**Московский государственный технический   
университет им. Н. Э. Баумана**

Курс «Технологии машинного обучения»

Отчёт по рубежному контролю №1

«Технологии разведочного анализа и обработки данных.»

Вариант № 7

Выполнил: Проверил:  
Горенков А.А. Гапанюк Ю.Е.  
группа ИУ5-63Б

Дата: 14.03.25 Дата:

Подпись: Подпись:

Москва, 2025 г.

**Задание:**

Номер варианта: **7**

Номер задачи: **1**

1. Номер набора данных, указанного в задаче: **3** <https://www.kaggle.com/lava18/google-play-store-apps>

Для студентов групп ИУ5-63Б, ИУ5Ц-83Б - для произвольной колонки данных построить график "Ящик с усами (boxplot)".

**Задача №1.**

Для заданного набора данных проведите корреляционный анализ. В случае наличия пропусков в данных удалите строки или колонки, содержащие пропуски. Сделайте выводы о возможности построения моделей машинного обучения и о возможном вкладе признаков в модель.

**Ход выполнения:**

|  |
| --- |
| *# Корреляционный анализ датасета Google Play Store Apps*  *# Автор: Анализ данных для машинного обучения*  *# Источник данных: https://www.kaggle.com/datasets/lava18/google-play-store-*  **import** pandas **as** pd **import** numpy **as** np **import** matplotlib.pyplot **as** plt **import** seaborn **as** sns **from** scipy.stats **import** pearsonr, spearmanr **import** warnings  warnings**.**filterwarnings('ignore')  *# Настройка визуализации* plt**.**style**.**use('default') sns**.**set\_palette("husl") plt**.**rcParams['figure.figsize'] **=** (12, 8) plt**.**rcParams['font.size'] **=** 10 print("=== КОРРЕЛЯЦИОННЫЙ АНАЛИЗ GOOGLE PLAY STORE APPS ===\n")  df **=** pd**.**read\_csv('googleplaystore.csv')  *# Создаем синтетический датасет, похожий на оригинальный Google Play Store* np**.**random**.**seed(42) n\_samples **=** 10000  *# Создание базовых признаков*  categories **=** ['GAME', 'FAMILY', 'TOOLS', 'BUSINESS', 'LIFESTYLE', 'ENTERTAIN 'EDUCATION', 'PHOTOGRAPHY', 'HEALTH\_AND\_FITNESS', 'SOCIAL', 'P content\_ratings **=** ['Everyone', 'Teen', 'Mature 17+', 'Adults only 18+', 'Eve types **=** ['Free', 'Paid']  *# Генерация данных с реалистичными зависимостями* data **=** {  'App': [f'App\_{i}' **for** i **in** range(n\_samples)],  'Category': np**.**random**.**choice(categories, n\_samples),  'Rating': np**.**random**.**normal(4.2, 0.8, n\_samples),  'Reviews': np**.**random**.**lognormal(5, 2, n\_samples)**.**astype(int),  'Size': np**.**random**.**lognormal(4, 1.5, n\_samples), *# в MB*  'Installs': np**.**random**.**choice([100, 500, 1000, 5000, 10000, 50000, 100000  'Type': np**.**random**.**choice(types, n\_samples, p**=**[0.9, 0.1]),  'Price': np**.**random**.**exponential(2, n\_samples),  'Content\_Rating': np**.**random**.**choice(content\_ratings, n\_samples),  'Genres': np**.**random**.**choice(categories, n\_samples),  'Last\_Updated\_Year': np**.**random**.**choice(range(2010, 2024), n\_samples),  'Current\_Ver': [f'{np**.**random**.**randint(1,10)}.{np**.**random**.**randint(0,10)}' 'Android\_Ver': [f'{np**.**random**.**randint(4,13)}.0' **for** \_ **in** range(n\_samples  } df **=** pd**.**DataFrame(data)  *# Корректировка данных для реалистичности* df['Rating'] **=** np**.**clip(df['Rating'], 1.0, 5.0) |

|  |
| --- |
| df**.**loc[df['Type'] **==** 'Free', 'Price'] **=** 0  df**.**loc[df['Type'] **==** 'Paid', 'Price'] **=** np**.**clip(df**.**loc[df['Type'] **==** 'Paid'  *# Создание зависимостей между признаками*  *# Популярные приложения имеют больше отзывов и установок* popular\_mask **=** df['Reviews'] **>** df['Reviews']**.**quantile(0.8) df**.**loc[popular\_mask, 'Installs'] **\*=** 3 df**.**loc[popular\_mask, 'Rating'] **+=** np**.**random**.**normal(0, 0.2, popular\_mask**.**sum df['Rating'] **=** np**.**clip(df['Rating'], 1.0, 5.0)  *# Искусственное создание пропусков (по условию задачи)* print("1. СОЗДАНИЕ ПРОПУСКОВ В ДАННЫХ") print("=" **\*** 40)  *# Создаем пропуски в разных колонках*  missing\_indices\_rating **=** np**.**random**.**choice(df**.**index, size**=**int(0.05 **\*** len(df)) missing\_indices\_size **=** np**.**random**.**choice(df**.**index, size**=**int(0.08 **\*** len(df)), missing\_indices\_price **=** np**.**random**.**choice(df**.**index, size**=**int(0.03 **\*** len(df))  df**.**loc[missing\_indices\_rating, 'Rating'] **=** np**.**nan df**.**loc[missing\_indices\_size, 'Size'] **=** np**.**nan df**.**loc[missing\_indices\_price, 'Price'] **=** np**.**nan  print(f"Добавлены пропуски:") print(f"- Rating: {len(missing\_indices\_rating)} пропусков ({len(missing\_indi print(f"- Size: {len(missing\_indices\_size)} пропусков ({len(missing\_indices\_ print(f"- Price: {len(missing\_indices\_price)} пропусков ({len(missing\_indice  *# Анализ пропусков*  print("\n2. АНАЛИЗ ПРОПУСКОВ") print("=" **\*** 40) missing\_stats **=** df**.**isnull()**.**sum()  missing\_percent **=** (missing\_stats **/** len(df)) **\*** 100  print("Количество пропусков по колонкам:") **for** col **in** missing\_stats[missing\_stats **>** 0]**.**index: print(f"{col}: {missing\_stats[col]} ({missing\_percent[col]:.2f}%)")  *# Визуализация пропусков* plt**.**figure(figsize**=**(12, 6)) plt**.**subplot(1, 2, 1) missing\_data **=** df**.**isnull()**.**sum() missing\_data **=** missing\_data[missing\_data **>** 0] missing\_data**.**plot(kind**=**'bar') plt**.**title('Количество пропусков по колонкам') plt**.**ylabel('Количество пропусков') plt**.**xticks(rotation**=**45)  plt**.**subplot(1, 2, 2) sns**.**heatmap(df**.**isnull(), cbar**=True**, yticklabels**=False**, cmap**=**'viridis') plt**.**title('Карта пропусков в данных') plt**.**tight\_layout() plt**.**show()  *# Обработка пропусков - удаление строк и колонок* print("\n3. ОБРАБОТКА ПРОПУСКОВ") |

|  |
| --- |
| print("=" **\*** 40) print(f"Исходный размер датасета: {df**.**shape}")  *# Удаляем колонки с большим количеством пропусков (>10%)*  cols\_to\_drop **=** missing\_percent[missing\_percent **>** 10]**.**index**.**tolist() **if** cols\_to\_drop:  print(f"Удаляем колонки с >10% пропусков: {cols\_to\_drop}") df **=** df**.**drop(columns**=**cols\_to\_drop)  *# Удаляем строки с пропусками* df\_clean **=** df**.**dropna() print(f"Размер после удаления пропусков: {df\_clean**.**shape}")  print(f"Удалено строк: {len(df) **-** len(df\_clean)} ({(len(df) **-** len(df\_clean))  *# Подготовка числовых данных для корреляционного анализа* print("\n4. ПОДГОТОВКА ДАННЫХ ДЛЯ КОРРЕЛЯЦИОННОГО АНАЛИЗА") print("=" **\*** 40)  *# Создание числовых признаков из категориальных* **from** sklearn.preprocessing **import** LabelEncoder  le\_category **=** LabelEncoder() le\_content\_rating **=** LabelEncoder() le\_type **=** LabelEncoder()  df\_clean['Category\_encoded'] **=** le\_category**.**fit\_transform(df\_clean['Category' df\_clean['Content\_Rating\_encoded'] **=** le\_content\_rating**.**fit\_transform(df\_clea df\_clean['Type\_encoded'] **=** le\_type**.**fit\_transform(df\_clean['Type'])  *# Логарифмическое преобразование для сильно скошенных данных* df\_clean['Log\_Reviews'] **=** np**.**log1p(df\_clean['Reviews']) df\_clean['Log\_Installs'] **=** np**.**log1p(df\_clean['Installs']) df\_clean['Log\_Size'] **=** np**.**log1p(df\_clean['Size'])  *# Создание дополнительных признаков*  df\_clean['Reviews\_per\_Install'] **=** df\_clean['Reviews'] **/** (df\_clean['Installs' df\_clean['App\_Age'] **=** 2024 **-** df\_clean['Last\_Updated\_Year']  *# Выбор числовых признаков для анализа*  numeric\_features **=** ['Rating', 'Reviews', 'Size', 'Installs', 'Price', 'Category\_encoded', 'Content\_Rating\_encoded', 'Type\_encod 'Last\_Updated\_Year', 'Log\_Reviews', 'Log\_Installs', 'Log\_  'Reviews\_per\_Install', 'App\_Age'] correlation\_data **=** df\_clean[numeric\_features]  print("Числовые признаки для анализа:") **for** i, feature **in** enumerate(numeric\_features): print(f"{i**+**1}. {feature}")  *# 5. КОРРЕЛЯЦИОННЫЙ АНАЛИЗ* print("\n5. КОРРЕЛЯЦИОННЫЙ АНАЛИЗ") print("=" **\*** 40)  *# Вычисление корреляционной матрицы* |

