**Анализ данных**

Крушение парохода «Титаник» представляет крупнейшую морскую катастрофу, произошедшей в ночь с 14 на 15 апреля 1912 года в северной части Атлантического океана. Трагедия случилась под конец пятого дня следования «Титаника» по трансатлантическому маршруту Саутгемптон Нью-Йорк. В 23часа 40 минут 14 апреля во время первого рейса самый большой на тот момент океанский лайнер с 2208 людьми на борту по касательной столкнулся с айсбергом и получил серьёзные повреждения обшивки корпуса. Спустя 2 часа 40 минут полностью ушёл под воду. Катастрофа унесла жизни, по разным данным, от 1495 до 1635 человек.

Выполните анализ вероятности спасения пассажиров Титаника.

К задаче прилагается файл: *titanic.csv* набор данных на основании, которого будет строиться и проверятся модель (с разделителем чисел – запятая, разделитель десятичной части – точка).

Пояснения по некоторым полям:

**PassengerId -** идентификатор пассажира;

**Survival -** поле в котором указано спасся человек (1) или нет (0);

**Pclass -** содержит социально-экономический статус: высокий, средний, низкий;

**Name -** имя пассажира;

**Sex -** пол пассажира;

**Age -** возраст пассажира;

**SibSp -** содержит информацию о количестве родственников 2-го по-рядка (муж, жена, братья, сестры);

**Parch -** содержит информацию о количестве родственников на борту 1-го порядка (мать, отец, дети);

**Ticket -** номер билета;

**Fare -** цена билета;

**Cabin -** каюта;

**Embarked -** порт посадки: C - Cherbourg, Q - Queenstown, S - South-ampton.

**Технология выполнение учебного здания**

*Ввод данных*

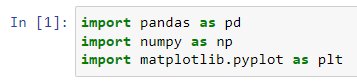
Создайте новый файл notebook. Для этого нажмите кнопку New в правом верхнем углу и выберите Python 3. Нажмите на заголовок и в открывшемся окне задайте имя Pract01.

Загружаем необходимые для решения задачи библиотеки pandas, numpy, matplotlib.

Для анализа понадобятся модули pandas, numpy и sklearn. С помощью pandas и numpy мы проведем начальный анализ данных, а sklearn поможет в вычислении прогнозной модели.

Библиотека pandas дает возможность строить сводные таблицы, выполнять группировки, предоставляет удобный доступ к табличным данным, а наличие пакета matplotlib дает возможность рисовать графики на полученных наборах данных

Библиотека numpy включает поддержку больших многомерных массивов и матриц, а также высокоуровневых математических функций для операций с этими массивами.



Для начала загрузим набор данных и посмотрим, как он выглядит.

Основными структурами хранения данных в Pandas являются Series и DataFrame. Series – это проиндексированный одномерный массив значений. Он похож на простой словарь, где имя элемента будет соответствовать индексу, а значение – значению записи.

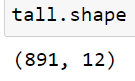
DataFrame - это таблица, представляющая проиндексированный многомерный массив значений.

Создать структуру DataFrame можно с помощью конструктора либо это можно сделать функцией read\_csv():



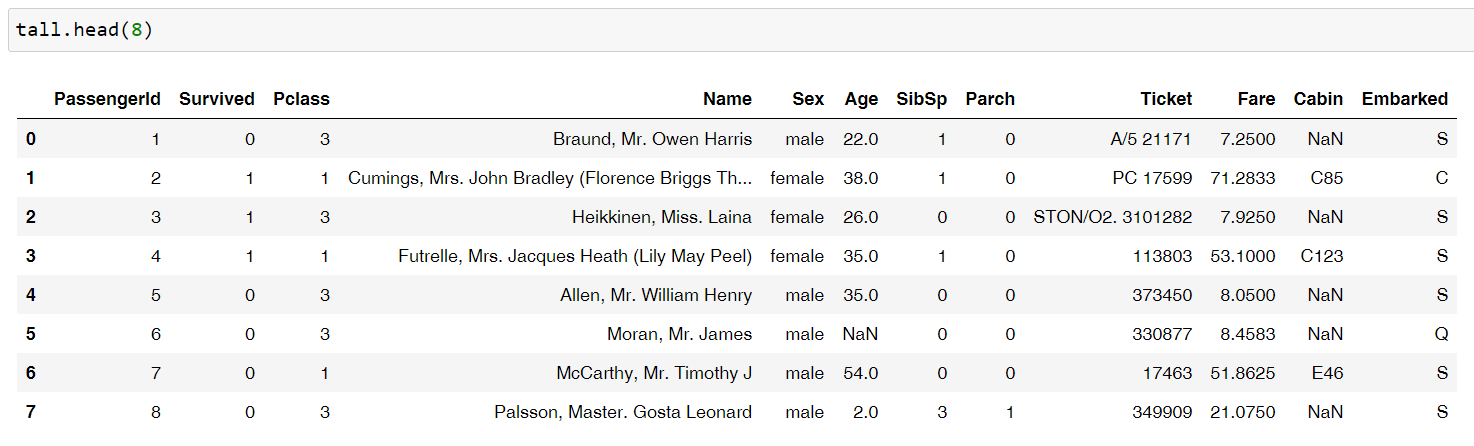
Обратите внимание, что вызов функции read\_csv() осуществляется через библиотеку pandas (сокращенно pd).

