«Визуализация данных»

В области машинного обучения визуализация данных - это не просто создание причудливой графики для отчетов; она широко используется в повседневной работе на всех этапах проекта.

Начнем с того, что визуальное изучение данных - это первое, что обычно делается при решении новой задачи. Мы проводим предварительные проверки и анализ с использованием графиков и таблиц, чтобы обобщить данные и опустить менее важные детали. Нам, людям, гораздо удобнее понимать основные моменты таким образом, чем путем чтения множества строк необработанных данных. Удивительно, сколько понимания можно получить из, казалось бы, простых диаграмм, созданных с помощью доступных инструментов визуализации.

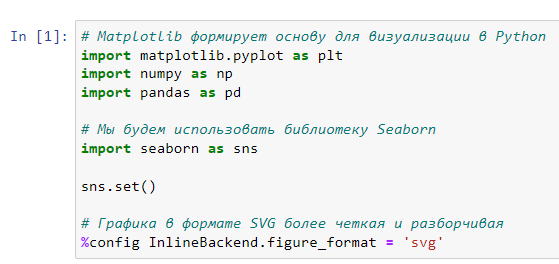
Далее, когда мы анализируем производительность модели или сообщаем результаты, мы также часто используем диаграммы и изображения. Иногда для интерпретации сложной модели нам необходимо спроецировать многомерные пространства на более визуально понятные 2D или 3D фигуры.

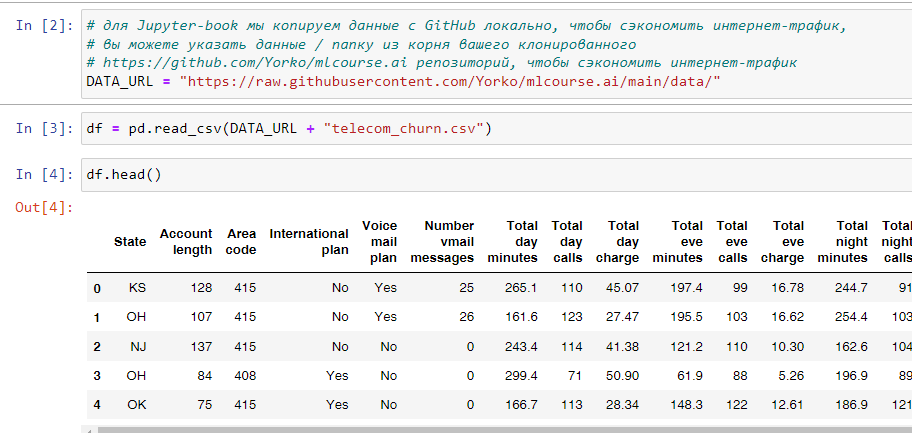
В целом, визуализация - это относительно быстрый способ узнать что-то новое о ваших данных. Таким образом, жизненно важно изучить ее наиболее полезные методы и сделать их частью вашего повседневного инструментария ML.

В этой работе мы собираемся получить практический опыт визуального исследования данных с использованием популярных библиотек, таких как pandas, matplotlib и seaborn.

**1. Набор данных**

Прежде чем мы перейдем к данным, давайте инициализируем нашу среду:





Последний столбец данных, Churn, является нашей целевой переменной. Он двоичный: True указывает, что компания в конечном итоге потеряла этого клиента, и False указывает, что клиент был сохранен. Позже мы построим модели, которые предсказывают эту функцию на основе остальных функций. Вот почему мы называем это target.

**2. Одномерная визуализация**

Одномерный анализ рассматривает по одному объекту за раз. Когда мы анализируем объект независимо, нас обычно в основном интересует распределение его значений и игнорируем другие объекты в наборе данных.

Ниже мы рассмотрим различные статистические типы объектов и соответствующие инструменты для их индивидуального визуального анализа.

**2.1 Количественные характеристики**

Количественные характеристики принимают упорядоченные числовые значения. Эти значения могут быть дискретными, как целые числа, или непрерывными, как вещественные числа, и обычно выражают количество или измерение.

Гистограммы и графики плотности.

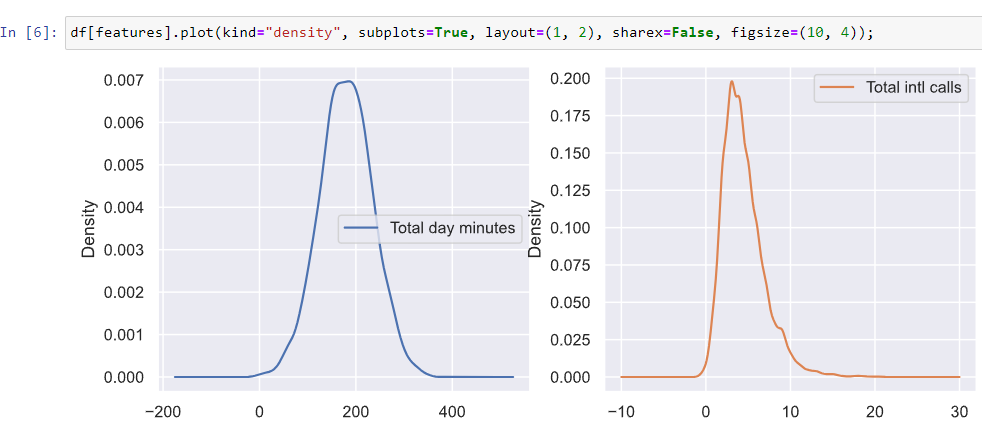
Самый простой способ взглянуть на распределение числовой переменной, чтобы построить его гистограмма использование DataFrame метод hist().