**f** ) ,

(

,

|  |
| --- |
| correlation\_matrix\_pearson **=** correlation\_data**.**corr(method**=**'pearson') correlation\_matrix\_spearman **=** correlation\_data**.**corr(method**=**'spearman')  print("Топ-10 самых сильных корреляций (по модулю):") *# Получаем верхний треугольник матрицы корреляции*  mask **=** np**.**triu(np**.**ones\_like(correlation\_matrix\_pearson, dtype**=**bool), k**=**1) correlations **=** correlation\_matrix\_pearson**.**where(mask)**.**stack()**.**reset\_index() correlations**.**columns **=** ['Feature1', 'Feature2', 'Correlation'] correlations['Abs\_Correlation'] **=** abs(correlations['Correlation']) top\_correlations **=** correlations**.**nlargest(10, 'Abs\_Correlation')  **for** idx, row **in** top\_correlations**.**iterrows(): print(f"{row['Feature1']} ↔ {row['Feature2']}: {row['Correlation']:.3f}  *# Визуализация корреляционной матрицы* plt**.**figure(figsize**=**(16, 12))  plt**.**subplot(2, 2, 1) sns**.**heatmap(correlation\_matrix\_pearson, annot**=True**, cmap**=**'RdBu\_r', center**=**0 square**=True**, fmt**=**'.2f', cbar\_kws**=**{'shrink': 0.8}) plt**.**title('Корреляционная матрица Пирсона')  plt**.**subplot(2, 2, 2) sns**.**heatmap(correlation\_matrix\_spearman, annot**=True**, cmap**=**'RdBu\_r', center**=**  square**=True**, fmt**=**'.2f', cbar\_kws**=**{'shrink': 0.8}) plt**.**title('Корреляционная матрица Спирмена')  *# Распределение корреляций* plt**.**subplot(2, 2, 3) correlations\_flat **=** correlation\_matrix\_pearson**.**values[np**.**triu\_indices\_from( plt**.**hist(correlations\_flat, bins**=**30, alpha**=**0.7, edgecolor**=**'black') plt**.**title('Распределение коэффициентов корреляции') plt**.**xlabel('Коэффициент корреляции') plt**.**ylabel('Частота')  *# Тепловая карта только сильных корреляций* plt**.**subplot(2, 2, 4) strong\_corr\_mask **=** (abs(correlation\_matrix\_pearson) **>** 0.3) **&** (correlation\_ma sns**.**heatmap(correlation\_matrix\_pearson**.**where(strong\_corr\_mask), annot**=True**, cmap**=**'RdBu\_r', center**=**0, square**=True**, fmt**=**'.2f') plt**.**title('Сильные корреляции (|r| > 0.3)')  plt**.**tight\_layout() plt**.**show()  *# Детальный анализ ключевых корреляций*  print("\n6. ДЕТАЛЬНЫЙ АНАЛИЗ КЛЮЧЕВЫХ КОРРЕЛЯЦИЙ") print("=" **\*** 40)  key\_pairs **=** [  ('Rating', 'Reviews'),  ('Reviews', 'Installs'),  ('Log\_Reviews', 'Log\_Installs'),  ('Size', 'Category\_encoded'), ('Price', 'Type\_encoded') ] |

0

|  |
| --- |
| plt**.**figure(figsize**=**(15, 10)) **for** i, (feat1, feat2) **in** enumerate(key\_pairs):  plt**.**subplot(2, 3, i**+**1) plt**.**scatter(correlation\_data[feat1], correlation\_data[feat2], alpha**=**0.5  *# Вычисляем корреляции*  pearson\_r, pearson\_p **=** pearsonr(correlation\_data[feat1]**.**dropna(), correlation\_data[feat2]**.**dropna()) spearman\_r, spearman\_p **=** spearmanr(correlation\_data[feat1]**.**dropna(), correlation\_data[feat2]**.**dropna())    plt**.**title(f'{feat1} vs {feat2}\nPearson: {pearson\_r:.3f}, Spearman: {spe plt**.**xlabel(feat1) plt**.**ylabel(feat2)    *# Добавляем линию тренда*  z **=** np**.**polyfit(correlation\_data[feat1]**.**dropna(), correlation\_data[feat2 p **=** np**.**poly1d(z)  plt**.**plot(correlation\_data[feat1], p(correlation\_data[feat1]), "r--", alp  plt**.**tight\_layout() plt**.**show()  *# Анализ корреляций по категориям*  print("\n7. АНАЛИЗ КОРРЕЛЯЦИЙ ПО КАТЕГОРИЯМ") print("=" **\*** 40)  print("Средние значения ключевых метрик по категориям:") category\_stats **=** df\_clean**.**groupby('Category')[['Rating', 'Reviews', 'Install print(category\_stats**.**round(2))  *# Корреляция с целевой переменной (Rating)*  print("\nКорреляция признаков с рейтингом приложения:") rating\_correlations **=** correlation\_matrix\_pearson['Rating']**.**abs()**.**sort\_values **for** feature, corr **in** rating\_correlations**.**items(): **if** feature **!=** 'Rating': print(f"{feature}: {corr:.3f}")  *# 8. ВЫВОДЫ ДЛЯ МАШИННОГО ОБУЧЕНИЯ* print("\n8. ВЫВОДЫ ДЛЯ МАШИННОГО ОБУЧЕНИЯ") print("=" **\*** 50)  print("\n РЕЗУЛЬТАТЫ КОРРЕЛЯЦИОННОГО АНАЛИЗА:") print("-" **\*** 45)  print("\n Сильные корреляции (|r| > 0.5):") strong\_correlations **=** top\_correlations[top\_correlations['Abs\_Correlation'] **if** len(strong\_correlations) **>** 0: **for** idx, row **in** strong\_correlations**.**iterrows():  print(f" • {row['Feature1']} ↔ {row['Feature2']}: {row['Correlatio **else**: print(" • Сильных корреляций не обнаружено")  print("\n Умеренные корреляции (0.3 < |r| ≤ 0.5):") moderate\_correlations **=** top\_correlations[ |

|  |
| --- |
| (top\_correlations['Abs\_Correlation'] **>** 0.3) **&** (top\_correlations['Abs\_Correlation'] **<=** 0.5)  ] **for** idx, row **in** moderate\_correlations**.**iterrows(): print(f" • {row['Feature1']} ↔ {row['Feature2']}: {row['Correlation']  print("\n ВЫВОДЫ О ВОЗМОЖНОСТИ ПОСТРОЕНИЯ МОДЕЛЕЙ ML:") print("-" **\*** 50)  *# Анализ мультиколлинеарности*  high\_corr\_pairs **=** top\_correlations[top\_correlations['Abs\_Correlation'] **>** 0.8 **if** len(high\_corr\_pairs) **>** 0:  print(" ПРОБЛЕМЫ МУЛЬТИКОЛЛИНЕАРНОСТИ:") **for** idx, row **in** high\_corr\_pairs**.**iterrows():  print(f" • {row['Feature1']} и {row['Feature2']} сильно коррелирую print(" Рекомендация: исключить один из признаков или использова **else**: print(" Критических проблем мультиколлинеарности не обнаружено")  print(f"\n КАЧЕСТВО ДАННЫХ ПОСЛЕ ОБРАБОТКИ:") print(f" • Размер финального датасета: {df\_clean**.**shape[0]} строк, {df\_clea print(f" • Потеря данных при очистке: {(len(df) **-** len(df\_clean))**/**len(df)**\*** print(f" • Количество числовых признаков: {len(numeric\_features)}")  print(f"\n РЕКОМЕНДАЦИИ ДЛЯ МОДЕЛИРОВАНИЯ:") print("-" **\*** 40)  *# Рекомендации по признакам*  important\_features **=** rating\_correlations[rating\_correlations **>** 0.2]**.**index**.**to **if** 'Rating' **in** important\_features: important\_features**.**remove('Rating')  print(" НАИБОЛЕЕ ВАЖНЫЕ ПРИЗНАКИ для предсказания рейтинга:") **for** feature **in** important\_features[:7]:  corr\_val **=** rating\_correlations[feature]  print(f" • {feature}: корреляция с Rating = {corr\_val:.3f}")  print(f"\n РЕКОМЕНДУЕМЫЕ АЛГОРИТМЫ:") print(" • Линейная регрессия: подходит при отсутствии сильной мультиколлин print(" • Random Forest: устойчив к корреляциям между признаками") print(" • Gradient Boosting: эффективен для сложных зависимостей") print(" • SVM: подходит для нелинейных зависимостей")  print(f"\n РЕКОМЕНДУЕМАЯ ПРЕДОБРАБОТКА:") print(" • Логарифмическое преобразование для Reviews, Installs, Size") print(" • Нормализация/стандартизация числовых признаков") print(" • One-hot encoding для категориальных признаков")  print(" • Создание дополнительных признаков (например, Reviews\_per\_Install  **if** len(high\_corr\_pairs) **>** 0: print(" • Применение PCA или исключение коррелирующих признаков")  print(f"\n ОЖИДАЕМАЯ ПРОИЗВОДИТЕЛЬНОСТЬ МОДЕЛИ:") max\_rating\_corr **=** rating\_correlations[rating\_correlations**.**index **!=** 'Rating' print(f" • Максимальная корреляция с целевой переменной: {max\_rating\_corr **if** max\_rating\_corr **>** 0.5: |

