Московский государственный университет им. М В Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра Исследования операций

Курсовая работа

по теме

Машинное обучение в моделировании экономики

Выполнила

Серебрякова Софья,

группа 312

С

Научный руководитель

Оленев Н Н

Москва, 2020

**Оглавление**

**1. Введение**

**2. Основная часть**

**2.1 Анализ данных**

**2.2 Построение модели**

**3. Выводы**

**4. Список литературы**

**Введение**

В данной работе я хочу проанализировать набор данных. Затем найти лучшую модель для их описания и предсказания. Таблица с данными, используемая мной, взята из [1]. Она находится в файле [TaiwanCreditDefaults.xls](http://localhost:8888/edit/PycharmProjects/msu_2019-2020/ML_coursework/TaiwanCreditDefaults.xls) и представляет собой информацию о выплате задолженности по кредитной карте, некоторые демографические показатели и историю предыдущих платежей жителей Тайваня в период с апреля по сентябрь 2005 года. В основной части работы я более подробно расскажу о структуре набора данных.

Анализ и построение модели я буду делать используя python3 а так же некоторые библиотеки:

1. Pandas – для работы с набором данных
2. Numpy – для работы с массивами
3. Matplotlib.pyplot и seaborn – для визуализации данных
4. Sklearn и xgboost – для использования моделей машинного обучения

Весь код написан в Jupyter Notebook и находится в файле Prediction\_of\_default\_of\_credit\_card.ipub.

**Основная часть**

Давайте подробнее остановимся на структуре таблицы. Она состоит из 25 столбцов.

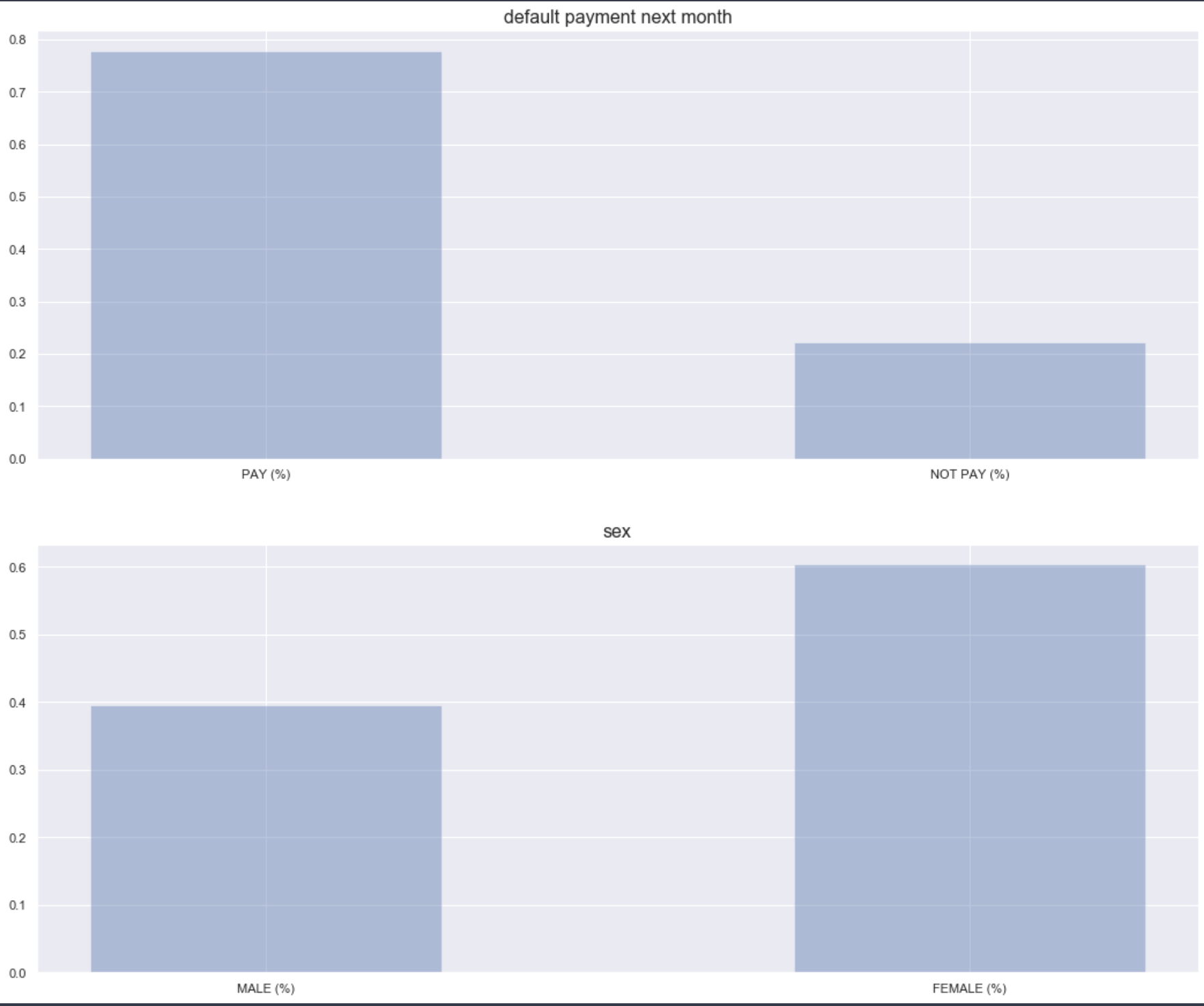
* ID: уникальный номер клиента
* LIMIT\_BAL: размер кредита в долларах
* SEX: пол (1 == мужчина, 2 == женщина)
* EDUCATION: Образование (1 == аспирантура, 2 == высшее, 3 == среднее, 4 == другое, 5 == нет данных, 6 == нет данных)
* MARRIAGE: Семейное положение (1 == замужем/женат, 2 == не замужем, 3 == другое)
* AGE: возраст
* PAY\_0: статус погашения в сентябре, 2005 (-1 == оплачен, 1 == отсрочка платежа на месяц, 2 == отсрочка на два месяца, … 9 == отсрочка на девять месяцев)
* PAY\_2: статус погашения в августе, 2005 (измеряется аналогично предыдущему)
* PAY\_3: статус погашения в июле, 2005 (измеряется аналогично предыдущему)
* PAY\_4: статус погашения в июне, 2005 (измеряется аналогично предыдущему)
* PAY\_5: статус погашения в мае, 2005 (измеряется аналогично предыдущему)
* PAY\_6: статус погашения в апреле, 2005 (измеряется аналогично предыдущему)
* BILL\_AMT1: сумма выписки по счету в сентябре, 2005 (в долларах)
* BILL\_AMT2: сумма выписки по счету в августе, 2005 (в долларах)
* BILL\_AMT3: сумма выписки по счету июле, 2005 (в долларах)
* BILL\_AMT4: сумма выписки по счету в июне, 2005 (в долларах)
* BILL\_AMT5: сумма выписки по счету в мае, 2005 (в долларах)
* BILL\_AMT6: сумма выписки по счету в апреле, 2005 (в долларах)
* PAY\_AMT1: сумма предыдущего платежа в сентябре, 2005 (в долларах)
* PAY\_AMT2: сумма предыдущего платежа в августе, 2005 (в долларах)
* PAY\_AMT3: сумма предыдущего платежа в июле, 2005(в долларах)
* PAY\_AMT4: сумма предыдущего платежа в июне, 2005 (в долларах)
* PAY\_AMT5 сумма предыдущего платежа в мае, 2005 (в долларах)
* PAY\_AMT6: сумма предыдущего платежа в апреле, 2005 (в долларах)
* Default payment next month: выплата по кредиту (1== да, 0 == нет)

**Анализ данных**

Для того, чтобы использовать ту или иную модель машинного обучения, надо провести предварительный анализ данных. По его завершению сделать выводы о том какие признаки являются важными для нас для установления факта оплаты по кредитной карте.

