

Ключевые показатели (KPI) анализа пользовательской активности

Активные пользователи (Active Users)

DAU (Daily Active Users) - ежедневные активные пользователи

WAU (Weekly Active Users) - еженедельные активные пользователи

MAU (Monthly Active Users) - ежемесячные активные пользователи

Новые пользователи

Количество новых установок или регистраций за выбранный период

Retention Rate (Удержание пользователей)

Процент пользователей, вернувшихся в приложение через 1, 7, 30 дней

Среднее время сессии

Средняя продолжительность одного сеанса пользователя

Количество сессий на пользователя

Среднее число запусков приложения одним пользователем за период

Конверсия в ключевые действия

Например, включение функции контроля заряда, совершение покупки, подписка

Отказы (Bounce Rate)

Процент пользователей, которые быстро покидают приложение или не взаимодействуют с ключевыми функциями

Важные срезы и фильтры для детализации

По времени

День, неделя, месяц, произвольный период

По сегментам пользователей

Новые vs. постоянные

География (страна, регион)

Устройство и ОС (Android/iOS, версия)

Возраст, пол (если доступны)

Источник трафика (органический, реклама, рефералы)

По поведению

Частота использования функции контроля заряда

Время использования функции

Реакция на уведомления (открытия, клики)

Визуальные элементы дашборда

Линейные графики

Для отображения динамики активных пользователей, удержания, конверсий

Столбчатые диаграммы

Для сравнения активности по сегментам (например, по регионам или устройствам)

Круговые диаграммы

Для распределения пользователей по категориям (например, по источникам трафика)

Таблицы с детализацией

Для просмотра конкретных метрик по выбранным фильтрам

Индикаторы (карточки KPI)

С ключевыми цифрами и процентным изменением относительно предыдущего периода

pip install dash pandas numpy plotly dash-bootstrap-components pip install jupyter-dash

```
In [1]: import pandas as pd
import numpy as np
import random
from datetime import datetime, timedelta
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import dash
import sys
from dash import Dash, dcc, html, Input, Output, dash_table
import dash_bootstrap_components as dbc
from IPython.display import display, IFrame
```

Определяем тип приложения в зависимости от среды

```
In [2]: app = Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])
app.title = "Аналитика активности пользователей"
```

Генерация синтетического датасета

```
In [3]: def generate_dataset(start_date, end_date, user_count=100000):
    date_range = pd.date_range(start_date, end_date)
    data = {
        'date': np.random.choice(date_range, user_count),
        'user_id': [f'user_{i}' for i in range(user_count)],
        'is_new_user': np.random.choice([True, False], user_count, p=[0.2, 0.8]),
        'country': np.random.choice(['Россия', 'Белоруссия', 'Украина', 'Армения'],
                                     user_count),
        'device': np.random.choice(['Android', 'iOS'], user_count, p=[0.6, 0.4]),
        'os_version': np.random.choice(['13', '12', '11', '10'], user_count),
        'age': np.random.randint(16, 65, user_count),
        'gender': np.random.choice(['Муж', 'Жен'], user_count, p=[0.55, 0.45]),
        'traffic_source': np.random.choice(['Organic', 'Paid Ads', 'Referral', 'Social'],
                                             user_count),
        'session_duration': np.random.exponential(120, user_count).astype(int),
        'sessions_per_user': np.random.poisson(3, user_count) + 1,
        'used_battery_control': np.random.choice([True, False], user_count, p=[0.4, 0.6]),
        'battery_control_time': np.random.exponential(300, user_count).astype(int),
        'notification_opened': np.random.choice([True, False], user_count, p=[0.3, 0.7]),
        'made_purchase': np.random.choice([True, False], user_count, p=[0.1, 0.9]),
        'subscribed': np.random.choice([True, False], user_count, p=[0.05, 0.95])
    }
    df = pd.DataFrame(data)
    df['retention_1d'] = np.random.choice([True, False], user_count, p=[0.6, 0.4])
    df['retention_7d'] = np.random.choice([True, False], user_count, p=[0.3, 0.7])
    df['retention_30d'] = np.random.choice([True, False], user_count, p=[0.15, 0.85])
    df['bounce'] = np.random.choice([True, False], user_count, p=[0.2, 0.8])
    return df
```

