|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |
| Институт информационных технологий (ИИТ) | |
| Кафедра Прикладной Математики (ПМ) | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2** | | | |
| **по дисциплине «Технологии и инструментарий анализа больших данных»** | | | |
|  | | | |
| Выполнил студент группы ИКБО-14-20 | | Вежновец Ф. Ю. | |
|  | |  | |
| Принял: асистент | | Горячев А. А. | |
| Практические работы выполнены | «\_\_\_»\_\_\_\_\_\_\_2023 г. | | (подпись студента) | |
| «Зачтено» | «\_\_\_»\_\_\_\_\_\_\_2023 г. | | (подпись руководителя) | |
|  |  | |  | |

Москва 2023

Задание 1

Найти и выгрузить многомерные данные (с большим количеством признаков – столбцов) с использованием библиотеки pandas. В отчёте описать найденные данные

Реализация

На сайте kaggle.com был найден датасет “Google Play Store Apps”. Состоящий из колонок:

* App - Название приложения
* Category - Категория, к которой относится приложение
* Rating - Общий рейтинг приложения пользователями (по состоянию на момент удаления)
* Reviews - Количество отзывов пользователей о приложении (по состоянию на момент удаления)
* Size - Размер приложения (как при очистке)
* Installs - Количество пользовательских загрузок/установок приложения (по состоянию на момент удаления)
* Type - Платный или бесплатный
* Price - Цена приложения (как при очистке)
* Content Rating - Возрастная группа, на которую ориентировано приложение - Дети / Зрелые 21+ / Взрослые
* Genres - Приложение может принадлежать к нескольким жанрам (помимо своей основной категории). Например, музыкальная семейная игра будет относиться к музыкальным, игровым, семейным жанрам.
* Last Updated - Дата последнего обновления приложения в Play Store (на момент удаления)
* Current Ver - Текущая версия приложения доступна в Play Store (на момент удаления)
* Android Ver - Минимальная требуемая версия Android (как при очистке)

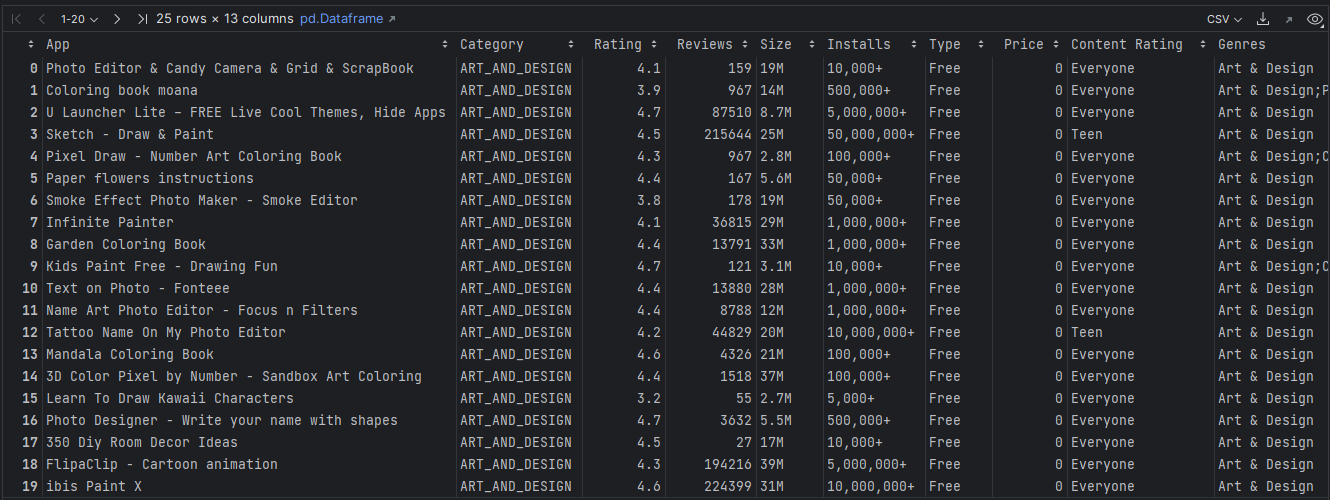


Рисунок 1 – Датасет

Задание 2

Вывести информацию о данных при помощи методов .info(), .head(). Проверить данные на наличие пустых значений. В случае их наличия удалить данные строки или интерполировать пропущенные значения. При необходимости дополнительно предобработать данные для дальнейшей работы с ними.

Реализация

Результат работы программы представлен на рисунке 2.

