# Тема 2: Визуализация данных c Python

Занятие посвящено визуализации данных в Python. Сначала мы посмотрим на основные методы библиотек Seaborn и Plotly, затем проанализируем знакомый нам по теме 1 набор данных по оттоку клиентов телеком-оператора и подглядим в n-мерное пространство с помощью алгоритма t-SNE.

1. Демонстрация основных методов Seaborn и Plotly
2. Пример визуального анализа данных
3. Подглядывание в n-мерное пространство с t-SNE
4. Домашнее задание №2

## Демонстрация основных методов Seaborn и Plotly

В начале настроим окружение: импортируем все необходимые библиотеки и немного настроим дефолтное отображение картинок.

# отключим предупреждения Anaconda

import warnings

warnings.simplefilter('ignore')

# будем отображать графики прямо в jupyter'e

%matplotlib inline

import seaborn as sns

import matplotlib.pyplot as plt

#графики в svg выглядят более четкими

%config InlineBackend.figure\_format = 'svg'

#увеличим дефолтный размер графиков

from pylab import rcParams

rcParams['figure.figsize'] = 8, 5

import pandas as pd

Загрузим в DataFrame данные, с которыми будем работать. Для примеров выбраны данные о продажах и оценках видео-игр из Kaggle Datasets.

df = pd.read\_csv('video\_games\_sales.csv')

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 16719 entries, 0 to 16718

Data columns (total 16 columns):

Name 16717 non-null object

Platform 16719 non-null object

Year\_of\_Release 16450 non-null float64

Genre 16717 non-null object

Publisher 16665 non-null object

NA\_Sales 16719 non-null float64

EU\_Sales 16719 non-null float64

JP\_Sales 16719 non-null float64

Other\_Sales 16719 non-null float64

Global\_Sales 16719 non-null float64

Critic\_Score 8137 non-null float64

Critic\_Count 8137 non-null float64

User\_Score 10015 non-null object

User\_Count 7590 non-null float64

Developer 10096 non-null object

Rating 9950 non-null object

dtypes: float64(9), object(7)

memory usage: 2.0+ MB

Некоторые признаки, которые pandas считал, как object, явно приведем к типам float или int.

df['User\_Score'] = df.User\_Score.astype('float64')

df['Year\_of\_Release'] = df.Year\_of\_Release.astype('int64')

df['User\_Count'] = df.User\_Count.astype('int64')

df['Critic\_Count'] = df.Critic\_Count.astype('int64')

Но данные есть не для всех игр, поэтому давайте оставим только те записи, в которых нет пропусков, с помощью метода dropna. Иначе предыдущая команда будет давать ошибку.

df = df.dropna()

print(df.shape)

(6825, 16)

Повторим преобразование данных:

df['User\_Score'] = df.User\_Score.astype('float64')

df['Year\_of\_Release'] = df.Year\_of\_Release.astype('int64')

df['User\_Count'] = df.User\_Count.astype('int64')

df['Critic\_Count'] = df.Critic\_Count.astype('int64')

Всего в таблице 6825 объектов и 16 признаков для них. Посмотрим на несколько первых записей c помощью метода head, чтобы убедиться, что все правильно. Для удобства я оставил только те признаки, которые мы будем в дальнейшем использовать.

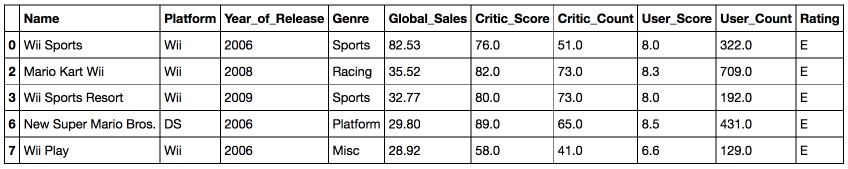
useful\_cols = ['Name', 'Platform', 'Year\_of\_Release', 'Genre',

'Global\_Sales', 'Critic\_Score', 'Critic\_Count',

'User\_Score', 'User\_Count', 'Rating'

]

df[useful\_cols].head()



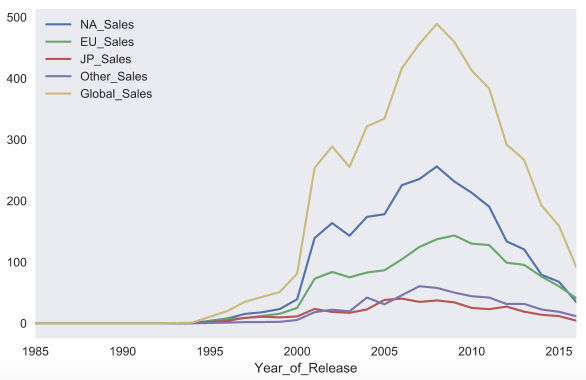
Прежде чем мы перейдем к рассмотрению методов библиотек seaborn и plotly, обсудим самый простой и зачастую удобный способ визуализировать данные из pandas DataFrame — это воспользоваться функцией plot.

Для примера построим график продаж видео игр в различных странах в зависимости от года. Для начала отфильтруем только нужные нам столбцы, затем посчитаем суммарные продажи по годам и у получившегося DataFrame вызовем функцию plot без параметров.

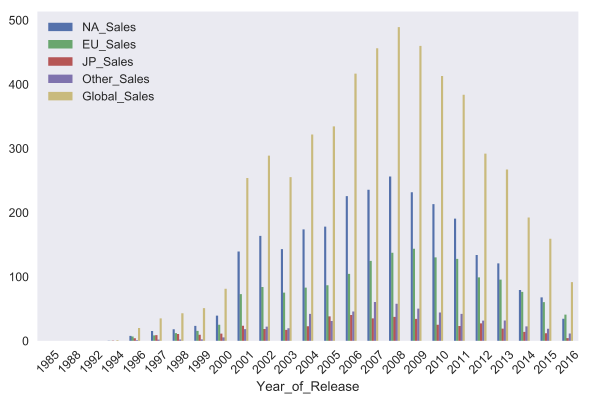
sales\_df = df[[x for x in df.columns if 'Sales' in x] + ['Year\_of\_Release']]

sales\_df.groupby('Year\_of\_Release').sum().plot()

Реализация функции plot в pandas основана на библиотеке matplotlib.



C помощью параметра kind можно изменить тип графика, например, на bar chart. Matplotlib позволяет очень гибко настраивать графики. На графике можно изменить почти все, что угодно, но потребуется порыться в документации и найти нужные параметры. Например, параметр rot отвечает за угол наклона подписей к оси *x*.

sales\_df.groupby('Year\_of\_Release').sum().plot(kind='bar', rot=45)

### Seaborn

Перейдем к библиотеке seaborn. Seaborn — это по сути более высокоуровневое API на базе библиотеки matplotlib. Seaborn содержит более адекватные дефолтные настройки оформления графиков. Также в библиотеке есть достаточно сложные типы визуализации, которые в matplotlib потребовали бы большого количество кода.