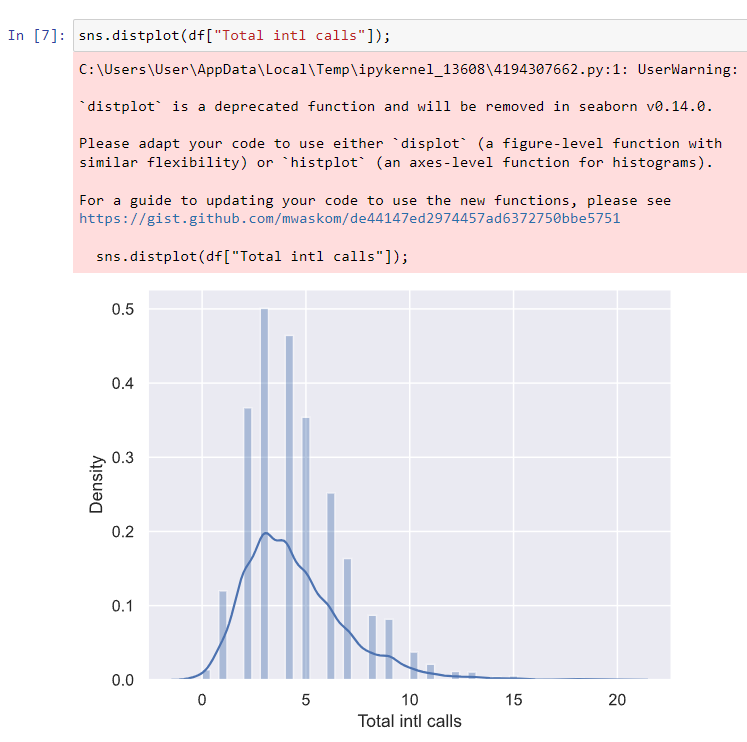
Гистограмма группирует значения в ячейки равного диапазона значений. Форма гистограммы может содержать подсказки о базовом типе распределения: гауссовском, экспоненциальном и т.д. Вы также можете обнаружить любую асимметрию в его форме, когда распределение почти регулярное, но имеет некоторые аномалии. Знание распределения значений функций становится важным, когда вы используете методы машинного обучения, которые предполагают определенный тип (чаще всего гауссовский).

На приведенном выше графике мы видим, что переменная Общее количество минут в день распределена нормально, в то время как Общее количество вызовов intl заметно смещено.

Существует также другой, часто более понятный способ понимания распределения: графики плотности или, более формально, Графики плотности ядра. Их можно рассматривать как сглаженную версию гистограммы. Их главное преимущество перед последними в том, что они не зависят от размера ячеек. Давайте создадим графики плотности для тех же двух переменных:



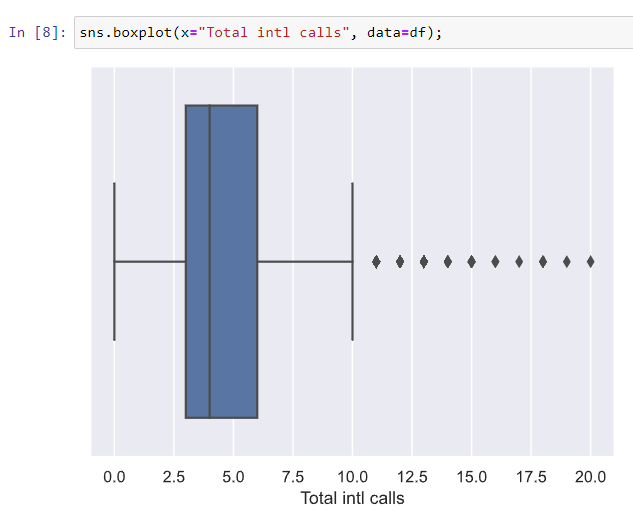
Также возможно построить распределение наблюдений с помощью seaborn’s distplot(). Например, давайте посмотрим на распределение Общего количества минут в день. По умолчанию на графике отображается гистограмма с оценкой плотности ядра (KDE) сверху.



Высота столбцов гистограммы здесь нормирована и показывает плотность, а не количество примеров в каждой ячейке.

**Прямоугольная диаграмма**

Другой полезный тип визуализации - это прямоугольная диаграмма box plot*.*seaborn здесь отлично справляется:



Давайте посмотрим, как интерпретировать прямоугольную диаграмму. Ее компонентами являются прямоугольная диаграмма, так называемые усы и ряд отдельных точек (выбросы).

Вставка сама по себе иллюстрирует межквартильный разброс распределения; его длина определяется 25th(Q1) и 75th(Q3) процентили. Вертикальная линия внутри прямоугольника обозначает медиану (50%) распределения.

Усы - это линии, отходящие от прямоугольника. Они представляют весь разброс точек данных, в частности, точек, попадающих в интервал (Q1-1.5\*IQR? Q3+1.5\*IQR), где IQR=Q3-Q1 это межквартильный диапазон.

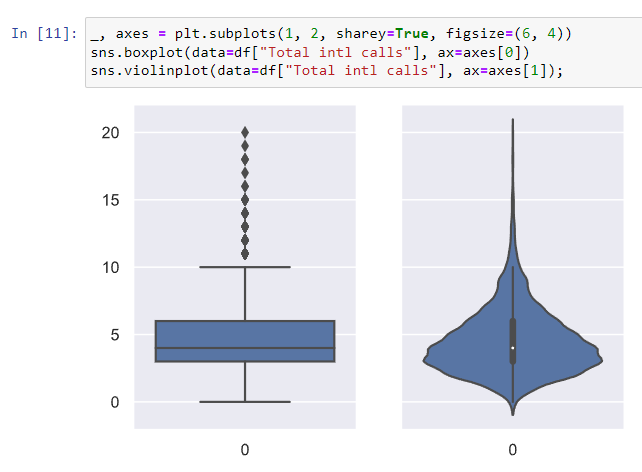
Выбросы, выходящие за пределы диапазона, ограниченного усами, отображаются по отдельности в виде черных точек вдоль центральной оси.

Мы можем видеть, что большое количество международных звонков довольно редко встречается в наших данных.