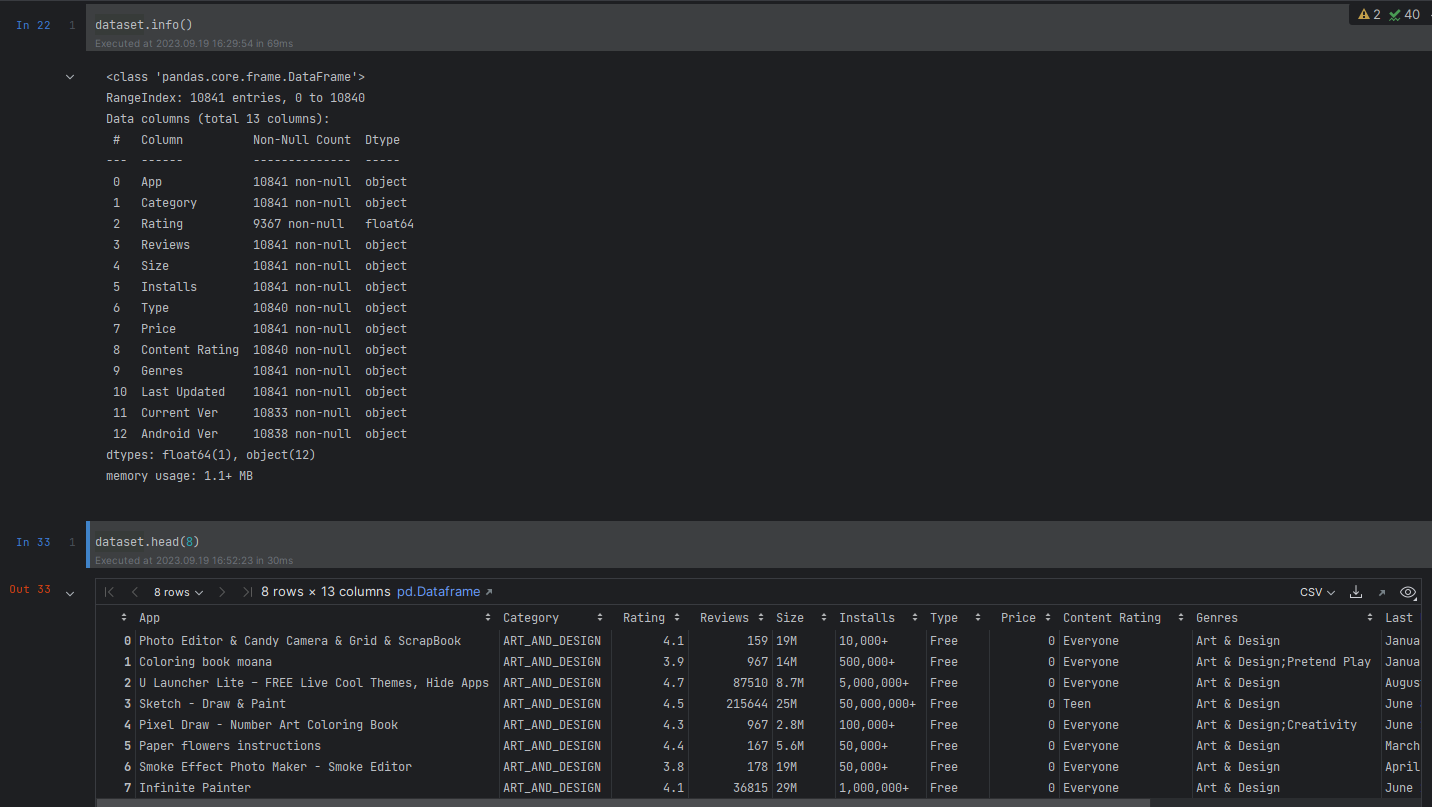


Рисунок 2 – Результат работы программы

Задание 3

Построить столбчатую диаграмму (.bar) с использованием модуля graph\_objs из библиотеки Plotly со следующими параметрами:

* По оси Х указать дату или название, по оси У указать количественный показатель.
* Сделать так, чтобы столбец принимал цвет в зависимости от значения показателя (marker=dict(color=признак, coloraxis="coloraxis")).
* Сделать так, чтобы границы каждого столбца были выделены чёрной линией с толщиной равной 2.
* Отобразить заголовок диаграммы, разместив его по центру сверху, с 20 размером текста.
* Добавить подписи для осей X и Y с размером текста, равным 16. Для оси абсцисс развернуть метки так, чтобы они читались под углом, равным 315.
* Размер текста меток осей сделать равным 14. 3.7. Расположить график во всю ширину рабочей области и присвоить высоту, равную 700 пикселей.
* Добавить сетку на график, сделать её цвет 'ivory' и толщину равную 2. (Можно сделать это при настройке осей с помощью gridwidth=2, gridcolor='ivory')
* Убрать лишние отступы по краям

Реализация

Для решения задачи был написан код (Листинг 1) Результат работы программы представлен на рисунке 3.

