Лабораторная работа «Получение и предобработка данных»

Задание: получите данные из файла формата .csv, выполните предобработку полученных данных.

1. Библиотека Pandas.

1). Получите доступ к библиотеке Pandas, используйте имя переменной pd.

2). Создайте список music с 5 парами «имя вашего любимого исполнителя - название его песни». 3). Создайте список entries с названиями для двух столбцов — artist и track. 4). Используя конструктор DataFrame(), создайте таблицу из списка ваших любимых исполнителей music и списка столбцов entries. Сохраните таблицу в переменной playlist и выведите эту сборную таблицу на экран.

2. Получение данных

1). Прочитайте файл music_log.csv и сохраните его в переменной df. Сохраните первые 5 строк с данными из music_log.csv в переменной music_head и выведите значение переменной на экран.

2). Прочитайте файл music_log.csv и сохраните его в переменной df. Сохраните последние 10 строк с данными из music_log.csv в переменной music_tail и выведите значение переменной на экран.

3. Объект DataFrame

1). Прочитайте файл music_log.csv и сохраните его в переменной df. Создайте переменную shape_table и сохраните в ней размеры таблицы music_log.csv. Напечатайте на экране размер таблицы в таком виде: Размер таблицы: ...

2). Сколько наблюдений в наборе данных? В переменной shape_table хранится кортеж.

Его первый элемент — количество наблюдений, который надо сохранить в переменной observations_table (не забывайте, что индексация элементов идёт с 0). Напечатайте на экране ответ в таком виде: Количество наблюдений: ...

3). Сравните полученные результаты в переменных observations_info_table и observations_table. Если значения переменных совпадают, то выведите количество наблюдений и сообщение: «Решение верно, количество наблюдений равно», observations_table. Если значения переменных не совпадают, то выведите сообщение: «Решение неверно, проверьте еще раз!»

4. Индексация в DataFrame

1). Получите таблицу, состоящую из столбцов genre и Artist. Сохраните её в переменной genre_fight. Посчитайте число прослушанных треков в жанре поп.

Сохраните результат в переменной genre_pop. Напечатайте ответ на экране в виде: Число прослушанных треков в жанре поп равно ...

2). Посчитайте число прослушанных треков в жанре рок. Допишите в код подсчёт, похожий на предыдущий, только с логическим условием df['genre'] == 'rock'. Сохраните результат в переменной genre_rock. Напечатайте ответ на экране в виде:

Число прослушанных треков в жанре поп равно ...

Число прослушанных треков в жанре рок равно ...

3). Напишите условную конструкцию, которая сравнивает полученные значения и выводит информацию о победителе в этом бою! Если победил жанр рок, то выведите сообщение "Рок победил!", а если победил жанр поп - сообщение "Попса forever!"

5. Объект Series

1). Получите таблицу только с жанром rock и сохраните её в переменной rock.

2). Выделите время прослушивания роковых композиций в особую структуру данных. Сохраните столбец 'total play' таблицы rock в переменной rock_time.

3). Обратитесь к новой Series с именем rock_time и посчитайте количество треков жанра рок, пропущенных в течение 5 секунд. Логическим условием укажите rock_time <= 5. Результат сохраните в переменной rock_haters и напечатайте на экране с пояснением:

Количество пропущенных треков жанра рок равно ...

4). Выберите из исходной таблицы только строки с жанром 'pop' и сохраните эту новую таблицу в переменной pop.

5). По аналогии с роком создайте Series, где хранятся только данные о времени воспроизведения композиций в жанре поп. Назовите его pop_time и сохраните в нём данные столбца 'total play' из таблицы pop .

6). По аналогии с роком обратитесь к Series pop_time, чтобы посчитать количество пропущенных в течение 5 секунд треков жанра поп. Используйте условие pop_time <= 5. Результат сохраните в переменной pop_haters и напечатайте на экране в таком виде:

Количество пропущенных треков жанра поп равно ...

7). Для обоих жанров посчитайте долю быстро пропущенных пользователями композиций в процентах. Разделите количество треков, которые пользователи пропустили — соответственно rock_haters и pop_haters — на общее количество треков жанра рок и жанра поп.

