Введение

Поскольку индустрия игр очень закрыта - данные о продажах компаний являются зачастую тайной, поэтому, не стоит удивляться количеству пропусков в датасетах. Эти данные продаются на более продинутых сайтах, но это стоит достаточно приличную сумму денег, поэтому использовалось, что имелось.

Гипотезы:

1. VGChatz подходит для анализа только старых игр, Стим перехватил на себя роль хранилища игровой информации
2. Самая популярная консоль по данным с WGChartz будет PC
3. Между ценой игры в стиме и числом покупателей этой игры существует обратная связь

Далее по тексту будут делаться другие выводы, основанные на визуальной интерпретации графиков, но не вынесенные в эту графу. Также будут проверены статистически иные гипотезы, выявленные в процессе более детального анализа данных.

Предустановка библиотек

In [ ]:
```
%pip install selenium
%pip install undetected-chromedriver
%pip install requests
%pip install bs4
%pip install lxml
%pip install Jinja2
%pip install networkx
%pip install seaborn
```

Импорт всех необходимых библиотек

In [17]:
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from bs4 import BeautifulSoup
import numpy as np
import pandas as pd
import undetected_chromedriver as uc
```

```
import urllib.request
import matplotlib as mpl
import matplotlib.pyplot as plt
import networkx as nt
import seaborn as sea
from scipy import stats
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import root_mean_squared_error
```

Настройка веб-браузера и скрапинга

Поскольку чистый webdriver разпознается браузером автоматически, мы устанавливает расширение uc, которое скрывает нашу активность. В нем есть возможность использовать прокси, но, к счастью, это не потреболось. Далее мы используем обычный функционал Селениума, чтобы скрэпить данные с сайта WGChartz, это было достаточно сложно, так как, к сожалению, не все данные находились на главной странице-ссылке. Поэтому, чтобы получить доступ к внутренней информации, мы вынуждены скрэпить дополнительно ссылки на игры, а потом каждую ссылку перебирать. Этот процесс был очень долгим, по моим подсчетам для скрэпинга 1300 страниц с 50 играми потребовалось бы не менее 12 часов, поэтому я нашел файл старого года производства и скрэпил только нововышедшие игры - 2024, 2023, 2022. Проверить работу скрипта можно, установив значение страницы на желаемом вам уровне. Данные мы переводим в csv формат.

Стоит заметить, что код, закоменченный мною, это код, которым можно заменить часть, где я использую request и BS, так как получилось, что этот вариант работает быстрее. Но и закомменченный код яляется рабочим.

In [ ]:
```
page = 3
```

In [ ]:
```
### FROM https://codeby.net/threads/metody-obxoda-zaschity-ot-avtomatizirovann

optio = Options()
optio.add_argument("--headless")
driver = uc.Chrome(options= optio)
url = f"https://www.vgchartz.com/games/games.php?page={page}"
url_plus = '&order=ReleaseDate&ownership=Both&showtotalsales=1&shownasales=1&s

### END FROM
```

```
game_links = []
game = []
console = []
publisher = []
critic_score = []
user_score = []
copies_sold = []
total_sales = []
na_sales = []
eu_sales = []
jp_sales = []
other_sales = []
year = []
genre = []

for i in range(1, page + 1):
    url = f"https://www.vgchartz.com/games/games.php?page={i}"
    url_plus = '&order=ReleaseDate&ownership=Both&showtotalsales=1&shownasales
    driver.get(url+url_plus)
    while driver.execute_script("return document.readyState") != "complete":
        pass
    bunch = driver.find_element(By.ID, 'generalBody')
    game_links += [str(x.get_attribute("href")) for x in driver.find_elements(
    console += [x.get_attribute('alt') for x in bunch.find_elements(By.CSS_SEL
    data_table = bunch.find_elements(By.CSS_SELECTOR, 'td')
    lis = [x.text for x in data_table]
    game += lis[2::14]
    publisher += lis[4::14]
    critic_score += [float(x) if x != "N/A" else np.nan for x in lis[5::14]]
    user_score += [float(x) if x != "N/A" else np.nan for x in lis[6::14]]
    copies_sold += [float(x[:-1]) if x != "N/A" else np.nan for x in lis[7::14
    total_sales += [float(x[:-1]) if x != "N/A" else np.nan for x in lis[8::14
    na_sales += [float(x[:-1]) if x != "N/A" else np.nan for x in lis[9::14]]
    eu_sales += [float(x[:-1]) if x != "N/A" else np.nan for x in lis[10::14]]
    jp_sales += [float(x[:-1]) if x != "N/A" else np.nan for x in lis[11::14]]
    other_sales += [float(x[:-1]) if x != "N/A" else np.nan for x in lis[12::1
    year += [x for x in lis[13::14]]
driver.quit()




for item in game_links:
    site = urllib.request.urlopen(item).read()
    soup = BeautifulSoup(site, 'lxml')
    tags = soup.find("div", {"id": "gameGenInfoBox"}).find_all('h2')
    for tag in tags:
        if str(tag.text) == "Genre":
            gen = tag.find_next_sibling().text
            genre.append(gen)
        else:
            continue
```

```
#opt = Options()
#opt.add_argument("--headless")
#driver1 = webdriver.Chrome(options= opt)
#for item in game_links:
#    driver1.get(item)
#    box = driver1.find_element(By.ID, 'gameGenInfoBox')
#    box_items = box.find_elements(By.CSS_SELECTOR, 'p')
#    genre.append(box_items[1].text)
#driver.quit()


print(game_links,
genre,
game,
console,
publisher,
critic_score,
user_score,
copies_sold,
total_sales,
na_sales,
eu_sales,
jp_sales,
other_sales,
year, sep="\n")
```

Формирование датасета и csv файла

Как уже сообщалось, итогом скрапинга служит файл разрешения csv, заметим, что обработку данных мы провели во время процесса скрэпинга, приведя данные в нужный нам формат.

In [ ]:
```
for i in range(0, len(year)):
    if year[i] == "N/A":
        year[i] = np.nan
    else:
        if int(year[i][-2:]) <= 40:
            year[i] = int("20" + year[i][-2:])
        else:
            year[i] = int("19" + year[i][-2:])

columns = {
    'Game': game,
    'Console': console,
    'Publisher': publisher,
    'Genre': genre,
    'Critic_Score': critic_score,
    'User_Score': user_score,
```