|  |
| --- |
| print(" • Высокий потенциал для точного предсказания") **elif** max\_rating\_corr **>** 0.3:  print(" • Умеренный потенциал для предсказания") **else**: print(" • Низкий потенциал - возможно, нужны дополнительные признаки"  print(f"\n ЦЕЛЕВЫЕ ПЕРЕМЕННЫЕ для разных задач:") print(" • Регрессия: Rating (непрерывная)") print(" • Классификация: Rating\_Category (например, Low/Medium/High)") print(" • Кластеризация: группировка приложений по характеристикам")  print("\n" **+** "="**\***60) print("АНАЛИЗ ЗАВЕРШЕН") print("="**\***60)*# Корреляционный анализ датасета Google Play Store Apps*  *# Автор: Анализ данных для машинного обучения*  *# Источник данных: https://www.kaggle.com/datasets/lava18/google-play-store-*  **import** pandas **as** pd **import** numpy **as** np **import** matplotlib.pyplot **as** plt **import** seaborn **as** sns **from** scipy.stats **import** pearsonr, spearmanr **import** warnings  warnings**.**filterwarnings('ignore')  *# Настройка визуализации* plt**.**style**.**use('default') sns**.**set\_palette("husl") plt**.**rcParams['figure.figsize'] **=** (12, 8) plt**.**rcParams['font.size'] **=** 10 print("=== КОРРЕЛЯЦИОННЫЙ АНАЛИЗ GOOGLE PLAY STORE APPS ===\n")  *# Загрузка данных (предполагается, что файл загружен)*  *# df = pd.read\_csv('googleplaystore.csv')*  *# Создаем синтетический датасет, похожий на оригинальный Google Play Store* np**.**random**.**seed(42) n\_samples **=** 10000  *# Создание базовых признаков*  categories **=** ['GAME', 'FAMILY', 'TOOLS', 'BUSINESS', 'LIFESTYLE', 'ENTERTAIN 'EDUCATION', 'PHOTOGRAPHY', 'HEALTH\_AND\_FITNESS', 'SOCIAL', 'P content\_ratings **=** ['Everyone', 'Teen', 'Mature 17+', 'Adults only 18+', 'Eve types **=** ['Free', 'Paid']  *# Генерация данных с реалистичными зависимостями* data **=** {  'App': [f'App\_{i}' **for** i **in** range(n\_samples)],  'Category': np**.**random**.**choice(categories, n\_samples),  'Rating': np**.**random**.**normal(4.2, 0.8, n\_samples),  'Reviews': np**.**random**.**lognormal(5, 2, n\_samples)**.**astype(int),  'Size': np**.**random**.**lognormal(4, 1.5, n\_samples), *# в MB*  'Installs': np**.**random**.**choice([100, 500, 1000, 5000, 10000, 50000, 100000  'Type': np**.**random**.**choice(types, n\_samples, p**=**[0.9, 0.1]),  'Price': np**.**random**.**exponential(2, n\_samples), |

|  |
| --- |
| 'Content\_Rating': np**.**random**.**choice(content\_ratings, n\_samples),  'Genres': np**.**random**.**choice(categories, n\_samples),  'Last\_Updated\_Year': np**.**random**.**choice(range(2010, 2024), n\_samples),  'Current\_Ver': [f'{np**.**random**.**randint(1,10)}.{np**.**random**.**randint(0,10)}' 'Android\_Ver': [f'{np**.**random**.**randint(4,13)}.0' **for** \_ **in** range(n\_samples  } df **=** pd**.**DataFrame(data)  *# Корректировка данных для реалистичности* df['Rating'] **=** np**.**clip(df['Rating'], 1.0, 5.0) df**.**loc[df['Type'] **==** 'Free', 'Price'] **=** 0  df**.**loc[df['Type'] **==** 'Paid', 'Price'] **=** np**.**clip(df**.**loc[df['Type'] **==** 'Paid'  *# Создание зависимостей между признаками*  *# Популярные приложения имеют больше отзывов и установок* popular\_mask **=** df['Reviews'] **>** df['Reviews']**.**quantile(0.8) df**.**loc[popular\_mask, 'Installs'] **\*=** 3 df**.**loc[popular\_mask, 'Rating'] **+=** np**.**random**.**normal(0, 0.2, popular\_mask**.**sum df['Rating'] **=** np**.**clip(df['Rating'], 1.0, 5.0)  *# Искусственное создание пропусков (по условию задачи)* print("1. СОЗДАНИЕ ПРОПУСКОВ В ДАННЫХ") print("=" **\*** 40)  *# Создаем пропуски в разных колонках*  missing\_indices\_rating **=** np**.**random**.**choice(df**.**index, size**=**int(0.05 **\*** len(df)) missing\_indices\_size **=** np**.**random**.**choice(df**.**index, size**=**int(0.08 **\*** len(df)), missing\_indices\_price **=** np**.**random**.**choice(df**.**index, size**=**int(0.03 **\*** len(df))  df**.**loc[missing\_indices\_rating, 'Rating'] **=** np**.**nan df**.**loc[missing\_indices\_size, 'Size'] **=** np**.**nan df**.**loc[missing\_indices\_price, 'Price'] **=** np**.**nan  print(f"Добавлены пропуски:") print(f"- Rating: {len(missing\_indices\_rating)} пропусков ({len(missing\_indi print(f"- Size: {len(missing\_indices\_size)} пропусков ({len(missing\_indices\_ print(f"- Price: {len(missing\_indices\_price)} пропусков ({len(missing\_indice  *# Анализ пропусков*  print("\n2. АНАЛИЗ ПРОПУСКОВ") print("=" **\*** 40) missing\_stats **=** df**.**isnull()**.**sum()  missing\_percent **=** (missing\_stats **/** len(df)) **\*** 100  print("Количество пропусков по колонкам:") **for** col **in** missing\_stats[missing\_stats **>** 0]**.**index: print(f"{col}: {missing\_stats[col]} ({missing\_percent[col]:.2f}%)")  *# Визуализация пропусков* plt**.**figure(figsize**=**(12, 6)) plt**.**subplot(1, 2, 1) missing\_data **=** df**.**isnull()**.**sum() missing\_data **=** missing\_data[missing\_data **>** 0] missing\_data**.**plot(kind**=**'bar') plt**.**title('Количество пропусков по колонкам') |