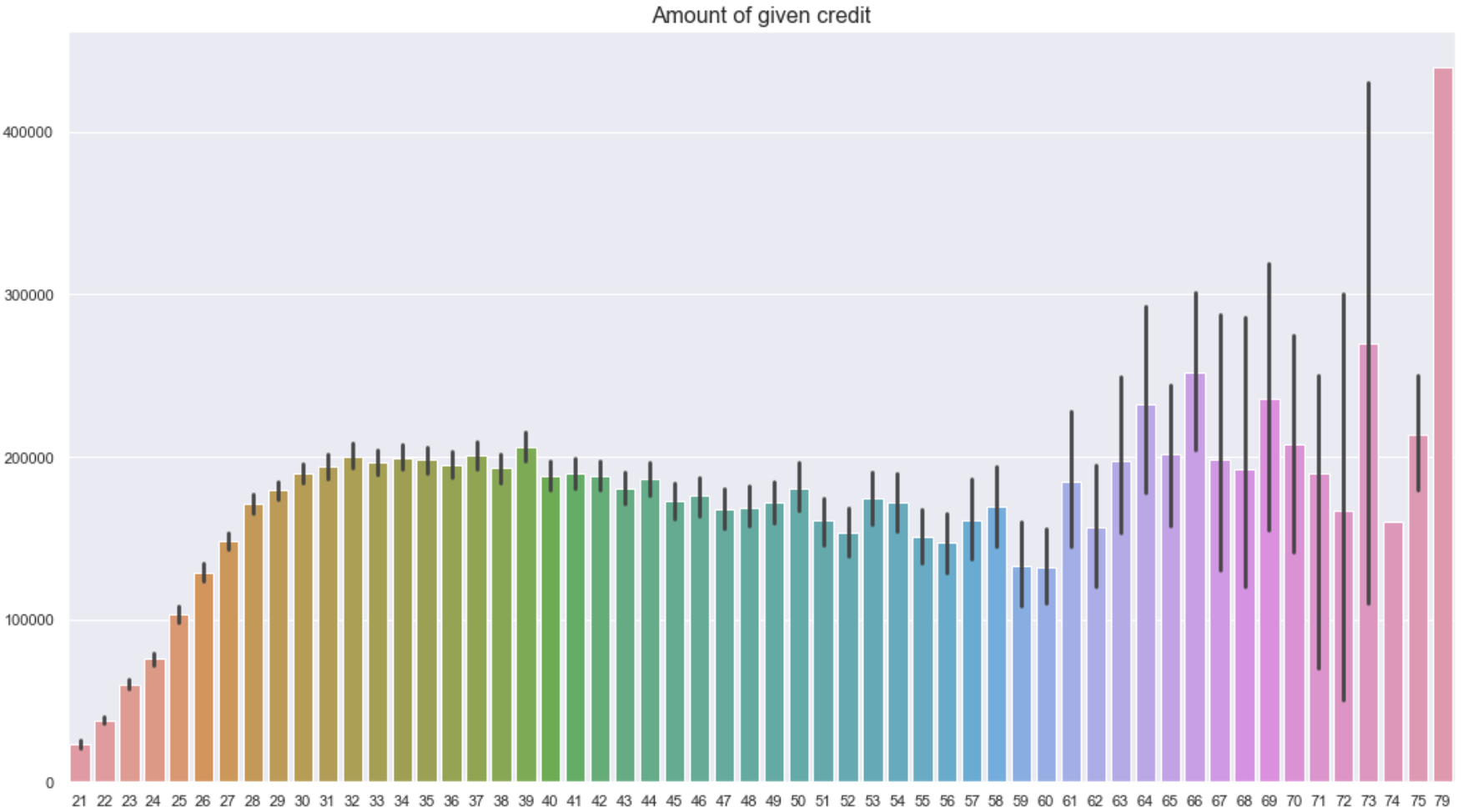
Для начала по данным таблицы я построила 2 гистограммы: 1) распределение количества людей (в процентах) заплативших по кредиту и 2) распределение людей по полу (в процентах).

Результаты приведены ниже.



Можно видеть, что около 77% жителей Тайваня исправно платят по задолженности. Так же на втором графике видно, что около 60% всех заемщиков – это женщины.

Далее я построила зависимость лимита по карте от возраста.

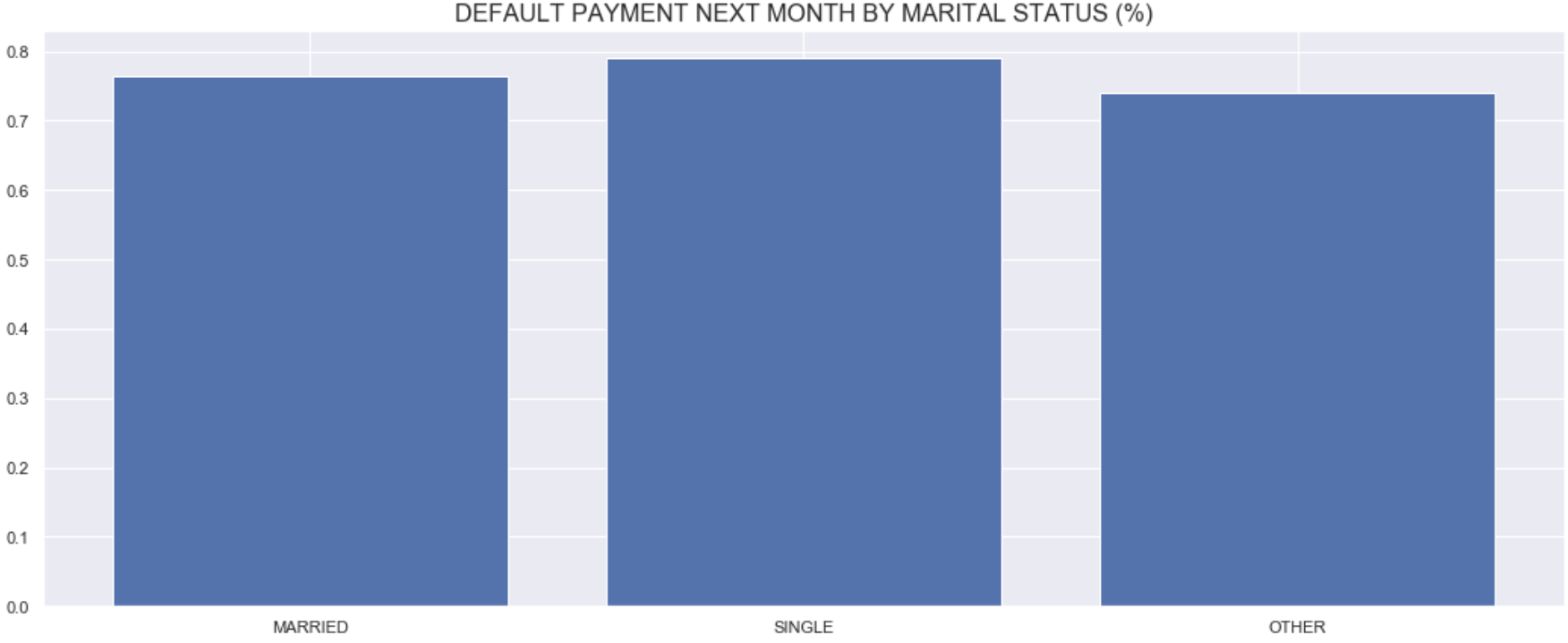


Видим, что люди в возрасте до 30 лет берут меньшие суммы в кредит, чем люди старше. Еще одно наблюдение, которое можно сделать из этого графика – это большой разброс значений выданного кредита у пожилых людей (60 – 75 лет)

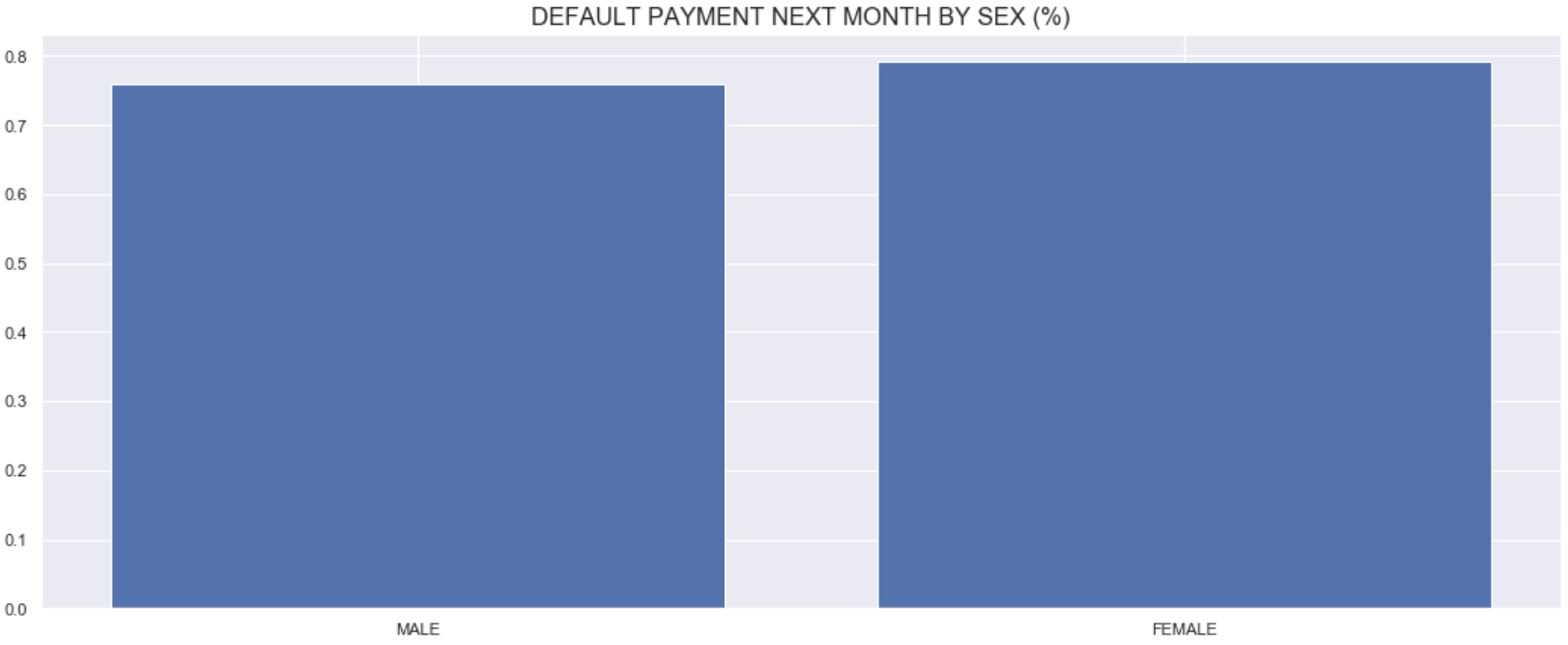
Теперь проверим, как отдельные параметры человека влияют на выплату по кредиту.

