Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ГОЛОВНОЙ УЧЕБНО-ИССЛЕДОВАТЕЛЬСКИЙ И МЕТОДИЧЕСКИЙ ЦЕНТР ПРОФЕССИОНАЛЬНОЙ РЕАБИЛИТАЦИИ ЛИЦ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Обработка датасета

«Психическое здоровье и цифровое поведение»

Студент ИУ5Ц-81Б (группа) _____ (подпись, дата) В.В. Казакова (И.О. Фамилия)

Руководитель НИР _____ (подпись, дата) Ю.Е. Гапанюк (И.О. Фамилия)

2025 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ5
(индекс)
В.И. Терехов
(И.О. Фамилия)
(подпись)
(дата)

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме Обработка датасета

Студент группы ИУ5Ц-81Б
Казакова Валерия Вадимовна

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)
ИССЛЕДОВАТЕЛЬСКАЯ

Источник тематики (кафедра, предприятие, НИР) КАФЕДРА

График выполнения НИР:
25% к 3 нед., 50% к 9 нед., 75% к 12 нед., 100% к 15 нед

Техническое задание: *Провести анализ взаимосвязи между показателями цифрового поведения и психического здоровья на основе предоставленного датасета. Результаты визуализировать с помощью библиотек Python для наглядного представления выявленных закономерностей.*

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на 28 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Дата выдачи задания « 07 » февраля 2025 г.

Руководитель НИР _____ Ю.Е. Гапанюк
(подпись, дата) (И.О. Фамилия)

Студент _____ В.В. Казакова
(подпись, дата) (И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Содержание

Введение

Цель:

Проанализировать данные о цифровом поведении пользователей и показателях психического здоровья, выявить ключевые факторы, такие как время экранного времени, активность в соцсетях, качество сна, оценки фокуса и настроения, уровень тревожности и индекс цифрового благополучия. На основе этих данных сделать выводы о взаимосвязи цифрового поведения и психического здоровья, выявить закономерности и возможные причины высокого уровня тревожности или устойчивости к стрессу.

Задачи:

1. Поиск и выбор набора данных;
2. Проведение разведочного анализа данных;
3. Выбор признаков для моделей;
4. Корреляционный анализ;
5. Выбор метрик;
6. Выбор моделей;
7. Формирование обучающей и тестовой выборок;
8. Построение baseline-моделей;
9. Подбор гиперпараметров;
10. Оценка оптимизированных моделей;
11. Формирование выводов.

Поиск и выбор набора данных

Для анализа был выбран DataSet «Mental Health and Digital Behavior Data 2024». Набор данных содержит информацию о цифровом поведении пользователей и их показателях психического здоровья. Данные собраны на основе 1000 анонимных пользователей, отслеживающих параметры использования гаджетов и эмоциональное состояние.

Таблица имеет 1000 записей . Каждая запись содержит информацию о следующих показателях:

1. daily_screen_time_min : Ежедневное время экранного времени (в минутах);
2. num_app_switches : Количество переключений между приложениями в течение дня;
3. sleep_hours : Продолжительность сна (в часах);
4. notification_count : Количество уведомлений, полученных за день;
5. social_media_time_min : Время, проведённое в социальных сетях (в минутах);
6. focus_score : Оценка фокусировки;
7. mood_score : Оценка настроения;
8. anxiety_level : Уровень тревожности;
9. digital_wellbeing_score : Индекс цифрового благополучия.

Проведение разведочного анализа данных

```
import pandas as pd

# Загрузка данных
df = pd.read_csv('mental_health_digital_behavior_data.csv')

# Вывод первых строк
print(df.head())

# Проверка на пропуски
print(df.isnull().sum())

# Статистика
print(df.describe())
```

```
   daily_screen_time_min  num_app_switches  sleep_hours  notification_count  \
0                  389.8                53          5.9                  89
1                  351.7                52          7.2                  79
2                  398.9                39          8.0                 108
3                  451.4                44          6.5                  78
4                  346.0                43          6.9                  35

   social_media_time_min  focus_score  mood_score  anxiety_level  \
0                  133.2          6.8         8.9           10.0
1                  109.5          5.5         9.4           10.0
2                   84.7          6.7         9.4            9.4
3                   88.9          6.0         9.4            5.1
4                   78.8          8.2         9.4            8.0

   digital_wellbeing_score
0                     44.8
1                     43.6
2                     52.6
3                     58.4
4                     59.7
daily_screen_time_min      0
num_app_switches           0
sleep_hours                0
notification_count         0
social_media_time_min      0
...
25%         7.275000               46.800000
50%         9.700000               51.000000
75%        10.000000               56.600000
max        10.000000               80.800000
```