Прежде всего, определим размеры таблицы tall, которая в Python описывается структурой DataFrame. С помощью атрибута shape структуры DataFrame можно определить число строк и столбцов.



После этого вы должны увидеть следующее: (891, 12) Это означает, что в нашем наборе данных 891 строк и 12 столбцов.

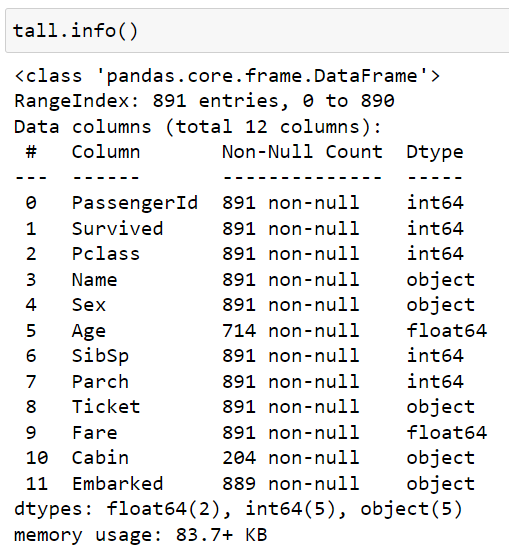
Функция head(n) структуры DataFrame используется для отображения заголовка файла, n – число строк в заголовке.



*Проверка данных*

Полученные данные могут иметь пропуски (незаполненные значения). Нам нужно найти такие значения и либо их заполнить, либо удалить.

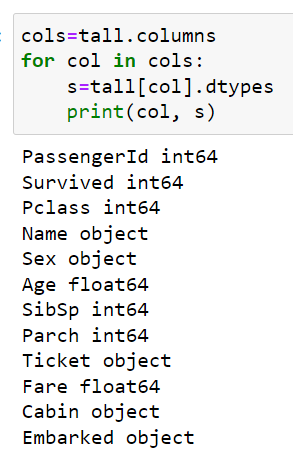
Сначала получим информацию о типах и количестве пропущенных значений данных, которые хранятся в нашей таблице tall. Для этого используем функцию info() структуры DataFrame. Код команды: tall.info()



Как видно из рисунка из 12 показателей, хранящихся в таблице, пять показателей имеют тип object, т.е. не числовой тип. Такие показатели нужно либо преобразовать в числовые показатели, либо удалять.

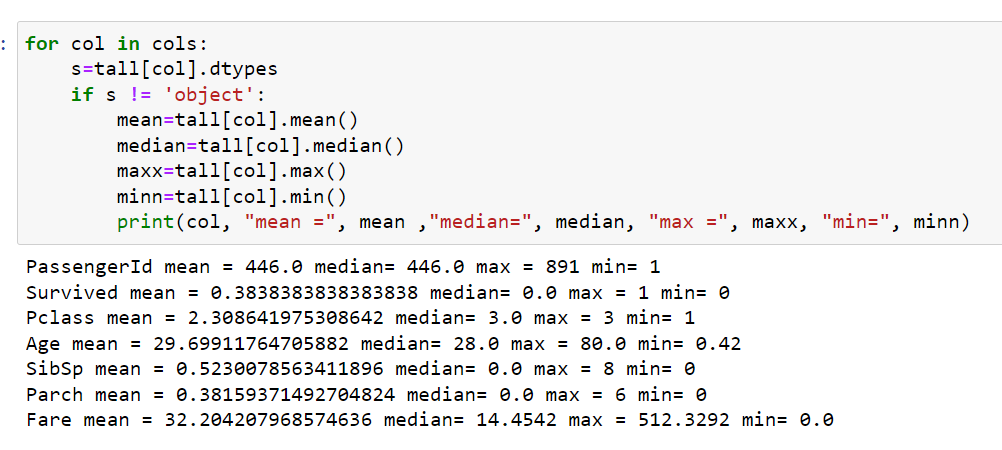
Мы можем увидеть, что пропуски есть для полей Age, Cabin, Embarked. Два пропусков найдено в поле Embarked, 177 значений пропущено для поля Age и 687 значений показателей не определены для поля Cabin.