```
    'Copies': copies_sold,
    'NA_Sales': na_sales,
    'EU_Sales': eu_sales,
    'JP_Sales': jp_sales,
    'Other_Sales': other_sales,
    'Total_Sales': total_sales,
    'Year': year
}

df = pd.DataFrame(columns)
print(df)
df.to_csv("scraped_data.csv", sep=",", encoding='utf-8', index=False)
```

Подготовка необходимых датасетов

Я нащел на сайте https://www.kaggle.com/datasets/gsimonx37/vgchartz, старые данные по VGChartz и в отдельном проекте их соединил. Я не включил этот процесс в этот проект, так как действия заключались лишь в перестановке колонок, их переименовывании, склеивании снизу и форматировании скачанного датасета тем же образом, что и я делал выше.

Датасет Steam был получен скрэпингом, с использованием API и включал в себя более 70000 игр, поэтому мной было принято решение просто скачать его и адаптировать под свои нужды.

Форматирование найденных датасетов

In [3]:
```
#FROM https://www.iditect.com/faq/python/pandas-how-to-read-csv-file-from-goog

url1 = 'https://drive.google.com/file/d/1XgMbAYTQy-dmvxqsW1joTJ6bYtvenHpE/view
url1_id = url1.split("/")[5]
urlf1 = f"https://drive.google.com/uc?id={url1_id}"
data_1 = pd.read_csv(urlf1)


# END FROM

data_1["Supported languages"] = [x.rstrip("']").lstrip("['").split("', '") if
data_1
```

Out[3]:

	AppID	Name	Release date	Estimated owners	Price	Supported languages	Windows	Ma
0	20200	Galactic Bowling	2008	10000.0	19.99	[English]	1	
1	655370	Train Bandit	2017	10000.0	0.99	[English, French, Italian, German, Spanish - S...	1	
2	1732930	Jolt Project	2021	10000.0	4.99	[English, Portuguese - Brazil]	1	
3	1355720	Henosisв„ў	2020	10000.0	5.99	[English, French, Italian, German, Spanish - S...	1	
4	1139950	Two Weeks in Painland	2020	10000.0	0.00	[English, Spanish - Spain]	1	
...	...	...	...	...	...	...	...	
74995	1301060	EreaDrone 2023	2023	0.0	19.99	[English, French]	1	
74996	2446340	Garestia	2023	0.0	0.00	[English]	1	
74997	2355170	Neon Nexus	2023	35000.0	1.69	[English]	1	
74998	2347960	D Life	2023	10000.0	5.94	[English, Japanese]	1	
74999	2400810	Cube Escape	2023	10000.0	3.59	[English]	1	

75000 rows × 13 columns

Мы должны взглянуть на датасет подробнее. Поскольку пропусков в графе Year относительно немного, мы можем избавиться от этих строчек. Остальные пропуски не представляется возможности закрыть, так как их очень много, и мы не можем предсказать хоть с какой-то точностью. Поскольку я совмещал данные, мы избавились от всех возможных дубликатов. Посмотрев на датасеты с помощью встроенных возможностей среды - кликнув на все переменные и выбрав датасеты, отсортировав, я увидел, что выбросов как таковых нет, у некоторых игр очень высокие продажи, что нормально для данной индустрии.

In [4]:
```
url2 = 'https://drive.google.com/file/d/1uMapnr_HBLYSl50xTyQ6PefkQ4oEVUQ8/view
url2_id = url2.split("/")[5]
urlf2 = f"https://drive.google.com/uc?id={url2_id}"
data_2 = pd.read_csv(urlf2)

data_2 = data_2.drop_duplicates(subset= ['Game', 'Year'])
print(data_2.isna().sum())
data_2['User_Score'].value_counts()
data_2.dropna(subset = ['Year'], inplace = True)
```

```
Game               0
Total_Sales     33457
Copies          42870
Publisher           0
Year             3421
Console             0
JP_Sales        41137
NA_Sales        38651
Other_Sales     36805
EU_Sales        38501
User_Score      46327
Critic_Score    42390
dtype: int64
```

Построение графических материалов для Steam

Стим - достаточно молодая платформа, поэтому можно ожидать, что в начале ее зарождения, на ней не было предсавлено достаточно много старых, но мало кому известных игр. Поэтому можно было ожидать, что количество релизов игр в стим будет небольшим в дальней перспективе. Однако по графику можно сделать более любопытный вывод:

Пик релизов игр находится в 2022 году, и мы можем объянить это тем, что за время ковидных ограничений, появилось огромное число инди-разработчиков, которые решались поробовать себя в производстве игр. Именно это с большой вероятностью привело к такому стремительному росту в ковидный период.

In [318…
```
year_count = data_1.groupby('Release date')['Release date'].agg('count')
graph_1 = year_count.plot(title= "Количество выпускаемых игр по годам на платф
plt.xlabel('Год')
plt.ylabel('Количество игр')
```

Out[318…
```
Text(0, 0.5, 'Количество игр')
```

Создадим корреляционную матрицу на основе полученных данных. Отдельно стоит отметить, что мы добавили новую колонку к датасету, представляюшую собой сумму, которая описывает мультиоперационность игры, ее совместимость с различными ОС. По данным мы видим, что все игры, выложенные в Стиме, совместимы, с Windows - основная платформа для игровых актиснотей. Нашей гипотезой было то, что чем больше игра адаптирована к разным платформам, тем больше она должна стоить, так как разработка кода для Линукса и Мака достаточно сложная задача. Однако наша гипотеза не подтвердилась, коэффициент корреляции близок к нулю.

In [219…

```
data_1_temp = data_1
data_1_temp["system_sum"] = data_1["Windows"] + data_1["Mac"] + data_1["Linux"
corr_1 = data_1_temp.iloc[:, [2,3,4,9,10,11,12,13]].corr()
corr_1.style.background_gradient(cmap='RdYlGn')
```

Out[219…

	Release date	Estimated owners	Price	Positive	Negative	Recomr
Release date	1.000000	-0.113035	-0.023543	-0.054340	-0.037885	
Estimated owners	-0.113035	1.000000	0.040631	0.668665	0.642202	
Price	-0.023543	0.040631	1.000000	0.033905	0.027924	
Positive	-0.054340	0.668665	0.033905	1.000000	0.784565	
Negative	-0.037885	0.642202	0.027924	0.784565	1.000000	
Recommendations	-0.054339	0.536811	0.049506	0.896506	0.793219	
Average playtime forever	-0.082182	0.234382	0.073531	0.205189	0.194797	
system_sum	-0.192623	0.047410	0.021188	0.030919	0.013352	

Никакой значимой зависимости на данных, кроме очевидной, мной не было замечено. Это может быть связано с тем, что мы рассматриваем огромный пласт инди-игр, которые портят статистику, поэтому огранчим данные числом тех, кто игру получил и проведем те же манипуляции (рассматриваем только очень популярные игры).

In [223…