Познакомимся с первым таким "сложным" типом графиков pair plot (scatter plot matrix). Эта визуализация поможет нам посмотреть на одной картинке, как связаны между собой различные признаки.

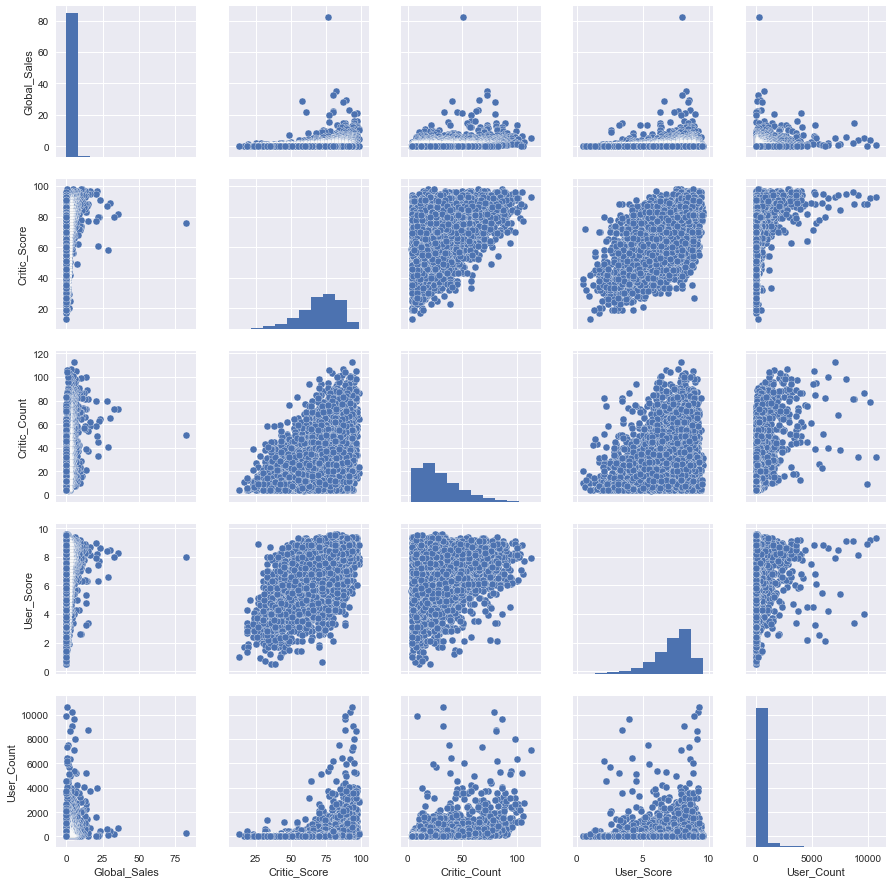
cols = ['Global\_Sales', 'Critic\_Score', 'Critic\_Count', 'User\_Score', 'User\_Count']

sns\_plot = sns.pairplot(df[cols])

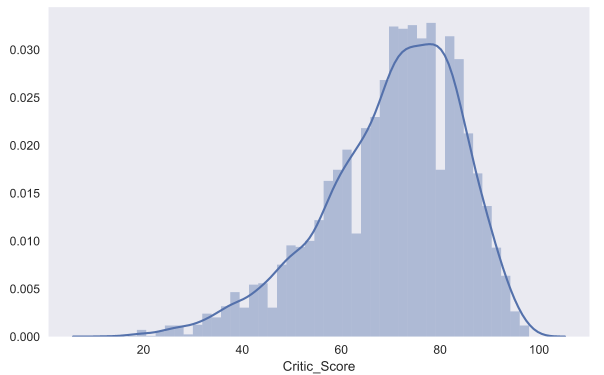
sns\_plot.savefig('pairplot.png')

Как можно видеть, на диагонали матрицы графиков расположены гистограммы распределений признака. Остальные же графики — это обычные scatter plots для соответствующих пар признаков.

Для сохранения графиков в файлы стоит использовать метод savefig.

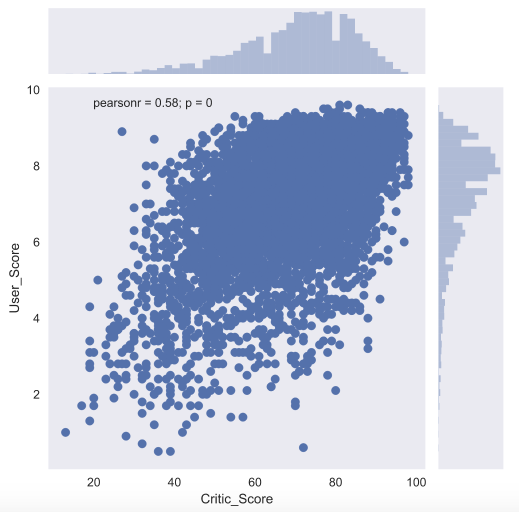
С помощью seaborn можно построить и распределение dist plot. Для примера посмотрим на распределение оценок критиков Critic\_Score. По умолчанию на графике отображается гистограмма и [kernel density estimation](https://en.wikipedia.org/wiki/Kernel_density_estimation)..

sns.distplot(df.Critic\_Score)



Для того, чтобы подробнее посмотреть на взаимосвязь двух численных признаков, есть еще и joint plot — это гибрид scatter plot и histogram. Посмотрим на то, как связаны между собой оценка критиков Critic\_Score и оценка пользователя User\_Score.

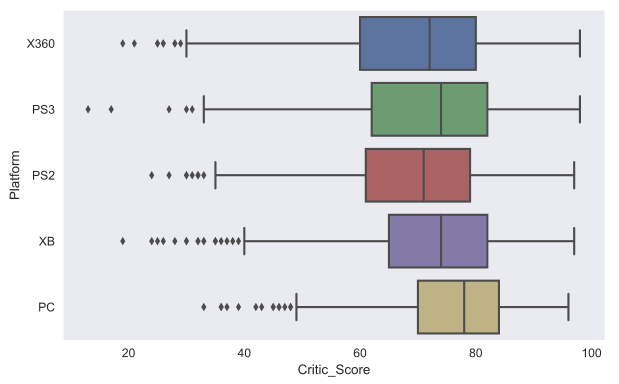
sns.jointplot(x=df.Critic\_Score, y=df.User\_Score)



Еще один полезный тип графиков — это box plot. Давайте сравним оценки игр от критиков для топ-5 крупнейших игровых платформ.