Изучим структуру таблицы. Будем использовать для этого цикл:



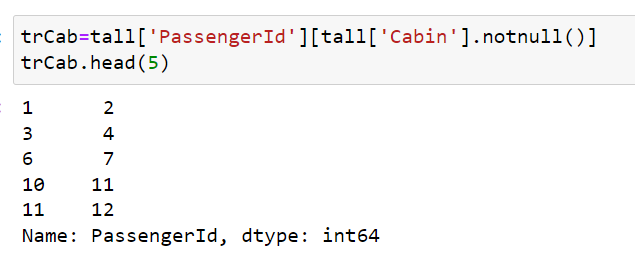
Таким образом, несколько полей (Survived, Pclass, Sex, SibSb, Parch, Em-barked) имеют небольшое число уникальных значений.

Далее вычислим для каждого числового поля простейшие статистические характеристики: среднее значение, медиану, максимальное и минимальное значение. Для этого используем соответствующие методы Data-Frame.

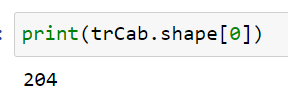


Рассмотрим поле Cabin, которое содержит номера кают. Для определения, у каких пассажиров отсутствует информация о номерах кают, Используйте метод notnull() структуры DataFrame, который выбирает записи с про-пущенными значениями указанного поля.

Используйте оператор, который выбирает значения поля PassengerId, которым соответствуют неопределенные значения поля Cabin.

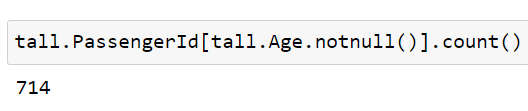


Число неопределенных значений можно определить с помощью оператора shape структуры DataFrame.



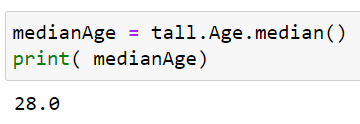
В итоге получаем подтверждение уже известной информации: заполнено всего 204 записи и 890, на основании этого можно сделать вывод, что данное поле при анализе может быть бесполезным.

Следующее поле, которое мы разберем, будет поле с возрастом (Age). Посмотрим на сколько оно заполнено:



Результат выполнения операции показывает, что данное поле практически все заполнено (714 непустых записей), хотя и есть пустые значения, которые не определены.

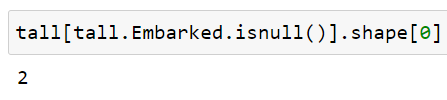
Пустые значения можно заполнить медианным значением возраста пассажиров. Это позволит создать более адекватную модель. Вычислим медиану возраста пассажиров. Для этого используем оператор:



Для замены пустых значений медианным значением используйте оператор:



Разберем поле Embarked, которое содержит порт посадки. Проверим, есть ли такие пассажиры, у которых порт не указан. Для этого выполним оператор:



Итак, у нас нашлось 2 таких пассажира. Давайте присвоим этим пассажирам порт, в котором село больше всего людей. Для этого нам нужно опять сформировать сводную таблицу, показывающей число пассажиров, севших на корабль в каждом порту. Можно использовать метод pivot\_table библиотеки pandas.

Синтаксис функции:

pivot\_table(data, values=None, index=None, columns=None, aggfunc='mean', fill\_value=None, margins=False, dropna=True, margins\_name='All')

Функция *pivot\_table* возвращает сводную таблицу в стиле *DataFrame*.

Аргумент *data* – это исходная таблица типа *DataFrame*.

Аргумент values задает имя столбцы, который агрегируется. Для нашей таблицы это будет столбец *PassengerId*.

Аргумент *index* задает имя столбца, по которому осуществляется группировка. Используется для идентификации строк агрегированной таблицы. В нашем случае будем использовать *Pclass*.

