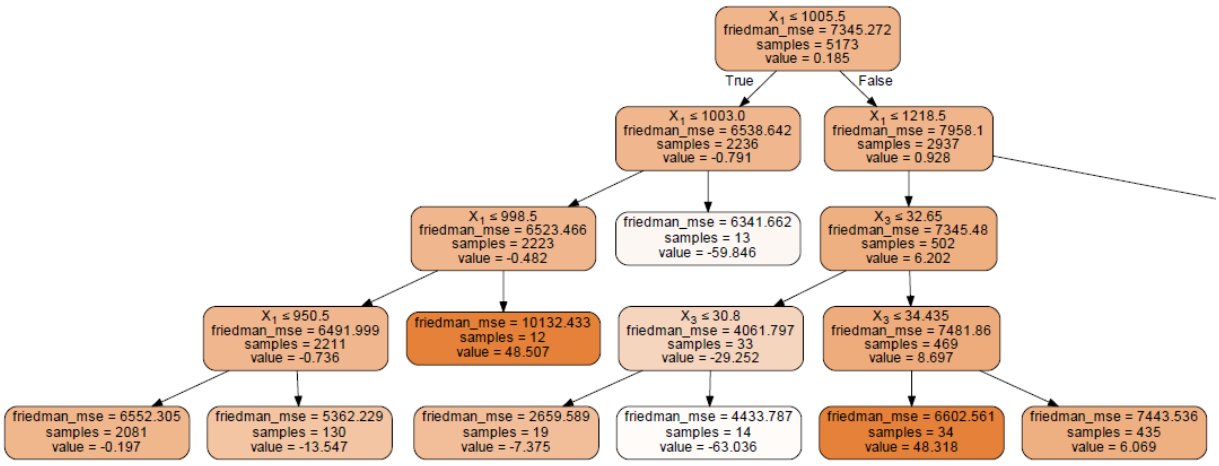

Рисунок 3.2.8 – Часть 317-ого дерева

Если внимательно посмотреть на Рисунок 3.2.8, то можно увидеть, что каждый узел имеет некоторые параметры. Эти параметры можно описать следующим образом:

1. X_i – i-ый фактор в обучающем множестве. Неудобство при анализе деревьев заключается в том, что факторы нумеруются, а не пишутся именем;
2. freedman_mse – среднеквадратичная ошибка по Фридману;
3. samples – количество объектов в узле;
4. value – корректирующий коэффициент.

Стоит отметить тот факт, что чем больше объектов (samples) в узле, тем меньше корректирующий коэффициент (value). Эта зависимость напрямую вытекает из смысла бустинга как такового: следующая модель должна учится на ошибках прошлой модели. Так как ошибочно предсказанных объектов всегда меньше (иначе, модель уж очень плохо работает), то в узлах с меньшим количеством объектов всегда корректирующих коэффициент больше, ибо эти объекты плохо смоделировались на прошлой модели.

Помимо некоторых технических способов интерпретации, в процессе построения модели возникла идея проверить, как работают корректировки на различных факторах. При этом, сразу же был найден относительно простой способ, как построить любую корректировку. Первой проверенной была корректировка на время. Алгоритм получения корректировки для одного объекта по построенной модели следующий:

1. Создать массив с датами, на которые необходимо оценить объект (вплоть до дня).
2. Взять один объект из выборки и зафиксировать все признаки, кроме даты сделки (возраста сделки).
3. Скопировать этот объект столько раз, сколько задали дат сделки в п.1.
4. Для получившихся объектов заполнить дату сделки из п.1. Таким образом, мы исследуем один объект в различные промежутки времени.
5. Отображаем на графике.
6. Далее, нежно взять ещё один объект и выполнить шаги 2-5.

Используя такой способ, но изменяя фактор, можно строить абсолютно любые корректировки. Таким образом, на Рисунках 3.2.9-3.2.12 можно наблюдать, как распределяется стоимости объектов во времени для тестовой выборки в Минске и Кобрине.