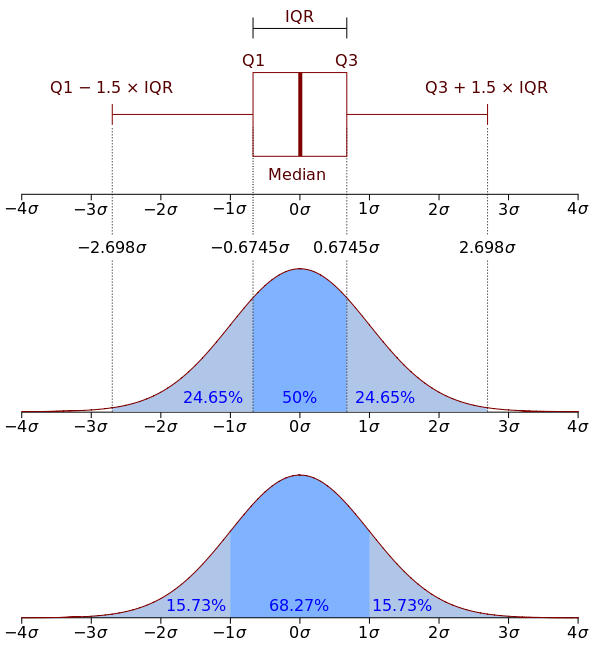
top\_platforms = df.Platform.value\_counts().sort\_values(ascending = False).head(5).index.values

sns.boxplot(y="Platform", x="Critic\_Score", data=df[df.Platform.isin(top\_platforms)], orient="h")

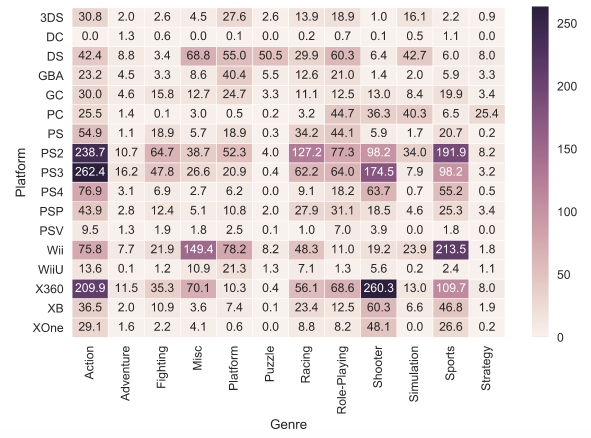


Думаю, стоит обсудить немного подробнее, как же понимать box plot. Box plot состоит из коробки (поэтому он и называется box plot), усиков и точек. Коробка показывает интерквартильный размах распределения, то есть соответственно 25% (Q1) и 75% (Q3) перцентили. Черта внутри коробки обозначает медиану распределения.

С коробкой разобрались, перейдем к усам. Усы отображают весь разброс точек кроме выбросов, то есть минимальные и максимальные значения, которые попадают в промежуток (Q1 - 1.5\*IQR, Q3 + 1.5\*IQR), где IQR = Q3 - Q1 — интерквартильный размах. Точками на графике обозначаются выбросы (outliers) — те значения, которые не вписываются в промежуток значений, заданный усами графика.

Для понимания лучше один раз увидеть, поэтому вот еще и картинка с [Wikipedia](https://en.wikipedia.org/wiki/Box_plot):

И еще один тип графиков — это heat map. Heat map позволяет посмотреть на распределение какого-то численного признака по двум категориальным. Визуализируем суммарные продажи игр по жанрам и игровым платформам.



**Plotly**

Мы рассмотрели визуализации на базе библиотеки matplotlib. Однако это не единственная опция для построения графиков на языке python. Познакомимся также с библиотекой plotly. Plotly — это open-source библиотека, которая позволяет строить интерактивные графики в jupyter.notebook'e без необходимости зарываться в javascript код.

Прелесть интерактивных графиков заключается в том, что можно посмотреть точное численное значение при наведении мыши, скрыть неинтересные ряды в визуализации, приблизить определенный участок графика и т.д.

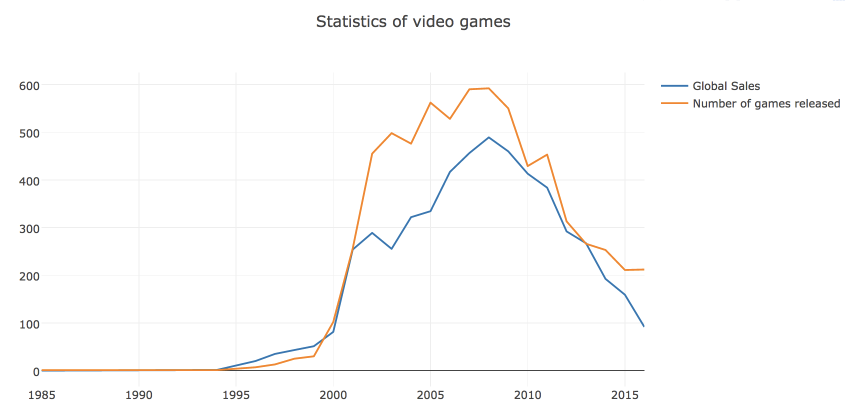
Перед началом работы импортируем все необходимые модули и инициализируем plotly с помощью команды init\_notebook\_mode.

from plotly.offline import download\_plotlyjs, init\_notebook\_mode, plot, iplot

import plotly

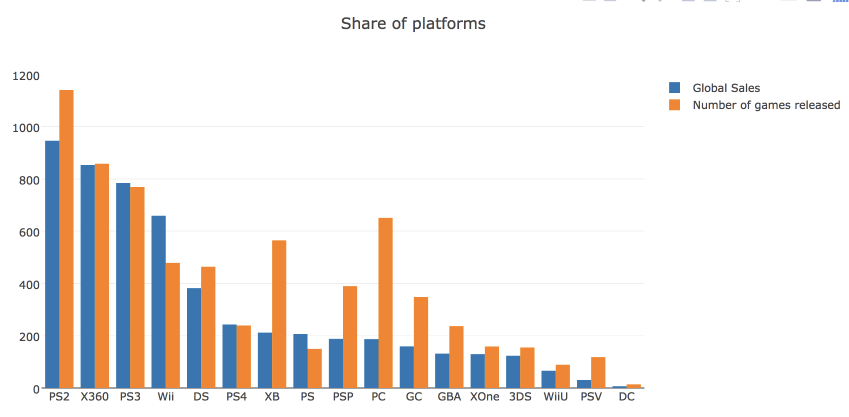
import plotly.graph\_objs as go

init\_notebook\_mode(connected=True)

Для начала построим line plot с динамикой числа вышедших игр и их продаж по годам.

Можно сразу сохранить график в виде html-файла.

plotly.offline.plot(fig, filename='years\_stats.html', show\_link=False)

Посмотрим также на рыночную долю игровых платформ, рассчитанную по количеству выпущенных игр и по суммарной выручке. Для этого построим bar chart.

В plotly можно построить и box plot. Рассмотрим распределения оценок критиков в зависимости от жанра игры.