**График скрипки (Violin plot)**

Последний тип распределения участков, которые мы будем рассматривать это скрипка участок.

Посмотреть на рисунках ниже. Слева мы видим уже знакомая прямоугольная диаграмма (box plot.). Направо, там скрипка участок (Violin plot) с оценкой плотности ядра с обеих сторон.

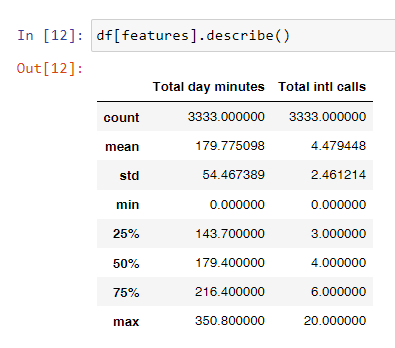


Разница между графиками box и violin заключается в том, что первый иллюстрирует определенную статистику, касающуюся отдельных примеров в наборе данных, в то время как график violin больше концентрируется на сглаженном распределении в целом.

В нашем случае график violin не предоставляет никакой дополнительной информации о данных, поскольку все ясно только из графика box.

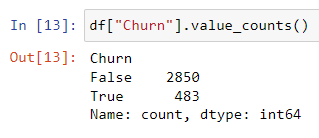
**describe()**

В дополнение к графическим инструментам, чтобы получить точную числовую статистику распределения, мы можем использовать метод describe() DataFrame:



**Таблица частот**

Давайте проверим баланс классов в нашем наборе данных, посмотрев на распределение целевой переменной: коэффициент оттока. Сначала мы получим таблицу частот, которая показывает, насколько частым является каждое значение категориальной переменной. Для этого мы будем использовать value\_counts() метод:



По умолчанию записи в выходных данных сортируются от наиболее часто встречающихся значений к наименее.

В нашем случае данные не сбалансированы; то есть два наших целевых класса, лояльные и нелояльные клиенты, неравномерно представлены в наборе данных. Лишь небольшая часть клиентов отменила подписку на телекоммуникационную услугу.

**Гистограмма**

Гистограмма представляет собой графическое представление таблицы частот. Самый простой способ ее создания - использовать seaborn функцию countplot() ’s'. В seaborn есть еще одна функция, которая называется несколько запутанно barplot() и в основном используется для представления некоторой базовой статистики числовой переменной, сгруппированной по категориальному признаку.

Давайте построим графики распределений для двух категориальных переменных:



Хотя гистограммы, рассмотренные выше, и столбчатые графики могут выглядеть похожими, между ними есть несколько различий:

1. Гистограммы лучше всего подходят для просмотра распределения числовых переменных, в то время как столбчатые графики используются для категориальных объектов.

2. Значения по оси X на гистограмме являются числовыми; гистограмма может иметь любой тип значений по оси X: числа, строки, логические значения.

3. Ось X гистограммы является декартовой координатной осью, вдоль которой значения не могут быть изменены; порядок столбцов не предопределен. Тем не менее, полезно отметить, что столбцы часто сортируются по высоте, то есть по частоте значений. Кроме того, когда мы рассматриваем порядковые переменные (например, обращения в службу поддержки в наших данных), столбцы обычно упорядочены по значению переменной.

Левая диаграмма выше наглядно иллюстрирует дисбаланс в нашей целевой переменной. Столбчатый график для обращений в службу поддержки клиентов справа дает подсказку о том, что большинство клиентов решают свои проблемы максимум за 2-3 обращения. Но, поскольку мы хотим иметь возможность предсказывать класс меньшинства, нас может больше интересовать, как ведет себя меньше недовольных клиентов. Вполне может быть, что в конце этого столбчатого графика находится большая часть нашего оттока. Пока это всего лишь гипотезы, поэтому давайте перейдем к некоторым более интересным и мощным визуальным техникам.

**3. Многомерная визуализация**

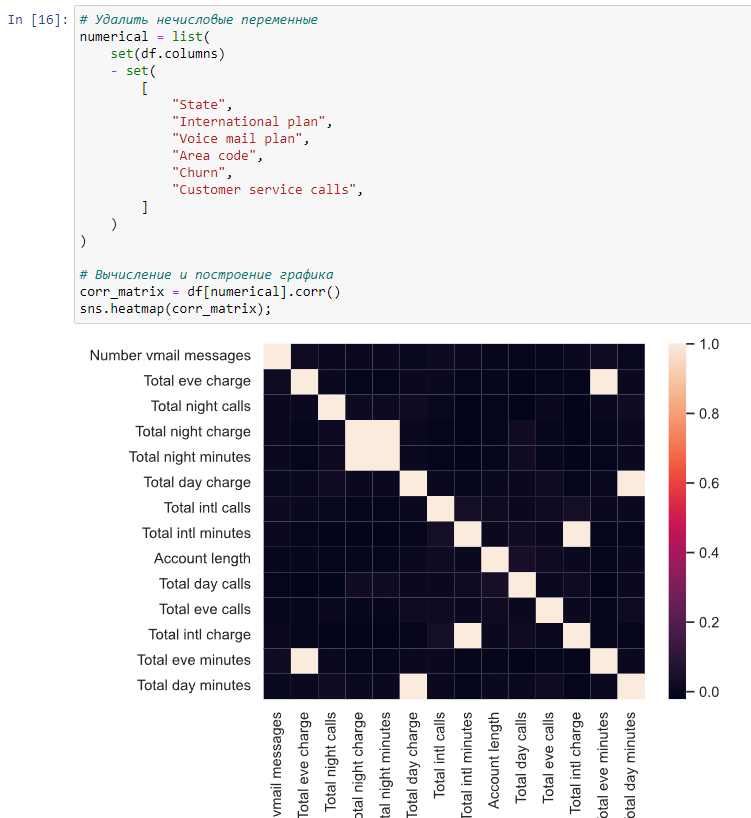
Многомерные графики позволяют нам видеть взаимосвязи между двумя и более различными переменными, все на одном рисунке. Как и в случае одномерных графиков, конкретный тип визуализации будет зависеть от типов анализируемых переменных.