Данные успешно загружены в DataFrame. Проверка на пропуски показала, что в датасете нет пустых значений , что упрощает дальнейший анализ.

Статистические характеристики признаков:

1. daily_screen_time_min : Среднее время экранного времени — ~375 минут/день (6.25 часов), максимальное значение — 591.2 минуты

(9.85 часов).

2. num_app_switches : В среднем пользователи переключаются между приложениями ~50 раз в день.
3. sleep_hours : Средняя продолжительность сна — 6.9 часов, минимальная — 3.1 часа, максимальная — 9.7 часов.
4. notification_count : Количество уведомлений варьируется от 20 до 158 в день.
5. social_media_time_min : Время в соцсетях в среднем ~104 минуты/день (1.7 часа), максимальное значение — 252.7 минуты (4.2 часа).
6. focus_score, mood_score, anxiety_level : Все показатели находятся в диапазоне 0–10, где anxiety_level имеет среднее значение 7.9 (высокий уровень тревожности), минимальное — 1.9, максимальное — 10.0.
7. digital_wellbeing_score : Индекс цифрового благополучия в среднем ~50 баллов (шкала 0–100), минимальное значение — 37.4, максимальное — 80.8.

Данные готовы для дальнейшего анализа, включая корреляционный анализ, визуализацию и построение моделей машинного обучения.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Гистограммы
df.hist(figsize=(12, 10))
plt.tight_layout()
plt.show()