Я рассмотрела следующие параметры – 1) семейное положение, 2) возраст, 3) пол и 4) образование.

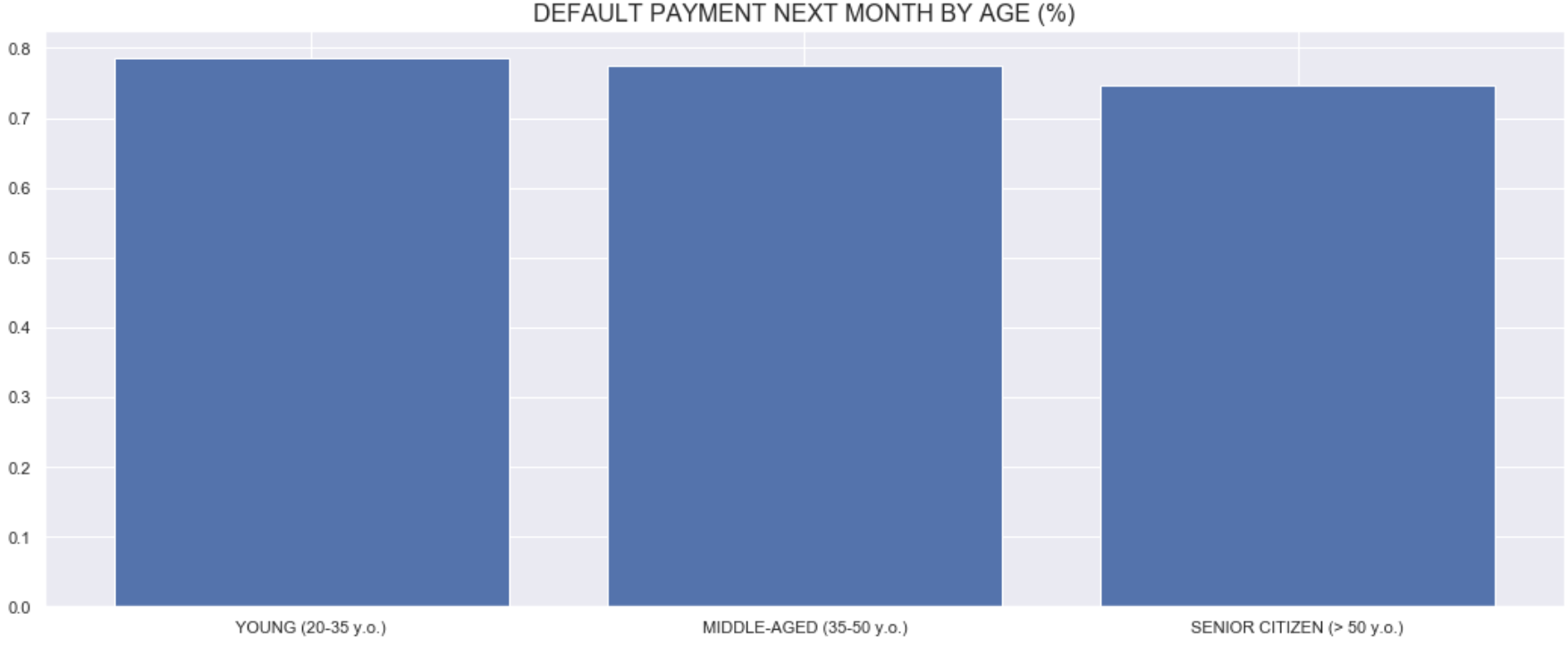
На графике ниже приведены сводные данные по количеству людей (в процентах), выплативших кредит в конце месяца. Люди разбиты на три группы по их семейному положению. Как видно сильного различия между группами людей по семейному положению нет. Однако одинокие жители Тайваня выплачивают задолженность по кредиту более регулярно, чем замужние/женатые.



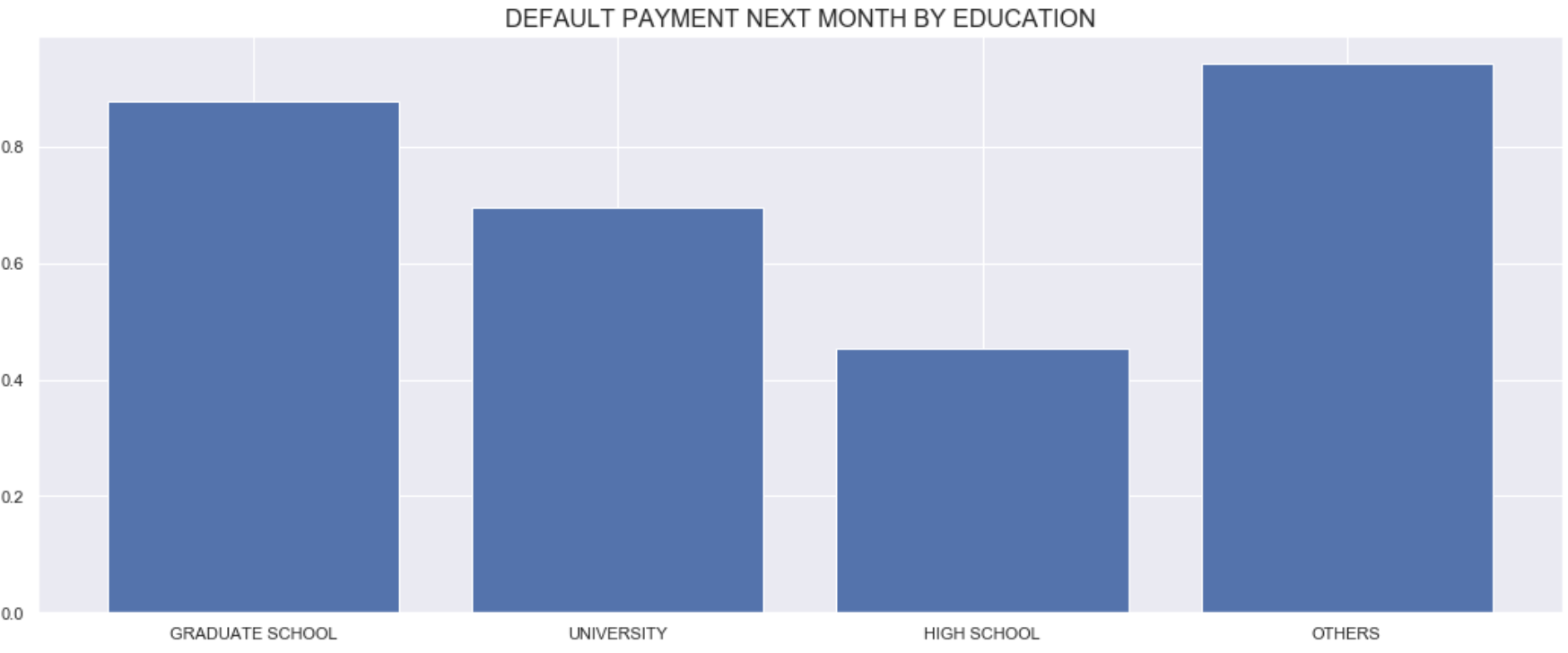
На графике ниже приведены сводные данные по количеству людей (в процентах), выплативших кредит в конце месяца. Люди разбиты на две группы по полу. Как видно сильного различия между группами нет. Однако женщины производят выплаты более исправно.