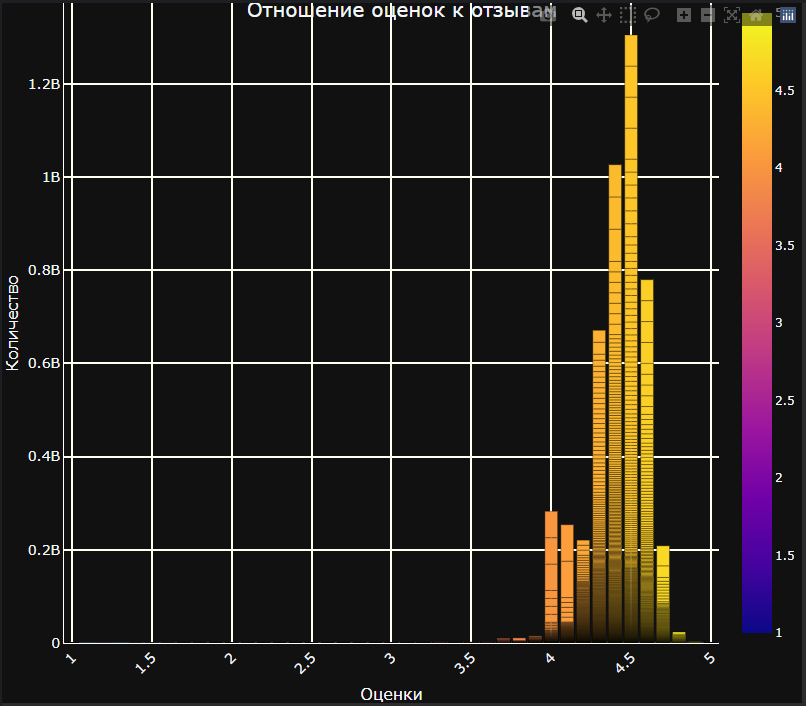


Рисунок 3 – Результат работы программы

Листинг 1

fig = go.Figure()

fig.add\_trace(

go.Bar(x=dataset['Rating'], y=dataset['Reviews'], marker=dict(color=dataset['Rating'], coloraxis="coloraxis"),

marker\_line=dict(color='black', width=0.5)))

fig.update\_layout(title=dict(text='Отношение оценок к скачиванию', x=0.5, y=1, font=dict(size=20)),

xaxis=dict(title=dict(text='Количество', font=dict(size=16)), tickangle=315, tickfont=dict(size=14),

gridwidth=2, gridcolor='ivory'),

yaxis=dict(title=dict(text='Оценки', font=dict(size=16)), tickfont=dict(size=14), gridwidth=2,

gridcolor='ivory'),

width=1200, height=700,

margin=dict(l=0, r=0, t=0, b=0),

xaxis\_showgrid=True, yaxis\_showgrid=True,

xaxis\_linecolor='white', yaxis\_linecolor='white',

showlegend=False)

fig.show()

Задание 4

Построить круговую диаграмму (go.Pie), использовав данные и стиль оформления из предыдущего графика. Сделать так, чтобы границы каждой доли были выделены чёрной линией с толщиной, равной 2 и категории круговой диаграммы были читаемы (к примеру, объединить часть объектов)

Реализация

Для решения задачи был написан код (Листинг 2) Результат работы программы представлен на рисунке 4.