**3.1 Количественный против количественного**

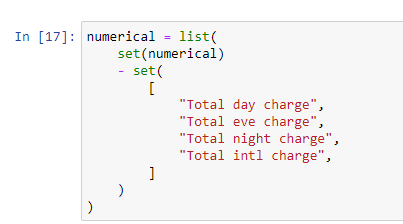
Корреляционная матрица

Давайте посмотрим на корреляции между числовыми переменными в нашем наборе данных. Эту информацию важно знать, поскольку существуют алгоритмы машинного обучения (например, линейная и логистическая регрессия), которые плохо обрабатывают сильно коррелированные входные переменные.

Сначала мы будем использовать метод corr() на DataFrame, который вычисляет корреляцию между каждой парой объектов. Затем мы передаем полученную корреляционную матрицу в heatmap() from seaborn, которая отображает цветовую матрицу для предоставленных значений:



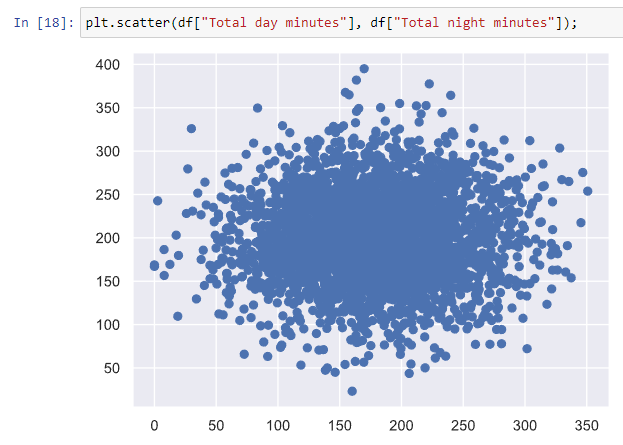
Из цветной корреляционной матрицы, сгенерированной выше, мы можем видеть, что существуют 4 переменные, такие как Общая дневная плата которые были рассчитаны непосредственно из количества минут, потраченных на телефонные звонки (Общее количество минут в день). Они называются зависимыми переменными и поэтому могут быть опущены, поскольку не вносят никакой дополнительной информации. Давайте избавимся от них:



**Точечный график**

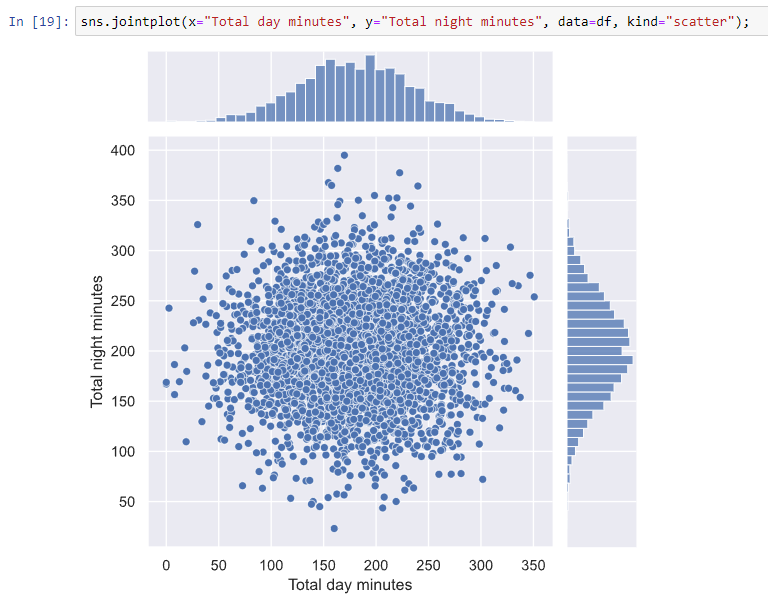
Точечный график отображает значения двух числовых переменных в виде Декартовых координат в 2D пространстве. Точечные графики в 3D также возможны.

Давайте попробуем функция scatter() от matplotlib библиотека:



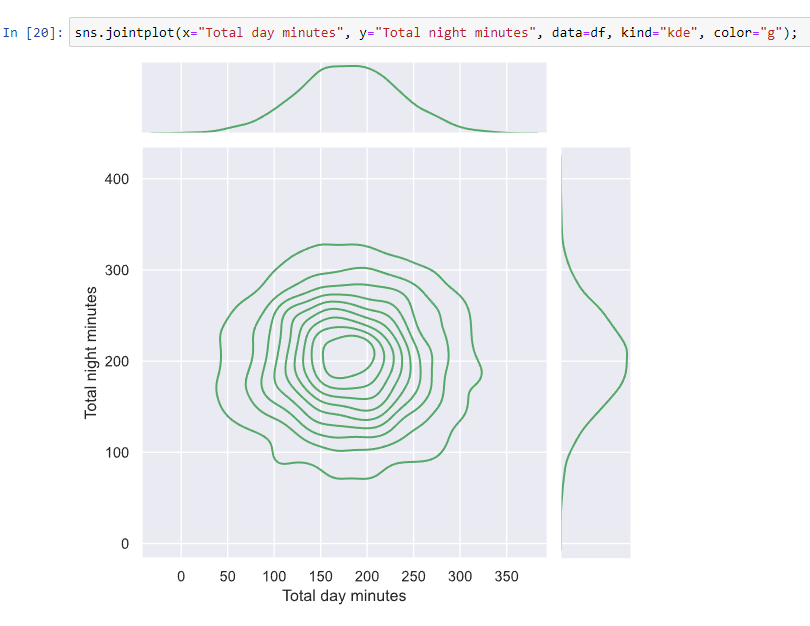
Мы получаем неинтересную картину двух нормально распределенных переменных. Кроме того, кажется, что эти объекты некоррелированы, потому что форма, подобная эллипсу, выровнена с осями.