Генерация данных за последний год

```
In [4]: end_date = datetime.now()
start_date = end_date - timedelta(days=365)
df = generate_dataset(start_date, end_date, 100000)
```

Инициализация Dash приложения

```
In [5]: app = dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])
app.title = "Аналитика активности пользователей"
```

Определение layout дашборда

```
In [6]: app.layout = dbc.Container([
    dbc.Row([
        dbc.Col(html.H1("Аналитика активности пользователей", className="text-center"), width=12),
    ]),

    # Фильтры

    dbc.Row([
        dbc.Col([
            html.Label("Интервал"),
            dcc.DatePickerRange(
                id='date-range',
                min_date_allowed=df['date'].min(),
                max_date_allowed=df['date'].max(),
                start_date=df['date'].max() - timedelta(days=30),
                end_date=df['date'].max()
            )
        ], width=3),

        dbc.Col([
            html.Label("Пользователи"),
            dcc.Dropdown(
                id='user-segment',
                options=[
                    {'label': 'Все пользователи', 'value': 'Все'},
                    {'label': 'Новые пользователи', 'value': 'Новые'},
                    {'label': 'Повторные пользователи', 'value': 'Повторные'}
                ],
                value='Все'
            )
        ], width=2),

        dbc.Col([
            html.Label("Страны"),
            dcc.Dropdown(
                id='country-filter',
                options=[{'label': c, 'value': c} for c in ['Все'] + sorted(df['country'].unique())],
                value='Все'
            )
        ], width=2),
    ])
```

```

dbc.Col([
    html.Label("ОС (Device)"),
    dcc.Dropdown(
        id='device-filter',
        options=[{'label': d, 'value': d} for d in ['Bce', 'Android', 'iOS']
        value='Bce'
    )
], width=2),

dbc.Col([
    html.Label("Источник трафика"),
    dcc.Dropdown(
        id='traffic-filter',
        options=[{'label': t, 'value': t} for t in ['Bce'] + sorted(df['tra
        value='Bce'
    )
], width=3)
], className="mb-4"),

# Карточки с KPI

dbc.Row([
    dbc.Col(dbc.Card([
        dbc.CardHeader("DAU"),
        dbc.CardBody([
            html.H4(id='dau-value', className="card-title"),
            html.P(id='dau-change', className="card-text")
        ])
    ], color="primary", outline=True), width=2),

    dbc.Col(dbc.Card([
        dbc.CardHeader("WAU"),
        dbc.CardBody([
            html.H4(id='wau-value', className="card-title"),
            html.P(id='wau-change', className="card-text")
        ])
    ], color="primary", outline=True), width=2),

    dbc.Col(dbc.Card([
        dbc.CardHeader("MAU"),
        dbc.CardBody([
            html.H4(id='mau-value', className="card-title"),
            html.P(id='mau-change', className="card-text")
        ])
    ], color="primary", outline=True), width=2),

    dbc.Col(dbc.Card([
        dbc.CardHeader("Удержание за неделю"),
        dbc.CardBody([
            html.H4(id='retention-value', className="card-title"),
            html.P(id='retention-change', className="card-text")
        ])
    ], color="success", outline=True), width=2),

    dbc.Col(dbc.Card([
        dbc.CardHeader("Среднее время взаимодействия на сеанс (Avg Session)"),

```

```

        dbc.CardBody([
            html.H4(id='session-value', className="card-title"),
            html.P(id='session-change', className="card-text")
        ])
    ], color="info", outline=True), width=2),

    dbc.Col(dbc.Card([
        dbc.CardHeader("Процент отказов (Bounce Rate)"),
        dbc.CardBody([
            html.H4(id='bounce-value', className="card-title"),
            html.P(id='bounce-change', className="card-text")
        ])
    ], color="danger", outline=True), width=2)
], className="mb-4"),

# Основные графики

dbc.Row([
    dbc.Col(dcc.Graph(id='active-users-chart'), width=6),
    dbc.Col(dcc.Graph(id='retention-chart'), width=6)
], className="mb-4"),

dbc.Row([
    dbc.Col(dcc.Graph(id='conversion-chart'), width=6),
    dbc.Col(dcc.Graph(id='device-distribution'), width=6)
], className="mb-4"),

dbc.Row([
    dbc.Col(dcc.Graph(id='traffic-source-chart'), width=6),
    dbc.Col(dcc.Graph(id='country-activity-chart'), width=6)
], className="mb-4"),

# Таблица с детализацией

dbc.Row([
    dbc.Col(html.H4("Подробно"), width=12),
    dbc.Col(dash_table.DataTable(
        id='detail-table',
        columns=[{"name": i, "id": i} for i in ['date', 'user_id', 'country', '
        page_size=10,
        style_table={'overflowX': 'auto'}
    ), width=12)
])
], fluid=True)