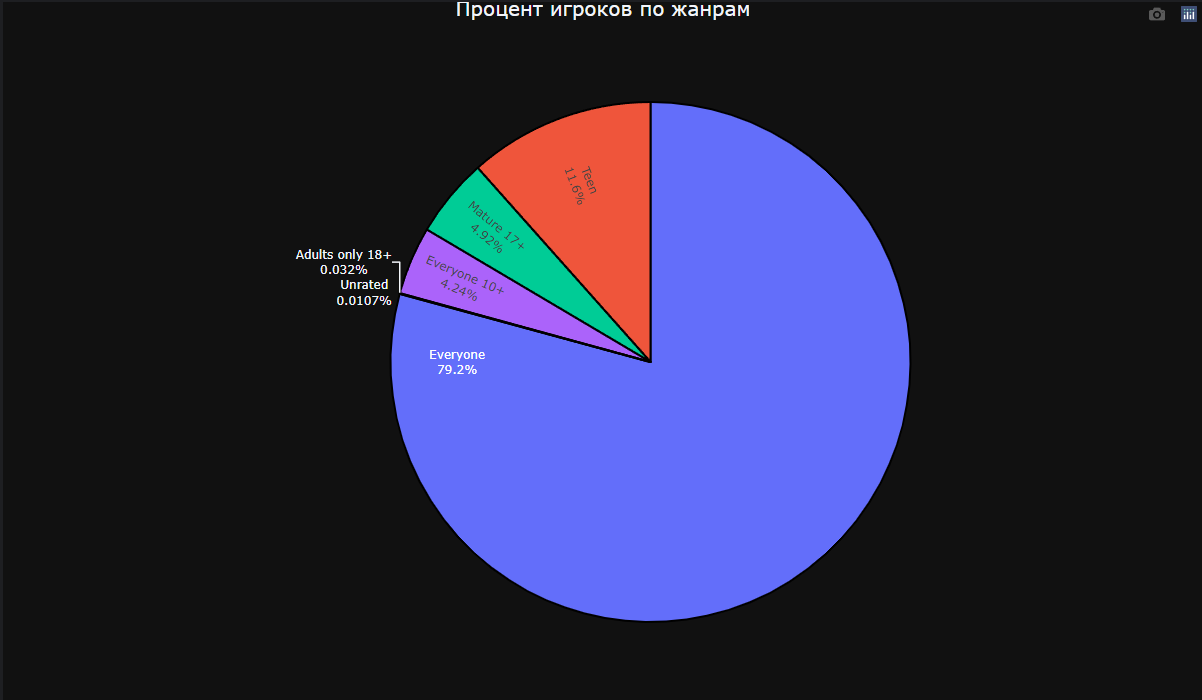


Рисунок 4 – Результат работы программы

Листинг 2

fig\_pi = go.Figure()

fig\_pi.add\_trace(

go.Pie(labels=result\_df['Имя'], values=result\_df['Количество'],

marker=dict(line=dict(color='black', width=2)),

textinfo='label+percent',

insidetextorientation='radial'))

fig\_pi.update\_layout(

title=dict(text='Процент игроков по жанрам', # Заголовок диаграммы

x=0.5, # Положение заголовка по горизонтали

y=1, # Положение заголовка по вертикали

font=dict(size=20)), # Размер шрифта заголовка

showlegend=False, # Скрыть легенду

width=1200, height=700, # Размеры графика

)

fig\_pi.show()

Задание 5

Построить линейные графики, взять один из параметров и определить зависимость между другими несколькими (от 2 до 5) показателями с использованием библиотеки matplotlib. Сделать вывод.

* Сделать график с линиями и маркерами, цвет линии 'crimson', цвет точек 'white', цвет границ точек 'black', толщина границ точек равна 2.
* Добавить сетку на график, сделать её цвет 'mistyrose' и толщину равную 2. (Можно сделать это при настройке осей с помощью linewidth=2, color='mistyrose').

Реализация

Для решения задачи был написан код (Листинг 3) Результат работы программы представлен на рисунке 5.

|  |
| --- |
|  |

Рисунок 5 – Результат работы программы

Листинг 3

sorted\_second = dataset.copy().head(100)

sorted\_second['Size'] = sorted\_second['Size'].str.replace('Varies with device', '0')

def convert\_size(size\_str):

if 'k' in size\_str:

return float(size\_str.replace('k', '')) / 1000

elif 'M' in size\_str:

return float(size\_str.replace('M', ''))

else:

return float(size\_str)

sorted\_second['Size'] = sorted\_second['Size'].apply(convert\_size)

sorted\_second['Rating'] = pd.to\_numeric(sorted\_second['Rating'], errors='coerce').fillna(0).astype(float)

sorted\_second = sorted\_second.sort\_values(by=['Last Updated'])

sorted\_second.head(100)

plt.figure(figsize=(12, 8))

plt.plot(sorted\_second['Last Updated'], sorted\_second['Size'], label='Размер', color='blue', marker='s', markersize=6,

markeredgecolor='black', markerfacecolor='white', linewidth=2)

plt.plot(sorted\_second['Last Updated'], sorted\_second['Rating'], label='Рейтинг', color='green', marker='^',

markersize=6, markeredgecolor='black', markerfacecolor='white', linewidth=2)

plt.title('График')

plt.xlabel('Количество скачиваний')

plt.ylabel('Значение')

plt.grid(linewidth=2, color='mistyrose')

plt.legend()

plt.show()

Задание 6

Выполнить визуализацию многомерных данных, используя t-SNE. Необходимо использовать набор данных MNIST или fashion MNIST (можно использовать и другие готовые наборы данных, где можно наблюдать разделение объектов по кластерам). Рассмотреть результаты визуализации для разных значений перплексии.