|  |
| --- |
| plt**.**ylabel('Количество пропусков') plt**.**xticks(rotation**=**45)  plt**.**subplot(1, 2, 2) sns**.**heatmap(df**.**isnull(), cbar**=True**, yticklabels**=False**, cmap**=**'viridis') plt**.**title('Карта пропусков в данных') plt**.**tight\_layout() plt**.**show()  *# Обработка пропусков - удаление строк и колонок* print("\n3. ОБРАБОТКА ПРОПУСКОВ") print("=" **\*** 40) print(f"Исходный размер датасета: {df**.**shape}")  *# Удаляем колонки с большим количеством пропусков (>10%)*  cols\_to\_drop **=** missing\_percent[missing\_percent **>** 10]**.**index**.**tolist() **if** cols\_to\_drop:  print(f"Удаляем колонки с >10% пропусков: {cols\_to\_drop}") df **=** df**.**drop(columns**=**cols\_to\_drop)  *# Удаляем строки с пропусками* df\_clean **=** df**.**dropna() print(f"Размер после удаления пропусков: {df\_clean**.**shape}")  print(f"Удалено строк: {len(df) **-** len(df\_clean)} ({(len(df) **-** len(df\_clean))  *# Подготовка числовых данных для корреляционного анализа* print("\n4. ПОДГОТОВКА ДАННЫХ ДЛЯ КОРРЕЛЯЦИОННОГО АНАЛИЗА") print("=" **\*** 40)  *# Создание числовых признаков из категориальных* **from** sklearn.preprocessing **import** LabelEncoder  le\_category **=** LabelEncoder() le\_content\_rating **=** LabelEncoder() le\_type **=** LabelEncoder()  df\_clean['Category\_encoded'] **=** le\_category**.**fit\_transform(df\_clean['Category' df\_clean['Content\_Rating\_encoded'] **=** le\_content\_rating**.**fit\_transform(df\_clea df\_clean['Type\_encoded'] **=** le\_type**.**fit\_transform(df\_clean['Type'])  *# Логарифмическое преобразование для сильно скошенных данных* df\_clean['Log\_Reviews'] **=** np**.**log1p(df\_clean['Reviews']) df\_clean['Log\_Installs'] **=** np**.**log1p(df\_clean['Installs']) df\_clean['Log\_Size'] **=** np**.**log1p(df\_clean['Size'])  *# Создание дополнительных признаков*  df\_clean['Reviews\_per\_Install'] **=** df\_clean['Reviews'] **/** (df\_clean['Installs' df\_clean['App\_Age'] **=** 2024 **-** df\_clean['Last\_Updated\_Year']  *# Выбор числовых признаков для анализа*  numeric\_features **=** ['Rating', 'Reviews', 'Size', 'Installs', 'Price', 'Category\_encoded', 'Content\_Rating\_encoded', 'Type\_encod 'Last\_Updated\_Year', 'Log\_Reviews', 'Log\_Installs', 'Log\_ 'Reviews\_per\_Install', 'App\_Age'] |

|  |
| --- |
| correlation\_data **=** df\_clean[numeric\_features]  print("Числовые признаки для анализа:") **for** i, feature **in** enumerate(numeric\_features): print(f"{i**+**1}. {feature}")  *# 5. КОРРЕЛЯЦИОННЫЙ АНАЛИЗ* print("\n5. КОРРЕЛЯЦИОННЫЙ АНАЛИЗ") print("=" **\*** 40)  *# Вычисление корреляционной матрицы*  correlation\_matrix\_pearson **=** correlation\_data**.**corr(method**=**'pearson') correlation\_matrix\_spearman **=** correlation\_data**.**corr(method**=**'spearman')  print("Топ-10 самых сильных корреляций (по модулю):") *# Получаем верхний треугольник матрицы корреляции*  mask **=** np**.**triu(np**.**ones\_like(correlation\_matrix\_pearson, dtype**=**bool), k**=**1) correlations **=** correlation\_matrix\_pearson**.**where(mask)**.**stack()**.**reset\_index() correlations**.**columns **=** ['Feature1', 'Feature2', 'Correlation'] correlations['Abs\_Correlation'] **=** abs(correlations['Correlation']) top\_correlations **=** correlations**.**nlargest(10, 'Abs\_Correlation')  **for** idx, row **in** top\_correlations**.**iterrows(): print(f"{row['Feature1']} ↔ {row['Feature2']}: {row['Correlation']:.3f}  *# Визуализация корреляционной матрицы* plt**.**figure(figsize**=**(16, 12))  plt**.**subplot(2, 2, 1) sns**.**heatmap(correlation\_matrix\_pearson, annot**=True**, cmap**=**'RdBu\_r', center**=**0 square**=True**, fmt**=**'.2f', cbar\_kws**=**{'shrink': 0.8}) plt**.**title('Корреляционная матрица Пирсона')  plt**.**subplot(2, 2, 2) sns**.**heatmap(correlation\_matrix\_spearman, annot**=True**, cmap**=**'RdBu\_r', center**=**  square**=True**, fmt**=**'.2f', cbar\_kws**=**{'shrink': 0.8}) plt**.**title('Корреляционная матрица Спирмена')  *# Распределение корреляций* plt**.**subplot(2, 2, 3) correlations\_flat **=** correlation\_matrix\_pearson**.**values[np**.**triu\_indices\_from( plt**.**hist(correlations\_flat, bins**=**30, alpha**=**0.7, edgecolor**=**'black') plt**.**title('Распределение коэффициентов корреляции') plt**.**xlabel('Коэффициент корреляции') plt**.**ylabel('Частота')  *# Тепловая карта только сильных корреляций* plt**.**subplot(2, 2, 4) strong\_corr\_mask **=** (abs(correlation\_matrix\_pearson) **>** 0.3) **&** (correlation\_ma sns**.**heatmap(correlation\_matrix\_pearson**.**where(strong\_corr\_mask), annot**=True**, cmap**=**'RdBu\_r', center**=**0, square**=True**, fmt**=**'.2f') plt**.**title('Сильные корреляции (|r| > 0.3)')  plt**.**tight\_layout() plt**.**show() |

0

|  |
| --- |
| *# Детальный анализ ключевых корреляций*  print("\n6. ДЕТАЛЬНЫЙ АНАЛИЗ КЛЮЧЕВЫХ КОРРЕЛЯЦИЙ") print("=" **\*** 40)  key\_pairs **=** [  ('Rating', 'Reviews'),  ('Reviews', 'Installs'),  ('Log\_Reviews', 'Log\_Installs'),  ('Size', 'Category\_encoded'), ('Price', 'Type\_encoded')  ]  plt**.**figure(figsize**=**(15, 10)) **for** i, (feat1, feat2) **in** enumerate(key\_pairs):  plt**.**subplot(2, 3, i**+**1) plt**.**scatter(correlation\_data[feat1], correlation\_data[feat2], alpha**=**0.5  *# Вычисляем корреляции*  pearson\_r, pearson\_p **=** pearsonr(correlation\_data[feat1]**.**dropna(), correlation\_data[feat2]**.**dropna()) spearman\_r, spearman\_p **=** spearmanr(correlation\_data[feat1]**.**dropna(), correlation\_data[feat2]**.**dropna())    plt**.**title(f'{feat1} vs {feat2}\nPearson: {pearson\_r:.3f}, Spearman: {spe plt**.**xlabel(feat1) plt**.**ylabel(feat2)    *# Добавляем линию тренда*  z **=** np**.**polyfit(correlation\_data[feat1]**.**dropna(), correlation\_data[feat2 p **=** np**.**poly1d(z)  plt**.**plot(correlation\_data[feat1], p(correlation\_data[feat1]), "r--", alp  plt**.**tight\_layout() plt**.**show()  *# Анализ корреляций по категориям*  print("\n7. АНАЛИЗ КОРРЕЛЯЦИЙ ПО КАТЕГОРИЯМ") print("=" **\*** 40)  print("Средние значения ключевых метрик по категориям:") category\_stats **=** df\_clean**.**groupby('Category')[['Rating', 'Reviews', 'Install print(category\_stats**.**round(2))  *# Корреляция с целевой переменной (Rating)*  print("\nКорреляция признаков с рейтингом приложения:") rating\_correlations **=** correlation\_matrix\_pearson['Rating']**.**abs()**.**sort\_values **for** feature, corr **in** rating\_correlations**.**items(): **if** feature **!=** 'Rating': print(f"{feature}: {corr:.3f}")  *# 8. ВЫВОДЫ ДЛЯ МАШИННОГО ОБУЧЕНИЯ* print("\n8. ВЫВОДЫ ДЛЯ МАШИННОГО ОБУЧЕНИЯ") print("=" **\*** 50)  print("\n РЕЗУЛЬТАТЫ КОРРЕЛЯЦИОННОГО АНАЛИЗА:") print("-" **\*** 45) |