# создаем Box trace для каждого жанра из наших данных

data = []

for genre in df.Genre.unique():

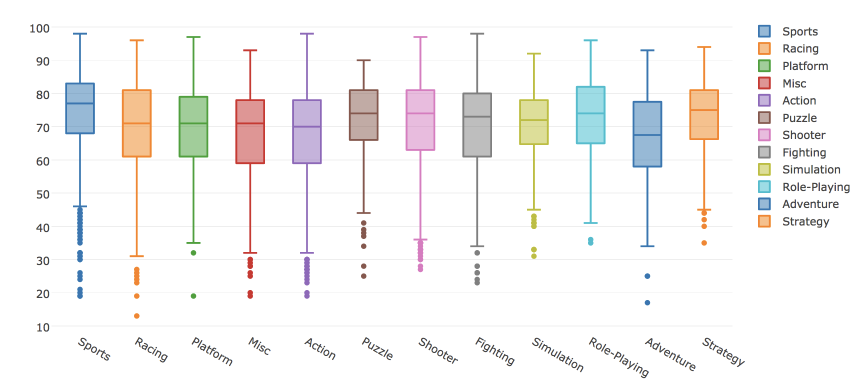
data.append(

go.Box(y=df[df.Genre==genre].Critic\_Score, name=genre)

)

# визуализируем данные

iplot(data, show\_link = False)



С помощью plotly можно построить и другие типы визуализаций. Графики получаются достаточно симпатичными с дефолтными настройками. Однако библиотека позволяет и гибко настраивать различные параметры визуализации: цвета, шрифты, подписи, аннотации и многое другое.

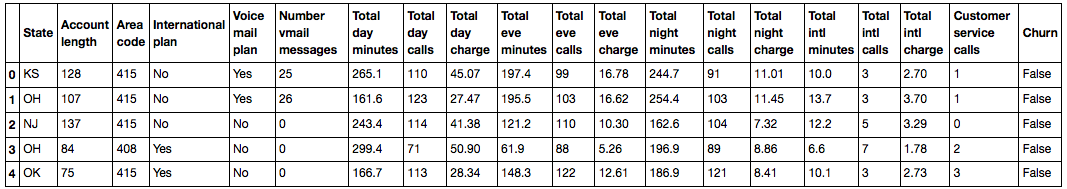
**Пример визуального анализа данных**

Считываем в DataFrame знакомые нам по первой теме данные по оттоку клиентов телеком-оператора.

df = pd.read\_csv('telecom\_churn.csv')

Проверим, все ли нормально считалось – посмотрим на первые 5 строк (метод head).

df.head()



Число строк (клиентов) и столбцов (признаков):

df.shape

(3333, 20)

Посмотрим на признаки и убедимся, что пропусков ни в одном из них нет – везде по 3333 записи.

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 3333 entries, 0 to 3332

Data columns (total 20 columns):

State 3333 non-null object

Account length 3333 non-null int64

Area code 3333 non-null int64

International plan 3333 non-null object

Voice mail plan 3333 non-null object

Number vmail messages 3333 non-null int64

Total day minutes 3333 non-null float64

Total day calls 3333 non-null int64

Total day charge 3333 non-null float64

Total eve minutes 3333 non-null float64

Total eve calls 3333 non-null int64

Total eve charge 3333 non-null float64

Total night minutes 3333 non-null float64

Total night calls 3333 non-null int64

Total night charge 3333 non-null float64

Total intl minutes 3333 non-null float64

Total intl calls 3333 non-null int64

Total intl charge 3333 non-null float64

Customer service calls 3333 non-null int64

Churn 3333 non-null bool

dtypes: bool(1), float64(8), int64(8), object(3)

memory usage: 498.1+ KB

**Описание признаков**

| **Название** | **Описание** | **Тип** |
| --- | --- | --- |
| **State** | Буквенный код штата | категориальный |
| **Account length** | Как долго клиент обслуживается компанией | количественный |
| **Area code** | Префикс номера телефона | количественный |
| **International plan** | Международный роуминг (подключен/не подключен) | бинарный |
| **Voice mail plan** | Голосовая почта (подключена/не подключена) | бинарный |
| **Number vmail messages** | Количество голосовых сообщений | количественный |
| **Total day minutes** | Общая длительность разговоров днем | количественный |
| **Total day calls** | Общее количество звонков днем | количественный |
| **Total day charge** | Общая сумма оплаты за услуги днем | количественный |
| **Total eve minutes** | Общая длительность разговоров вечером | количественный |
| **Total eve calls** | Общее количество звонков вечером | количественный |
| **Total eve charge** | Общая сумма оплаты за услуги вечером | количественный |
| **Total night minutes** | Общая длительность разговоров ночью | количественный |
| **Total night calls** | Общее количество звонков ночью | количественный |
| **Total night charge** | Общая сумма оплаты за услуги ночью | количественный |
| **Total intl minutes** | Общая длительность международных разговоров | количественный |
| **Total intl calls** | Общее количество международных разговоров | количественный |
| **Total intl charge** | Общая сумма оплаты за международные разговоры | количественный |
| **Customer service calls** | Число обращений в сервисный центр | количественный |

Посмотрим на распределение целевого класса – оттока клиентов.

df['Churn'].value\_counts()

False 2850

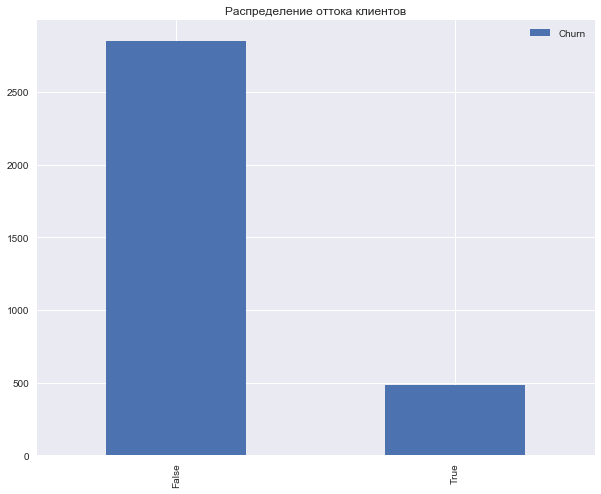
True 483

Name: Churn, dtype: int64

df['Churn'].value\_counts().plot(kind='bar', label='Churn')

plt.legend()

plt.title('Распределение оттока клиентов');



Выделим следующие группы признаков (среди всех кроме Churn ):