Есть вариант немного красивее, чтобы создать график рассеяния с seaborn библиотека:



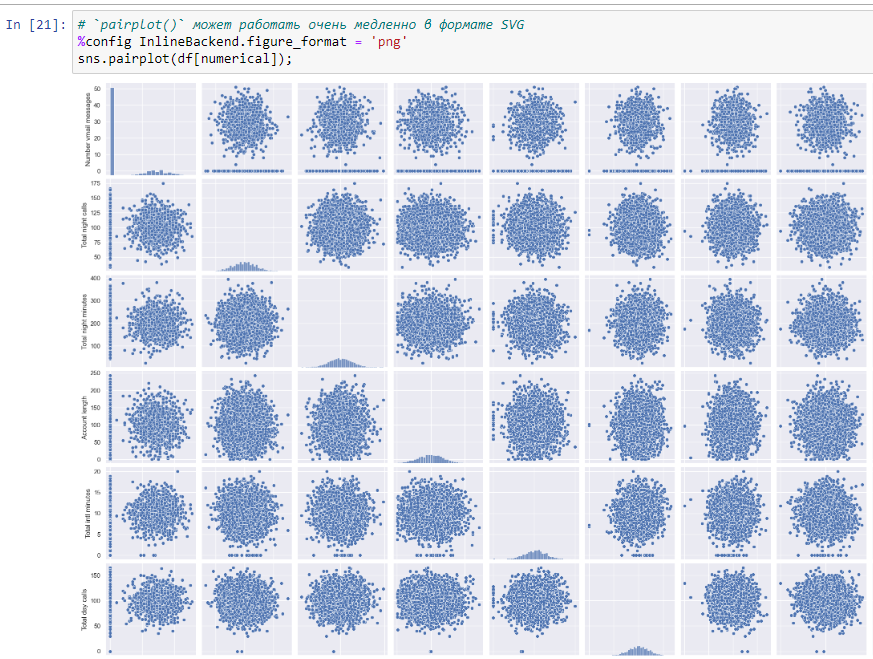
Функция jointplot() выводит две гистограммы, которые могут быть полезны в некоторых случаях.

Используя ту же функцию, мы также можем получить сглаженную версию нашего двумерного распределения:



**Матрица диаграммы рассеяния**

В некоторых случаях мы можем захотеть построить диаграмму рассеяния матрицы такую, как показано ниже. Ее диагональ содержит распределения соответствующих переменных, а точечные диаграммы для каждой пары переменных заполняют остальную часть матрицы.



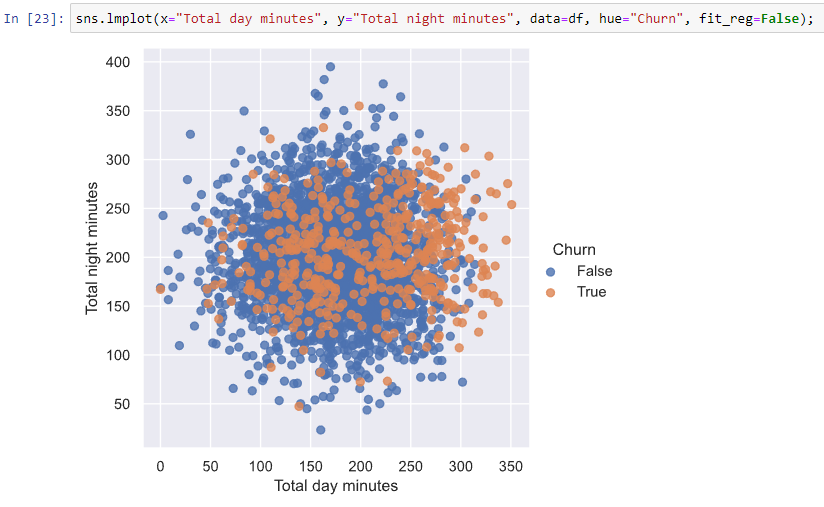
Иногда такая визуализация может помочь сделать выводы о данных; но в данном случае все довольно ясно и без сюрпризов.

**3.2 Количественный против Категориального**

В этом разделе мы сделаем наши простые количественные графики немного более захватывающими. Мы попытаемся получить новое представление о прогнозировании оттока из взаимодействий между числовыми и категориальными характеристиками.

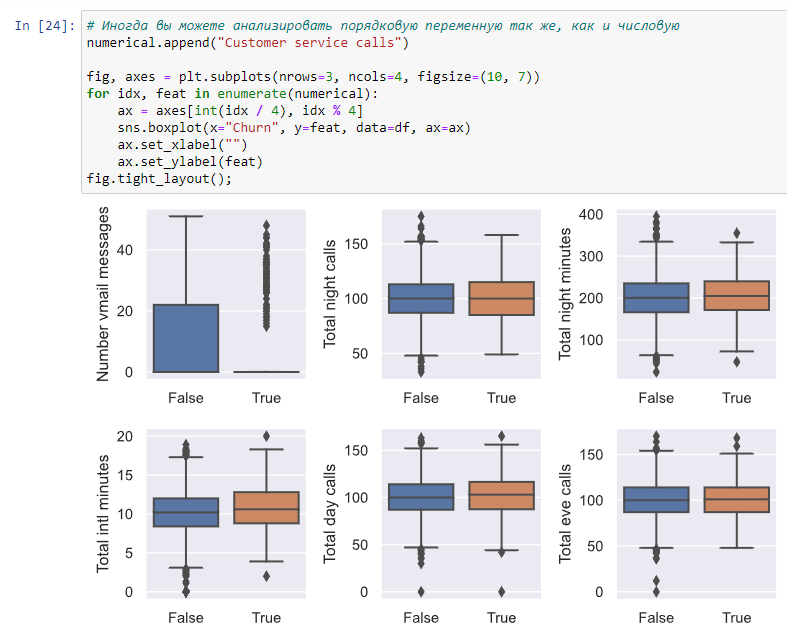
Более конкретно, давайте посмотрим, как входные переменные связаны с целевой переменной Churn.