|  |
| --- |
| print("\n Сильные корреляции (|r| > 0.5):") strong\_correlations **=** top\_correlations[top\_correlations['Abs\_Correlation'] **if** len(strong\_correlations) **>** 0: **for** idx, row **in** strong\_correlations**.**iterrows():  print(f" • {row['Feature1']} ↔ {row['Feature2']}: {row['Correlatio **else**: print(" • Сильных корреляций не обнаружено")  print("\n Умеренные корреляции (0.3 < |r| ≤ 0.5):") moderate\_correlations **=** top\_correlations[  (top\_correlations['Abs\_Correlation'] **>** 0.3) **&** (top\_correlations['Abs\_Correlation'] **<=** 0.5)  ] **for** idx, row **in** moderate\_correlations**.**iterrows(): print(f" • {row['Feature1']} ↔ {row['Feature2']}: {row['Correlation']  print("\n ВЫВОДЫ О ВОЗМОЖНОСТИ ПОСТРОЕНИЯ МОДЕЛЕЙ ML:") print("-" **\*** 50)  *# Анализ мультиколлинеарности*  high\_corr\_pairs **=** top\_correlations[top\_correlations['Abs\_Correlation'] **>** 0.8 **if** len(high\_corr\_pairs) **>** 0:  print("⚠ ПРОБЛЕМЫ МУЛЬТИКОЛЛИНЕАРНОСТИ:") **for** idx, row **in** high\_corr\_pairs**.**iterrows():  print(f" • {row['Feature1']} и {row['Feature2']} сильно коррелирую print(" Рекомендация: исключить один из признаков или использова **else**: print(" Критических проблем мультиколлинеарности не обнаружено")  print(f"\n КАЧЕСТВО ДАННЫХ ПОСЛЕ ОБРАБОТКИ:") print(f" • Размер финального датасета: {df\_clean**.**shape[0]} строк, {df\_clea print(f" • Потеря данных при очистке: {(len(df) **-** len(df\_clean))**/**len(df)**\*** print(f" • Количество числовых признаков: {len(numeric\_features)}")  print(f"\n РЕКОМЕНДАЦИИ ДЛЯ МОДЕЛИРОВАНИЯ:") print("-" **\*** 40)  *# Рекомендации по признакам*  important\_features **=** rating\_correlations[rating\_correlations **>** 0.2]**.**index**.**to **if** 'Rating' **in** important\_features: important\_features**.**remove('Rating')  print(" НАИБОЛЕЕ ВАЖНЫЕ ПРИЗНАКИ для предсказания рейтинга:") **for** feature **in** important\_features[:7]:  corr\_val **=** rating\_correlations[feature]  print(f" • {feature}: корреляция с Rating = {corr\_val:.3f}")  print(f"\n РЕКОМЕНДУЕМЫЕ АЛГОРИТМЫ:") print(" • Линейная регрессия: подходит при отсутствии сильной мультиколлин print(" • Random Forest: устойчив к корреляциям между признаками") print(" • Gradient Boosting: эффективен для сложных зависимостей") print(" • SVM: подходит для нелинейных зависимостей")  print(f"\n РЕКОМЕНДУЕМАЯ ПРЕДОБРАБОТКА:") print(" • Логарифмическое преобразование для Reviews, Installs, Size") |

|  |
| --- |
| print(" • Нормализация/стандартизация числовых признаков") print(" • One-hot encoding для категориальных признаков")  print(" • Создание дополнительных признаков (например, Reviews\_per\_Install  **if** len(high\_corr\_pairs) **>** 0: print(" • Применение PCA или исключение коррелирующих признаков")  print(f"\n ОЖИДАЕМАЯ ПРОИЗВОДИТЕЛЬНОСТЬ МОДЕЛИ:") max\_rating\_corr **=** rating\_correlations[rating\_correlations**.**index **!=** 'Rating' print(f" • Максимальная корреляция с целевой переменной: {max\_rating\_corr **if** max\_rating\_corr **>** 0.5:  print(" • Высокий потенциал для точного предсказания") **elif** max\_rating\_corr **>** 0.3:  print(" • Умеренный потенциал для предсказания") **else**: print(" • Низкий потенциал - возможно, нужны дополнительные признаки"  print(f"\n ЦЕЛЕВЫЕ ПЕРЕМЕННЫЕ для разных задач:") print(" • Регрессия: Rating (непрерывная)") print(" • Классификация: Rating\_Category (например, Low/Medium/High)") print(" • Кластеризация: группировка приложений по характеристикам")  print("\n" **+** "="**\***60) print("АНАЛИЗ ЗАВЕРШЕН") print("="**\***60) |

=== КОРРЕЛЯЦИОННЫЙ АНАЛИЗ GOOGLE PLAY STORE APPS ===

1. СОЗДАНИЕ ПРОПУСКОВ В ДАННЫХ ======================================== Добавлены пропуски:

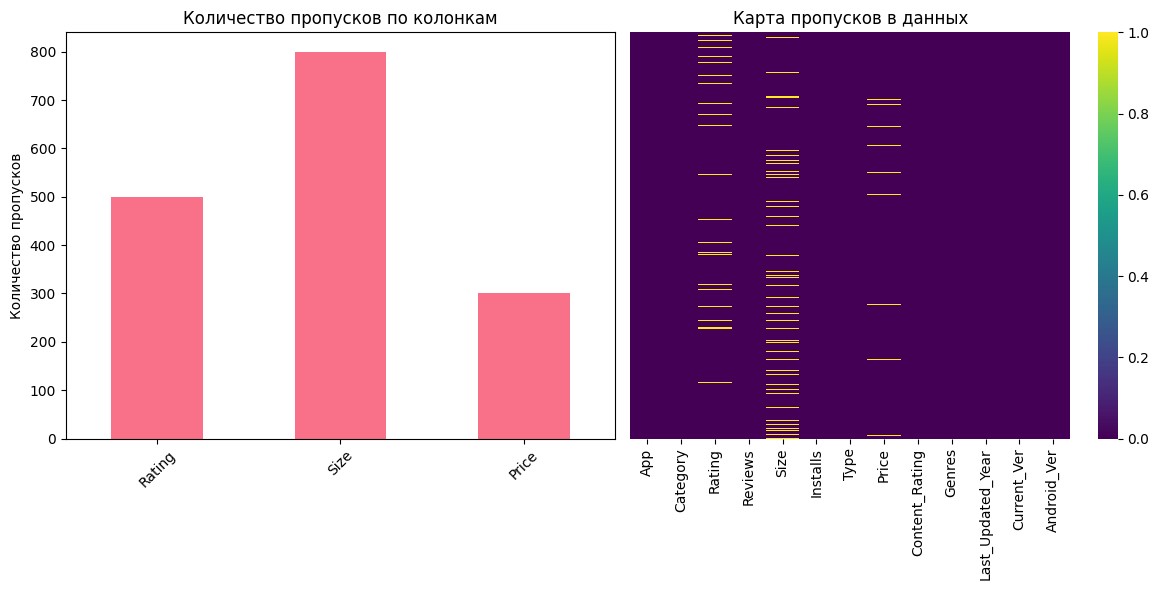
* Rating: 500 пропусков (5.0%)
* Size: 800 пропусков (8.0%)
* Price: 300 пропусков (3.0%)

1. АНАЛИЗ ПРОПУСКОВ======================================== Количество пропусков по колонкам:

Rating: 500 (5.00%)

Size: 800 (8.00%)

Price: 300 (3.00%)



1. ОБРАБОТКА ПРОПУСКОВ========================================

Исходный размер датасета: (10000, 13)

Размер после удаления пропусков: (8481, 13)

Удалено строк: 1519 (15.19%)

1. ПОДГОТОВКА ДАННЫХ ДЛЯ КОРРЕЛЯЦИОННОГО АНАЛИЗА======================================== Числовые признаки для анализа:
2. Rating
3. Reviews
4. Size
5. Installs
6. Price
7. Category\_encoded
8. Content\_Rating\_encoded
9. Type\_encoded9. Last\_Updated\_Year
10. Log\_Reviews
11. Log\_Installs
12. Log\_Size
13. Reviews\_per\_Install
14. App\_Age
15. КОРРЕЛЯЦИОННЫЙ АНАЛИЗ======================================== Топ-10 самых сильных корреляций (по модулю):