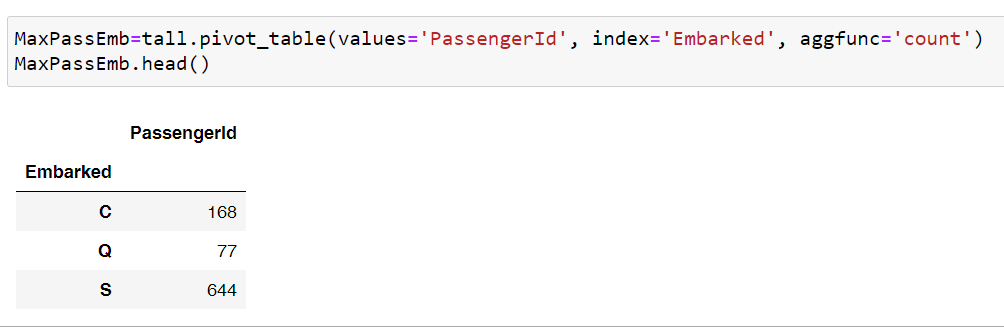
Аргумент *columns* задает имя столбца, по которому осуществляется группировка. Используется для идентификации столбцов агрегированной таблицы. В нашем случае будем использовать *Survived*.

Аргумент *aggfunc* – определяет имя функции агрегирования. В качестве имени может использоваться *mean* (среднее значение), *sum* (сумма значе-ний), *count* (количество значение).

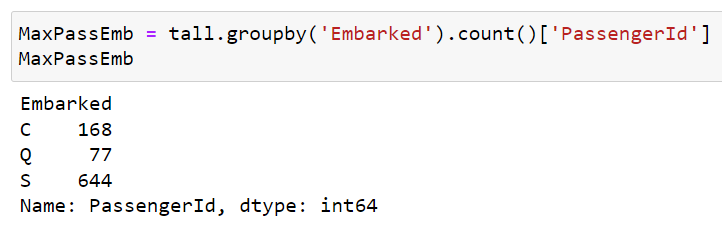
Аргумент *fill\_value* – позволяет заполнять заданными значениями ячейки таблицы типа *NAN* (неопределенное значение).

В качестве столбца, который агрегируется используйте поле *PassengerId*, а в качестве столбца, по которому осуществляется группировка, поле *Embarked*. функция агрегирования должна подсчитывать количество значе-ний поля *PassengerId*.

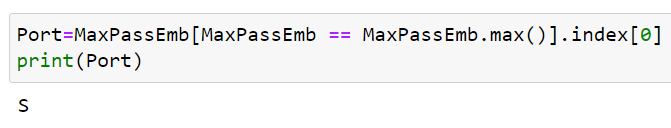
Таким образом, сводная таблица формируется с помощью следующей команды:



Для создания сводной таблицы также может использоваться функция groupby(), которая возвращает объект GroupBy. В нашем случае это сводная таблица. Функции типа count(), max(), min(), mean(), first(), last() могут быть быстро применены к объекту GroupBy для получения сводной статистики для каждой группы.



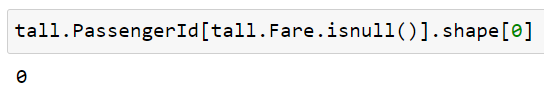
Результат будет аналогичен.



Далее присваиваем пропущенным данным название порта, в котором осуществило посадку максимальное число пассажиров.

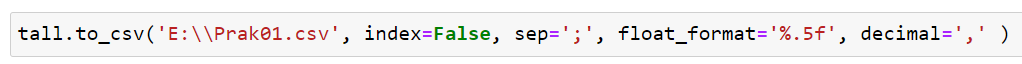


Из оставшихся полей нам интересна только цена (Fare), т.к. она в какой-то мере определяем ранжирование внутри классов поля Pclass. Например, люди внутри среднего класса могут быть разделены на тех, кто ближе к первому (высшему) классу, а кто к третьему(низший). Проверим это поле на пустые значения и, если таковые имеются, заменим цену медианой по цене из всей выборки:



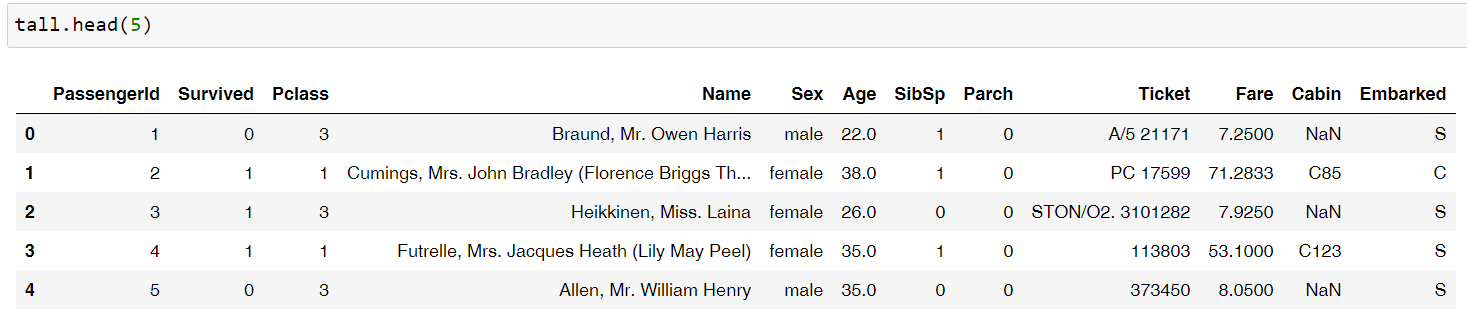
В нашем случае пустых записей нет.