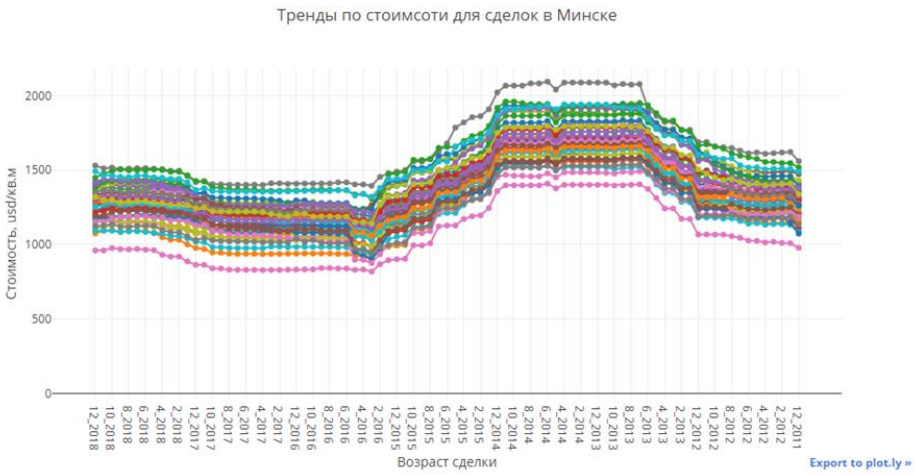

Рисунок 3.2.9 – Распределение стоимостей во времени для сделок в г Минск

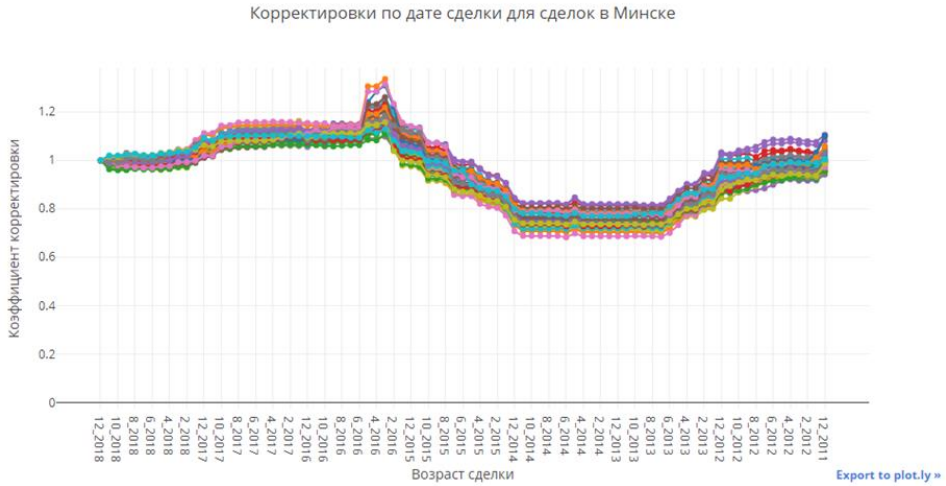

Рисунок 3.2.10 – Распределение Корректирующих коэффициентов во времени для сделок в г Минск