```
corr_1_1 = data_1_temp[data_1_temp["Estimated owners"] > 35000000].iloc[:, [2,
corr_1_1.style.background_gradient(cmap='RdYlGn')
```

Out[223…

	Release date	Estimated owners	Price	Positive	Negative	Recomr
Release date	1.000000	-0.105644	0.739510	-0.301517	0.117647	
Estimated owners	-0.105644	1.000000	-0.250000	-0.099782	-0.168954	
Price	0.739510	-0.250000	1.000000	-0.431432	-0.492496	
Positive	-0.301517	-0.099782	-0.431432	1.000000	0.581866	
Negative	0.117647	-0.168954	-0.492496	0.581866	1.000000	
Recommendations	-0.024462	-0.390341	-0.322075	0.891383	0.810460	
Average playtime forever	-0.182779	0.687232	-0.608306	0.566220	0.524742	
system_sum	-0.862582	0.408248	-0.612372	0.499781	-0.154405	

Мною были замечены следующие закономерности:

1. Чем больше дата релиза, тем выше цены. Это можно объяснить инфляцией, а также тем, что популярные игровые производители

затрачивают большие объемы средств на разработку более реалистичной графики и прочего.

2. Чем больше дата релиза, тем меньше мультиоперационность. Это можно объяснить огромным наплывом инди-игр, которые зачастую разработываются только для ОС Windows.

3. Чем больше цена, тем меньше среднее время игры. Это тоже объясняется достаточно просто. В Стиме можно вернуть деньги за игру, если проиграл в нее меньше установленного времени, поэтому многие люди просто бросают игры на этом временном промежутке, чтобы не тратить большие средства.

Рассмотрим игры с самым высоким количеством позитивных оценок:

In [226…
```
corr_1_2 = data_1_temp[data_1_temp["Positive"] >= 70000].iloc[:, [2,3,4,9,10,1
corr_1_2.style.background_gradient(cmap='RdYlGn')
```

Out[226…

	Release date	Estimated owners	Price	Positive	Negative	Recomr
Release date	1.000000	-0.132185	0.289080	-0.156008	-0.023689	
Estimated owners	-0.132185	1.000000	-0.305513	0.582120	0.601028	
Price	0.289080	-0.305513	1.000000	-0.175844	-0.170555	
Positive	-0.156008	0.582120	-0.175844	1.000000	0.755240	
Negative	-0.023689	0.601028	-0.170555	0.755240	1.000000	
Recommendations	-0.112953	0.372540	-0.085648	0.867400	0.768274	
Average playtime forever	-0.161475	0.702693	-0.055349	0.644049	0.658296	
system_sum	-0.465681	0.071608	-0.151019	0.106618	-0.053245	

Каких-либо серьзных выводов сделать нельзя, так как корреляции достаточно низкие.

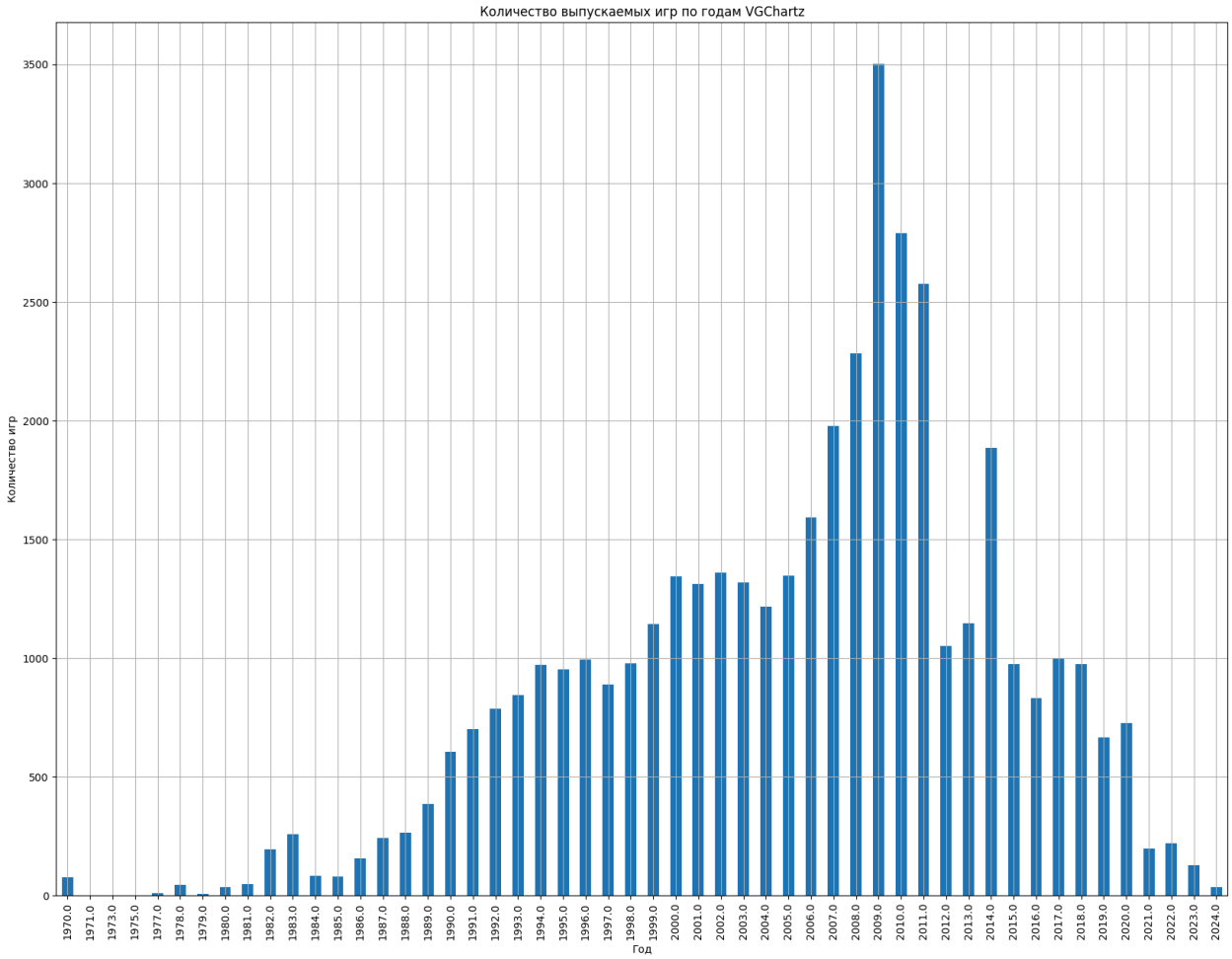
Построение графичсеких материалов для VGchartz

Рассмотрим тот же график, что и для Стима (годы релиза):

In [409…
```
year_count2 = data_2.groupby('Year')['Year'].agg('count')
```