Ранее вы узнали о точечных диаграммах. Кроме того, их точки могут быть закодированы по цвету или размеру, чтобы значения третьей категориальной переменной также были представлены на том же рисунке. Мы можем достичь этого с помощью scatter() функция, рассмотренная выше, но давайте попробуем новую функцию с именем lmplot() и используем параметр hue для обозначения интересующей нас категориальной характеристики:



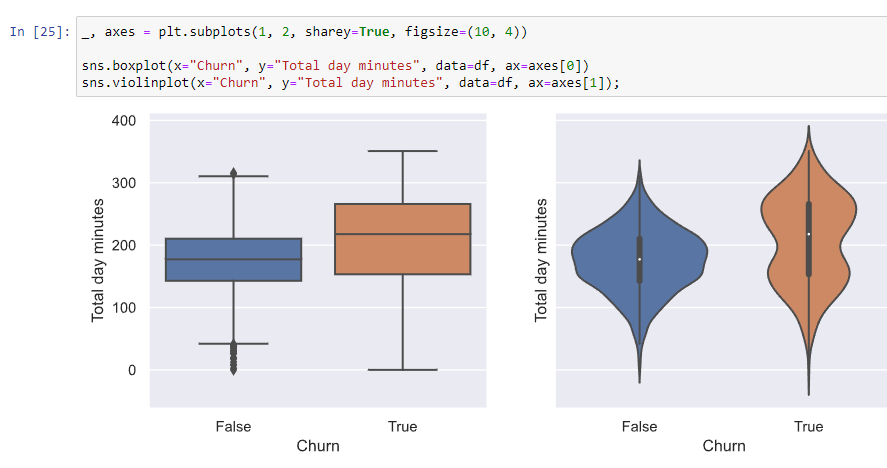
Похоже, что наша небольшая доля нелояльных клиентов склоняется к верхнему правому углу; то есть такие клиенты, как правило, проводят больше времени на телефоне как днем, так и ночью. Но это не совсем понятно, и мы не будем делать никаких окончательных выводов из этой диаграммы.

Теперь давайте создадим прямоугольные графики для визуализации статистики распределения числовых переменных по двум непересекающимся группам: постоянным клиентам (Churn=False) и тем, кто ушел (Churn=True).



Из этой диаграммы мы можем видеть, что наибольшее расхождение в распределении между двумя группами наблюдается по трем переменным: Общее количество минут в день, Звонки в службу поддержки и Количество сообщений электронной почты. Позже в этом курсе мы узнаем, как определить важность объекта при классификации, используя Случайный лес или Повышение градиента; здесь мы увидим, что первые две функции действительно очень важны для прогнозирования оттока.

Давайте рассмотрим распределение минут в день, заданное отдельно для лояльных и нелояльных клиентов. Мы создадим графики box и violin для Общего количества минут в день сгруппированных по целевой переменной.



В этом случае график violin не предоставляет никакой дополнительной информации о наших данных, поскольку все ясно только из графика box: нелояльные клиенты, как правило, больше разговаривают по телефону.

Интересное наблюдение: в среднем клиенты, которые расторгают свои контракты, являются более активными пользователями услуг связи. Возможно, они недовольны тарифами, поэтому возможной мерой предотвращения оттока могло бы стать снижение тарифов на звонки. Компании необходимо будет провести дополнительный экономический анализ, чтобы выяснить, будут ли такие меры выгодными.

Когда мы хотим проанализировать количественную переменную сразу в двух категориальных измерениях, для этого есть подходящая функция в seaborn библиотеке под названием catplot(). Например, давайте визуализируем взаимодействие между Общим количеством минут в день и две категориальные переменные на одном графике:

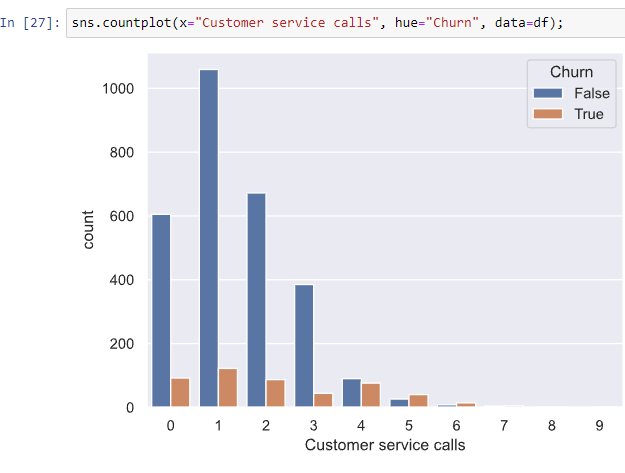


Исходя из этого, мы могли бы сделать вывод, что, начиная с 4 звонков, общее количество минут в день больше не может быть основным фактором оттока клиентов. Возможно, в дополнение к нашему предыдущему предположению о тарифах, есть клиенты, которые недовольны сервисом из-за других проблем, что может привести к уменьшению количества минут в день, потраченных на звонки.

**3.3 Категориальный против Категориальный**

Как мы видели ранее в этой статье, переменная вызовы службы поддержки имеет несколько уникальных значений и, таким образом, может считаться либо числовой, либо порядковой. Мы уже видели его распределение с помощью графика подсчета. Теперь нас интересует взаимосвязь между этим порядковым признаком и целевой переменной Churn.

Давайте посмотрим на распределение количества обращений в службу поддержки клиентов, снова используя график подсчета. На этот раз давайте также передадим параметр hue=Churn, который добавляет категориальное измерение к графику:



Наблюдение: уровень оттока значительно возрастает после 4 или более звонков в службу поддержки клиентов.

Теперь давайте рассмотрим взаимосвязь между оттоком и бинарными функциями, международным планом и планом голосовой почты.