```

Callbacks для обновления данных

```

In [7]: @app.callback(
    [Output('dau-value', 'children'),
    Output('dau-change', 'children'),
    Output('wau-value', 'children'),
    Output('wau-change', 'children'),
    Output('mau-value', 'children'),
    Output('mau-change', 'children'),
    Output('retention-value', 'children'),

```

```

    Output('retention-change', 'children'),
    Output('session-value', 'children'),
    Output('session-change', 'children'),
    Output('bounce-value', 'children'),
    Output('bounce-change', 'children'),
    Output('active-users-chart', 'figure'),
    Output('retention-chart', 'figure'),
    Output('conversion-chart', 'figure'),
    Output('device-distribution', 'figure'),
    Output('traffic-source-chart', 'figure'),
    Output('country-activity-chart', 'figure'),
    Output('detail-table', 'data')],
    [Input('date-range', 'start_date'),
     Input('date-range', 'end_date'),
     Input('user-segment', 'value'),
     Input('country-filter', 'value'),
     Input('device-filter', 'value'),
     Input('traffic-filter', 'value')]
)
def update_all(start_date, end_date, user_segment, country_filter, device_filter, t
    return update_dashboard(start_date, end_date, user_segment, country_filter, dev

```

In [8]: `def update_dashboard(start_date, end_date, user_segment, country, device, traffic_s`

```

# Фильтрация данных

filtered_df = df[(df['date'] >= start_date) & (df['date'] <= end_date)]

if user_segment == 'new':
    filtered_df = filtered_df[filtered_df['is_new_user']]
elif user_segment == 'returning':
    filtered_df = filtered_df[~filtered_df['is_new_user']]

if country != 'Bce':
    filtered_df = filtered_df[filtered_df['country'] == country]

if device != 'Bce':
    filtered_df = filtered_df[filtered_df['device'] == device]

if traffic_source != 'Bce':
    filtered_df = filtered_df[filtered_df['traffic_source'] == traffic_source]

# Расчет KPI

# DAU

dau = filtered_df['user_id'].nunique()

prev_start_date = (pd.to_datetime(start_date) - timedelta(days=30)).strftime('%Y-%m-%d')
prev_end_date = (pd.to_datetime(end_date) - timedelta(days=30)).strftime('%Y-%m-%d')
prev_period_df = df[(df['date'] >= prev_start_date) & (df['date'] <= prev_end_date)]
prev_dau = prev_period_df['user_id'].nunique()
dau_change = ((dau - prev_dau) / prev_dau * 100) if prev_dau > 0 else 0

# WAU

```

```

wau = filtered_df['user_id'].nunique()
prev_wau = prev_period_df['user_id'].nunique()
wau_change = ((wau - prev_wau) / prev_wau * 100) if prev_wau > 0 else 0

# MAU

mau = filtered_df['user_id'].nunique()
prev_mau = prev_period_df['user_id'].nunique()
mau_change = ((mau - prev_mau) / prev_mau * 100) if prev_mau > 0 else 0

# Retention

retention = filtered_df['retention_7d'].mean() * 100
prev_retention = prev_period_df['retention_7d'].mean() * 100
retention_change = retention - prev_retention

# Avg Session Duration

avg_session = filtered_df['session_duration'].mean()
prev_avg_session = prev_period_df['session_duration'].mean()
session_change = ((avg_session - prev_avg_session) / prev_avg_session * 100) if

# Bounce Rate

bounce_rate = filtered_df['bounce'].mean() * 100
prev_bounce_rate = prev_period_df['bounce'].mean() * 100
bounce_change = bounce_rate - prev_bounce_rate

# График активных пользователей

daily_users = filtered_df.groupby('date')['user_id'].nunique().reset_index()
active_users_fig = px.line(daily_users, x='date', y='user_id',
                           title='Количество уникальных пользователей за сутки (
                           labels={'user_id': 'Users', 'date': 'Date'})

# График удержания

retention_data = filtered_df[['retention_1d', 'retention_7d', 'retention_30d']]
retention_fig = px.bar(x=['День', 'Неделя', 'Месяц'], y=retention_data,
                       title='Показатель удержания клиентов (User Retention Rate
                       labels={'x': 'Retention Period', 'y': 'Retention Rate (%)

# График конверсий

conversion_data = {
    'Action': ['Контроль батареи', 'Уведомление открыто', 'Покупка', 'Подписка']
    'Rate': [
        filtered_df['used_battery_control'].mean() * 100,
        filtered_df['notification_opened'].mean() * 100,
        filtered_df['made_purchase'].mean() * 100,
        filtered_df['subscribed'].mean() * 100
    ]
}
conversion_fig = px.bar(conversion_data, x='Action', y='Rate',
                        title='Коэффициенты конверсии для ключевых действий (Con
                        labels={'Rate': 'Conversion Rate (%)'})