```
plt.figure(figsize = (20, 15))
graph_2 = year_count2.plot(title= "Количество выпускаемых игр по годам VGChart
plt.xlabel('Год')
plt.ylabel('Количество игр')
```

Out[409… Text(0, 0.5, 'Количество игр')

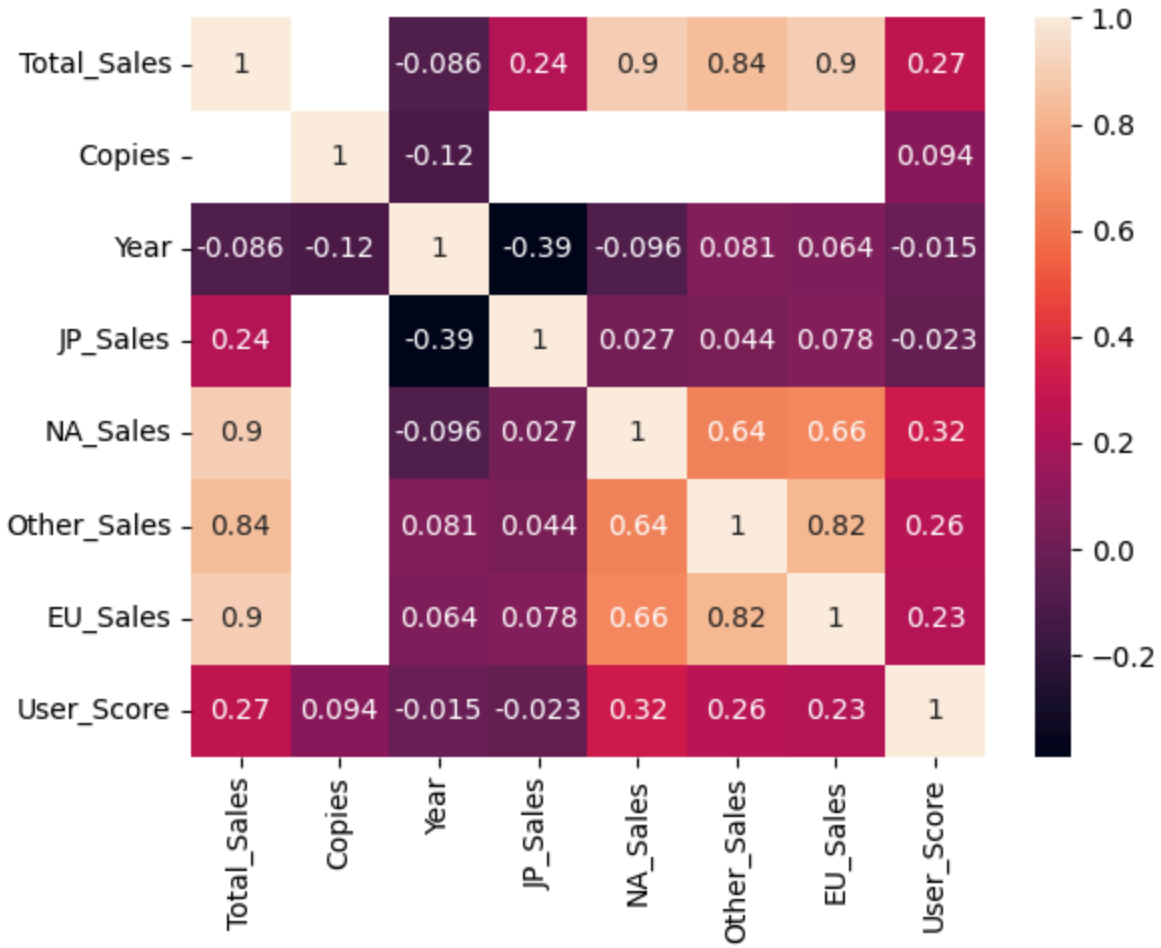


Как мы видим, сайт VGChartz начал сдавать свои позиции как одного из лучших информационных игровых сайтов, еще в 2021 - пик развития Стима, как мы и предполагали во введении. Также мы пожем константировать, что пиком развития игровой индустрии был 2009 год. Другой пик, который мы получили на данных стима был 2022. Между пиками прошло 13 лет, за это время индустрия игр серьезно изменилась, перейдя от пиксельных игр к высокотехнологичеким.

Попробуем найти значимые корреляции в данных:

In [329…
```
sea.heatmap(data_2.iloc[:,[1,2,4,6,7,8,9,10]].corr(), annot= True)
```

Out[329… <Axes: >

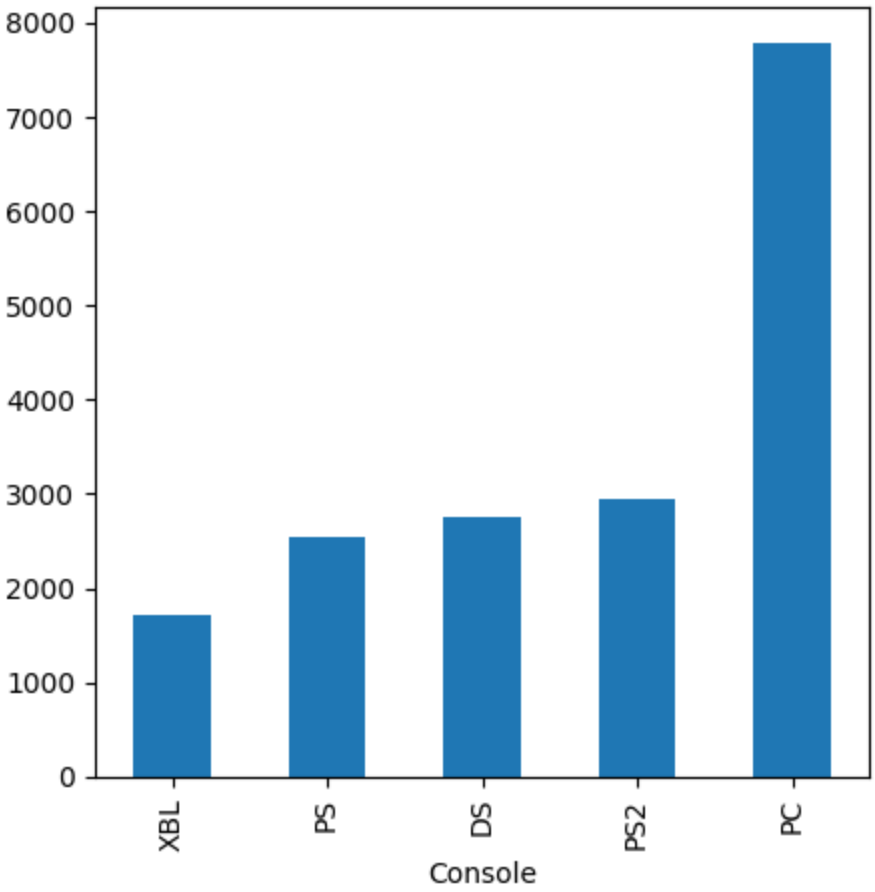


Кроме очевидной положительной корреляции между продажами, никакой другой значимой корреляции найти не удалось.

В этом блоке мы нашли 5 самых популярных среди производителей игр платформ в промежутке от 90 до 20ых годов (активное время работы сайта VGChartz). Не удивительно, что большинство игр разрабатывались под ПК, так как консоли контролировались большими корпорациями, а остальную долю рынка занимали более мелкие компании, которые не могли себе позволить разработку консолей.

In [406…
```
plt.figure(figsize = (5, 5))
best_console = list(data_2.groupby("Console")["Console"].agg("count").sort_val
data_2.groupby("Console")["Console"].agg("count").sort_values().tail().plot(ki
```

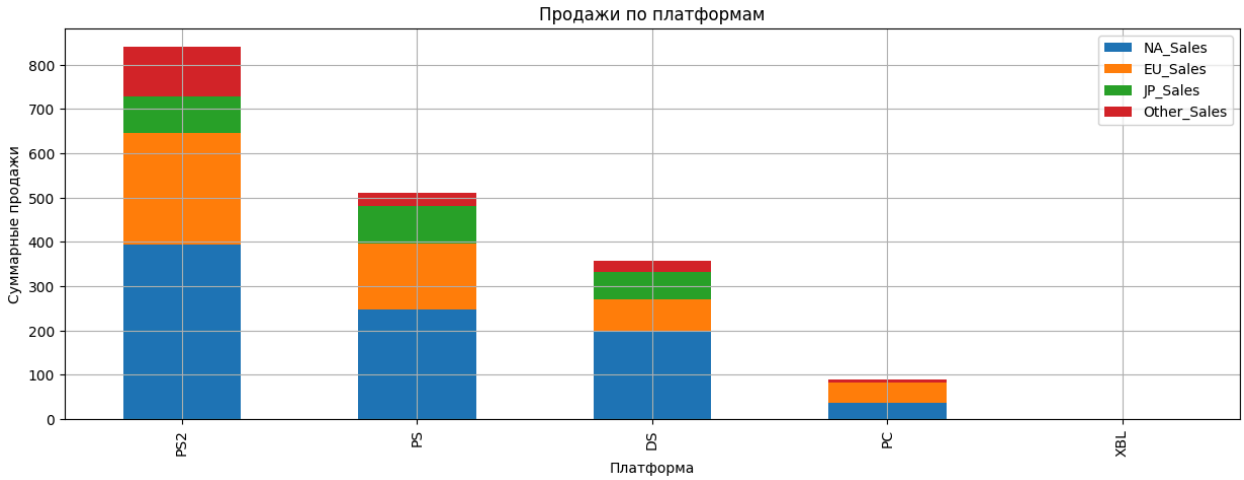
Out[406…
```
<Axes: xlabel='Console'>
```



Однако взгялянем на продажи - как мы видим, игры на консолях продавались больше всего, так как компьютеры были не у всех, позволить себе его мог не каждый, а консоли были очень удобными, с качественными инрыми от опытных разработчиков. Неудивительно, что лидером по продажам дого времени была именно PlayStation2 - самая популярная консоль в мире.

In [408…