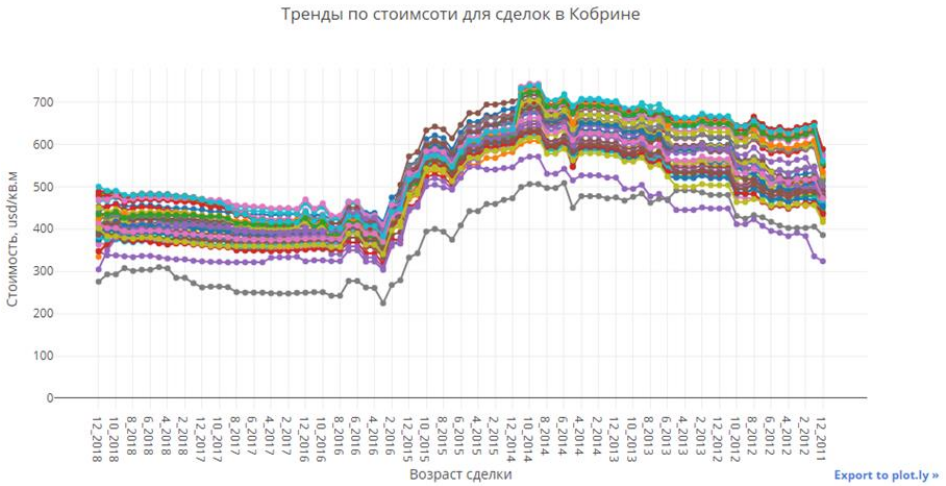

Рисунок 3.2.11 – Распределение стоимостей во времени для сделок в г Кобрин

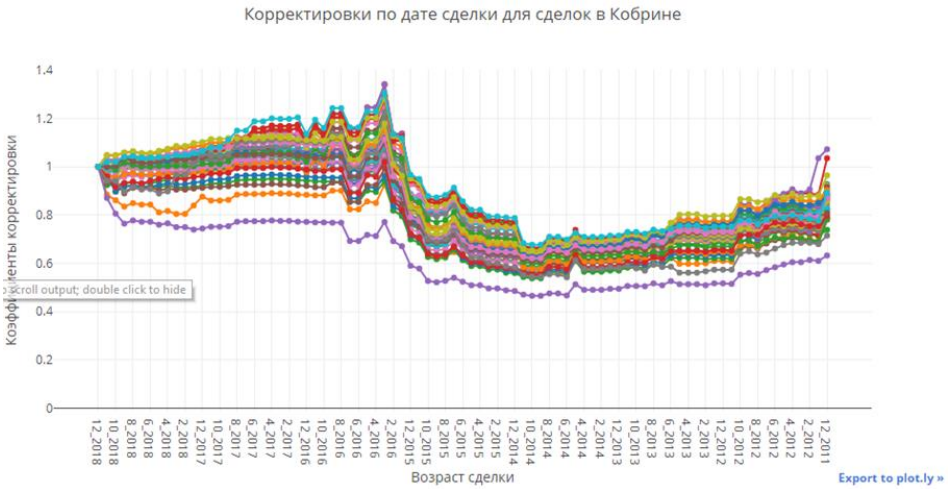

Рисунок 3.2.11 – Распределение Корректирующих коэффициентов во времени для сделок в г Кобрин

Чтобы получить общий тренд по стоимости и корректировкам, необходимо усреднить стоимости и корректировки для каждой даты. Сравнение с корректировками на время, рассчитанными в ходе анализа рынка многоквартирной недвижимости (Подраздел 2.3.2 Раздела 2 настоящего Отчета) представлено на Рисунках 3.2.12-3.2.13.