```



```

# Распределение по устройствам

device_data = filtered_df['device'].value_counts().reset_index()
device_fig = px.pie(device_data, values='count', names='device',
                    title='Распределение пользователей по устройствам')

# Источники трафика

traffic_data = filtered_df['traffic_source'].value_counts().reset_index()
traffic_fig = px.pie(traffic_data, values='count', names='traffic_source',
                    title='Распределение пользователей по источнику трафика')

# Активность по странам

country_data = filtered_df.groupby('country').agg(
    users=('user_id', 'nunique'),
    avg_session=('session_duration', 'mean'),
    bounce_rate=('bounce', 'mean')
).reset_index()
country_fig = px.bar(country_data, x='country', y='users',
                    title='Активность пользователей по странам',
                    labels={'users': 'Number of Users', 'country': 'Country'})

# Подготовка данных для таблицы

table_data = filtered_df[['date', 'user_id', 'country', 'device', 'session_dura
table_data = table_data.sort_values('date', ascending=False)
table_data['session_duration'] = table_data['session_duration'].apply(lambda x:

return (
    f"{dau:,.1f}",
    f"{dau_change:,.1f}% {'↑' if dau_change > 0 else '↓'} по сравнению с предыду",
    f"{wau:,.1f}",
    f"{wau_change:,.1f}% {'↑' if wau_change > 0 else '↓'} по сравнению с предыду",
    f"{mau:,.1f}",
    f"{mau_change:,.1f}% {'↑' if mau_change > 0 else '↓'} по сравнению с предыду",
    f"{retention:,.1f}%",
    f"'+' if retention_change > 0 else ''}{retention_change:,.1f}pp по сравнени",
    f"{avg_session//60}m {avg_session%60}s",
    f"{session_change:,.1f}% {'↑' if session_change > 0 else '↓'} по сравнению с",
    f"{bounce_rate:,.1f}%",
    f"'+' if bounce_change > 0 else ''}{bounce_change:,.1f}pp по сравнению с пр",
    active_users_fig,
    retention_fig,
    conversion_fig,
    device_fig,
    traffic_fig,
    country_fig,
    table_data.to_dict('records')
)

```

if 'ipykernel' in sys.modules: app.run(mode='inline', debug=True) else: app.run(debug=True)

```

In [9]: if __name__ == '__main__':
        app.run(debug=True, port=8051)

```

```
if 'ipykernel' in sys.modules:  
    display(IFrame(src="http://localhost:8051", width='100%', height=800))
```

Аналитика активности пользователей

Интервал

04/21/2025
→
05/21/2025

Пользователи

...

Страны

...

ОС
(Device)

...

Источник
трафика

Все

DAU	WAU	MAU	Удержани е за неде лю	Сред нее врем я взаи моде йстви я на сеан с (Avg Sessi on)	Проц ент отказ ов (Bou nce Rate)
8,5 47	8,5 47	8,5 47	30. 3%		20. 7%
5.0%	5.0%	5.0%	-0.0p р по срав нени ю с		+0.5 pp по
↑ по срав нени ю с пред ыду щим пери одом	↑ по срав нени ю с пред ыду щим пери одом	↑ по срав нени ю с пред ыду щим пери			

Errors

Callbacks

v3.0.4

Server

Аналитика активности пользователей

Интервал

04/21/2025

→

05/21/2025

Пользователи

Страны

ОС
(Device)

Источник
трафика

...