Сохраним полученную таблицу в файле *Prak01.scv*. Для этого используйте функцию *to\_csv().*



Обратите внимание, что мы сохраняем таблицу в формате, который ис-пользуется в России: в качестве разделителей чисел и десятичной части ис-пользуется точка с запятой и запятая.

Для проверки правильности записанного файла используется оператор *tall.head(5).*

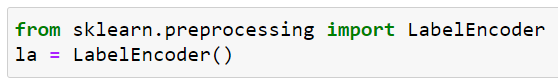


Преобразование тестовых полей в числовые поля.

Предварительный анализ данных завершен, и по его результатам у нас получилась некая выборка, в которой содержатся несколько полей и можно преступить к построению модели, если бы не одно «но»: наши данные содержат не только числовые, но и текстовые данные.

Поэтому перед тем, как строить модель, нужно превратить наши текстовые поля в числовые. В реальных данных не так редки категориальные признаки, принимающие дискретные значения, такие как да/нет или январь/февраль/.../декабрь. В нашем случае таким категориальными признаками (показателями) являются поля Sex и Embarked.

Преобразовать категориальные признаки в числовые можно с помощью класса LabelEncoder(). Суть функции преобразования данного класса заключается в том, что на вход ей подается список значений, который надо закодировать (например, текстовые переменные), а на выходе получается спи-сок индексов, каждый из которых являются кодом уникальных значений поданного на вход списка. Первым делом нужно объявить класс LabelEncoder.

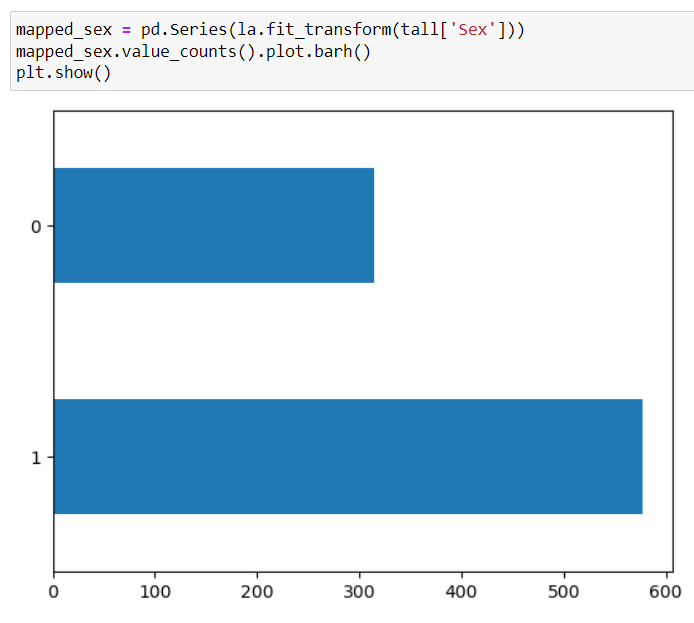


Показатель Sex представляет массив с двумя уникальными значениями female и male. Таким образом, если обозначить один индексом 0, а другой индексом 1, то мы легко можем преобразовать это поле в числовое. Для начала проверим как преобразуются наши данные. Для этого будем использовать функцию *fit\_transform().* В результате получим массив, который отформатируем как структуру Series.

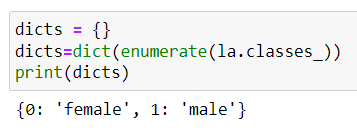
Структура Series представляет из себя объект, похожий на одномерный массив (список, например), но отличительной его чертой является наличие ассоциированных меток, так называемых индексов, вдоль каждого элемента из списка.

В строковом представлении объекта Series, индекс находится слева, а сам элемент справа. Объект Series напоминает словарь, где ключом является индекс, а значением сам элемент.

Для того чтобы определить количество уникальных значений используйте функцию value\_counts(), которая является методом объекта Series. Функция value\_counts() возвращает объект Series, который содержит количество уникальных значений массива mapped\_sex. Для графически отображения используйте функцию plot.barh().