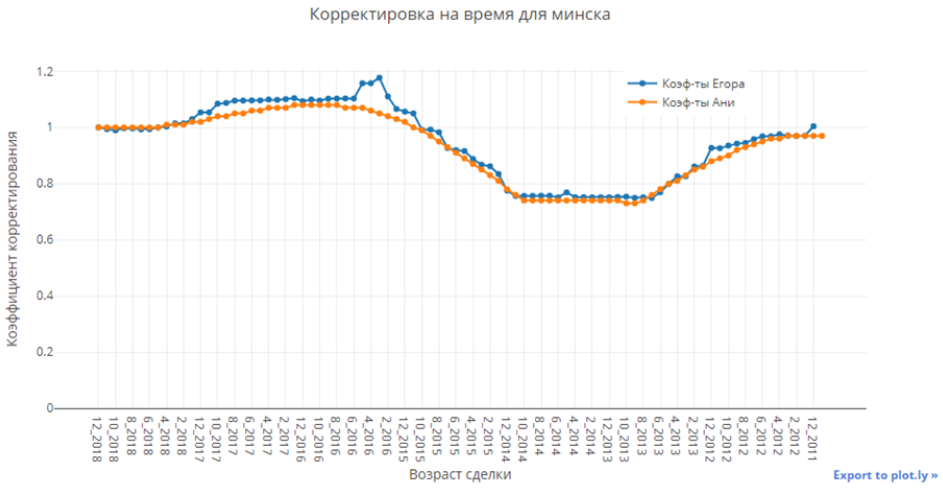

Рисунок 3.2.12 – Сравнение корректировок на время в г. Минск

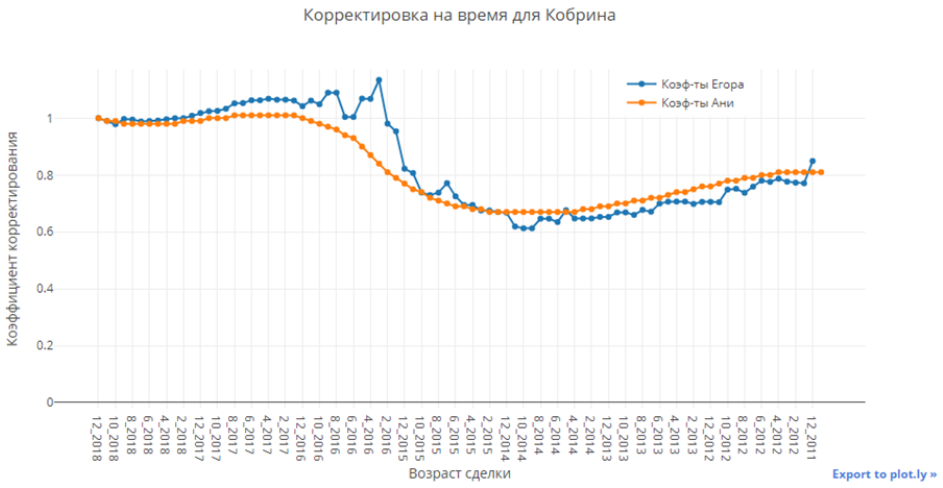

Рисунок 3.2.13 – Сравнение корректировок на время в г. Кобрин

Из Рисунков 3.2.12-3.2.13 сразу сделали вывод о том, что есть точки, когда модели срабатывают абсолютно противоположно. Проанализировав данные за эти периоды, обнаружили, что при построении модели градиентного бустинга использовалось много сделок, которые были в одном капитальном строении. Убрав часть этих сделок из обучающей выборки, перестроили модель и получили следующий результат: Незначительно уменьшились общая ошибка по модели и ошибка по Минску, но по Кобрину ошибка возросла на 2%. Это связано с тем, что в Кобрине удалили больше ста объектов, которые продавались в одном капитальном строении. Но количество объектов, которые попадают в допустимые диапазоны (±15% и ±20%) возросло на ~2% для Минска и уменьшилось на ~0.5% для Кобрина. Для получившейся модели перестроили корректировки на время и получили следующий результат (Рисунок 30-31): В Минске корректировка изменилась незначительно и есть периоды (с 2016 по 2018 годы), где можно пересмотреть корректировку, которую используют в кадастровой оценке. В Кобрине существенно изменился вид корректировки, где был «крутой» пик (он стал намного меньше).

Если обобщить, то по построению корректировки моделью, видно, что в Минске коэффициенты очень похожи, но отличаются (от 0.07 до 0.16). Если принимать такое расхождение нормальным, то можно усреднять коэффициенты для получения одной линии корректировки в Минске. Что касается Кобрина, то ситуация не такая радужная. Коэффициенты отличаются намного существеннее (от 0.2 до 0.45), чем в Минске. Из этого можно сделать вывод о том, что, возможно, нужно строить не одну, а несколько корректировок для различных объектов.