Общее количество треков жанра равно количеству наблюдений в таблицах rock и pop, т.е. значению атрибута shape[0] этих таблиц.

Результаты сохраните в переменных rock_skip и pop_skip. Выведите значения новых переменных в процентах с точностью до одного знака после запятой в форме:

Доля пропущенных композиций жанра рок равна: ...

Доля пропущенных композиций жанра поп равна: ...

Лабораторная работа «Раздельный сбор информационного мусора»

Задание: получите данные из файла формата .csv, удалите информационный мусор.

1. Получение данных

1). Прочитайте файл music_log.csv и сохраните его в переменной df.

2). Просмотрите информацию о наборе данных, воспользовавшись методом info().

2. Переименование столбцов

1). Выведите список столбцов.

2). Подготовьте список new_names с новыми именами для столбцов:

- user_id → user_id

- total play → total_play_seconds

- Artist → artist_name

- genre → genre_name

- track → track_name

3). Переименуйте столбцы таблицы, которая хранится в переменной df.

4). Проверьте, что получилось, запросив для структуры данных df атрибут columns.

3. Обработка пропущенных значений

1). Посчитайте количество пропущенных значений в наборе данных и выведите его на экран.

2). Заполните отсутствующие значения столбца 'track_name' строкой 'unknown'. 3). Заполните отсутствующие значения столбца 'artist_name' строкой 'unknown'.

4). Удалите пропущенные значения из столбца 'genre_name'.

5). Проверьте полученный результат. Просмотрите информацию о наборе данных:

воспользуйтесь методом info().

4. Обработка дубликатов

1). Сохраните текущий размер таблицы в переменной shape_table.

2). Посчитайте и выведите на экран суммарное количество дубликатов в таблице.

3). Удалите дубликаты. Используйте метод reset_index() для сохранения порядка индексов.

4). Сохраните в переменную shape_table_update размер таблицы после удаления дубликатов.

5). Сравните переменные shape_table и shape_table_update. Если они равны, выведите сообщение 'Размер таблицы не изменился, текущий размер: ' и значение переменной shape_table_update. В ином случае сообщение должно быть таким:

'Таблица уменьшилась, текущий размер: ' и значение переменной shape_table_update.

6). Получите уникальные значения столбца 'genre_name', используйте метод unique().

Просмотрите результат и найдите название жанра, которое выпадает из общего ряда.

7). Оцените изменения: пересчитайте количество значений 'электроника' в столбце 'genre_name'. Если удалось всё заменить, результат должен быть равен 0.

Сохраните этот результат в переменной genre_final_count, выведите на экран. Примените к отобранным по логическому условию df['genre_name'] == 'электроника' значениям столбца 'genre_name' метод count() для подсчёта. Результат сохраните в переменной genre_final_count, значение которой напечатайте на экране.

5. Сделайте выводы по проделанной работе.

Лабораторная работа «Анализ данных и оформление результатов»

Задание: получите данные из файла формата .csv, выполните расчеты, подготовьте презентацию по результатам работы.

1. Знакомство с набором данных

1). Прочитайте данные из файла music_log_2.csv и выведите первые 10 строк (music_log_2.csv — обновлённый файл с данными, которые прошли предобработку).

2). Получите список названий столбцов, запросив атрибут columns. Результат выведите на экран.

3). Посчитайте количество пустых значений в наборе данных, сохраните результат в переменной na_number. Выведите её значение на экран.

4). Посчитайте количество дубликатов в наборе данных, сохраните результат в переменной duplicated_number. Выведите её значение на экран.

2. Группировка данных

1). Узнайте user_id меломанов. Для этого сгруппируйте данные по каждому пользователю, чтобы собрать жанры прослушанных им композиций. Сгруппируйте DataFrame по столбцу user_id, сохраните полученный результат в переменной genre_grouping. Посчитайте количество жанров, которые выбрали пользователи, методом count(), указав, что выбираем один столбец genre_name. Сохраните результат в переменной genre_counting и выведите первые 30 строк этой таблицы.