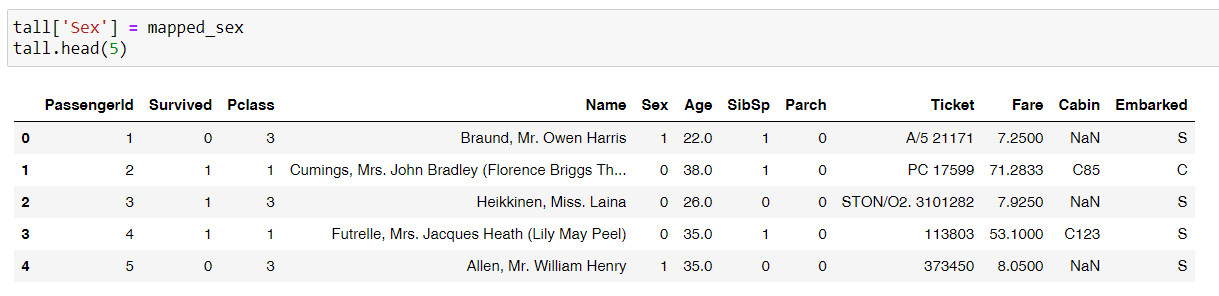


Распечатаем уникальные индексы поля Sex и их текстовые значения. Для потребуется словарь, поэтому создадим словарь с именем dicts.

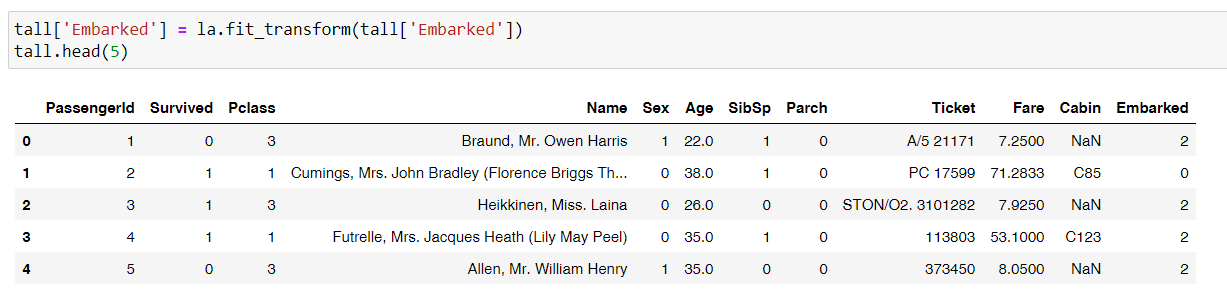


В результате мы получили mapped\_sex (объект типа Series) содержащий 0 и 1.

Полученный массив числовых индексов поля Sex загружаем в таблицу tall заменяя текстовые значения поля.

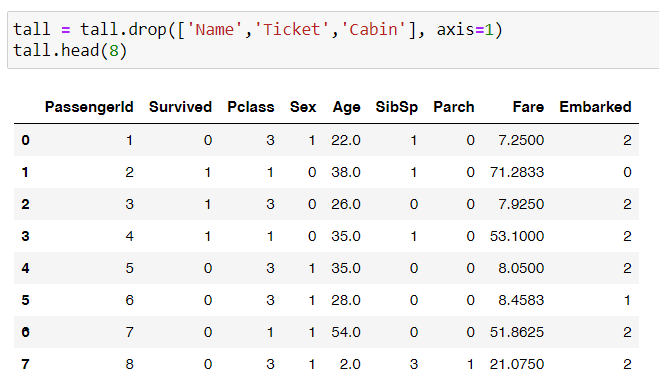


Преобразуйте значения поля Embarked в числовые значения.



Как показывает предварительный анализ номер билета и имя пассажира нам никак не помогут, т.к. имя - это просто справочная информация, а по полю номер билета очень много пропущенной информации. Поэтому их нужно просто удалить. Для этого используем функцию drop() класса Data-Frame.

Аргумент axis используется для указания оси удаления. В нашем случае это столбцы, поэтому axis=1. Теперь, после удаления всех ненужных полей, наш набор выглядит так как показано на рисунке.



Можно предположить, что чем выше социальный статус пассажира, тем выше вероятность его спасения. Давайте проверим это предположение. Для этого вычислим, сколько пассажиров каждого класса спаслись и сколько утонули. Такой анализ можно легко выполнить путем построения сводной таблицы с помощью функции *pivot\_table* библиотеки pandas. Синтаксис функции:

pivot\_table*(data, values=None, index=None, columns=None, ag-gfunc='mean', fill\_value=None, margins=False, dropna=True, mar-gins\_name='All'*)

Функция *pivot\_table* возвращает сводную таблицу в стиле DataFrame.

Аргумент *data* – это исходная таблица типа DataFrame.