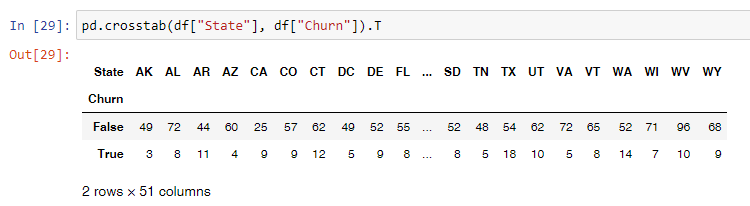


Наблюдение: когда включен международный план, уровень оттока намного выше; использование международного плана заказчиком является сильной стороной. Мы не наблюдаем того же эффекта с планом голосовой почты.

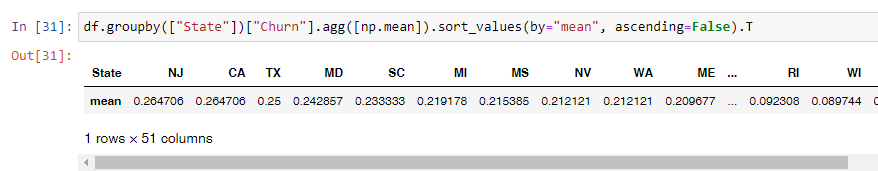
**Таблица непредвиденных обстоятельств**

В дополнение к использованию графических средств для категориального анализа существует традиционный инструмент статистики: таблица непредвиденных обстоятельств, также называемая перекрестной таблицей. В табличной форме показано многомерное частотное распределение категориальных переменных. В частности, это позволяет нам видеть распределение одной переменной в зависимости от другой, просматривая столбец или строку.

Давайте попробуем увидеть, как Churn связан с категориальной переменной State, создав перекрестную таблицу:



В случае Состояние число различных значений достаточно высока: 51. Мы видим, что есть только несколько точек данных для каждого отдельного государства – только 3 до 17 лет клиенты в каждом государстве отказались от оператора. Давайте на секунду проигнорируем это и рассчитаем коэффициент оттока для каждого состояния, отсортировав его от высокого к низкому:



На первый взгляд кажется, что уровень оттока в Нью-Джерси и Калифорнии превышает 25% и менее 6% на Гавайях и Аляске. Однако эти выводы основаны на слишком малом количестве примеров, и наше наблюдение может быть простым свойством нашего конкретного набора данных. Мы можем подтвердить это с помощью Мэтьюза и Крамера гипотезы о корреляции.

**4. Визуализация всего набора данных**

**4.1 Наивный подход**

Мы рассматривали различные аспекты нашего набора данных, угадывая интересные функции и выбирая небольшое их количество одновременно для визуализации. Мы имели дело только с двумя-тремя переменными одновременно и смогли легко наблюдать структуру и взаимосвязи в данных. Но что, если мы хотим отобразить все функции и при этом иметь возможность интерпретировать полученную визуализацию?

Мы могли бы использовать hist() или создать диаграмму рассеяния с помощью pairplot() для всего набора данных, чтобы просматривать все наши функции одновременно. Но, когда количество функций достаточно велико, этот вид визуального анализа быстро становится медленным и неэффективным. Кроме того, мы по-прежнему анализировали бы наши переменные попарно, а не все сразу.

**4.2 Уменьшение размерности**

Большинство реальных наборов данных имеют множество функций, иногда их много тысяч. Каждую из них можно рассматривать как измерение в пространстве точек данных. Следовательно, чаще всего мы имеем дело с многомерными наборами данных, где полная визуализация довольно сложна.

Чтобы посмотреть на набор данных в целом, нам нужно уменьшить количество измерений, используемых в визуализации, без потери большой информации о данных. Эта задача называется уменьшение размерности и является примером обучение без учителя проблема в том, что нам нужно извлекать новые низкоразмерные функции из самих данных без какого-либо контролируемого ввода.

Одним из хорошо известных методов уменьшения размерности является Анализ главных компонент (PCA), который мы изучим позже в этом курсе. Его ограничение заключается в том, что он является линейным алгоритм, который подразумевает определенные ограничения на данные.

Существует также множество нелинейных методов, которые в совокупности называются Обучение по многообразию. Одним из наиболее известных из них является t-SNE.

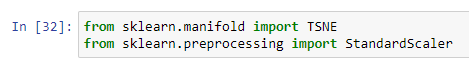
**4.3 t-SNE**

Давайте создадим t-SNE представление тех же данных об оттоке, которые мы использовали.

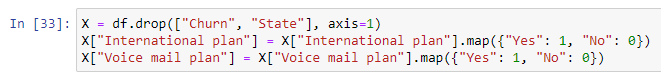
Название метода выглядит сложным и немного пугающим: t-распределенное стохастическое встраивание соседей. Его математика также впечатляет (мы не будем углубляться в это здесь, но, если вы чувствуете смелость, вот оригинальная статья Лоренса ван дер Маатена и Джеффри Хинтона из JMLR). Его основная идея проста: найдите проекцию для многомерного пространства объектов на плоскость (или трехмерную гиперплоскость, но это почти всегда 2D), такую, что те точки, которые были далеко друг от друга в исходном n-мерном пространстве, окажутся далеко друг от друга на плоскости. Те, которые изначально были близки, останутся близкими друг к другу.

По сути, встраивание по соседству это поиск нового и менее размерного представления данных, которое сохраняет соседство примеров.