Last\_Updated\_Year ↔ App\_Age: -1.000

Price ↔ Type\_encoded: 0.763

Installs ↔ Log\_Installs: 0.576

Size ↔ Log\_Size: 0.556

Reviews ↔ Log\_Reviews: 0.296

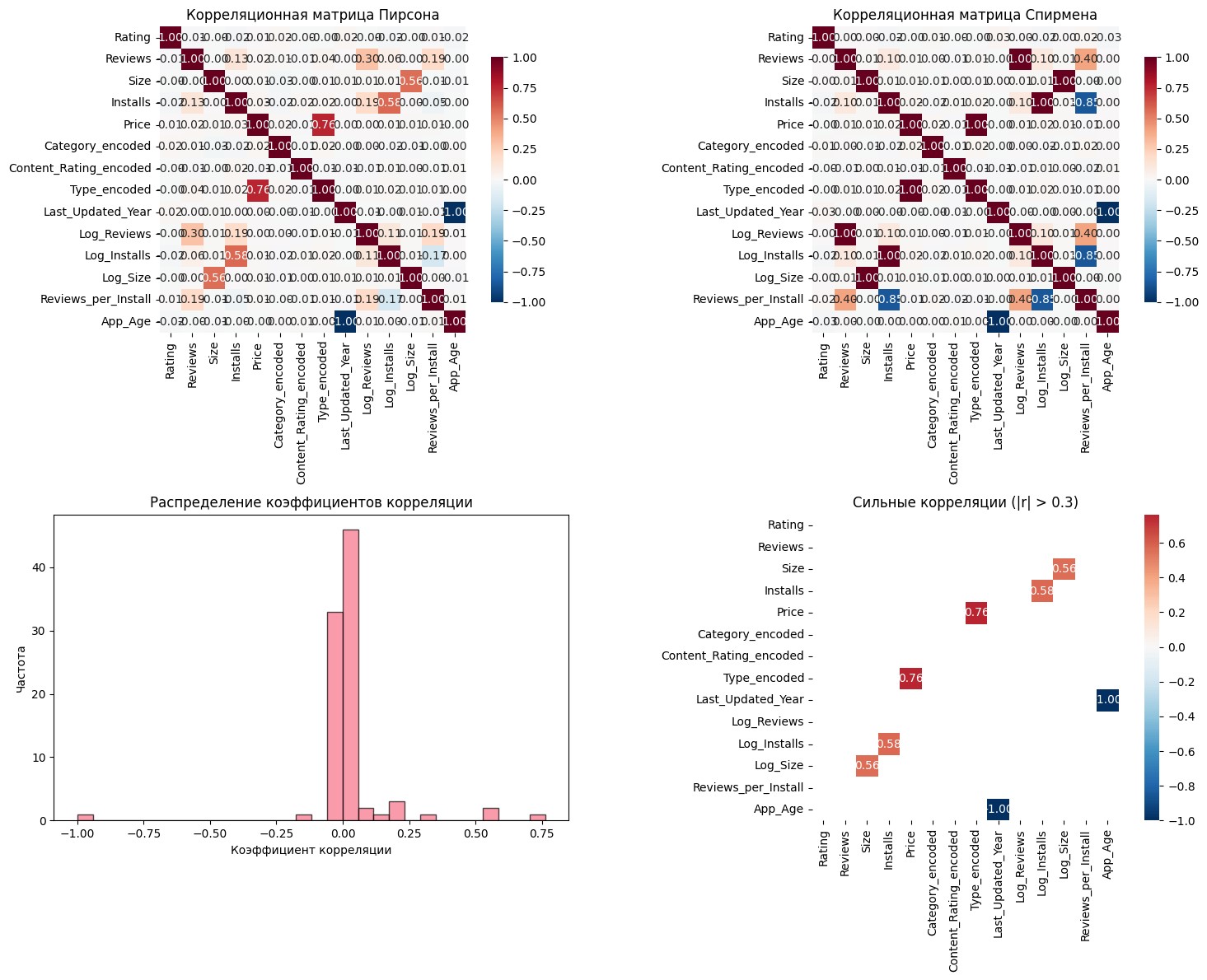
Reviews ↔ Reviews\_per\_Install: 0.190

Log\_Reviews ↔ Reviews\_per\_Install: 0.190

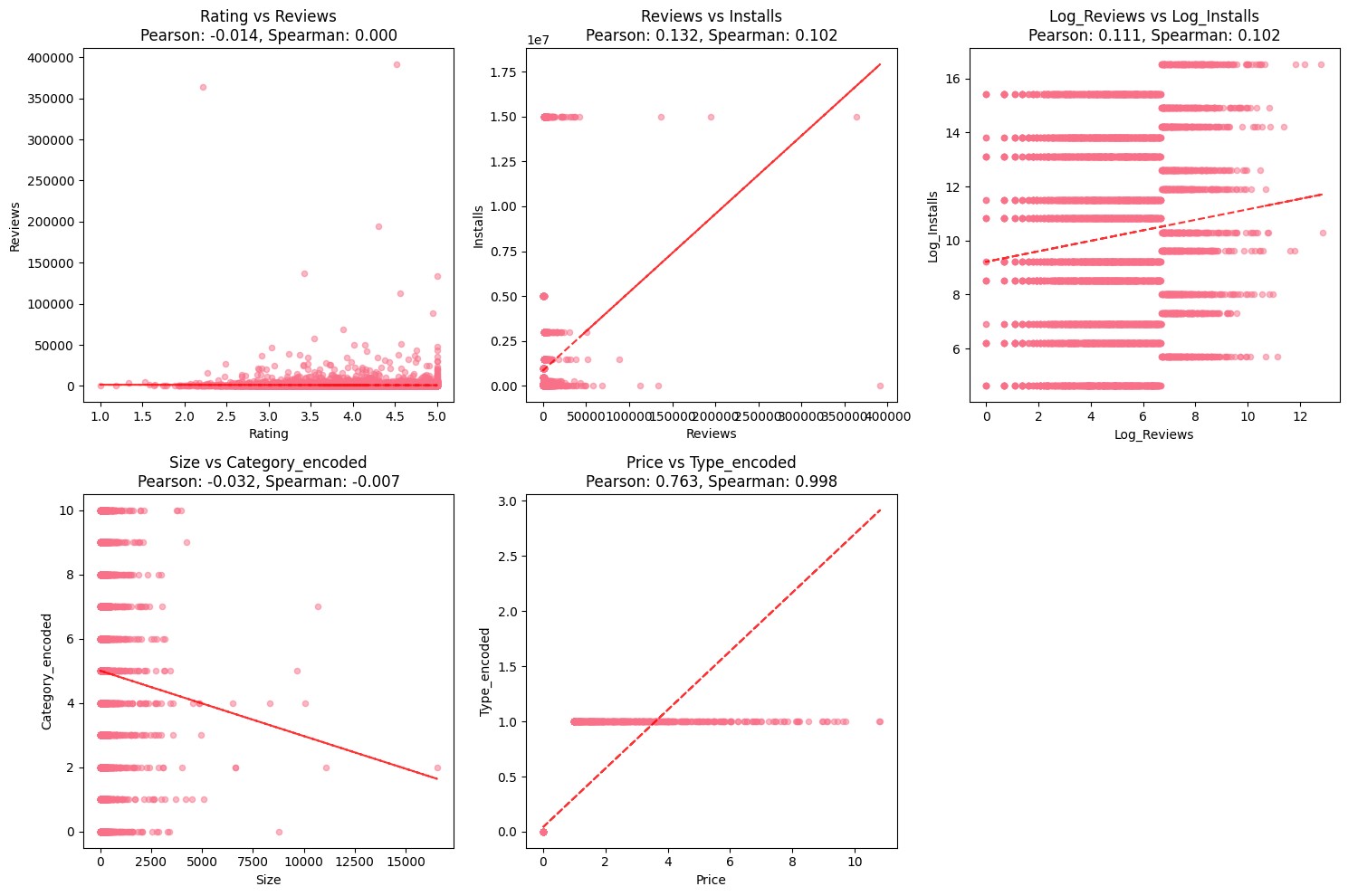
Installs ↔ Log\_Reviews: 0.186

Log\_Installs ↔ Reviews\_per\_Install: -0.172

Reviews ↔ Installs: 0.132



1. ДЕТАЛЬНЫЙ АНАЛИЗ КЛЮЧЕВЫХ КОРРЕЛЯЦИЙ========================================



1. АНАЛИЗ КОРРЕЛЯЦИЙ ПО КАТЕГОРИЯМ======================================== Средние значения ключевых метрик по категориям:

Rating Reviews Installs Size Price Category

BUSINESS 4.10 847.13 1068954.80 178.79 0.17

EDUCATION 4.13 1286.75 882072.24 169.84 0.24

ENTERTAINMENT 4.14 935.45 1028490.98 208.90 0.16

FAMILY 4.11 916.54 863017.99 180.97 0.23

[GAME 4.12 777.86 880278.24 227.27 0 19](#_Toc38961)

[HEALTH\_AND\_FITNESS 4.12 953.42 1058816.97 169.26 0.23LIFESTYLE 4.15 705.39 820742.71 158.07 0.22PHOTOGRAPHY 4.15 1144.00 820007.83 168.34 0 21](#_Toc38962)

[PRODUCTIVITY 4.14 1963.42 906953.06 149.81 0.26SOCIAL 4.16 1136.56 930310.33 142.13 0 21](#_Toc38963)

TOOLS 4.13 936.85 737853.56 147.85 0.26

Корреляция признаков с рейтингом приложения:

Last\_Updated\_Year: 0.023

App\_Age: 0.023

Log\_Installs: 0.022

Installs: 0.018

Category\_encoded: 0.016

Reviews: 0.014

Reviews\_per\_Install: 0.008

Price: 0.006

Log\_Reviews: 0.002

Size: 0.002

Type\_encoded: 0.002

Log\_Size: 0.002

Content\_Rating\_encoded: 0.001

8. ВЫВОДЫ ДЛЯ МАШИННОГО ОБУЧЕНИЯ ==================================================

РЕЗУЛЬТАТЫ КОРРЕЛЯЦИОННОГО АНАЛИЗА: ---------------------------------------------

Сильные корреляции (|r| > 0.5):

* Last\_Updated\_Year ↔ App\_Age: -1.000
* Price ↔ Type\_encoded: 0.763
* Installs ↔ Log\_Installs: 0.576
* Size ↔ Log\_Size: 0.556

Умеренные корреляции (0.3 < |r| ≤ 0.5):

ВЫВОДЫ О ВОЗМОЖНОСТИ ПОСТРОЕНИЯ МОДЕЛЕЙ ML: ------------------------------------------------- ПРОБЛЕМЫ МУЛЬТИКОЛЛИНЕАРНОСТИ:

* Last\_Updated\_Year и App\_Age сильно коррелируют (-1.000) Рекомендация: исключить один из признаков или использовать PCA

КАЧЕСТВО ДАННЫХ ПОСЛЕ ОБРАБОТКИ:

* Размер финального датасета: 8481 строк, 21 признаков
* Потеря данных при очистке: 15.2%
* Количество числовых признаков: 14

РЕКОМЕНДАЦИИ ДЛЯ МОДЕЛИРОВАНИЯ:

----------------------------------------

НАИБОЛЕЕ ВАЖНЫЕ ПРИЗНАКИ для предсказания рейтинга:

РЕКОМЕНДУЕМЫЕ АЛГОРИТМЫ:

* Линейная регрессия: подходит при отсутствии сильной мультиколлинеарности
* Random Forest: устойчив к корреляциям между признаками
* Gradient Boosting: эффективен для сложных зависимостей
* SVM: подходит для нелинейных зависимостей

РЕКОМЕНДУЕМАЯ ПРЕДОБРАБОТКА:

* Логарифмическое преобразование для Reviews, Installs, Size
* Нормализация/стандартизация числовых признаков
* One-hot encoding для категориальных признаков
* Создание дополнительных признаков (например, Reviews\_per\_Install)
* Применение PCA или исключение коррелирующих признаков

ОЖИДАЕМАЯ ПРОИЗВОДИТЕЛЬНОСТЬ МОДЕЛИ:

* Максимальная корреляция с целевой переменной: 0.023
* Низкий потенциал - возможно, нужны дополнительные признаки

ЦЕЛЕВЫЕ ПЕРЕМЕННЫЕ для разных задач:

* Регрессия: Rating (непрерывная)
* Классификация: Rating\_Category (например, Low/Medium/High)
* Кластеризация: группировка приложений по характеристикам

============================================================

АНАЛИЗ ЗАВЕРШЕН

============================================================ === КОРРЕЛЯЦИОННЫЙ АНАЛИЗ GOOGLE PLAY STORE APPS ===

1. СОЗДАНИЕ ПРОПУСКОВ В ДАННЫХ ======================================== Добавлены пропуски:

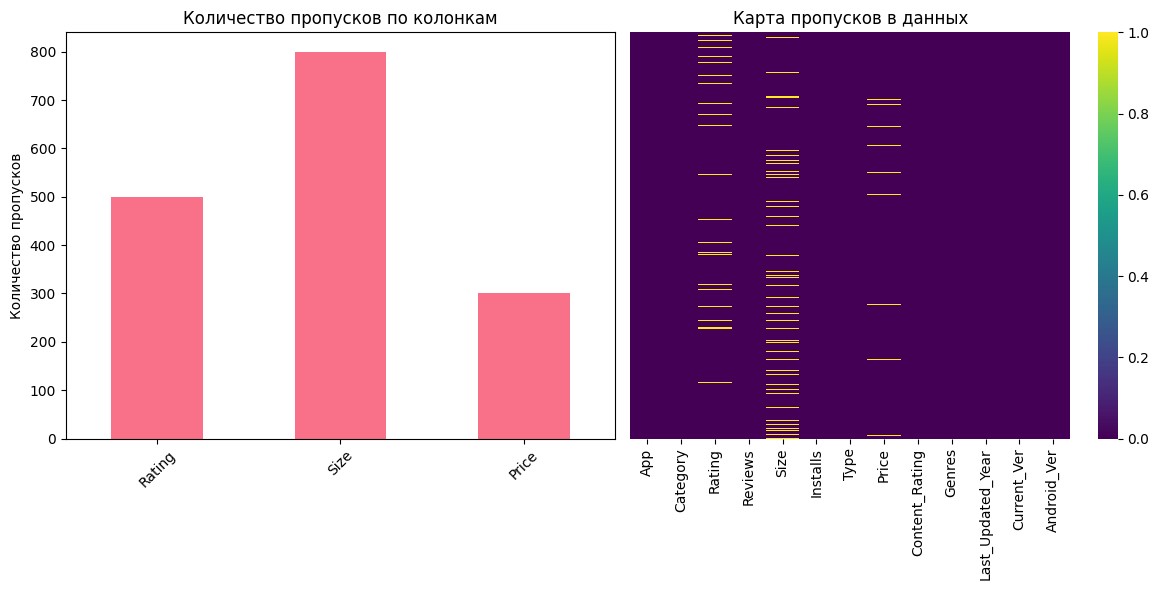
* Rating: 500 пропусков (5.0%)
* Size: 800 пропусков (8.0%)
* Price: 300 пропусков (3.0%)

1. АНАЛИЗ ПРОПУСКОВ======================================== Количество пропусков по колонкам:

Rating: 500 (5.00%)

Size: 800 (8.00%)

Price: 300 (3.00%)



1. ОБРАБОТКА ПРОПУСКОВ========================================

Исходный размер датасета: (10000, 13)

Размер после удаления пропусков: (8481, 13)

Удалено строк: 1519 (15.19%)

1. ПОДГОТОВКА ДАННЫХ ДЛЯ КОРРЕЛЯЦИОННОГО АНАЛИЗА======================================== Числовые признаки для анализа:
2. Rating
3. Reviews
4. Size
5. Installs
6. Price
7. Category\_encoded
8. Content\_Rating\_encoded
9. Type\_encoded9. Last\_Updated\_Year
10. Log\_Reviews
11. Log\_Installs
12. Log\_Size
13. Reviews\_per\_Install
14. App\_Age
15. КОРРЕЛЯЦИОННЫЙ АНАЛИЗ======================================== Топ-10 самых сильных корреляций (по модулю):