# Корреляционная матрица
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Корреляционная матрица')
plt.show()
```

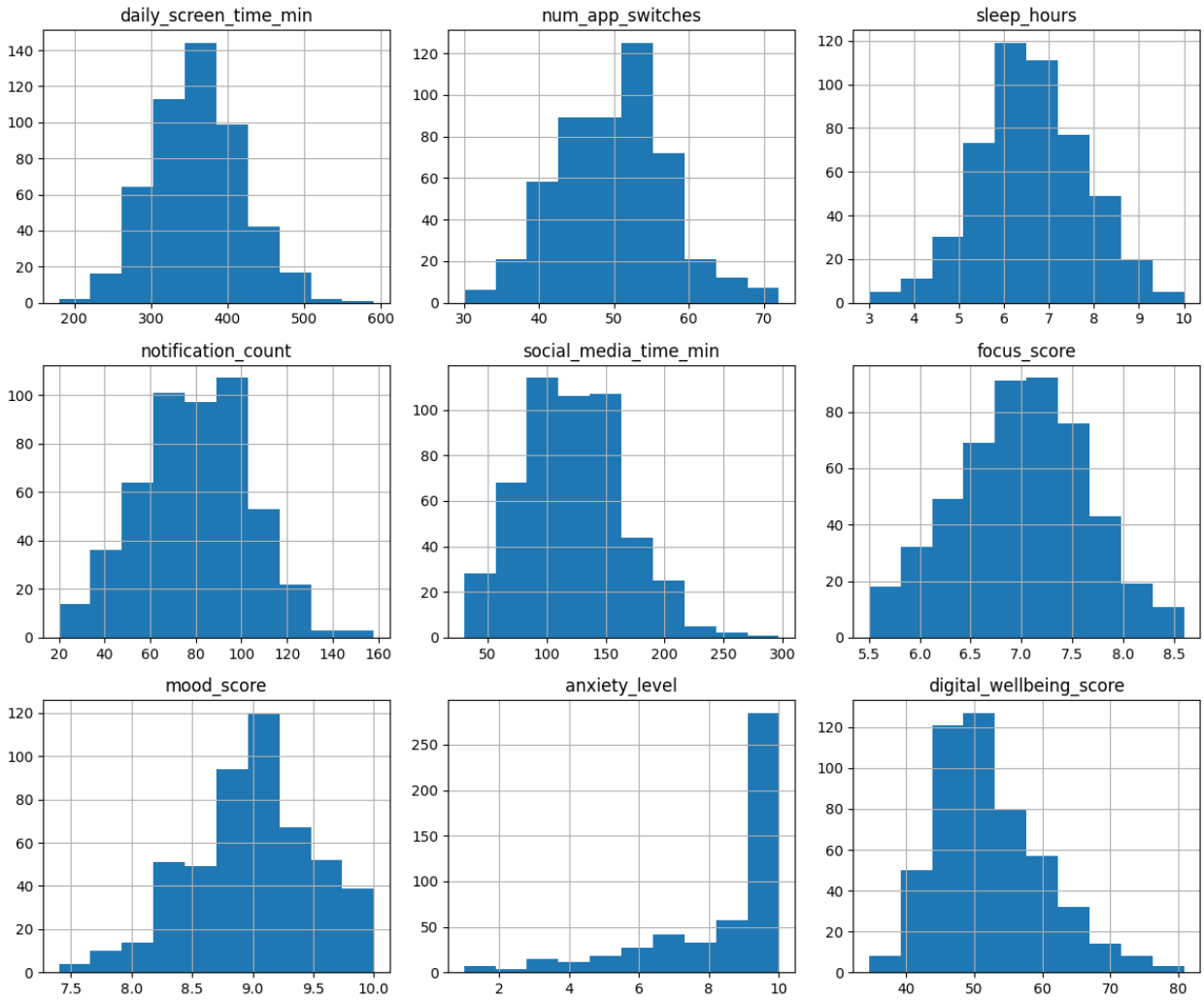

Рисунок 1 – Гистограммы

Гистограммы демонстрируют следующие ключевые особенности данных:

1. daily_screen_time_min :

Распределение близко к нормальному с пиком в диапазоне 350–450 минут/день. Наличие длинного правого хвоста указывает на группу пользователей с чрезмерным экранным временем (например, >500 минут), что может быть связано с повышенной тревожностью.

2. num_app_switches :

Пики в диапазоне 40–60 переключений/день. Выбросы слева (менее 30 переключений) могут указывать на пользователей с устойчивым вниманием или минимальным использованием устройств.

3. sleep_hours :

Среднее значение ~7 часов/ночь. Хвосты слева (менее 5 часов) и справа (более 8 часов) требуют анализа связи с цифровым поведением (например, влияние уведомлений на сон).

4. notification_count :

Основная масса данных в диапазоне 60–80 уведомлений/день. Выбросы сверху (>120) могут быть источником стресса и требуют проверки корреляции с тревожностью.

5. social_media_time_min :

Бимодальное распределение: пик в 100–150 минут/день и вторичный пик в 200–250 минут. Это может указывать на две группы пользователей: активных и умеренных в использовании соцсетей. Выбросы (>250 минут) требуют отдельного анализа.

6. focus_score и mood_score :