Аргумент *values* задает имя столбцы, который агрегируется. Для нашей таблицы это будет столбец PassengerId.

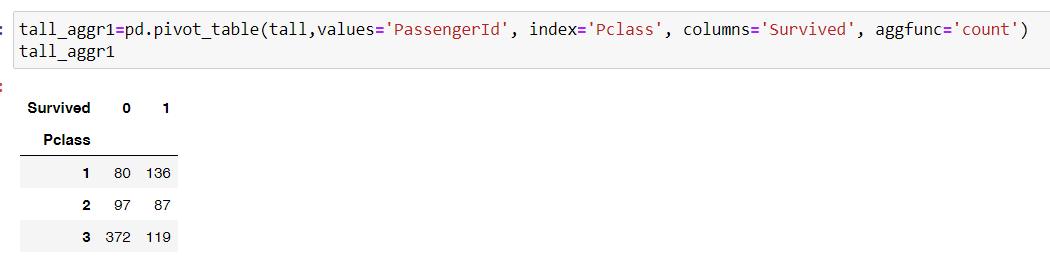
Аргумент *index* задает имя столбца, по которому осуществляется группировка. Используется для идентификации строк агрегированной таблицы. В нашем случае будем использовать Pclass.

Аргумент *columns* задает имя столбца, по которому осуществляется группировка. Используется для идентификации столбцов агрегированной таблицы. В нашем случае будем использовать Survived.

Аргумент *aggfunc* – определяет имя функции агрегирования. В качестве имени может использоваться mean (среднее значение), sum (сумма значений), count (количество значение).

Аргумент *fill\_value* – позволяет заполнять заданными значениями ячейки таблицы типа NAN (неопределенное значение).

Для агрегирования таблицы tall следующий оператор:



Таким образом, агрегированная таблица показывает, что 136 из 226 пассажиров 1-го класса спаслись.

Попробуйте записать оператор немного по-другому, использую функцию pivot\_table как метод класса DataFrame.

Отобразим данные сводной таблицы в виде диаграммы. Для этого будем использовать метод plot структуры DataFrame. Описание структуры Data-Frame:

DataFrame.plot(x=None, y=None, kind='line', ax=None, subplots=False, sharex=None, sharey=False, layout=None, figsize=None, use\_index=True, ti-tle=None, grid=None, legend=True, style=None, logx=False, logy=False, log-log=False, xticks=None, yticks=None, xlim=None, ylim=None, rot=None, font-size=None, colormap=None, table=False, yerr=None, xerr=None, second-ary\_y=False, sort\_columns=False, \*\*kwds)¶

Аргумент *kind* задает тип графического отображения и может принимать значения:

*line* – график линейный (по умолчанию);

*bar* – вертикальная столбчатая диаграмма;

*barh* – горизонтальная столбчатая диаграмма;

*hist* – гистограмма.

Аргумент *figsize* задает размеры рисунка.

Аргумент *fontsize* задает размеры шрифта.

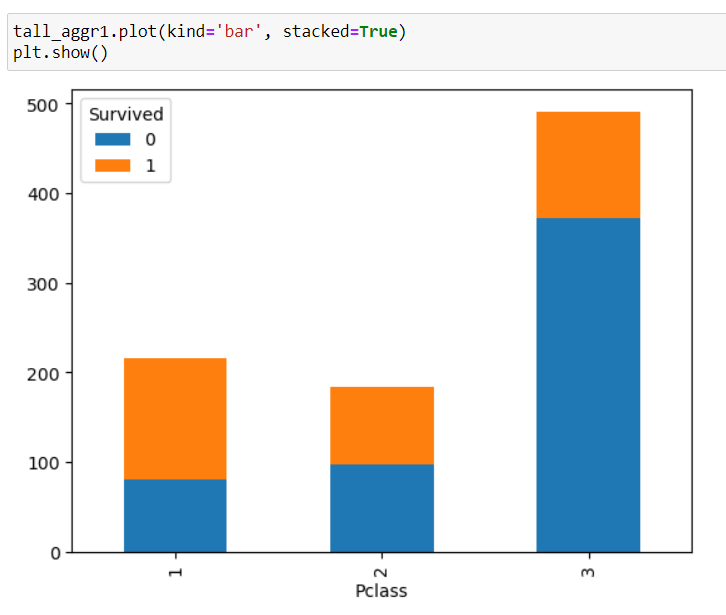
Аргумент *colormap* принимает значения либо палитры Matplotlib, либо строки, которая является именем colormap, зарегистрированным в Matplotlib.

Примеры имен colormap: gist\_rainbow, gist\_gray, winter, spring, summer, autumn, spectral, cubehelix, prism.