```
data_cut = data_2.query("Console in @best_console")
plot = data_cut.groupby("Console")[['NA_Sales','EU_Sales','JP_Sales','Other_Sa
plot.set_xlabel('Платформа')
plot.set_ylabel('Суммарные продажи')
plt.show()
```



Отдельно можно отметить, что продажи выше всего были в Америке - стране производителе большинства игр и консолей.

Построение графа

В этой теме мы построим двудольный граф, который связывает игру с языками, на которые она была переведена, таким образом, мы узнаем наиболее популярные языки в игровом мире. В коллекции по ключу (id) мы присваем список языков, доступных для этой игры. Также, формируем список уникльных языков и избавляемся от небольших ошибках в данных (из-за разной структуры ннкоторых стимовских игр). Далее, используя функции из документации, мы разбиваем вершины на две категории и строим связи между ними. Поскольку данных у нас огромное количество, мы рассмотрели slice исходного датафрейма. Можно менять в первой строке кода.

In [298…

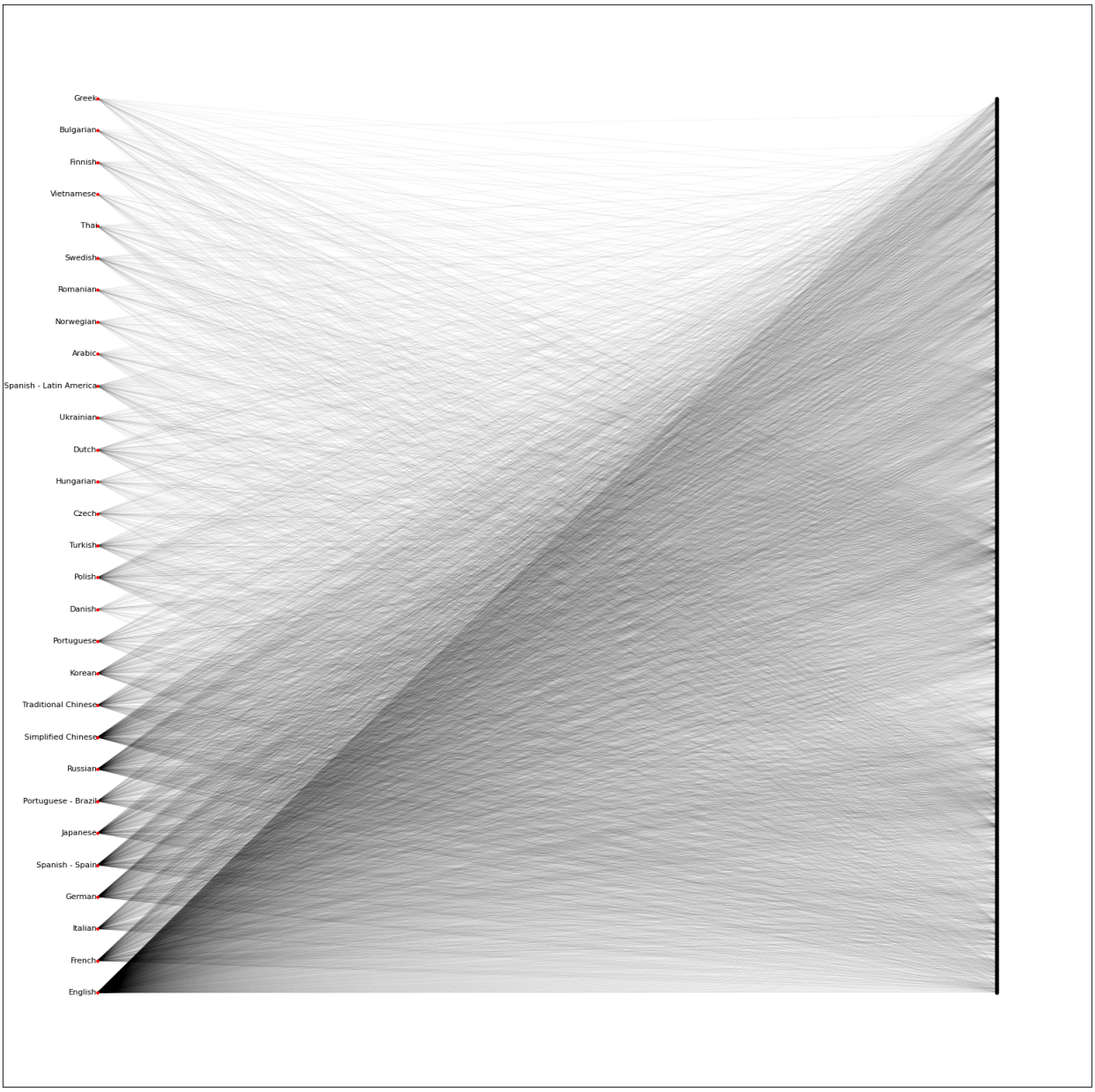
```
slice_of_data = data_1.loc[0:2000,:]