бинарные: International plan, Voice mail plan

категориальные: State

порядковые: Customer service calls

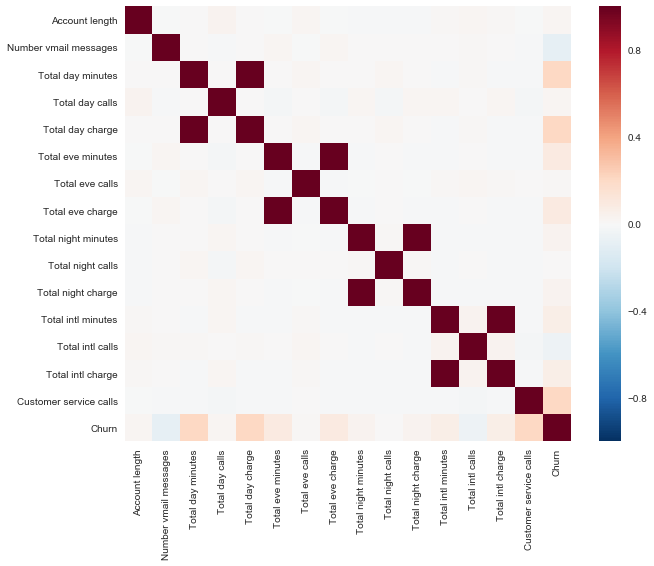
количественные: все остальные

Посмотрим на корреляции количественных признаков. По раскрашенной матрице корреляций видно, что такие признаки как Total day charge считаются по проговоренным минутам (Total day minutes). То есть 4 признака можно выкинуть, они не несут полезной информации.

corr\_matrix = df.drop(['State', 'International plan', 'Voice mail plan',

'Area code'], axis=1).corr()

sns.heatmap(corr\_matrix);

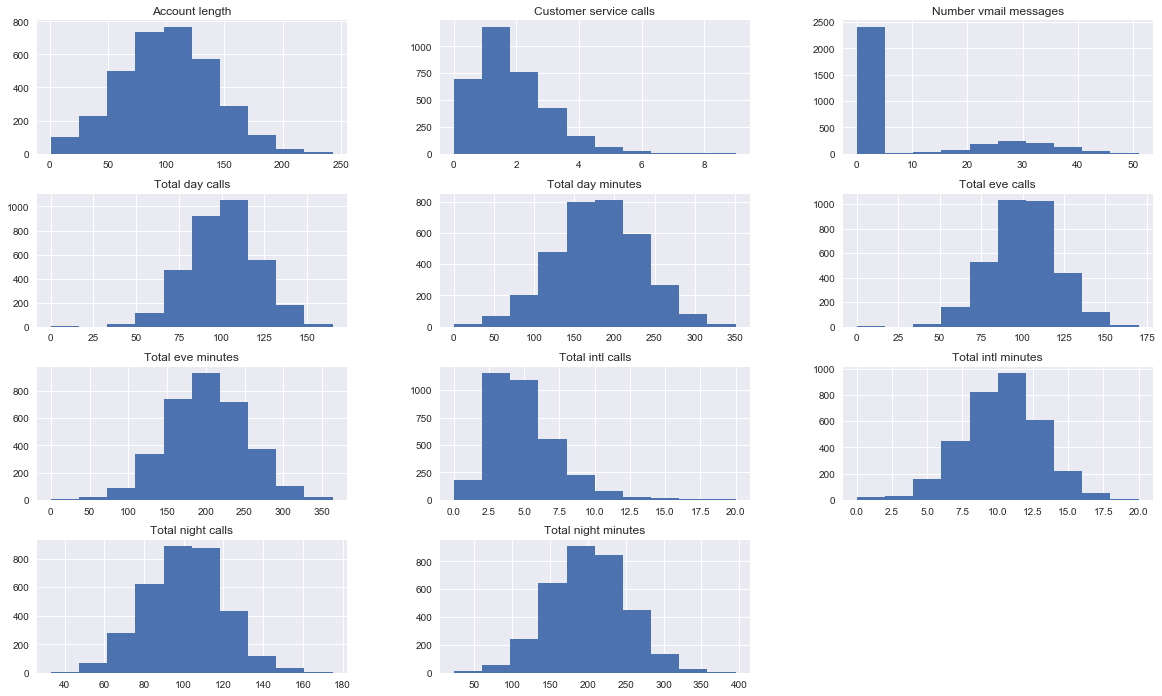


Теперь посмотрим на распределения всех интересующих нас количественных признаков. На бинарные/категориальные/порядковые признаки будем смотреть отдельно.

features = list(set(df.columns) - set(['State', 'International plan', 'Voice mail plan', 'Area code',

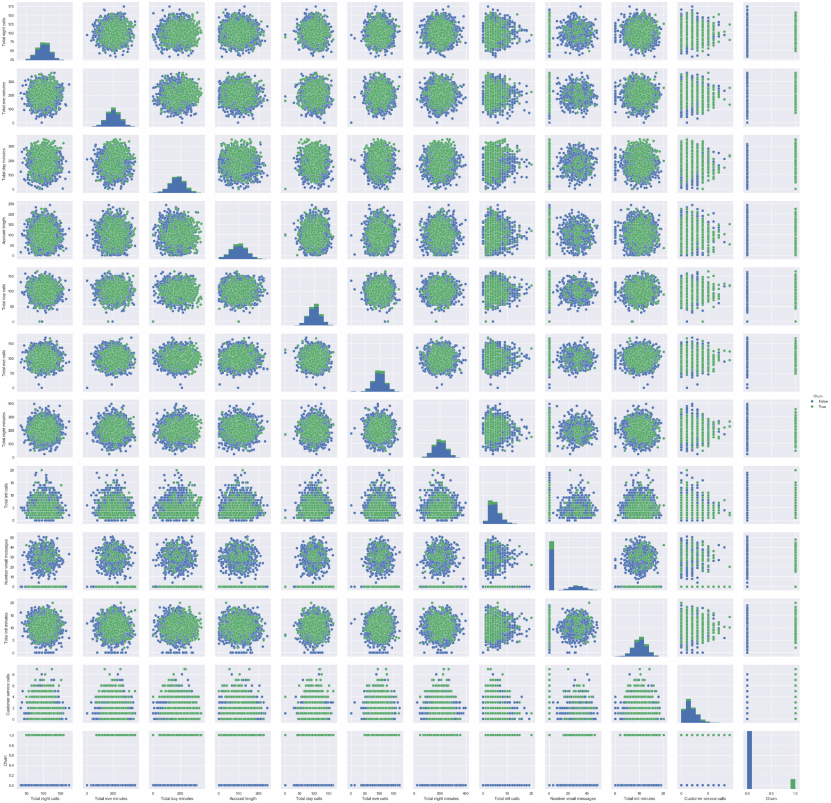
'Total day charge', 'Total eve charge', 'Total night charge',

'Total intl charge', 'Churn']))

df[features].hist(figsize=(20,12));

Видим, что большинство признаков распределены нормально. Исключения – число звонков в сервисный центр (Customer service calls) (тут больше подходит пуассоновское распределение) и число голосовых сообщений (Number vmail messages, пик в нуле, т.е. это те, у кого голосовая почта не подключена). Также смещено распределение числа международных звонков (Total intl calls).