Аргумент *stacked* позволяет задавать сложные диаграммы. Для этого он должен иметь значение True.

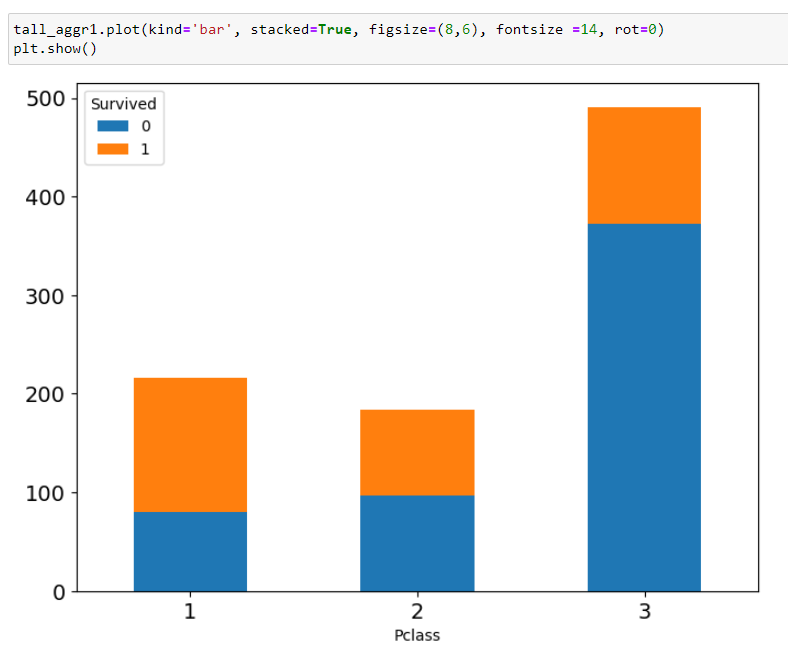
Аргумент *title* позволяет задать название графика или диаграммы, которое расположено в верхней части рисунка.

Построите диаграмму, используя оператор:

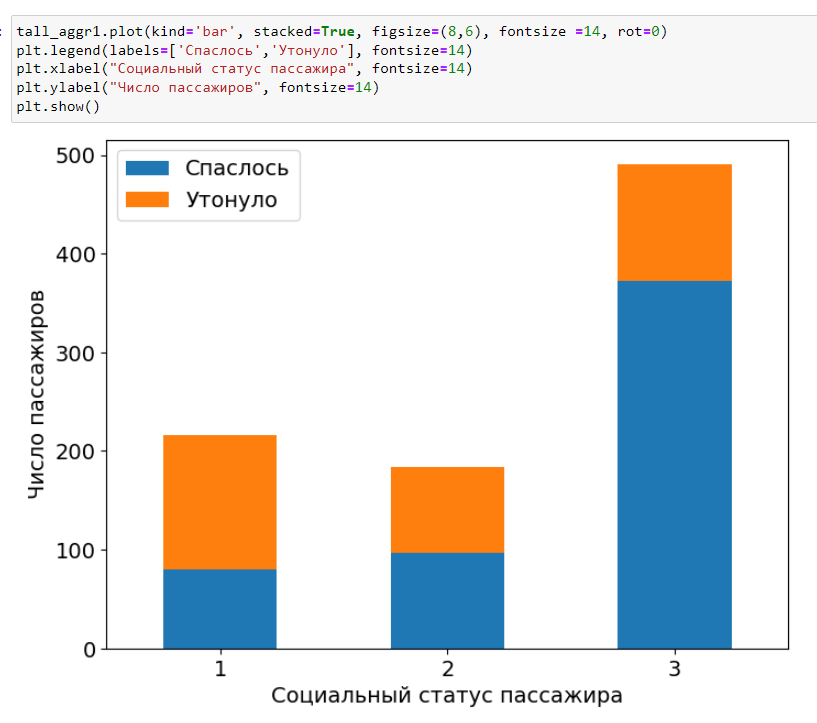


Вышеописанное предположение про то, что чем выше у пассажиров их социальное положение, тем выше их вероятность спасения, оказалось правильным.

Для изменения размеров рисунка используйте аргументы figsize, fontsize. Кроме того, угол поворота текста зададим равным 0, код будет выглядеть следующим образом:



Задайте подписи по осям. Для этого будем использовать функции xlabel() и ylabel(). В качестве аргументов указываем строку с названием оси и размер шрифта. Далее откорректируем легенду с помощью функции legend(). В качестве аргументов задаем labels и fontsize.



Отчет сохранить в виде HTML файла.