Last\_Updated\_Year ↔ App\_Age: -1.000

Price ↔ Type\_encoded: 0.763

Installs ↔ Log\_Installs: 0.576

Size ↔ Log\_Size: 0.556

Reviews ↔ Log\_Reviews: 0.296

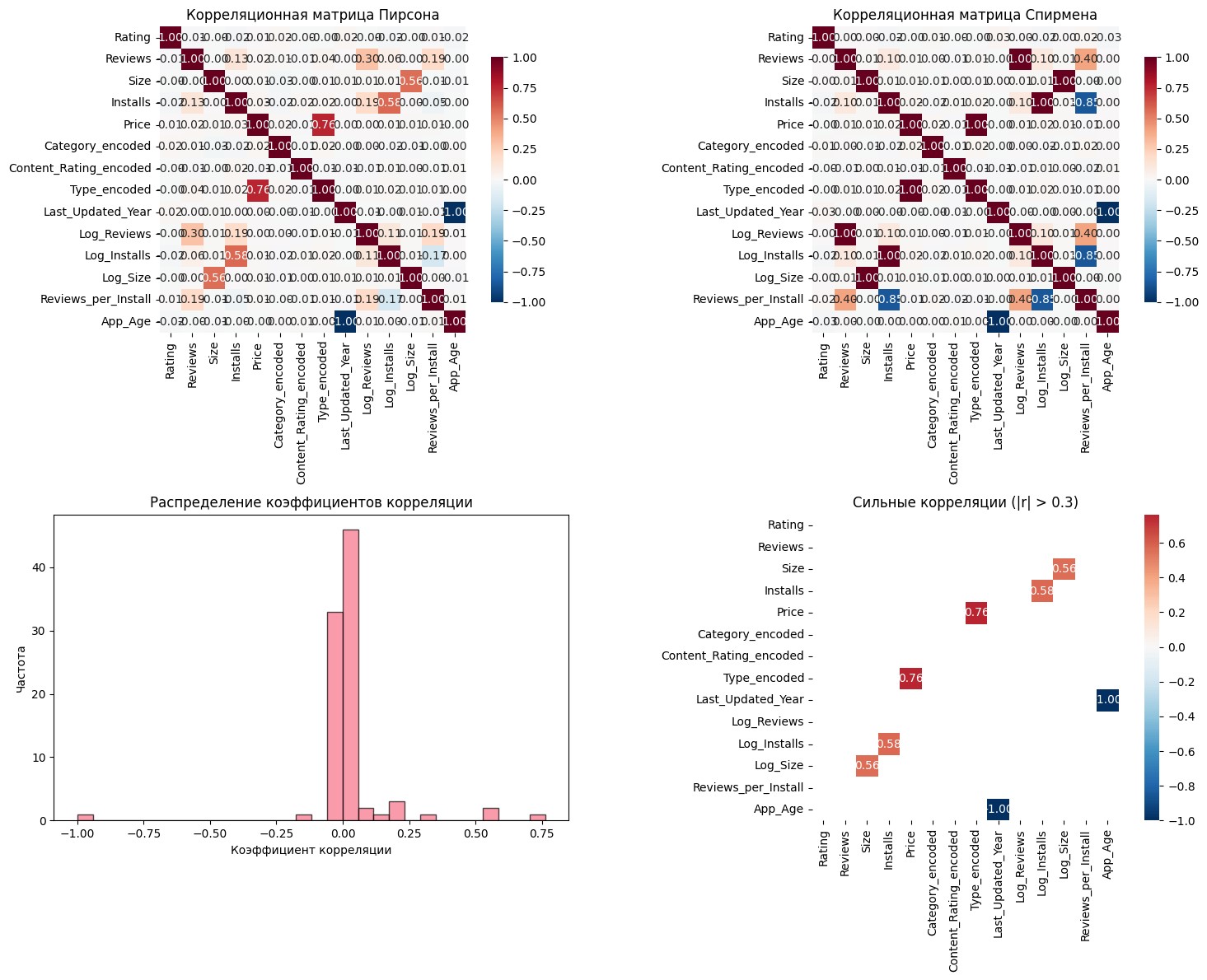
Reviews ↔ Reviews\_per\_Install: 0.190

Log\_Reviews ↔ Reviews\_per\_Install: 0.190

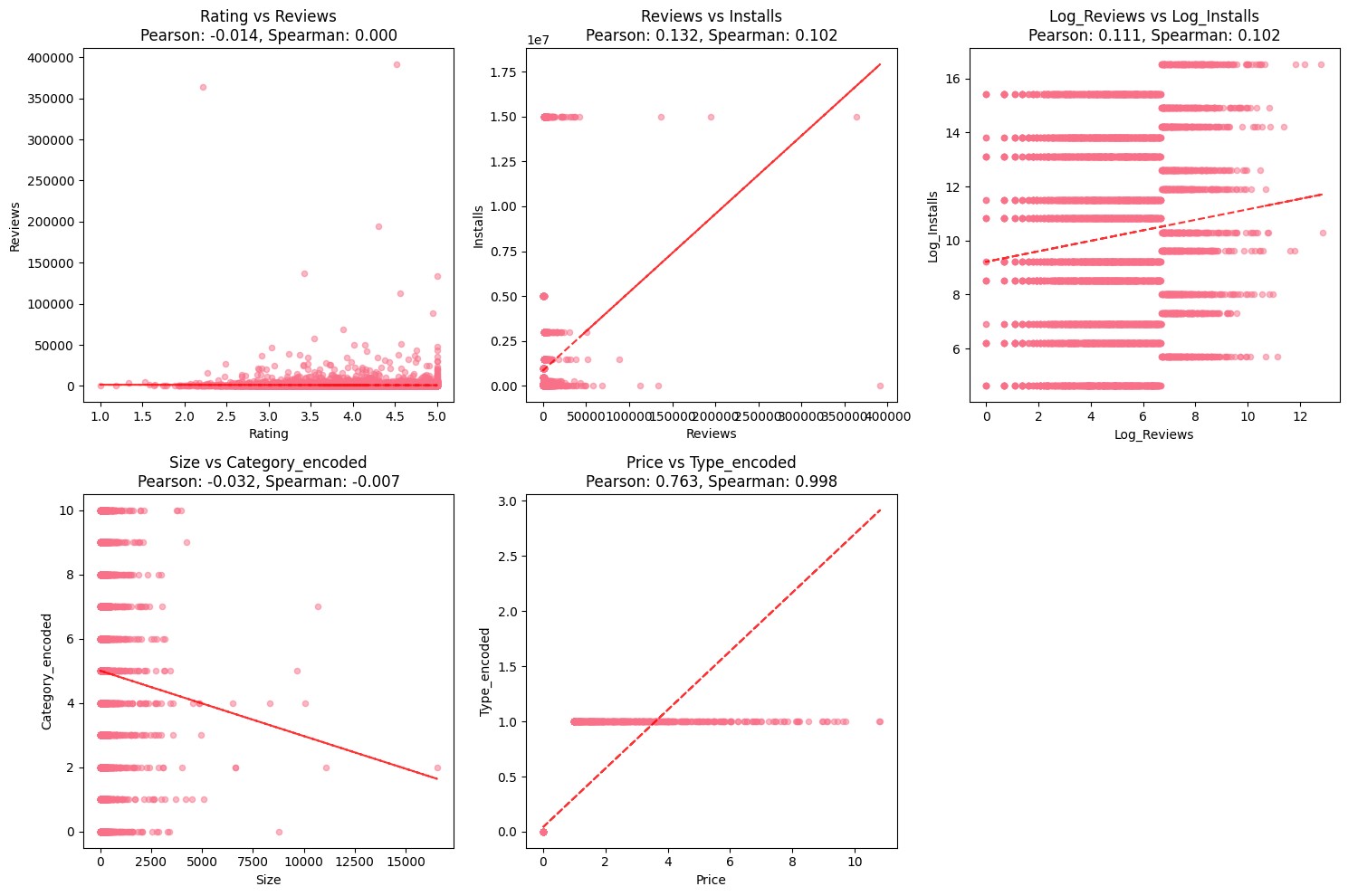
Installs ↔ Log\_Reviews: 0.186

Log\_Installs ↔ Reviews\_per\_Install: -0.172

Reviews ↔ Installs: 0.132



1. ДЕТАЛЬНЫЙ АНАЛИЗ КЛЮЧЕВЫХ КОРРЕЛЯЦИЙ========================================



1. АНАЛИЗ КОРРЕЛЯЦИЙ ПО КАТЕГОРИЯМ======================================== Средние значения ключевых метрик по категориям:

Rating Reviews Installs Size Price Category

BUSINESS 4.10 847.13 1068954.80 178.79 0.17

EDUCATION 4.13 1286.75 882072.24 169.84 0.24

ENTERTAINMENT 4.14 935.45 1028490.98 208.90 0.16

FAMILY 4.11 916.54 863017.99 180.97 0.23

# GAME 4.12 777.86 880278.24 227.27 0.19

# HEALTH\_AND\_FITNESS 4.12 953.42 1058816.97 169.26 0.23

LIFESTYLE 4.15 705.39 820742.71 158.07 0.22

PHOTOGRAPHY 4.15 1144.00 820007.83 168.34 0.21

# PRODUCTIVITY 4.14 1963.42 906953.06 149.81 0.26

SOCIAL 4.16 1136.56 930310.33 142.13 0.21

TOOLS 4.13 936.85 737853.56 147.85 0.26

Корреляция признаков с рейтингом приложения:

Last\_Updated\_Year: 0.023

App\_Age: 0.023

Log\_Installs: 0.022

Installs: 0.018

Category\_encoded: 0.016

Reviews: 0.014

Reviews\_per\_Install: 0.008

Price: 0.006

Log\_Reviews: 0.002

Size: 0.002

Type\_encoded: 0.002

Log\_Size: 0.002

Content\_Rating\_encoded: 0.001

8. ВЫВОДЫ ДЛЯ МАШИННОГО ОБУЧЕНИЯ ==================================================

РЕЗУЛЬТАТЫ КОРРЕЛЯЦИОННОГО АНАЛИЗА: ---------------------------------------------

Сильные корреляции (|r| > 0.5):

* Last\_Updated\_Year ↔ App\_Age: -1.000
* Price ↔ Type\_encoded: 0.763
* Installs ↔ Log\_Installs: 0.576
* Size ↔ Log\_Size: 0.556

Умеренные корреляции (0.3 < |r| ≤ 0.5):

ВЫВОДЫ О ВОЗМОЖНОСТИ ПОСТРОЕНИЯ МОДЕЛЕЙ ML: ------------------------------------------------- ПРОБЛЕМЫ МУЛЬТИКОЛЛИНЕАРНОСТИ:

* Last\_Updated\_Year и App\_Age сильно коррелируют (-1.000) Рекомендация: исключить один из признаков или использовать PCA

КАЧЕСТВО ДАННЫХ ПОСЛЕ ОБРАБОТКИ:

* Размер финального датасета: 8481 строк, 21 признаков
* Потеря данных при очистке: 15.2%
* Количество числовых признаков: 14

РЕКОМЕНДАЦИИ ДЛЯ МОДЕЛИРОВАНИЯ:

----------------------------------------

НАИБОЛЕЕ ВАЖНЫЕ ПРИЗНАКИ для предсказания рейтинга:

РЕКОМЕНДУЕМЫЕ АЛГОРИТМЫ:

* Линейная регрессия: подходит при отсутствии сильной мультиколлинеарности
* Random Forest: устойчив к корреляциям между признаками
* Gradient Boosting: эффективен для сложных зависимостей
* SVM: подходит для нелинейных зависимостей

РЕКОМЕНДУЕМАЯ ПРЕДОБРАБОТКА:

* Логарифмическое преобразование для Reviews, Installs, Size
* Нормализация/стандартизация числовых признаков
* One-hot encoding для категориальных признаков
* Создание дополнительных признаков (например, Reviews\_per\_Install)
* Применение PCA или исключение коррелирующих признаков

ОЖИДАЕМАЯ ПРОИЗВОДИТЕЛЬНОСТЬ МОДЕЛИ:

* Максимальная корреляция с целевой переменной: 0.023
* Низкий потенциал - возможно, нужны дополнительные признаки

ЦЕЛЕВЫЕ ПЕРЕМЕННЫЕ для разных задач:

* Регрессия: Rating (непрерывная)
* Классификация: Rating\_Category (например, Low/Medium/High)
* Кластеризация: группировка приложений по характеристикам

============================================================

АНАЛИЗ ЗАВЕРШЕН

============================================================