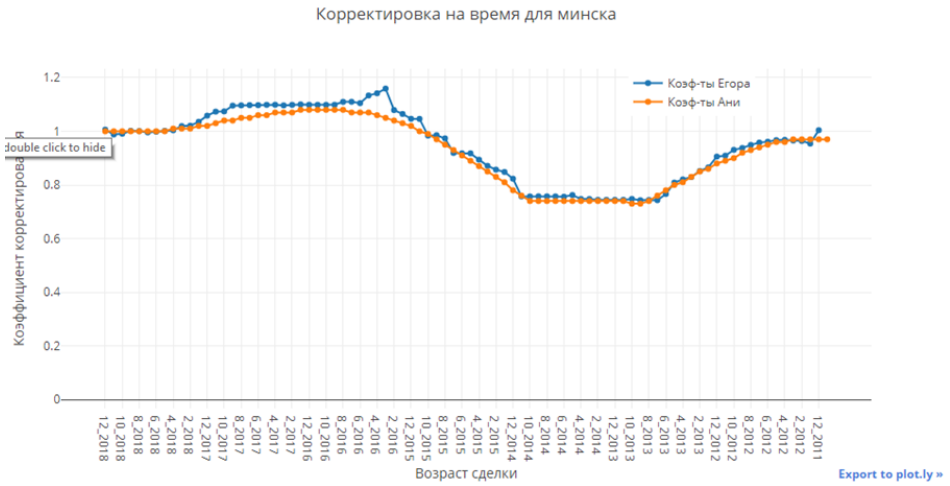

Рисунок 3.2.14 – Сравнение корректировок на время в г. Минск после удаления сделок в одном КС

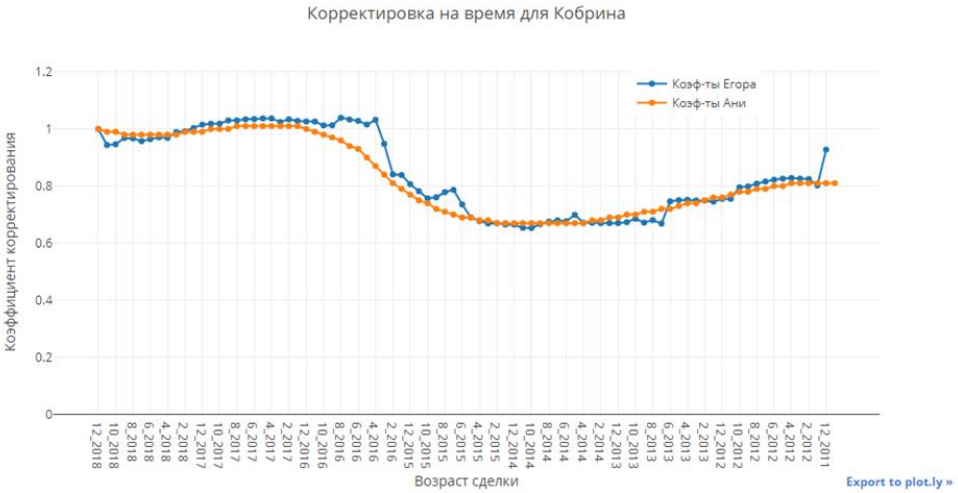

Рисунок 3.2.15 – Сравнение корректировок на время в г. Кобрин после удаления сделок в одном КС

Ещё одной корректировкой, которую удалось построить, является корректировка на площадь. При этом, из сделок отбирались объекты с разным количеством комнат и для них строились различные корректировки. На Рисунках 3.2.16-3.2.17 (для Минска) и Рисунках 3.2.18-3.2.19 (для Кобрина) изображены средние смоделированные стоимости за кв.м и усреднённые коэффициенты корректирования для объектов в разрезе количества комнат.