...

...

Все

DAU	WAU	MAU	Удержани е за неде лю	Сред нее врем я взаи моде йств ия на сеан с (Avg Sessi on)	Проц ент отказ ов (Bou nce Rate)
8,547	8,547	8,547	30.3%		20.7%
5.0% ↑ по срав нени ю с пред ыду щим пери одом	5.0% ↑ по срав нени ю с пред ыду щим пери одом	5.0% ↑ по срав нени ю с пред ыду щим пери одом	-0.0p р по срав нени ю с пред ыду щим пери одом		+0.5 pp по срав нени ю с пред ыду щим
				1.0m 57.230724	

```
C:\Users\VAIO\AppData\Local\Programs\Python\Python39\lib\site-packages\_plotly_utils\
basevalidators.py:105: FutureWarning:
```

The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python datetime objects instead of an ndarray. To retain the old behavior, call `np.array` on the result

```
C:\Users\VAIO\AppData\Local\Programs\Python\Python39\lib\site-packages\_plotly_utils\
basevalidators.py:105: FutureWarning:
```

The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python datetime objects instead of an ndarray. To retain the old behavior, call `np.array` on the result

```
C:\Users\VAIO\AppData\Local\Programs\Python\Python39\lib\site-packages\_plotly_utils\
basevalidators.py:105: FutureWarning:
```

The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python datetime objects instead of an ndarray. To retain the old behavior, call `np.array` on the result

```
C:\Users\VAIO\AppData\Local\Programs\Python\Python39\lib\site-packages\_plotly_utils\
basevalidators.py:105: FutureWarning:
```

The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python datetime objects instead of an ndarray. To retain the old behavior, call `np.array` on the result

```
C:\Users\VAIO\AppData\Local\Programs\Python\Python39\lib\site-packages\_plotly_utils\
basevalidators.py:105: FutureWarning:
```

The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python datetime objects instead of an ndarray. To retain the old behavior, call `np.array` on the result

C:\Users\VAIO\AppData\Local\Programs\Python\Python39\lib\site-packages_plotly_utils\basevalidators.py:105: FutureWarning: The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python datetime objects instead of an ndarray. To retain the old behavior, call `np.array` on the result

C:\Users\VAIO\AppData\Local\Programs\Python\Python39\lib\site-packages_plotly_utils\basevalidators.py:105: FutureWarning: The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python datetime objects instead of an ndarray. To retain the old behavior, call `np.array` on the result Эти ошибки связаны с изменениями в библиотеке Pandas и тем, как она взаимодействует с Plotly при обработке данных о времени - в будущих версиях Pandas метод DatetimeProperties.to_pydatetime будет возвращать объект Series, а не массив NumPy (ndarray) Есть несколько способов решить эту проблему: 1. Явное преобразование в NumPy array: Измените код, чтобы явно преобразовывать результат to_pydatetime() в массив NumPy с помощью np.array(). import numpy as np import pandas as pd series = pd.Series(pd.date_range('20230101', periods=3)) datetime_array = np.array(series.dt.to_pydatetime()) Этот способ обеспечит совместимость со старым поведением и устранил предупреждение. 2. Адаптация к Pandas Series: Измените код, чтобы он работал с Pandas Series вместо массивов NumPy. Этот способ потребует изменений в коде, который использует результаты to_pydatetime(), чтобы он мог обрабатывать объекты Series. import pandas as pd series = pd.Series(pd.date_range('20230101', periods=3)) datetime_series = series.dt.to_pydatetime() 3. Подавление предупреждения: Вы можете временно подавить

предупреждение, используя модуль warnings. Однако это не рекомендуется в качестве долгосрочного решения, так как не решает основную проблему совместимости. `import warnings` `import pandas as pd` `warnings.filterwarnings("ignore", category=FutureWarning)` `series = pd.Series(pd.date_range('20230101', periods=3))` `datetime_array = series.dt.to_pydatetime()` 4. Обновление Plotly: Убедитесь, что вы используете последнюю версию Plotly. В версиях Plotly $\geq 5.18.0$ эта проблема должна быть исправлена