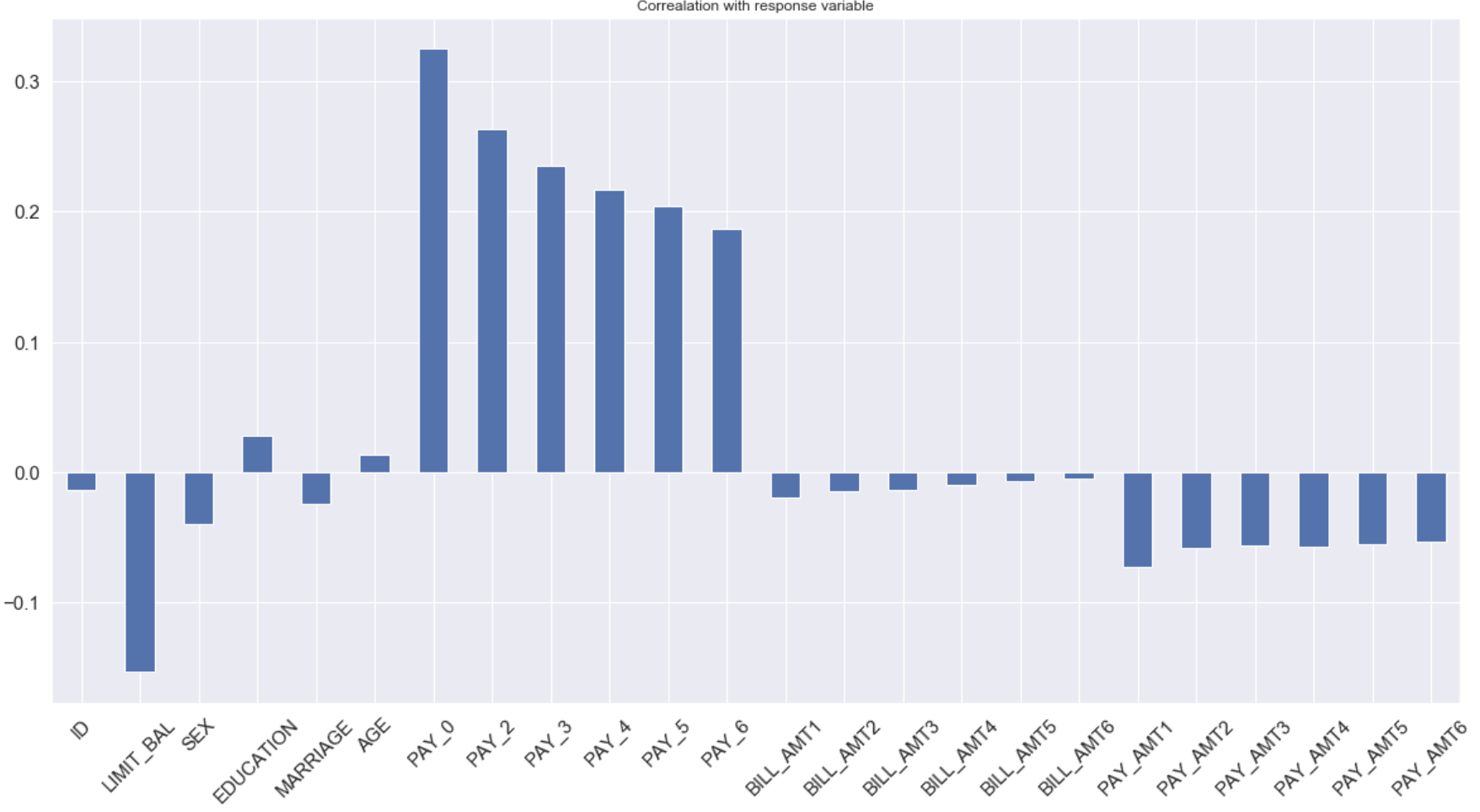
Что касается зависимости выплат от возраста, тут можно заметить, что пожилые люди производят выплаты реже. Возможно, это связано с тем, что они специально берут кредиты, рассчитывая на то, что можно не возвращать деньги в связи с тем, что им осталось недолго жить.



И самый на мой взгляд показательный график – это зависимость погашения долга от уровня образования. Тут явно видно что, чем выше уровень образования, тем более исправно человек платит.



И последние, что я сделала, проводя анализ данных, – это построила график корреляции выплаты по кредитной карте с другими данными из таблицы

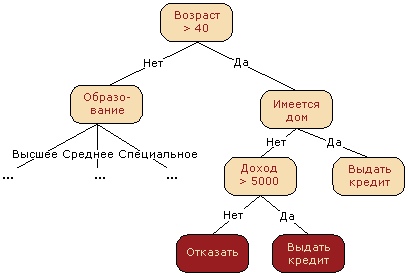


**Построение модели**

Уберем из таблицы лишние данные. Оставим те, по которым можно построить модель.

В данной работе я решила рассмотреть несколько алгоритмов машинного обучения. Начнем с самого просто – дерево решений. Для начала расскажу немного о сути этого алгоритма.

Это семейство алгоритмов, которое занимает важное место в машинном обучении особенно часто его применяют в задачах кредитного скоринга. Связано это с тем, что дерево решений представляет собой интуитивно понятную модель. Оно работает по аналогии с человеческим мышлением. Ведь когда человек хочет понять ту или иную вещь, он будет задавать последовательность из простых вопросов, которые в итоге приведут его к ответу. Давайте разберем простой пример. Допустим в банк приходит клиент, хочет взять кредит. Задача банка по некоторым известным данным о клиенте сделать вывод отдаст ли он деньги. Сначала сотрудник банка спрашивает, сколько клиенту лет, какое у него образование, семейное положение, брали ли клиент кредиты до этого. Задачу можно решить следующим решающим деревом.



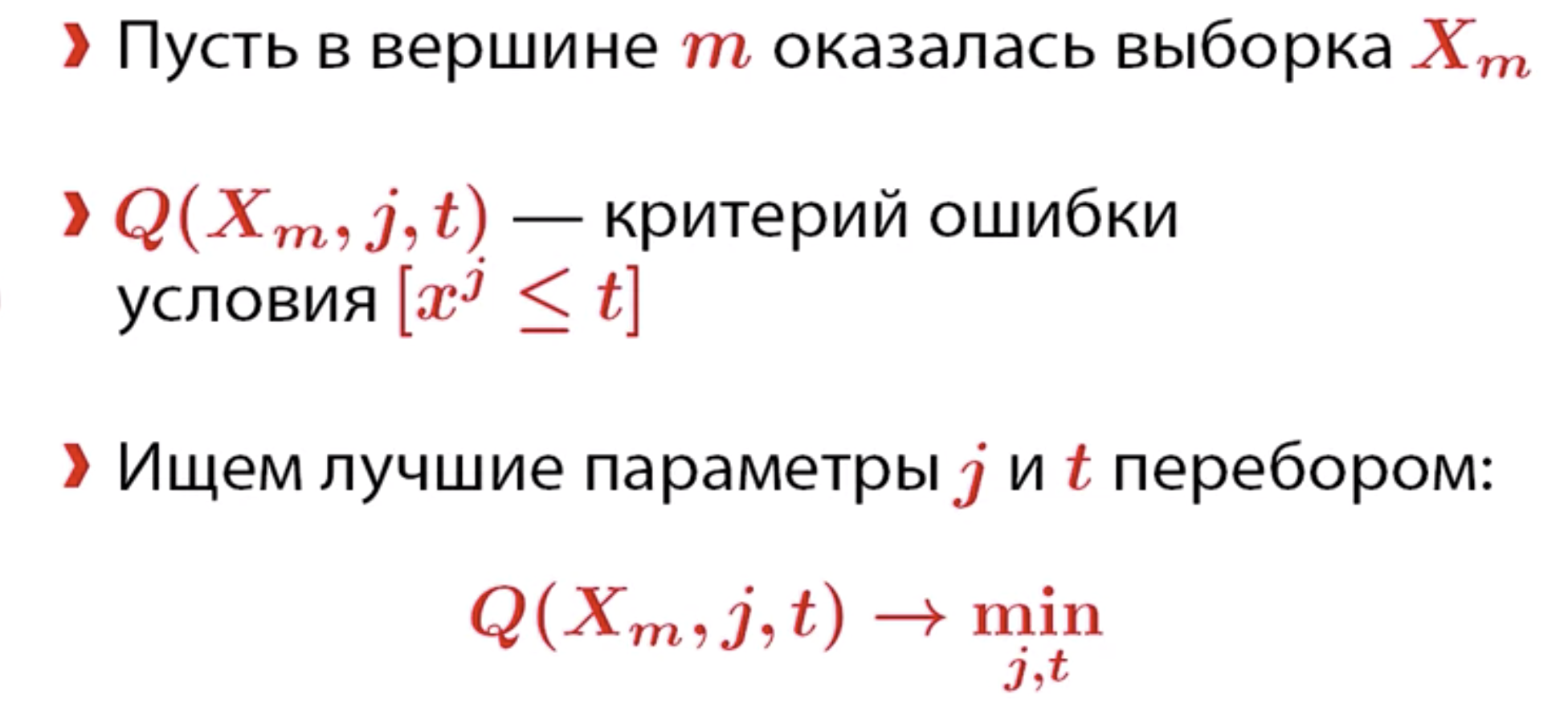
Итак, мы рассмотрели пример решающего дерева. Обычно это некоторое бинарное дерево, у которого в каждой внутренней вершине записано простое условие. В зависимости от того выполнено оно или нет, мы будем идти направо или налево.