Еще полезно строить вот такие картинки, где на главной диагонали рисуются распределения признаков, а вне главной диагонали – диаграммы рассеяния для пар признаков. Бывает, что это приводит к каким-то выводам, но в данном случае все примерно понятно, без сюрпризов.

sns.pairplot(df[features + ['Churn']], hue='Churn');

Дальше посмотрим, как признаки связаны с целевым – с оттоком.

Построим boxplot-ы, описывающее статистики распределения количественных признаков в двух группах: среди лояльных и ушедших клиентов.

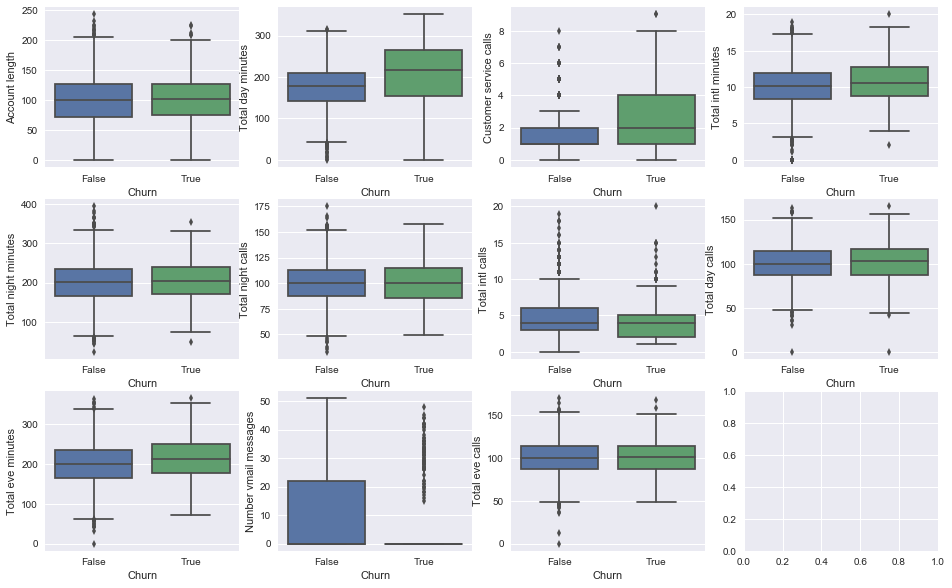
fig, axes = plt.subplots(nrows=3, ncols=4, figsize=(16, 10))

for idx, feat in enumerate(features):

sns.boxplot(x='Churn', y=feat, data=df, ax=axes[idx / 4, idx % 4])

axes[idx / 4, idx % 4].legend()

axes[idx / 4, idx % 4].set\_xlabel('Churn')

 axes[idx / 4, idx % 4].set\_ylabel(feat);

На глаз наибольшее отличие мы видим для признаков Total day minutes, Customer service calls и Number vmail messages. Впоследствии мы научимся определять важность признаков в задаче классификации с помощью случайного леса (или градиентного бустинга), и окажется, что первые два – действительно очень важные признаки для прогнозирования оттока.

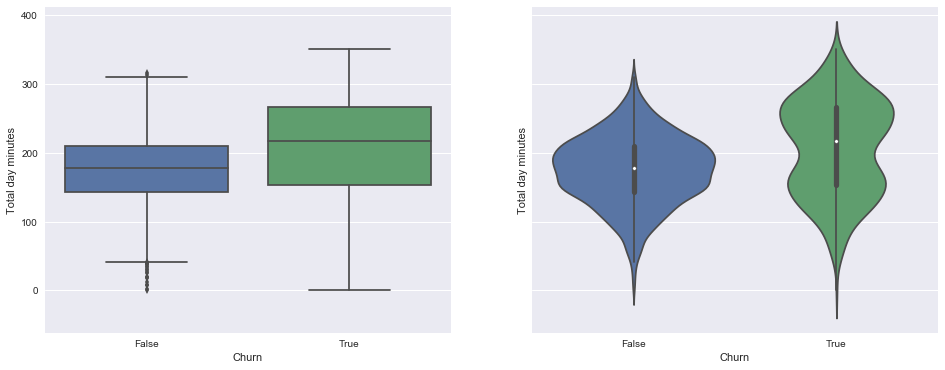
Посмотрим отдельно на картинки с распределением кол-ва проговоренных днем минут среди лояльных/ушедших. Слева — знакомые нам боксплоты, справа – сглаженные гистограммы распределения числового признака в двух группах (скорее просто красивая картинка, все и так понятно по боксплоту).

Интересное наблюдение: в среднем ушедшие клиенты больше пользуются связью. Возможно, они недовольны тарифами, и одной из мер борьбы с оттоком будет понижение тарифных ставок (стоимости мобильной связи). Но это уже компании надо будет проводить дополнительный экономический анализ, действительно ли такие меры будут оправданы.

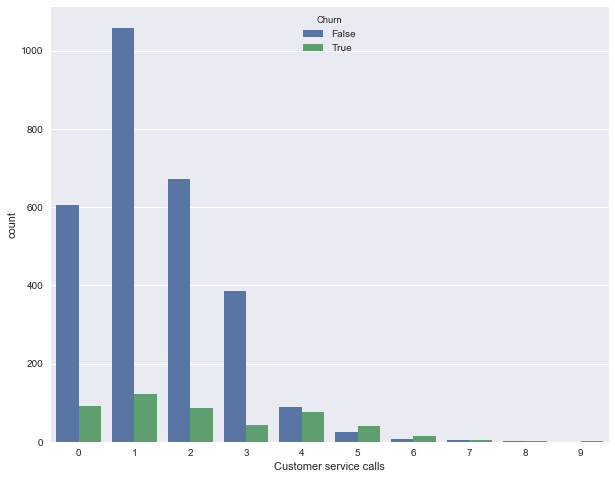
\_, axes = plt.subplots(1, 2, sharey=True, figsize=(16,6))

sns.boxplot(x='Churn', y='Total day minutes', data=df, ax=axes[0]);

sns.violinplot(x='Churn', y='Total day minutes', data=df, ax=axes[1]);



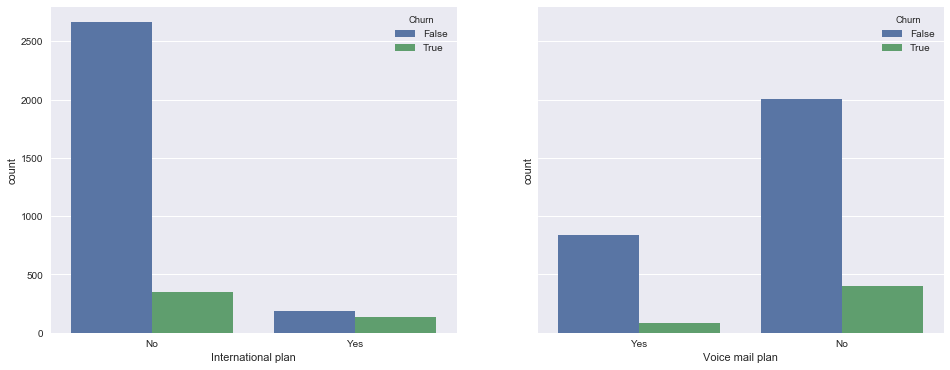
Теперь изобразим распределение числа обращений в сервисный центр (такую картинку мы строили в первой статье). Тут уникальных значений признака не много (признак можно считать, как количественным целочисленным, так и порядковым), и наглядней изобразить распределение с помощью countplot. Наблюдение: доля оттока сильно возрастает, начиная с 4 звонков в сервисный центр.

sns.countplot(x='Customer service calls', hue='Churn', data=df);

Теперь посмотрим на связь бинарных признаков International plan и Voice mail plan с оттоком. Наблюдение: когда роуминг подключен, доля оттока намного выше, т.е. наличие международного роуминга – сильный признак. Про голосовую почту такого нельзя сказать.