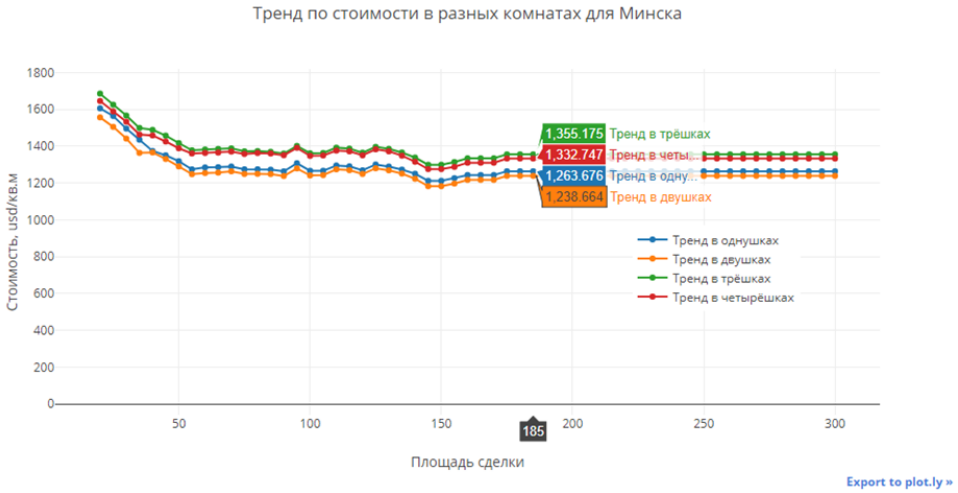

Рисунок 3.2.16 – Тренд стоимости по площади в разрезе 1-, 2-х, 3-х и 4-х комнатных квартир для Минска

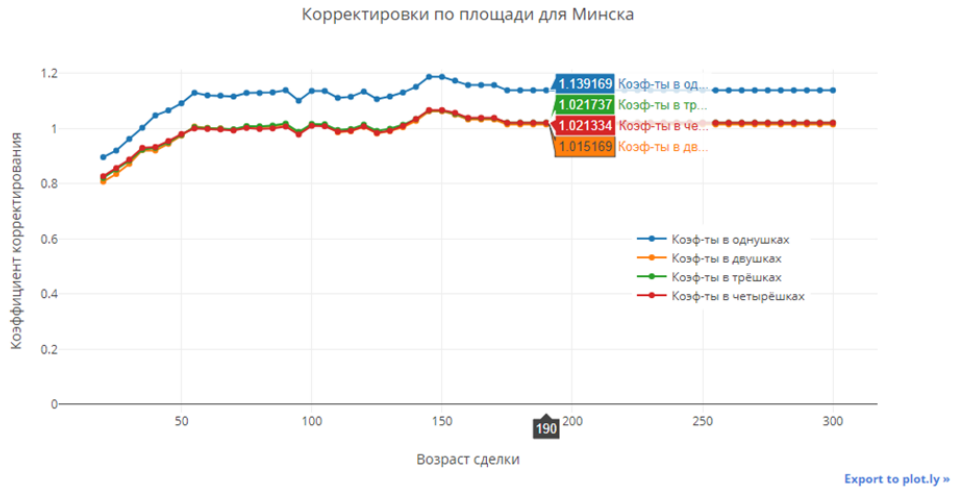

Рисунок 3.2.17 – Коэффициенты корректирования по площади в разрезе 1-, 2-х, 3-х и 4-х комнатных квартир для Минска

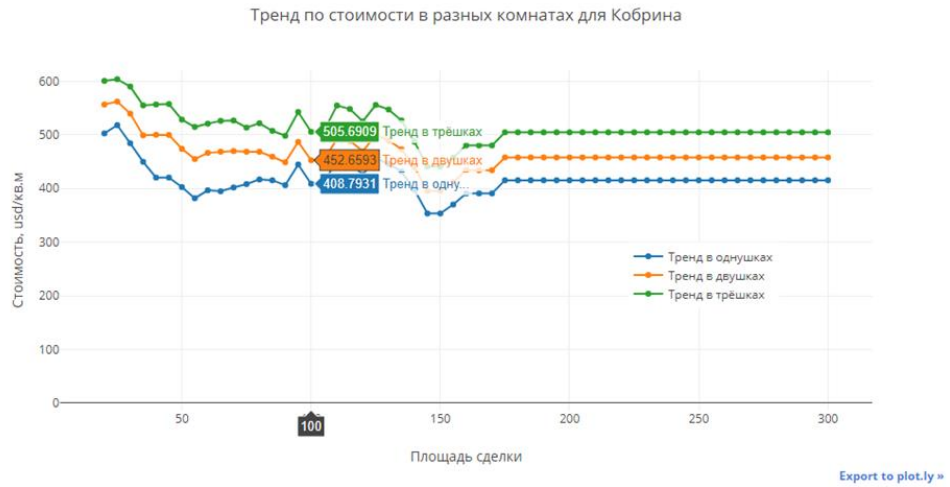

Рисунок 3.2.18 – Тренд стоимости по площади в разрезе 1-, 2-х и 3-х комнатных квартир для Кобрина