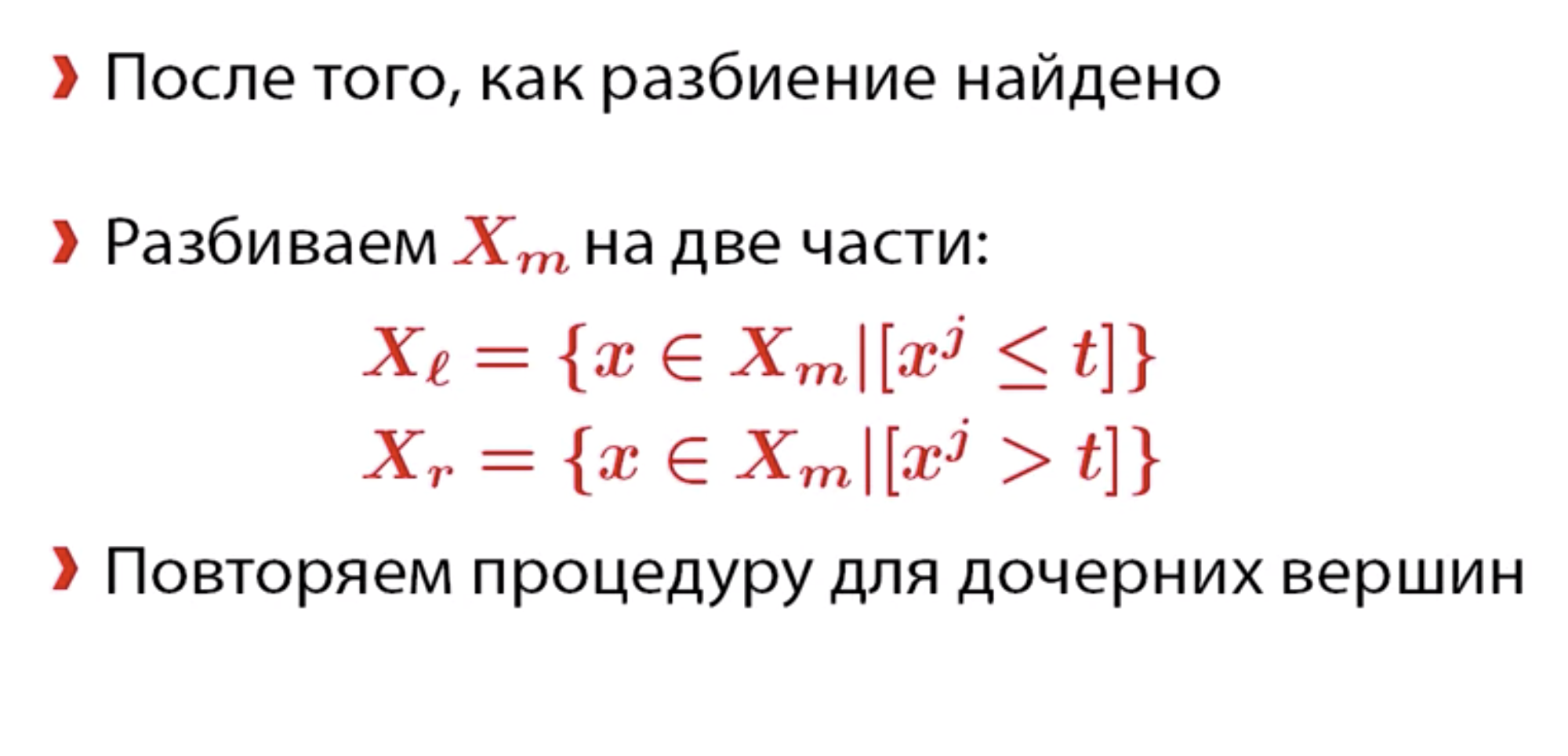
Таким образом, мы берем некий объект, стартуем из корня и движемся по дереву, проверяя условие в текущей вершине. В зависимости от его выполнения идем либо влево, либо вправо. В конце концов мы попадаем в лист, в котором записан прогноз, который и выдается в качестве ответа модели. Можно строить и более сложные небинарные деревья, но, как правило, используются именно бинарные. Этого достаточно, чтобы решать большинство задач. Условия во внутренних вершинах также используются крайне простые. Наиболее частый вариант —мы проверяем, находится ли значение j-того признака левее, чем некоторый порог. То есть мы берем у объекта j-тый признак, сравниваем с порогом t, и если оно меньше порога, мы идем влево, если больше порога, мы идем вправо. Это опять же очень простое условие, которое зависит всего от одного признака, но его достаточно, чтобы решать многие сложные задачи. Прогноз в листе будет вещественным, если это регрессия, и он будет пытаться как можно лучше приблизить истинный ответ. Если это классификация, то есть два варианта. Дерево может выдавать либо номер класса (тогда в каждом листе будет записан просто тот или иной класс), либо распределение вероятности на классах. В этом случае в каждом листе будет записан некоторый вектор длины k, если k — это число классов, который будет говорить, насколько вероятно, что объект относится к тому или иному классу.

Теперь поговорим о том, как строить решающие деревья, как обучать их на конкретную выборку. Решающие деревья имеют недостаток, они очень легко переобучаются. Легко построить дерево, у которого каждый лист будет соответствовать одному объекту обучающей выборки. Поскольку, дерево может достичь нулевой ошибки на обучающей выборке, нам не подходит любое дерево. В принципе, скорее всего, каждую задачу можно решить деревом, которое будет иметь нулевую ошибку и при этом не быть переобученным, и, скорее всего, это будет минимальное дерево из всех, у которых нулевая ошибка. Минимальным оно может быть, например, в смысле количества листьев, то есть можно поставить задачу построить такое решающее дерево, которое не ошибается на данной выборке и при этом имеет меньше всего листьев. К сожалению, эта задача NP-полная, то есть ее невозможно решить за разумное время, поэтому в машинном обучении пользуются гораздо более простым подходом: строят дерево жадно, последовательно, от корня к листьям.

Начинаем с пустого дерева, дальше выбираем каким-то образом корень, который разбивает нашу выборку на две, дальше разбиваем потомков этого корня и так далее. Ветвим дерево до тех пор, пока не решим, что этого достаточно. Давайте выясним, как именно можно разбивать конкретную вершину на две, на два потомка. Итак, как мы договаривались,что будем использовать в качестве условия для разбиения очень простую штуку: будем брать один из признаков, j-тый, и сравнивать его с порогом t. Если значение j-того признака меньше порога, отправляем объект в одну сторону, например, влево, если больше порога, то вправо.