\_, axes = plt.subplots(1, 2, sharey=True, figsize=(16,6))

sns.countplot(x='International plan', hue='Churn', data=df, ax=axes[0]);

sns.countplot(x='Voice mail plan', hue='Churn', data=df, ax=axes[1]);

Наконец, посмотрим, как с оттоком связан категориальный признак *State*. С ним уже не так приятно работать, поскольку число уникальных штатов довольно велико – 51. Можно в начале построить сводную табличку или посчитать процент оттока для каждого штата. Но данных по каждом штату по отдельности маловато (ушедших клиентов всего от 3 до 17 в каждом штате), поэтому, возможно, признак *State* впоследствии не стоит добавлять в модели классификации из-за риска *переобучения* (но мы это будем проверять на *кросс-валидации*, stay tuned!).

Доли оттока для каждого штата:

df.groupby(['State'])['Churn'].agg([np.mean]).sort\_values(by='mean', ascending=False).T

https://habrastorage.org/files/e08/460/918/e0846091885244e9b124207a834e908f.png

https://habrastorage.org/r/w1560/files/3c4/d55/e55/3c4d55e55f9044f0836fc513598de442.png

Видно, что в Нью-Джерси и Калифорнии доля оттока выше 25%, а на Гавайях и в Аляске меньше 5%. Но эти выводы построены на слишком скромной статистике и возможно, это просто особенности имеющихся данных (тут можно и гипотезы попроверять про корреляции Мэтьюса и Крамера, но это уже за рамками данной статьи).

**Подглядывание в n-мерное пространство с t-SNE**

Построим t-SNE представление все тех же данных по оттоку. Название метода сложное – t-distributed Stohastic Neighbor Embedding, математика тоже крутая, но основная идея проста, как дверь: найдем такое отображение из многомерного признакового пространства на плоскость (или в 3D, но почти всегда выбирают 2D), чтоб точки, которые были далеко друг от друга, на плоскости тоже оказались удаленными, а близкие точки – также отобразились на близкие. То есть neighbor embedding – это своего рода поиск нового представления данных, при котором сохраняется соседство.

Немного деталей: выкинем штаты и признак оттока, бинарные Yes/No-признаки переведем в числа (pd.factorize). Также нужно масштабировать выборку – из каждого признака вычесть его среднее и поделить на стандартное отклонение, это делает StandardScaler.

from sklearn.manifold import TSNE

from sklearn.preprocessing import StandardScaler

# преобразуем все признаки в числовые, выкинув штаты

X = df.drop(['Churn', 'State'], axis=1)

X['International plan'] = pd.factorize(X['International plan'])[0]

X['Voice mail plan'] = pd.factorize(X['Voice mail plan'])[0]

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

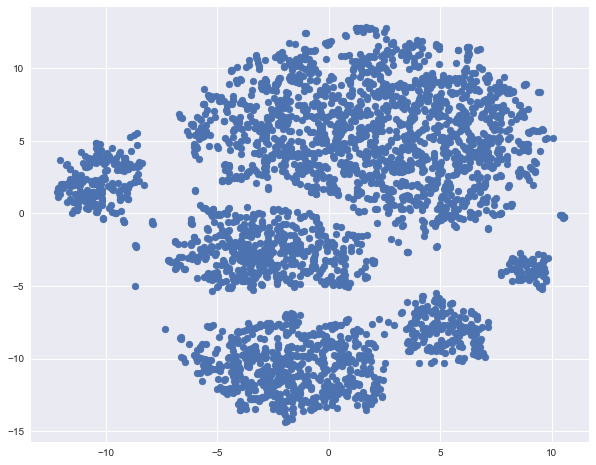
%%time

tsne = TSNE(random\_state=17)

tsne\_representation = tsne.fit\_transform(X\_scaled)

CPU times: user 20 s, sys: 2.41 s, total: 22.4 s

Wall time: 21.9 s

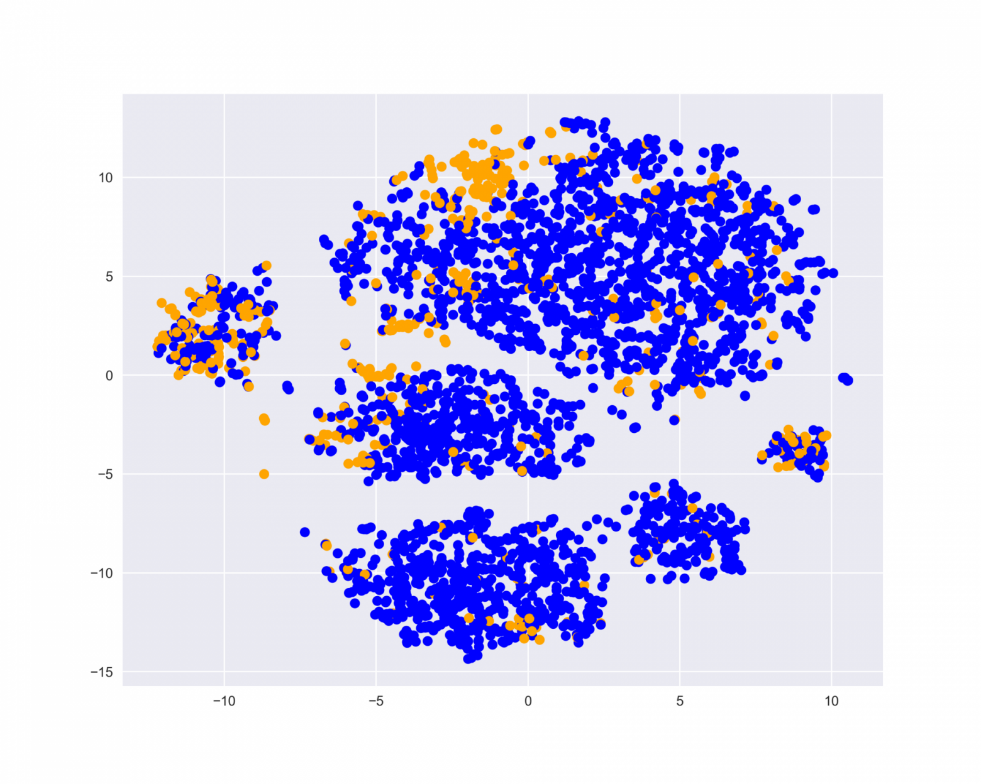
plt.scatter(tsne\_representation[:, 0], tsne\_representation[:, 1]);

Раскрасим полученное t-SNE представление данных по оттоку (синие – лояльные, оранжевые – ушедшие клиенты).

plt.scatter(tsne\_representation[:, 0], tsne\_representation[:, 1],

c=df['Churn'].map({0: 'blue', 1: 'orange'}));

Видим, что ушедшие клиенты преимущественно "кучкуются" в некоторых областях признакового пространства.