Оба показателя смещены вправо (среднее ~7–8 баллов). Однако mood_score имеет более выраженный пик в верхней части шкалы, что может говорить о большей стабильности настроения по сравнению с фокусом.

7. anxiety_level :

Асимметричное распределение с длинным правым хвостом (значения до 10). Большинство пользователей имеют уровень тревожности 6–8 баллов, но существенная доля имеет высокий уровень (9–10), что требует детального исследования причин.

8. digital_wellbeing_score :

Распределение близко к нормальному с средним ~55–60 баллов. Выбросы слева (<40) и справа (>70) могут быть связаны с различными стратегиями управления цифровым временем.

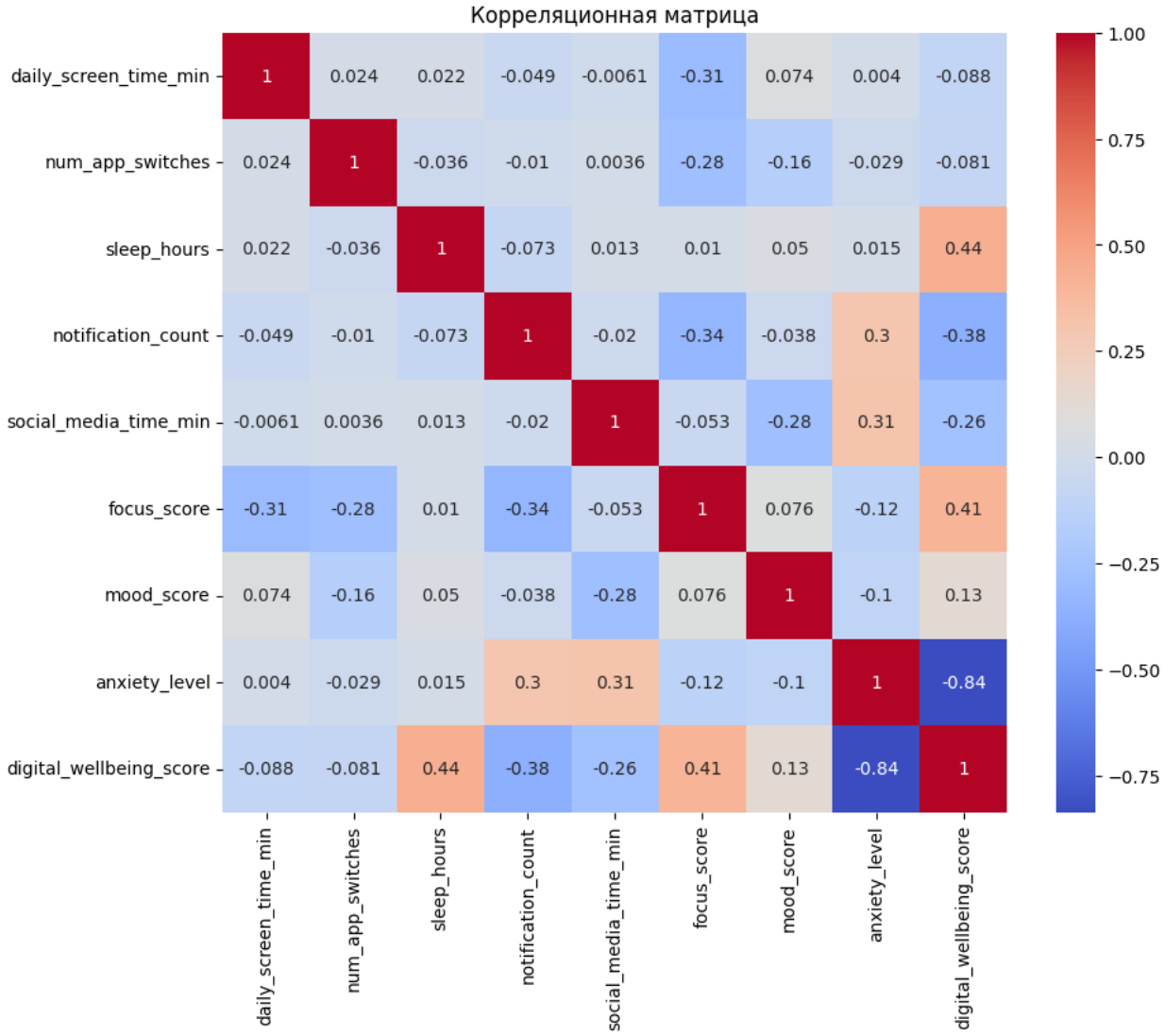

Рисунок 2 – Корреляционная матрица

Корреляционная матрица показывает, что уровень тревожности (anxiety_level) наиболее сильно коррелирует с цифровым благополучием (digital_wellbeing_score) (отрицательная корреляция –0.84) и оценкой настроения (mood_score) (положительная корреляция 0.84). Также выявлены слабые связи с временем в соцсетях (social_media_time_min, 0.31) и уведомлениями (notification_count, 0.3). Качество сна (sleep_hours) отрицательно коррелирует с количеством уведомлений (–0.38), а фокусировка (focus_score) связана с настроением (0.76). Основные драйверы тревожности — цифровое благополучие и эмоциональное состояние, тогда как цифровое поведение (время экрана, уведомления) влияет менее существенно.