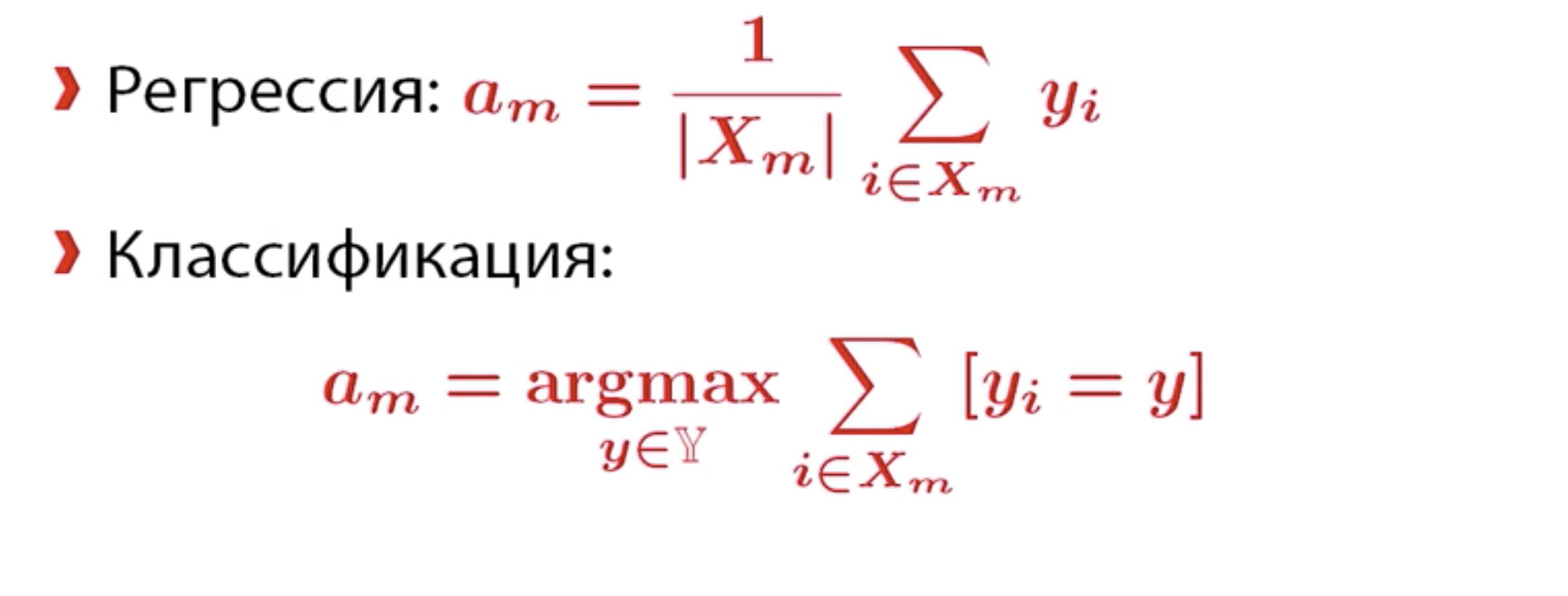
Допустим, мы сейчас находимся в вершине m, и в нее попало некоторое количество объектов обучающей выборки, например, xm, будем так обозначать это подмножество объектов. Мы будем использовать некоторый критерий ошибки Q, который зависит от того, какие объекты попали в данную вершину, то есть xm, и от параметров разбиения j и t, то есть на основе какого признака мы разбиваем и с каким порогом мы сравниваем значение этого признака. Будем выбирать параметры j и t-разбиения так, чтобы они минимизировали данный критерий ошибки Q. Подбирать параметр j можно перебором, поскольку признаков у нас конечное число, а из всех возможных значений параметра t, порога, можно рассматривать только те, при которых получаются различные разбиения. Можно показать, что этих значений t столько, сколько различных значений признака j на обучающей выборке. Например, можно отсортировать все значения j-того признака и брать пороги между этими значениями. После того как мы выбрали конкретное разбиение, выбрали оптимальные значения параметров j и t, мы разбиваем нашу вершину на две: левую и правую. При этом часть объектов, а именно те, на которых j-ый признак меньше или равен порогу t, отправляются влево, и будем обозначать это подмножество как xl, а часть объектов из xm, те, у которых значение j-того признака больше порога t, отправляются вправо, и это подмножество обозначается как xr. Эту процедуру можно повторить дальше для двух дочерних вершин, тем самым углубляя наше дерево. Схематично все вышесказанное можно представить так:





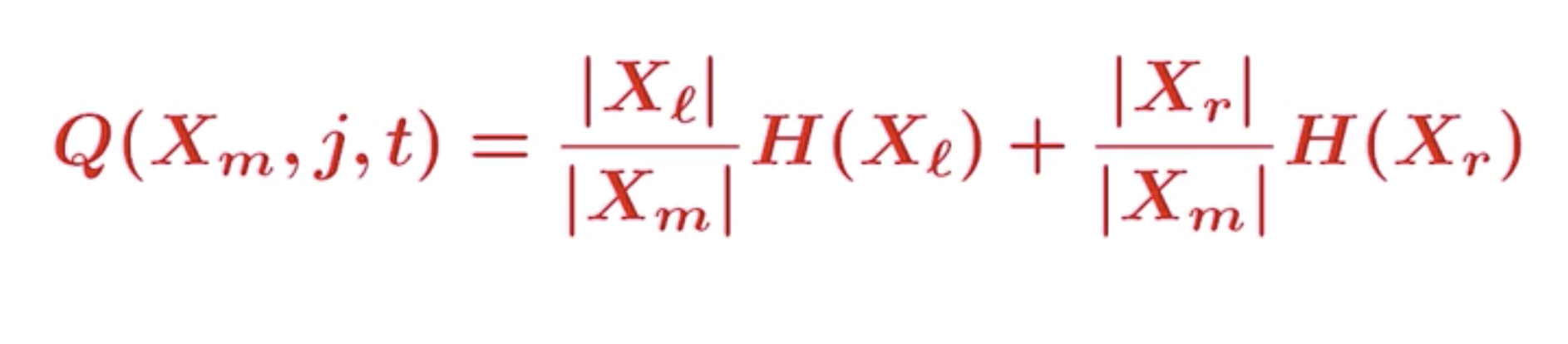
В какой-то момент нам все же придется остановиться. Как понять, что данную вершину уже разбивать не нужно, что ее можно объявить листом и выдавать прогноз для того объекта, который в нее попал? Критериев останова очень много. Например, можно смотреть, сколько объектов находится в данной вершине. Если там всего один объект обучающей выборки, понятно, что дальше разбивать не имеет смысла; если же больше, можно разбивать дальше. Или, например, можно смотреть, какие объекты попали в эту вершину: если они все относятся к одному классу в задаче классификации, можно прекратить разбиение; если же есть несколько классов, можно разбивать дальше. Или, например, можно следить за глубиной дерева и останавливать разбиение, если глубина превышает некоторый порог, например, 10.

Теперь надо разобраться, как выбирать ответ в листе, если мы решили вершину объявить листом. Итак, в данный лист попала некоторая подвыборка xm, некоторое подмножество объектов обучающей выборки, и нужно выбрать какой-то один прогноз, который будет оптимален для данной подвыборки. В случае с регрессией мы знаем, что если функционал – среднеквадратичная ошибка, то оптимально выдавать средний ответ по этой подвыборке, то есть мы суммируем ответы по всем объектам i, которые попали в данную вершину, и делим на количество объектов в этой вершине. Это и будет оптимальным прогнозом в случае с задачей регрессии среднеквадратичного функционала. Если мы решаем задачу классификации, то наиболее логичным выбором будет возвращать тот класс, который наиболее популярен в выборке xm. То есть мы для каждого класса y считаем, сколько объектов этого класса попало в данную вершину, и возвращаем тот, который максимален, которого больше всего. Если же мы хотим возвращать вероятности классов в данной вершине, это тоже очень легко сделать. Вероятность k-того класса оценивается как доля объектов k-того класса в данной вершине среди всех объектов, впавших в эту вершину, то есть среди всех объектов из xm.



У нас остались открытыми два вопроса: как именно разбивать, как задавать критерии разбиения, как оценивать ошибку разбиения в данной вершине и как выбирать критерий останова?

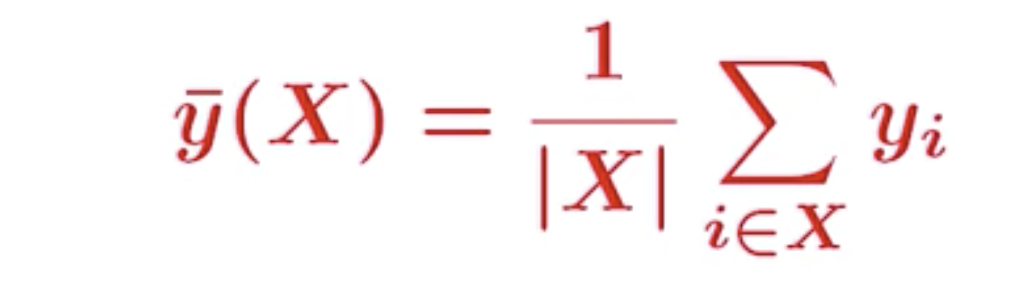
Итак, решающее дерево будем строить простое, у которого в каждой вершине записано условие, которое берет значение j-го признака, сравнивает его с порогом t, и если значение признака меньше порога t, то объект идет в левое поддерево, если больше, то в правое поддерево. Допустим, у нас есть некоторая вершина с номером m, в которую попала подвыборка обучающей выборки X с индексом m, и мы хотим разбить эту вершину на два поддерева. Мы уже говорили, что будем делать это с помощью критерия ошибки, который показывает, насколько качественно данное условие, данная пара (признак j и порог t) разбивает объект, попавший в эту выборку, на две подвыборки. Этот критерий обозначается буквой Q. После того, как конкретный критерий выбран, мы разбиваем выборку Xm на две части. Xl — это те объекты, у которых значения j-го признака меньше или равно порога t, и Xr — это те объекты, у которого значения j-го признака больше порога t. Выборка Xl отправляется в левое поддерево, выборка Xr отправляется в правое поддерево. После этого можно повторить процедуру для левого и правого листа, который мы породили из данной вершины. Критерий ошибки будем записывать вот в таком виде:



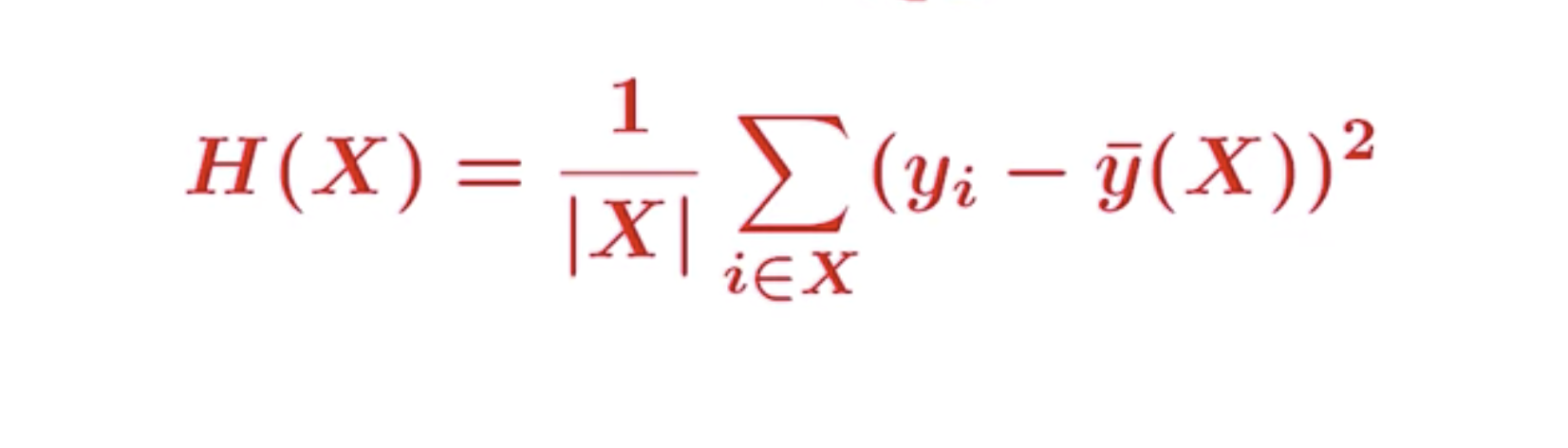
Давайте разберемся, что означают его части. Он состоит из двух слагаемых. В первом слагаемом используется функция H, которой на вход передается выборка Xl, то есть та часть объектов, которая пойдет в левое поддерево. Функция H должна измерять качество этого подмножества, то есть насколько сильный разброс ответов имеет место при попадании выборки Xl в левое поддерево. Аналогично второе слагаемое измеряет то же самое для правого поддерева, в нем функцию H передает выборка Xr, и она должна измерить, насколько силен разброс ответов в подмножестве Xr. Обратите внимание, что значение функции H на Xl и Xr нормируется, они домножаются на коэффициенты, которые равны доли объектов, которая идет влево, и доли объектов, которая идет вправо. Зачем это нужно? Представьте, что у нас в вершине m находится 1000 объектов, и из них 990 идут в левое поддерево, и 10 — в правое поддерево. При этом 990 объектов, которые идут влево, оказываются одного класса, то есть это очень хорошее поддерево, а 10 объектов, которые идут вправо, относятся ко всем возможным классам. Распределение классов там равномерное, то есть эта подвыборка получается плохой, но при этом в ней всего 10 объектов, и нам не так страшно, что она получилась плохой, при том, что 990 попали в правильную вершину. Поэтому нам важно домножать значение качества подмножества на размер этого подмножества.

Итак, функция H(X) называется критерием информативности, и она должна измерять, насколько силен разброс ответов в выборке X. По сути, эта функция зависит от того, какие ответы имеют объекты из множества X. Ее значение должно быть тем меньше, чем меньше разброс этих ответов.

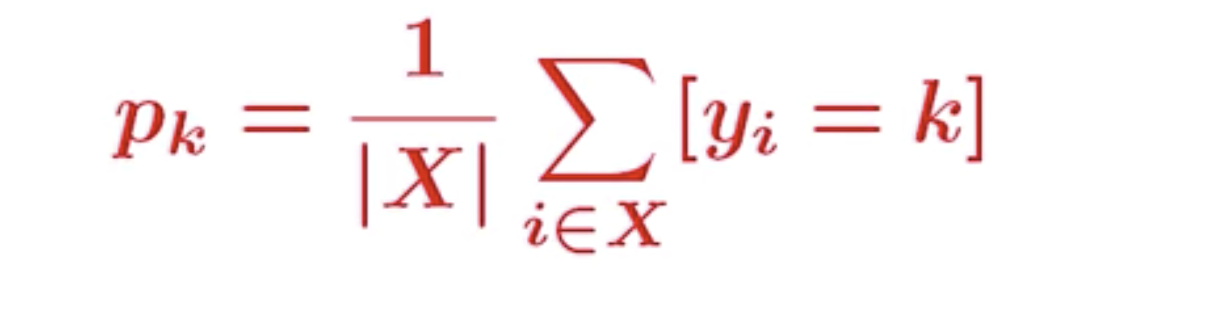
Давайте разберем несколько примеров критерия информативности для задач регрессии и классификации. Начнем с регрессии. Понятно, что в случае с регрессией измерить разброс довольно просто — это просто дисперсия ответов этой выборки. Чтобы ее измерить, сначала вычислим средний ответ выборки X, который обозначается буквой y с верхней чертой. Он вычисляется просто, как среднее значение.



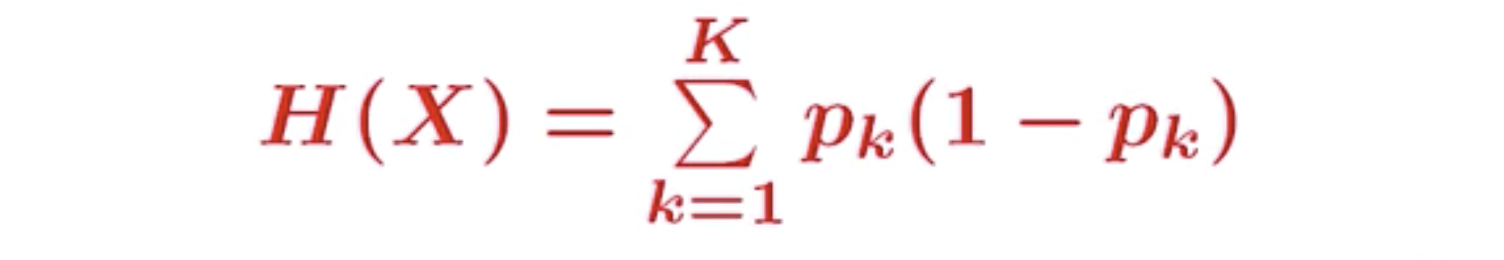
А затем вычислим дисперсию выборки, которая вычисляется как среднее значение квадрата отклонения ответа на объекте от среднего ответа по выборке.



Перейдем к классификации. В случае с классификацией все немного сложнее. Нам понадобится вспомогательная величина, которая показывает для k-го класса, какова доля объектов класса k в выборке X. Будем обозначать эту величину как pk-тое. И она, собственно, вычисляется по этой формуле, смысл которой как раз таки доля объектов класса k в выборке X.

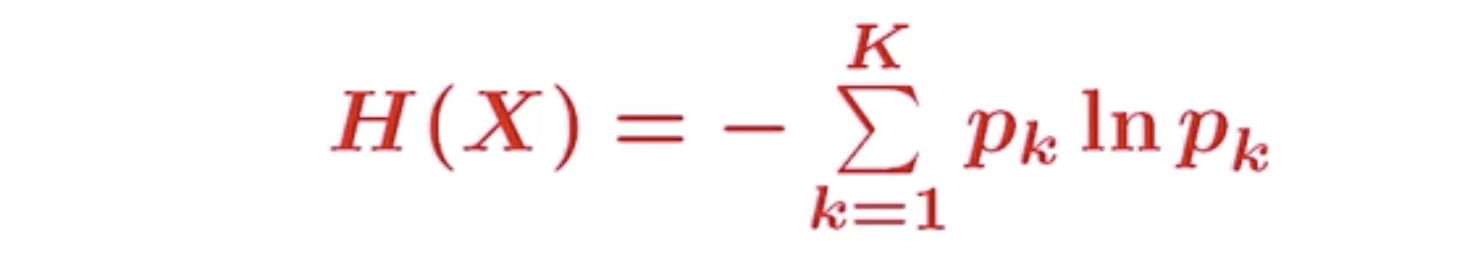


На основе этих чисел pk вводятся критерии информативности для классификации, и первый из них — это критерий Джини. Он вычисляется по такой формуле.