Чтоб лучше понять картинку, можно также раскрасить ее по остальным бинарным признакам – по роумингу и голосовой почте. Синие участки соответствуют объектам, обладающим этим бинарным признаком.

\_, axes = plt.subplots(1, 2, sharey=True, figsize=(16,6))

axes[0].scatter(tsne\_representation[:, 0], tsne\_representation[:, 1],

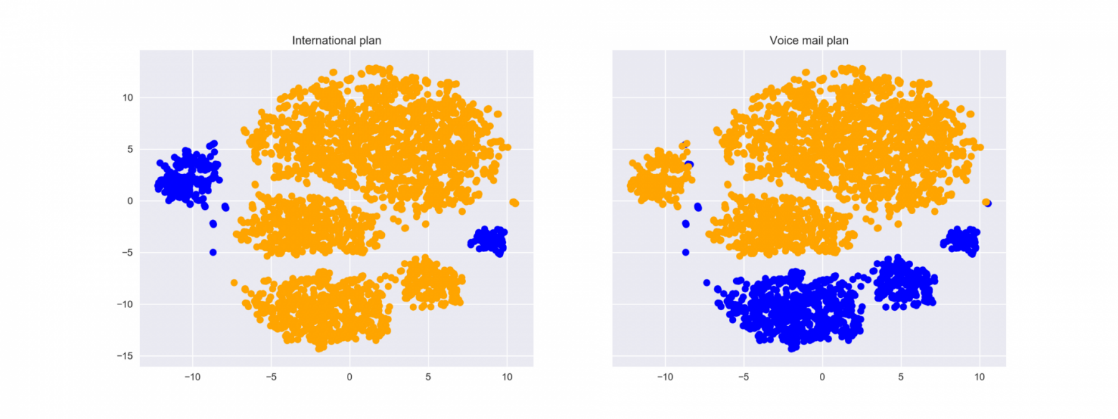
c=df['International plan'].map({'Yes': 'blue', 'No': 'orange'}));

axes[1].scatter(tsne\_representation[:, 0], tsne\_representation[:, 1],

c=df['Voice mail plan'].map({'Yes': 'blue', 'No': 'orange'}));

axes[0].set\_title('International plan');

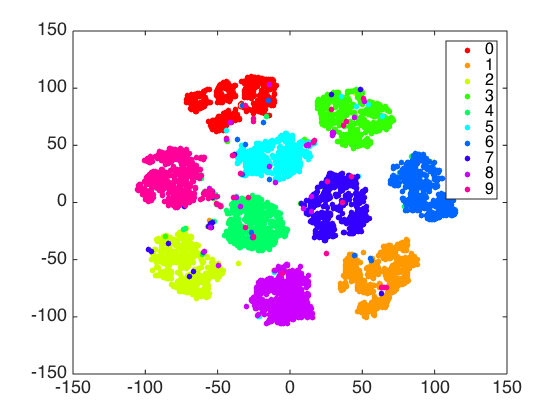
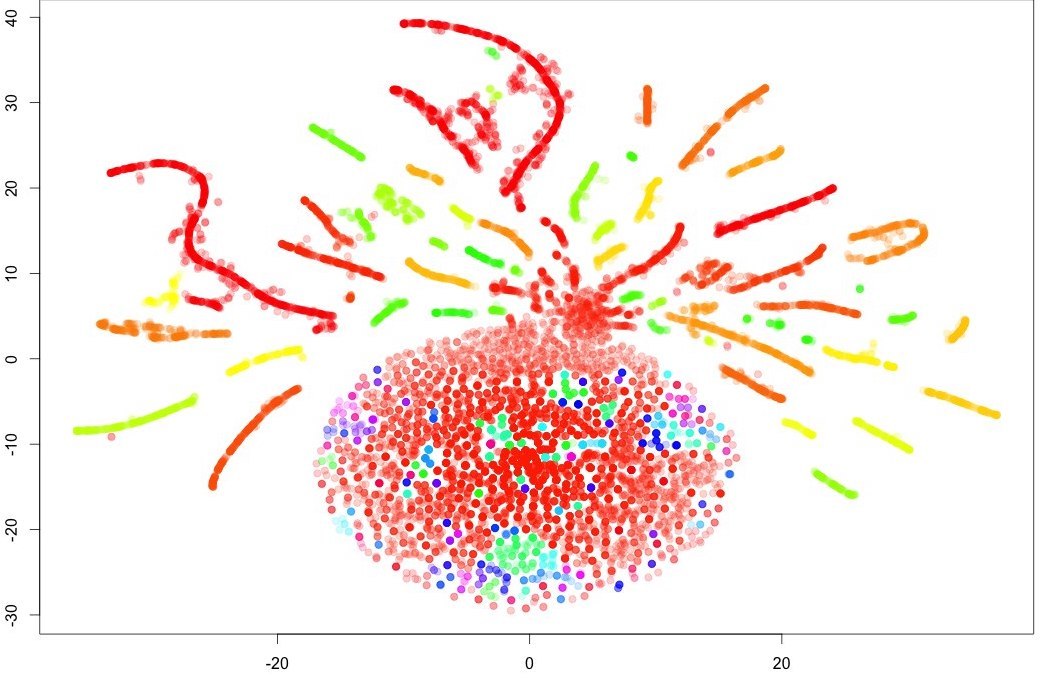
axes[1].set\_title('Voice mail plan');



Теперь понятно, что, например, много ушедших клиентов кучкуется в левом кластере людей с поключенным роумингом, но без голосовой почты.

Отметим минусы t-SNE:

* большая вычислительная сложность. Вот эта (http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html) реализация sklearn скорее всего не поможет в вашей реальной задаче, на больших выборках стоит посмотреть в сторону Multicore-TSNE;
* картинка может сильно поменяться при изменении random seed, это усложняет интерпретацию. В целом по таким картинкам не стоит делать далеко идущих выводов – не стоит гадать по кофейной гуще. Иногда что-то бросается в глаза и подтверждается при изучении, но это не часто происходит.

И еще пара картинок. С помощью t-SNE можно действительно получить хорошее представление о данных, а можно просто нарисовать елочную игрушку.