2). Быть может, те, кто за день слушает больше 50 песен, имеют более широкие предпочтения. Чтобы найти такого, напишите функцию user_genres, которая принимает некую группировку как свой аргумент group. Функция должна перебирать группы, входящие в эту группировку. В каждой группе два элемента — имя группы с индексом 0 и список значений с индексом 1. Обнаружив такую группу, в которой список (элемент с индексом 1) содержит более 50 значений, функция возвращает имя группы (значение элемента с индексом 0).

3). Вызовите функцию user_genres, как аргумент передайте ей genre_grouping.

Результат – user_id неведомого нам любителя музыки – сохраните в переменной search_id и выведите значение на экран.

3. Сортировка данных

1). Выполняя предыдущее задание, вы обнаружили меломана с уникальными данными. Он за день послушал больше 50 композиций. Получите таблицу с прослушанными им треками. Для этого запросите из структуры данных df строки, отвечающие сразу двум условиям: 1) значение в столбце 'user_id' должно быть равно значению переменной search_id; 2) время прослушивания, т.е. значение в столбце 'total_play_seconds', не должно равняться 0. Сохраните результат в переменной music_user.

2). Узнайте, сколько времени он слушал музыку каждого жанра. Сгруппируйте данные таблицы music_user по столбцу 'genre_name' и получите сумму значений столбца

'total_play_seconds'. Сохраните результат в переменной sum_music_user и выведите её значение на экран.

3). Важно знать, сколько треков каждого жанра он включил. Сгруппируйте данные по столбцу genre_name и посчитайте, сколько значений в столбце genre_name. Сохраните результат в переменной count_music_user и выведите её значение на экран.

4). Чтобы предпочтения были видны сразу, нужно крупнейшие значения расположить наверху. Отсортируйте данные в группировке sum_music_user по убыванию. Внимание: когда применяете метод sort_values() к Series с единственным столбцом, аргумент by указывать не нужно, только порядок сортировки. Сохраните результат в переменной final_sum и выведите её значение на экран.

5). Теперь то же самое надо сделать с числом прослушанных меломаном композиций. Отсортируйте данные группировки count_music_user по убыванию. Сохраните результат в переменной final_count, значение которой выведите на экран.

4. Описательная статистика

1). Получите таблицу с композициями самого популярного жанра — pop, исключив пропущенные треки. Сохраните результат в переменной pop_music.

2). Найдите максимальное время прослушивания песни в жанре pop. Сохраните результат в переменной pop_music_max_total_play и выведите её значение на экран.

3). Получите строку таблицы pop_music с информацией о самой длинной по времени прослушивания песне жанра 'pop' и сохраните её в переменной pop_music_max_info. Выведите эту строку на экран.

4). Найдите минимальное ненулевое время прослушивания композиции в жанре pop.

Сохраните его в переменной pop_music_min_total_play, значение выведите на экран.

5). Выведите на экран информацию о композиции жанра pop, которую запустили, но быстрее всех остальных выключили. Результат сохраните в переменную pop_music_min_info и выведите на экран.

6). Рассчитайте медиану времени прослушивания произведений жанра pop. Сохраните результат в переменной pop_music_median и выведите на экран.

7). Рассчитайте среднее арифметическое времени прослушивания произведений жанра pop. Сохраните результат в переменной pop_music_mean и выведите на экран.

5. Решение задачи и оформление результатов

1). Рассчитайте метрику happiness для всего набора данных. Сохраните полученный результат в переменной current_happiness и выведите на экран.

Метрика happiness рассчитывается так: считаем, как долго каждый пользователь слушал музыку. Для этого сгруппируем DataFrame по пользователю. Посчитаем общее время прослушивания музыки. Находим медианное значение для суммы прослушиваний по пользователю.

2). Рассчитайте разность двух значений метрики happiness до (57.456 секунд) и после эксперимента. Сделайте выводы об изменении удовлетворенности пользователей сервисом.