Реализация

Для решения задачи был написан код (Листинг 4) Результат работы программы представлен на рисунке 6.

|  |
| --- |
|  |

Рисунок 6 – Результат работы программы

Листинг 4

perplexities = [5, 25, 50, 100]

# Создайте график с несколькими подграфиками для каждой перплексии

plt.figure(figsize=(12, 8))

for i, perplexity in enumerate(perplexities):

# Выполните t-SNE с текущим значением перплексии

tsne = TSNE(n\_components=2, perplexity=perplexity, random\_state=42)

X\_embedded = tsne.fit\_transform(data)

# Создайте подграфик для текущей перплексии

plt.subplot(2, 2, i + 1)

plt.scatter(X\_embedded[:, 0], X\_embedded[:, 1], c=dataset['Type'], cmap='viridis')

plt.title(f't-SNE с perplexity={perplexity}')

plt.colorbar()

plt.tight\_layout()

plt.show()

Задание 7

Выполнить визуализацию многомерных данных, используя UMAP с различными параметрами n\_neighbors и min\_dist. Рассчитать время работы алгоритма с помощью библиотеки time и сравнить его с временем работы t-SNE.

Реализация

Для решения задачи был написан код (Листинг 5, Листинг 6) Результат работы программы представлен на рисунке 7 – 16.

|  |
| --- |
|  |
| Рисунок 7 – Результат работы программы |
| Рисунок 8 – Результат работы программы |
| Рисунок 9 – Результат работы программы |
| Рисунок 10 – Результат работы программы    Рисунок 11 – Результат работы программы |
| Рисунок 12 – Результат работы программы |
| Рисунок 13 – Результат работы программы    Рисунок 14 – Результат работы программы |
| Рисунок 15 – Результат работы программы |

Листинг 5

n\_neighbors\_values = [5, 25, 50]

min\_dist\_values = [0.1, 0.5, 0.9]

# Сравнение времени работы UMAP и t-SNE

tsne\_time = []

umap\_time = []

for n\_neighbors in n\_neighbors\_values:

for min\_dist in min\_dist\_values:

# Выполнение t-SNE и измерение времени

start\_time = time.time()

tsne = TSNE(n\_components=2, perplexity=30, random\_state=42)

X\_embedded\_tsne = tsne.fit\_transform(data)

tsne\_time.append(time.time() - start\_time)

# Выполнение UMAP и измерение времени

start\_time = time.time()

umap = umap\_.UMAP(n\_neighbors=n\_neighbors, min\_dist=min\_dist, n\_components=2, random\_state=42)

X\_embedded\_umap = umap.fit\_transform(data)

umap\_time.append(time.time() - start\_time)

# Создайте график для текущих параметров UMAP

plt.figure(figsize=(8, 6))

plt.scatter(X\_embedded\_umap[:, 0], X\_embedded\_umap[:, 1], c=dataset['Type'], cmap='viridis')

plt.title(f'UMAP с n\_neighbors={n\_neighbors}, min\_dist={min\_dist}')

plt.colorbar()

plt.show()

# Вывод времени работы

for i in range(len(n\_neighbors\_values)):

for j in range(len(min\_dist\_values)):

idx = i \* len(min\_dist\_values) + j

print(

f'UMAP (n\_neighbors={n\_neighbors\_values[i]}, min\_dist={min\_dist\_values[j]}): {umap\_time[idx]:.2f} seconds')

print(f't-SNE: {tsne\_time[idx]:.2f} seconds')

print()

Листинг 6

UMAP (n\_neighbors=25, min\_dist=0.5): 17.30 seconds

t-SNE: 41.91 seconds

UMAP (n\_neighbors=25, min\_dist=0.9): 17.18 seconds

t-SNE: 45.63 seconds

UMAP (n\_neighbors=50, min\_dist=0.1): 22.78 seconds

t-SNE: 42.88 seconds

UMAP (n\_neighbors=50, min\_dist=0.5): 22.79 seconds

t-SNE: 42.16 seconds

UMAP (n\_neighbors=50, min\_dist=0.9): 22.46 seconds

t-SNE: 42.49 seconds