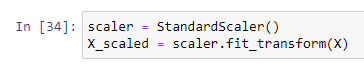
Теперь давайте немного потренируемся. Сначала нам нужно импортировать некоторые дополнительные классы:



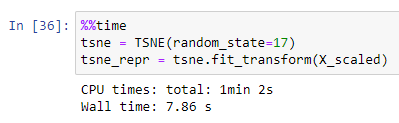
Мы опустим Состояние и Отток объектов и преобразуем значения “Да” / ”Нет" двоичных объектов в числовые значения с помощью pandas.Series.map():



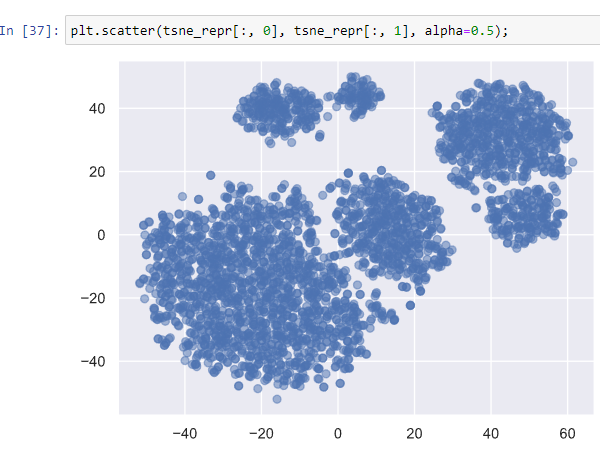
Нам также необходимо нормализовать данные. Для этого мы вычтем среднее значение из каждой переменной и разделим его на ее стандартное отклонение. Все это можно сделать с помощью StandardScaler.



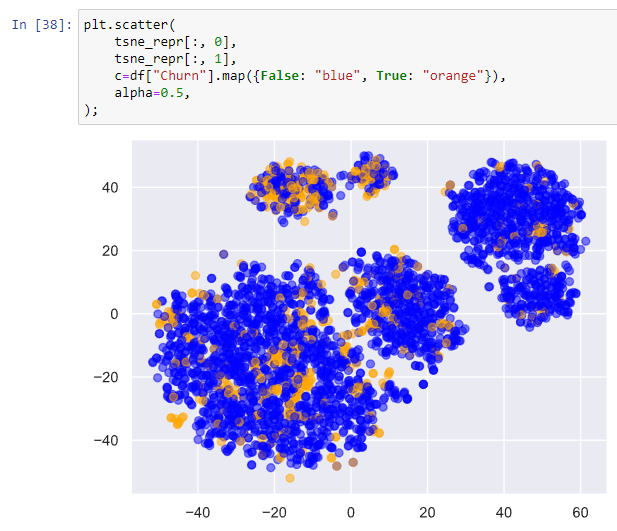
Теперь давайте построим представление t-SNE:



и изобразите это:

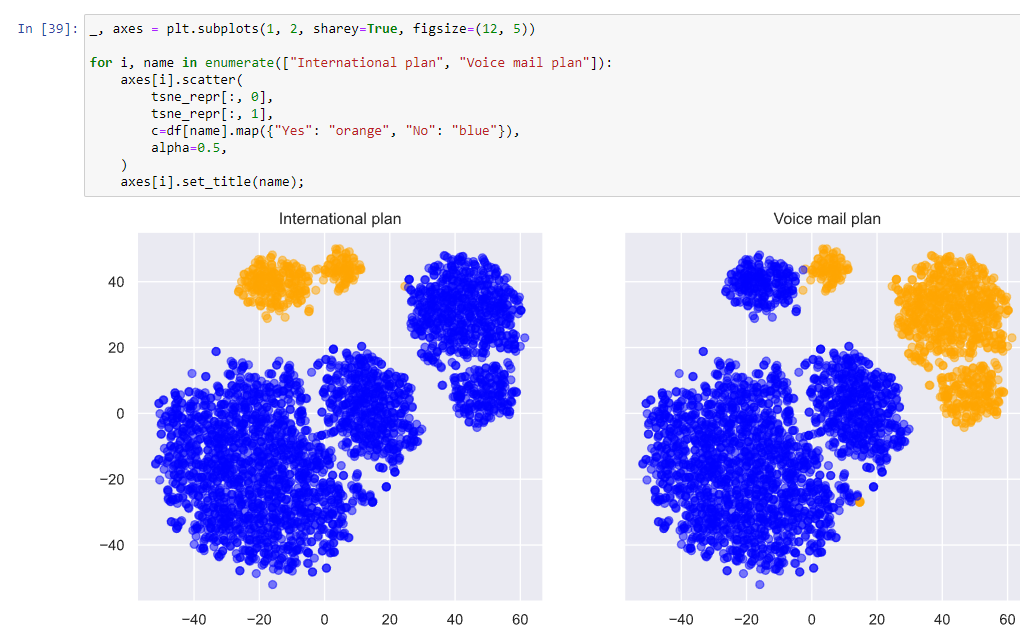


Давайте раскрасим это представление t-SNE в соответствии с оттоком (синий для постоянных клиентов и оранжевый для тех, кто отторгся).



Мы можем видеть, что клиенты, которых сбили с толку, сосредоточены в нескольких областях пространства объектов с более низкой размерностью.

Чтобы лучше понять картинку, мы также можем раскрасить ее оставшимися двоичными функциями: Международным планом и голосовой почтой. Оранжевые точки здесь указывают экземпляры, которые являются положительными для соответствующего двоичного объекта.



Теперь ясно, что, например, многие недовольные клиенты, отменившие подписку, собраны в один кластер, представляющий людей с международным тарифным планом, но без голосовой почты.

Наконец, давайте отметим некоторые недостатки t-SNE:

- Высокая вычислительная сложность. Реализация в scikit-learn вряд ли будет осуществима в реальной задаче. Если у вас большое количество образцов, вам следует попробовать Multicore-TSNE вместо этого.

- График может сильно меняться в зависимости от случайного начального числа, что усложняет интерпретацию. Вот хороший учебник по t-SNE. В общем, вам не следует делать никаких далеко идущих выводов на основе таких графиков, потому что это может быть равносильно простому угадыванию. Конечно, некоторые находки в изображениях t-SNE могут вдохновить на идею и быть подтверждены более тщательными исследованиями в дальнейшем, но это случается не очень часто.