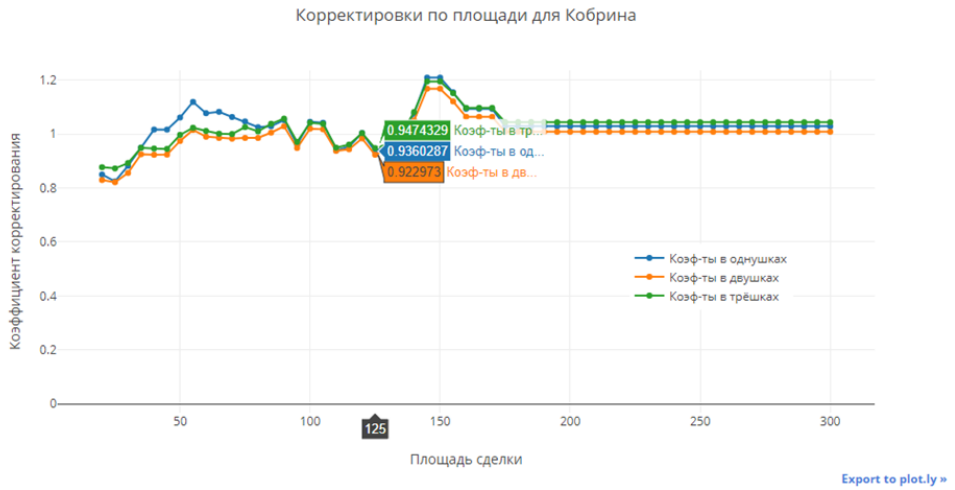

Рисунок 3.2.19 – Коэффициенты корректирования по площади в разрезе 1-, 2-х и 3-х комнатных квартир для Кобрина

К сожалению, времени для того, чтобы сравнить полученные корректировки с корректировками по модели оценки квартир, представленной в разделе 2,3, не хватило. Однако, из полученных трендов по стоимости можно сделать вывод о том, что однушки стоят

практически также, как двушки, а трёшки – почти также, как четырёшки, но из полученных корректировок можно сделать вывод о том, что модель не нашла существенных отличий в Минске между 2-х, 3-х и 4-х комнатными квартирами. В Кобрине же ситуация немного иная: тренды по стоимости отличаются существенно и коэффициенты корректирования отличаются до 90 кв.м, дальше – становятся практически равными.

3.2.4 Анализ результатов

Предложения по усовершенствованию модели

Одним из предложений было то, что нужно проанализировать факторы оценки и выявить (если такие есть) те, которые можно убрать без существенной потери качества модели. Чтобы полностью решить эту задачу, не хватило времени проекта. Однако, некоторые результаты всё же были получены. Экспериментируя с удалением факторов, было выявлено, что можно избавиться от следующих признаков: «Доступность центра», «Санитарно-защитная зона», «Тип конструкций», «Свободная планировка», «Зона рекреации», «Видеонаблюдение». При этом, точность модели не падает и количество объектов, попадающих в допустимые диапазоны по ошибкам, не уменьшается. Подробнее, в Приложении 1, файл «Удаление факторов из модели.xlsx».

При построении модели было замечание о том, что нужно заменить фактор «Оценочная зона». Сам по себе этот признак очень хорошо улучшает модель, так как описывает положение объектов внутри одного города. Но неизбежна проблема расширения выборки при масштабировании модели на территорию всей Республики. На последних этапах проекта, была выдвинута идея о том, чтобы попробовать вместо оценочных зон использовать координаты объектов. Присвоили координаты всем объектам в обучающей выборке и удалили факторы «Оценочная зона» и «Ранг». Полученная модель стала точнее оценивать объекты в Минске и Кобрине на ~1%. Количество объектов, которое попадает в допустимые диапазоны по ошибке, осталось на прежнем уровне, но выросло количество объектов, у которых ошибка находится в окрестности нуля (подробнее – в приложении 1, файл «Удаление факторов из модели.xlsx»). Таким образом, при добавлении двух факторов (x и y координаты), избавились от двух «проблемных» факторов. Однако, стоит проверить эту теорию на всей территории Республики, чтобы точно удостовериться в сделанных выводах. Эти предложения от части уже решены при анализе результатов, однако часть вопросов осталась незакрытой.

4. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Интернет-портал. Статья «Машинное обучение: методы и способы». [https://www.cio.ru/articles/210918-Mashinnoe-obuchenie-metody-i-sposoby]

2. Интернет-портал. Статья «Обучение с подкреплением на языке Python». [https://habr.com/ru/company/piter/blog/434738/]

3. Интернет-портал. Статья «Как выбирать алгоритмы для машинного обучения Microsoft Azure». [https://habr.com/ru/company/microsoft/blog/317512/]

4. Интернет-портал. Статья «Predicting House Prices Using Linear Regression». [https://medium.com/africa-creates/predicting-house-prices-using-linear-regression-fe699d091e04]

5. Интернет-портал. Статья «Create a model to predict house prices using Python» [https://towardsdatascience.com/create-a-model-to-predict-house-prices-using-python-d34fe8fad88f]

6. Интернет-портал. Статья «Machine Learning: Regression–predict house price (lesson 2)». [https://towardsdatascience.com/regression-predict-house-price-lesson-2-5e23bec1c09d]

7. Интернет-портал. Статья «Machine Learning Fundamentals: Predicting Airbnb Prices». [https://www.dataquest.io/blog/machine-learning-tutorial/]

8. Интернет-портал. Статья «Machine Learning Project: Predicting Boston House Prices With Regression». [https://towardsdatascience.com/machine-learning-project-predicting-boston-house-prices-with-regression-b4e47493633d]

9. Интернет-портал. Статья «Покупка оптимальной квартиры». [https://habr.com/ru/post/264407/]

10. Интернет-портал. Статья «Predicting House Price Using Regression Algorithm for Machine Learning». [https://yalantis.com/blog/predictive-algorithm-for-house-price/]

11. Интернет-портал. Статья «Открытый курс машинного обучения. Тема 3. Классификация, деревья решений и метод ближайших соседей». [https://habr.com/ru/company/ods/blog/322534/]

12. Интернет-портал. Статья «Открытый курс машинного обучения. Тема 4. Линейные модели классификации и регрессии». [https://habr.com/ru/company/ods/blog/323890/]

13. Интернет-портал. Статья «Линейная регрессия в машинном обучении». [https://neurohive.io/ru/osnovy-data-science/linejnaja-regressija/]

14. Интернет-портал. Статья «10 главных алгоритмов машинного обучения». [http://ru.datasides.com/code/algorithms-machine-learning/]

15. Интернет-портал. Статья «Regression in Machine Learning». [https://medium.com/datadriveninvestor/regression-in-machine-learning-296caae933ec]

16. Интернет-портал. Статья «Как легко понять логистическую регрессию». [https://habr.com/ru/company/io/blog/265007/]

17. Интернет-портал. Статья «Логистическая регрессия (Logistic Regression)». [https://wiki.loginom.ru/articles/logistic-regression.html]

18. Интернет-портал. Статья «Классификатор kNN». [https://habr.com/ru/post/149693/]

19. Интернет-портал. Статья «K Nearest Neighbors - Regression». [https://www.saedsayad.com/k_nearest_neighbors_reg.htm]

20. Интернет-портал. Статья «Градиентный спуск: всё, что нужно знать». [https://neurohive.io/ru/osnovy-data-science/gradient-descent/]

21. Интернет-портал. Статья «Градиентый бустинг — просто о сложном». [https://neurohive.io/ru/osnovy-data-science/gradientyj-busting/]

22. Интернет-портал. Статья «Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес». [https://habr.com/ru/company/ods/blog/324402/]

23. Интернет-портал. Статья «Открытый курс машинного обучения. Тема 10. Градиентный бустинг». [https://habr.com/ru/company/ods/blog/327250/]

24. Интернет-портал. Статья «Случайный лес (Random Forest)». [https://dyakonov.org/2016/11/14/%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D0%B9-%D0%BB%D0%B5%D1%81-random-forest/]