G = nt.Graph()
dict_lang = {}
unique_lang = []
ban_list = ["\\", "&", "(", ")", "/", ",", ";", "["]
id_count = 0
for item in slice_of_data["Supported languages"]:
    if type(item) is not float:
        dict_lang[id_count] = item
    else:
        continue
    for i in range(0, len(item)):
        if item[i] not in unique_lang and ban_list[1] not in item[i] and ban_l
            unique_lang.append(item[i])
        else:
            continue
    id_count += 1
```

```
G.add_nodes_from(list(dict_lang.keys()), bipartite= 1)
G.add_nodes_from(unique_lang, bipartite= 0)
for key,value in dict_lang.items():
    for item in unique_lang:
        if item in value:
            G.add_edge(item, key)
        else:
            continue
```

Визуализируем граф с помощью функций из matplotlib и networkx

In [410…

```
nodes_0 = [x for x in G.nodes if G.nodes[x]['bipartite'] == 0]
plt.figure(figsize = (20, 20))
layout = nt.bipartite_layout(G, nodes_0)
nt.draw_networkx_nodes(G, layout, nodelist= list(dict_lang.keys()), node_size
nt.draw_networkx_nodes(G, layout, nodelist= unique_lang, node_size = 5, node_c
nt.draw_networkx_labels(G, layout, labels= {n: n for n in unique_lang}, font_s
nt.draw_networkx_edges(G, layout, node_size= 2, alpha=0.5, width= 0.05)
plt.show()
```



Закономерно получили, что язык гейм индустрии - английский, чуть менее популярны европейские языки (Русский достаточно сильно поулярен среди других), также популярны японский и китайский - страна-производитель игр и страна с одним из самых больших населений в мире.

Проверка гипотез

In [ ]:
```
data_1['price_num'] = pd.to_numeric(data_1['Price'], errors='coerce')
data_1['owners_num'] = pd.to_numeric(data_1['Estimated owners'], errors='coerc

data_1['positive'] = pd.to_numeric(data_1['Positive'], errors='coerce').fillna
data_1['negative'] = pd.to_numeric(data_1['Negative'], errors='coerce').fillna
data_1['positive_rate'] = data_1['positive'] / (data_1['positive'] + data_1['r
```

```
data_1['log_owners'] = np.log1p(data_1['owners_num'])
```

Гипотеза 1: Бесплатные против платных игр

Мы проверили, отличается ли число владельцев между бесплатными и платными играми с помощью критерия Манна–Уитни.
Результаты показали статистически значимое различие (p-value < 0.001).

В целом, результат ожидаем: в среднем платные игры имеют больше владельцев, чем бесплатные. Это может объясняться тем, что в выборке бесплатных игр много малопопулярных проектов, тогда как среди платных больше крупных тайтлов. За качественные проекты производители обычно требуют денег.

In [ ]:
```
free = data_1.loc[data_1['price_num'] == 0, 'log_owners'].dropna()
paid = data_1.loc[data_1['price_num'] > 0, 'log_owners'].dropna()

# Непараметрический тест Манна-Уитни
stat, p = stats.mannwhitneyu(free, paid, alternative='two-sided')

print(f"Mann-Whitney test: U={stat}, p-value={p:.4f}")
print(f"Среднее log(owners) Free: {free.mean():.2f}, Paid: {paid.mean():.2f}")
```

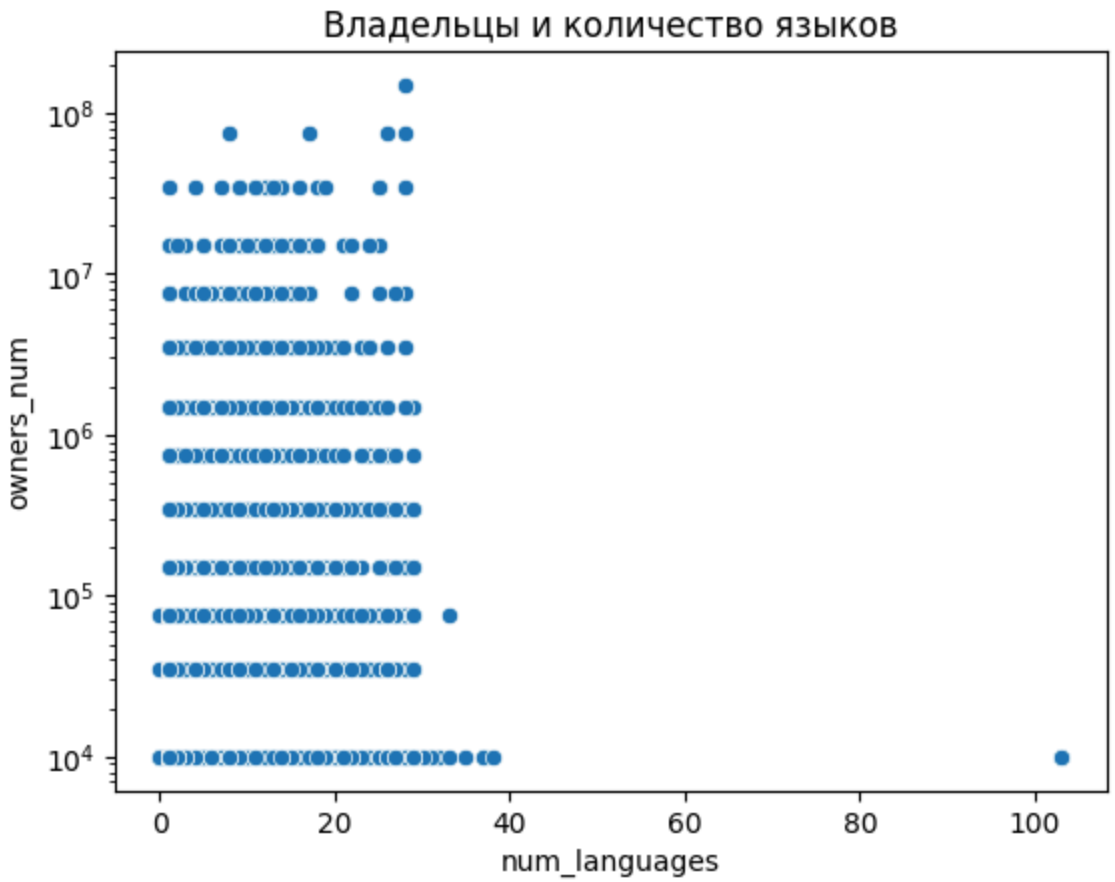
```
Mann-Whitney test: U=270787777.0, p-value=0.0000
Среднее log(owners) Free: 5.24, Paid: 9.45
```

Гипотеза 2: Влияние количества языков на популярность

Корреляция Спирмена показала положительную зависимость (0.294, p < 0.001).
Чем больше языков поддерживает игра, тем больше у неё владельцев.
Количество языков положительно ассоциируется с популярностью. Впрочем, весьма вероятна обратная ассоциированность, что популярность порождает большую локализацию. Локализация повышает потенциальный охват аудитории.

In [39]:
```
data_1['num_languages'] = data_1['Supported languages'].astype(str).apply(lamb