В нем стоит суммирование по всем классам, от первого до K, и для каждого класса вычисляется произведение pk на (1 − pk), где pk — это доля объектов k-го класса в вершине. Обратите внимание, что все числа в этой сумме положительные, поэтому критерий Джини всегда не отрицательный, он не меньше нуля. При этом, если в нашей выборке X все объекты относятся к какому-то одному классу, например к первому, то все слагаемые в этой сумме будут нулевыми, значит и сам критерий Джини будет равен 0. Это означает, что его оптимум достигается в том случае, если все объекты в подвыборке относятся к одному классу. У него есть много интерпретаций. И одна из них следующая: критерий Джини равен вероятности ошибки случайного классификатора, где случайный классификатор устроен так, что он выдает случайный класс от 1 до K, при этом вероятность выдать класс (какое-то k) равна pk, то есть равна пропорции этого класса в общей подвыборке X.

Еще один пример критерия информативности для классификации, это энтропийный критерий. Он вычисляется по такой формуле.



В нем суммируются следующие слагаемые: берется вероятность pk и домножается на логарифм этой вероятности, и все это берется со знаком минус. При этом мы считаем, что если вероятность равна 0, то ноль умножить на логарифм нуля — это то же ноль. Причем это можно доказать по непрерывности, если взять просто предел функции X In X. Для этого критерия выполнено то же самое свойство: если в выборке X находятся объекты ровно одного класса, например первого, то значение энтропийного критерия будет равно 0. И какое бы не было распределение на классах, значение энтропийного критерия не отрицательное. У него так же есть очень интересный физический смысл. По сути, энтропийный критерий — это мера отличия распределения классов от вырожденного. Если распределение вырожденное, то энтропия равна 0, в этом распределении нет ничего неожиданного, мы всегда знаем, что мы будем получать из него. Если же это распределение равномерное, то есть вероятность получить каждый класс в этой выборке одинаковая, то энтропия будет максимальна. У этого распределения максимальный уровень неожиданности. Мы не можем предсказать, что мы получим. Итак, мы выяснили, что критерий ошибки должен минимизировать разброс ответов в обоих поддеревьях после разбиения по какому-то условию. И выражается он через критерий информативности, который измеряет разброс ответов в одном из поддеревьев. В регрессии это просто среднеквадратичная ошибка или дисперсия. В классификации это может быть критерий Джини или энтропийный критерий. Нам осталось узнать, как выбирать критерий останова.

**Критерий останова**

Критерий останова показывает, нужно ли останавливать процесс построения дерева. Например, когда мы разбили дерево до какой-то степени и находимся в определенной вершине, мы хотим понять, нужно ли ее разбивать дальше или же стоит оделить ее листом? Критерий останова должен отвечать на этот вопрос. Худший случай решающего дерева — это дерево, у которого каждый лист соответствует одному объекту обучающей выборки. В этом случае дерево будет максимально переобученным, оно не будет обобщать информацию, полученную из обучающей выборки. Критерий останова, грамотно подобранный критерий останова — это способ борьбы с таким переобучением.

Самый простой критерий останова проверяет, все ли объекты, которые находятся в данной вершине, относятся к одному классу. Понятно, что это работает только для классификации. Это простой и понятный критерий останова, но при этом он будет выполнятся в нетривиальных случаях только на простых выборках. Если выборка и зависимости в ней сложные, то, скорее всего, этот критерий сработает только тогда, когда в каждом листе останется по одному объекту. Гораздо более устойчивый и полезный критерий проверяет, сколько объектов оказалось в данной вершине. Если их больше, чем n, то разбиение продолжается, а если меньше или равно, чем n, то процесс построения останавливается в этой вершине, и n — это некоторый параметр, который нужно подбирать. Если n = 1, мы получаем худший случай с деревом, где в каждом листе по одному объекту. Выбирать n нужно так, чтобы по n объектам, которые попали в вершину, можно было устойчиво построить прогноз, можно было надежно оценить, какой прогноз выдавать на этих объектах. Существует рекомендация, что n нужно брать равным 5, по идее, при 5 точках, если у нас 5 объектов попали в вершину, можно уже более или менее надежно оценить, какой ответ на них нужно выдавать.

Еще один критерий, гораздо более грубый — это ограничение на глубину дерева. Если мы видим, что эта вершина находится на 5-м уровне дерева и максимальная глубина равна 5 мы останавливаем построение независимо ни от чего — ни от распределения классов в этой вершине, ни от числа объектов в ней — просто останавливаем построение. Критерий довольно грубый, но при этом он хорошо себя зарекомендовал при построении композиций, то есть когда мы объединяем много решающих деревьев в один сложный алгоритм.

**Композиция деревьев**

Итак, мы узнали, что такое решающие деревья, и выяснили, что они могут восстанавливать очень сложные закономерности, из-за чего они склонны к переобучению. Решающее дерево слишком легко подгоняется под обучающую выборку и получается непригодным для построения прогнозов. Бороться с переобучением довольно сложно. Надо либо использовать критерии остановок, которые слишком простые и не всегда помогают, либо делать стрижку деревьев, которая, наоборот, слишком сложная. Зачем же мы тогда тратили на них время?

Оказывается решающие деревья очень хорошо подходят для объединения в композиции, для построения одного непереобученного алгоритма на основе большого количества решающих деревьев. Итак, у решающих деревьев есть два недостатка. Первый — они очень сильно переобучаются, а второй — они очень неустойчивы, они очень сильно меняются даже при небольших изменениях в выборке. И на самом деле второй пункт можно обратить в их достоинство с помощью композиции, но об этом чуть позже.

А пока давайте поговорим в целом о том, что такое композиция алгоритмов. Итак, композиция — это объединение n алгоритмов в один. Представьте, что мы каким-то образом нашли N большое алгоритмов b1, ..., bn. Пока не важно, откуда мы их взяли. Просто представьте, что мы их как-то обучили. Чтобы объединить их в композицию, мы усредняем их ответы, то есть суммируем ответы всех этих алгоритмов b1, ..., bn на объекте x, и делим на N большое, то есть на количество этих алгоритмов. Если мы решаем задачу классификации, то далее мы берем знак от этого среднего; если регрессии, то просто возвращаем это среднее как ответ. Алгоритм a(x), который возвращает знак среднего или просто среднее и называется композицией n алгоритмов. А алгоритмы b1, ..., bn, которые мы объединяем в композицию, называются базовыми алгоритмами.

Итак, для того чтобы строить композицию, нужно обучить n базовых алгоритмов. При этом понятно, что нельзя их обучать на всей обучающей выборке. Они получатся одинаковыми, и в их усреднении не будет никакого смысла. Нужно делать их немного различными. Как этого достичь? Например, с помощью рандомизации, то есть обучать их по разным подвыборкам обучающей выборки. Поскольку решающие деревья очень сильно меняются даже при небольших изменениях обучающей выборки, такая рандомизация с помощью подвыборок будет очень хорошо влиять на их различность.

**Вывод**

Из проведенного анализа данных, можно выяснить, что следующие признаки наиболее сильно влияют на не выплату по кредиту: размер выданного кредита, пол, образование, статус погашения и сумма погашения. Также видно несколько интересных моментов в данных. По графикам видно, что женщины более склонны к тому, чтобы вносить платеж во-время. Можно сказать, что люди состоящие в браке склонны не платить вовремя. Размер выданного кредита был довольно случайным после 60 лет. До этого размер кредита возрастал до 39 лет, там он достигает локального минимума и далее снижается. Именно поэтому возраст не был выбран в качестве признака для модели обучения.

После анализа данных я использовала некоторые модели машинного обучения для предсказания данных. Как и ожидалось модели построенные из композиции случайных деревьев показывают лучшее качество, чем модель решающего дерева.

**Список литературы**

[1] <https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset>

[2] <https://loginom.ru/blog/decision-tree-p1>

[3] [https://dyakonov.org](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/)

[4] <https://habr.com/ru/company/ods/blog/324402/>

[5] <http://www.machinelearning.ru/wiki/images/c/cc/PZAD2016_09_rf.pdf>