Лабораторная работа «Предобработка и анализ данных»

Вы работаете в интернет-магазине «Стримчик», который продаёт по всему миру компьютерные игры. Из открытых источников доступны исторические данные о продажах игр, оценки пользователей и экспертов, жанры и платформы (например, Xbox или PlayStation). Вам нужно выявить определяющие успешность игры закономерности. Это позволит сделать ставку на потенциально популярный продукт и спланировать рекламные кампании.

Перед вами данные до 2016 года. Представим, что сейчас декабрь 2016 г., и вы планируете кампанию на 2017-й. Нужно отработать принцип работы с данными. Неважно, прогнозируете ли вы продажи на 2017 год по данным 2016-го или же 2027-й — по данным 2026 года.

В наборе данных попадается аббревиатура ESRB (Entertainment Software Rating Board) — это ассоциация, определяющая возрастной рейтинг компьютерных игр. ESRB оценивает игровой контент и присваивает ему подходящую возрастную категорию, например, «Для взрослых», «Для детей младшего возраста» или «Для подростков».

Шаг 1. Откройте файл с данными *games.csv* **и изучите общую информацию**

Шаг 2. Подготовьте данные

- Замените названия столбцов (приведите к нижнему регистру);
- Преобразуйте данные в нужные типы. Опишите, в каких столбцах заменили тип данных и почему;
- Обработайте пропуски при необходимости:
    - Объясните, почему заполнили пропуски определённым образом или почему не стали это делать;
    - Опишите причины, которые могли привести к пропускам;
    - Обратите внимание на аббревиатуру 'tbd' в столбце с оценкой пользователей. Отдельно разберите это значение и опишите, как его обработать;
- Посчитайте суммарные продажи во всех регионах и запишите их в отдельный столбец.

Шаг 3. Проведите исследовательский анализ данных

- Посмотрите, сколько игр выпускалось в разные годы. Важны ли данные за все периоды?
- Посмотрите, как менялись продажи по платформам. Выберите платформы с наибольшими суммарными продажами и постройте распределение по годам. За какой характерный срок появляются новые и исчезают старые платформы?
- Возьмите данные за соответствующий **актуальный период.** Актуальный период определите самостоятельно в результате исследования предыдущих вопросов. Основной фактор — эти данные помогут построить прогноз на 2017 год.
- Не учитывайте в работе данные за **предыдущие годы**.
- Какие платформы лидируют по продажам, растут или падают? Выберите несколько потенциально прибыльных платформ.

- Постройте график «ящик с усами» по глобальным продажам игр в разбивке по платформам. Опишите результат.
- Посмотрите, как влияют на продажи внутри одной популярной платформы отзывы пользователей и критиков. Постройте диаграмму рассеяния и посчитайте корреляцию между отзывами и продажами. Сформулируйте выводы.
- Соотнесите выводы с продажами игр на других платформах.
- Посмотрите на общее распределение игр по жанрам. Что можно сказать о самых прибыльных жанрах? Выделяются ли жанры с высокими и низкими продажами?

Шаг 4. Составьте портрет пользователя каждого региона
Определите для пользователя каждого региона (*NA, EU, JP*):

- Самые популярные платформы (топ-5). Опишите различия в долях продаж.
- Самые популярные жанры (топ-5). Поясните разницу.
- Влияет ли рейтинг ESRB на продажи в отдельном регионе?

Шаг 5. Напишите общий вывод

Описание данных

- *Name* — название игры
- *Platform* — платформа
- *Year_of_Release* — год выпуска
- *Genre* — жанр игры
- *NA_sales* — продажи в Северной Америке (миллионы проданных копий)
- *EU_sales* — продажи в Европе (миллионы проданных копий)
- *JP_sales* — продажи в Японии (миллионы проданных копий)
- *Other_sales* — продажи в других странах (миллионы проданных копий)
- *Critic_Score* — оценка критиков (максимум 100)
- *User_Score* — оценка пользователей (максимум 10)
- *Rating* — рейтинг от организации *ESRB* (англ. *Entertainment Software Rating Board).* Эта ассоциация определяет рейтинг компьютерных игр и присваивает им подходящую возрастную категорию.

Данные за 2016 год могут быть неполными.