corr = stats.spearmanr(data_1['num_languages'], data_1['owners_num'], nan_poli
print(f"Корреляция Спирмена num_languages vs owners: {corr.correlation:.3f}, p

sea.scatterplot(x='num_languages', y='owners_num', data=data_1)
plt.yscale('log')
plt.title("Владельцы и количество языков")
plt.show()
```

```
Корреляция Спирмена num_languages vs owners: 0.294, p-value=0.0000
```

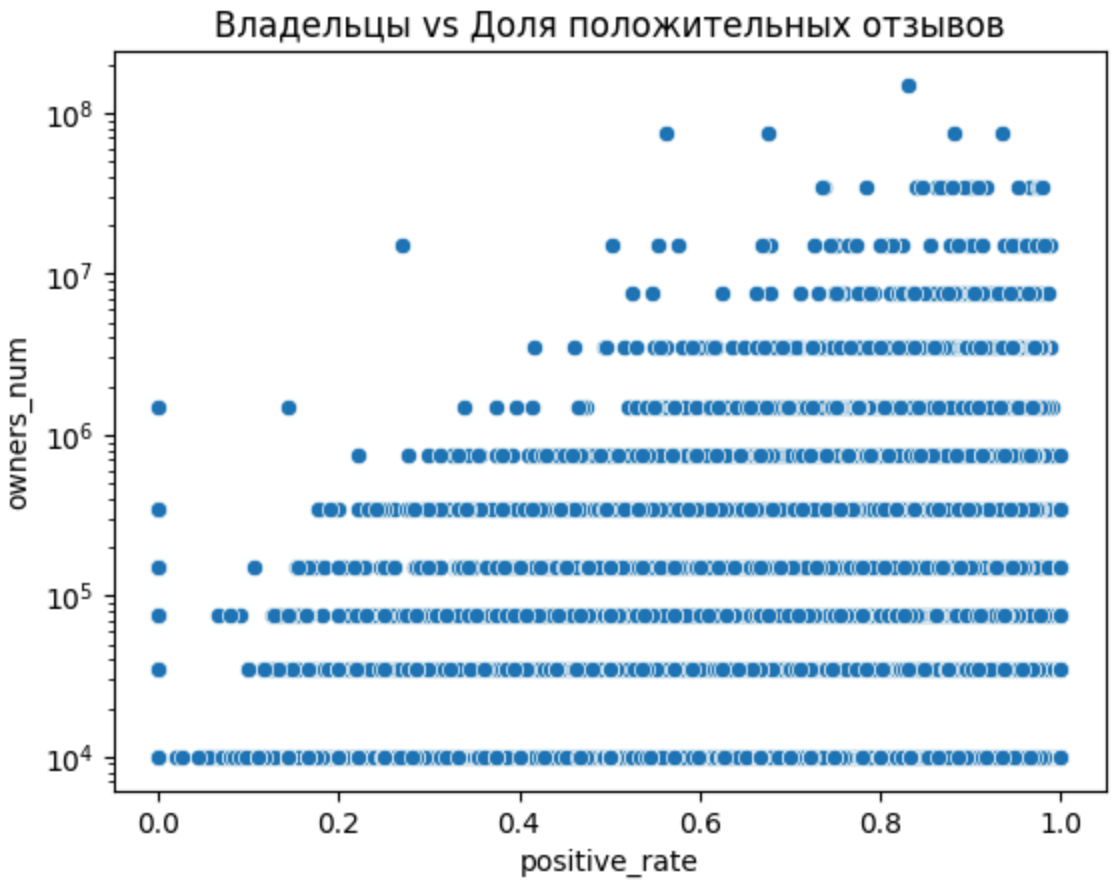


Гипотеза 3: Влияние отзывов на число владельцев

Корреляция Спирмена оказалась близка к нулю (–0.029).
Хотя результат статистически значим из-за большого объёма выборки, практически зависимость отсутствует.
Это значит, что количество владельцев напрямую не определяется долей положительных отзывов.

In [ ]:
```
corr = stats.spearmanr(data_1['positive_rate'], data_1['owners_num'], nan_poli
print(f"Корреляция positive_rate vs owners: {corr.correlation:.3f}, p-value={c

sea.scatterplot(x='positive_rate', y='owners_num', data=data_1)
plt.yscale('log')
plt.title("Владельцы и доля рельных отзывов")
plt.show()
```

```
Корреляция positive_rate vs owners: -0.029, p-value=0.0000
```



Линейная регрессия

Модель объясняет около 6% вариации в данных (R^2 = 0.058).
Факторы влияют так:

- Количество языков (coef = +0.0492) положительно связано с числом владельцев.
- Доля положительных отзывов (coef = +0.2548) также положительно влияет.
- Цена неожиданно имеет положительный эффект (coef = +0.0116), что можно объяснить тем, что более дорогие игры часто являются крупными проектами с широкой аудиторией.

Вывод: линейная модель фиксирует значимые факторы, но слабая объяснительная сила указывает на то, что нужно большее количество фичей для объяснения таргета или более сложная нелинейная модель.

In [26]:
```
ml_data = data_1[['log_owners','price_num','num_languages','positive_rate']].c

# Простая линейная регрессия
model = smf.ols('log_owners ~ price_num + num_languages + positive_rate', data
print(model.summary())
```

```python
y_true = ml_data['log_owners']
y_pred = model.fittedvalues  # предсказания из OLS

rmse_ols = root_mean_squared_error(y_true, y_pred)
print(f"OLS RMSE:, {rmse_ols:.3f}")
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             log_owners   R-squared:                       0.058
Model:                            OLS   Adj. R-squared:                  0.058
Method:                 Least Squares   F-statistic:                     1201.
Date:                Sun, 24 Aug 2025   Prob (F-statistic):               0.00
Time:                        18:24:28   Log-Likelihood:                -94569.
No. Observations:               58851   AIC:                         1.891e+05
Df Residuals:                   58847   BIC:                         1.892e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      9.4294      0.016    572.937      0.000       9.397       9.462
price_num      0.0116      0.000     25.281      0.000       0.011       0.012
num_languages  0.0492      0.001     48.564      0.000       0.047       0.051
positive_rate  0.2548      0.020     12.490      0.000       0.215       0.295
==============================================================================
Omnibus:                    20612.135   Durbin-Watson:                   1.973
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            67671.493
Skew:                           1.811   Prob(JB):                         0.00
Kurtosis:                       6.804   Cond. No.                         71.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
OLS RMSE:, 1.207
```

Random Forest

Нелинейная модель показала слегка лучшее качество (RMSE = 1.117). Это означает, что проблема лежит не в функциональной форме модели, а в недостатке объясняющих переменных. Важность факторов:

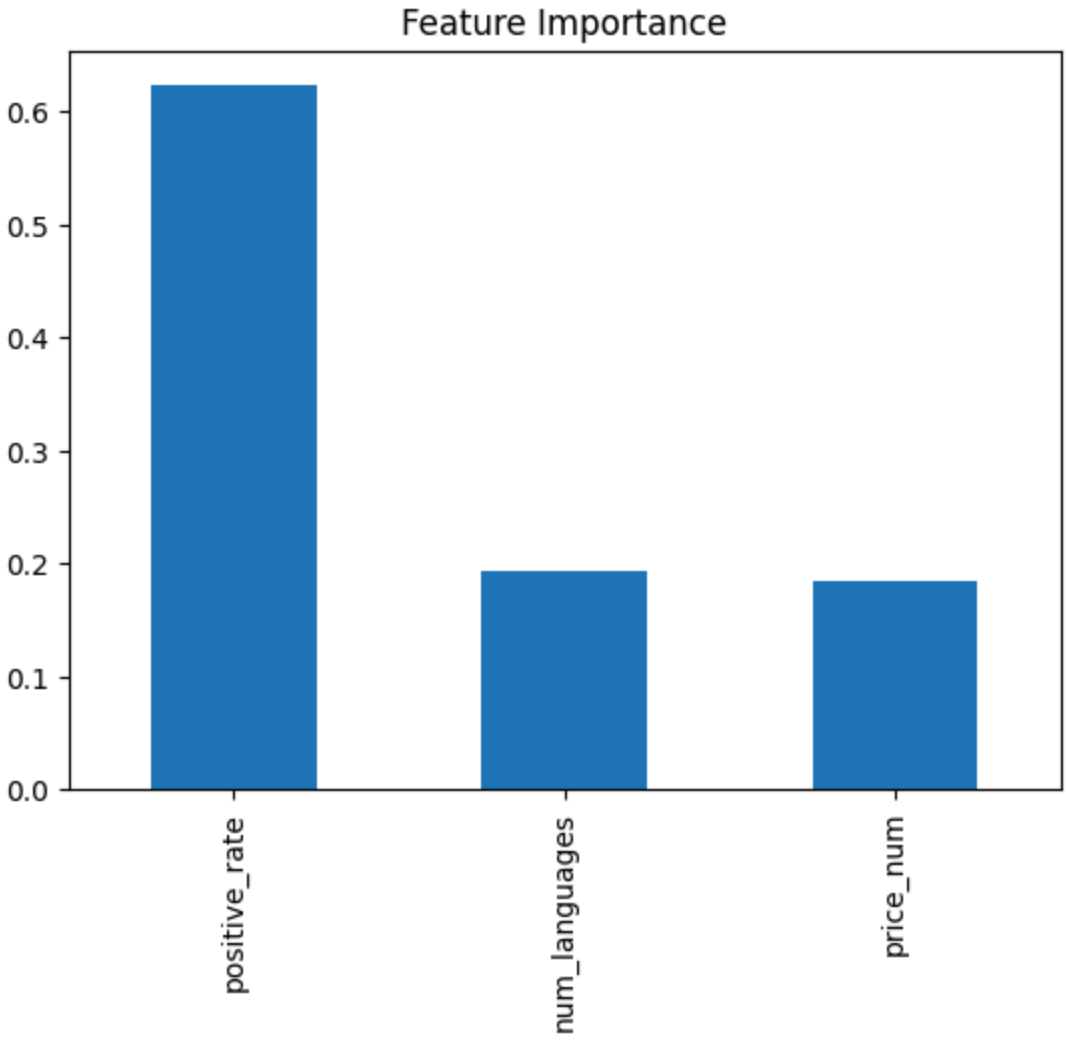
1. Положительные отзывы - главный предиктор числа владельцев.
2. Количество языков также важно.

3. Цена имеет наименьшее влияние.

Вывод: для прогнозирования популярности игр отзывы играют ключевую роль, тогда как локализация и цена играют меньшую роль.

In [40]:
```
X = ml_data[['price_num','num_languages','positive_rate']]
y = ml_data['log_owners']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

rf = RandomForestRegressor(n_estimators=100, random_state=1)
rf.fit(X_train, y_train)

preds = rf.predict(X_test)
rmse = root_mean_squared_error(y_test, preds)
print(f"RandomForest RMSE: {rmse:.3f}")

feat_imp = pd.Series(rf.feature_importances_, index=X.columns).sort_values(asc
feat_imp.plot(kind='bar', title='Feature Importance')
plt.show()
```

```
RandomForest RMSE: 1.117
```



Выводы

Касаемо гипотез, поставленных в этом проекте:

1. VGChatz подходит для анализа только старых игр, Стим перехватил на себя роль хранилища игровой информации
2. Самая популярная консоль по данным с WGChartz будет PC
3. Между ценой игры в стиме и числом покупателей этой игры существует обратная связь
4. Платные игры в среднем имеют больше владельцев, чем бесплатные
5. Локализация положительно ассоциируется с успеом игры
6. Отзывы напрямую почти не влияют на число владельцев. Но в моделях ML положительная роль отзывов всё же видна - они важнее цены и локализации.

7. Цена имеет двойственный эффект. В корреляции её связь с владельцами отрицательная, но в линейной модели положительная (возможно, дорогие игры чаще являются крупными хитами).

1 - потвердилась с опорой на соответствующие графики. Стим стал информационной геймингговой базой нашего времени. 2 - гипотеза подтвердилась, но с оговоркой, что популярна для разработчиков. Для обычных геймеров того времени самая популярная консоль - PS2, потом PS, что показали нам данные о продажах игр на этой консоли. 3 - к большому удивлению, гипотеза была опровегнута отсутсвующей корреляцией. Можно принять альтернативную гипотезу, что для покупателей в Стиме цена не является определяющим фактором. Найти определяющий фактор не удалось, посольку высоких корреляций по цене не было. Остальные гипотезы быили описаны выше.

Игровая индустрия очень интересна для исследования, но поскольку данные о ней либо платные, либо рассеяны по интернету, для бюджетных проектов тяжело делать какие-либо серьзные выводы. Например, одной из интересных тем было бы рассмотреть появление Nintendo Swith на рынке портативных консолей, ее влияние на рынок, но эти данные найти чрезвычайно сложно.