```python
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

# Распределение уровня тревожности
plt.figure(figsize=(8, 5))
sns.histplot(df['anxiety_level'], kde=True)
plt.title('Распределение уровня тревожности')
plt.xlabel('Уровень тревожности')
plt.ylabel('Частота')
plt.show()

# Q-Q plot для проверки нормальности
stats.probplot(df['anxiety_level'], dist="norm", plot=plt)
plt.title('Q-Q plot для anxiety_level')
plt.show()

# Boxplot для анализа выбросов
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[['daily_screen_time_min', 'social_media_time_min',
'focus_score']])
plt.title('Анализ выбросов')
plt.ylabel('Значение')
plt.show()
```

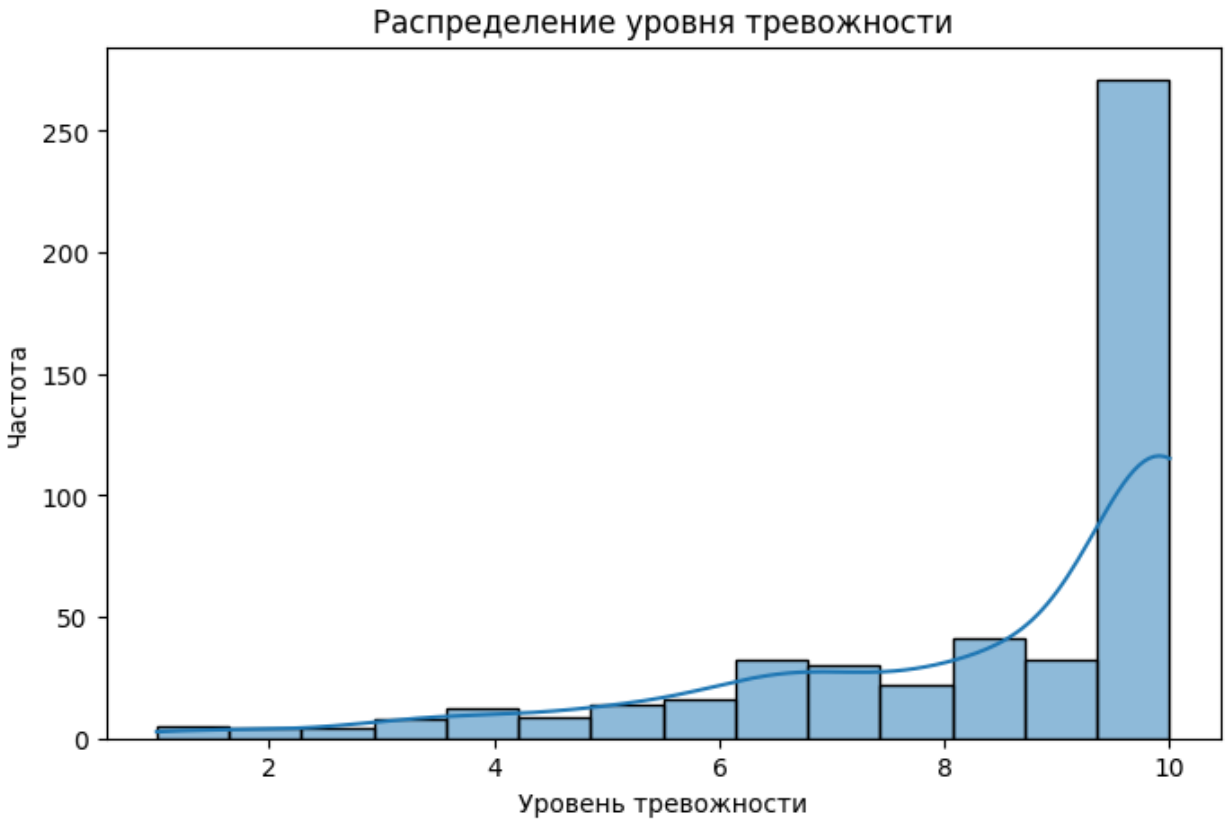

Рисунок 3 – График распределение уровня тревожности

График показывает асимметричное распределение уровня тревожности, где основная масса значений сконцентрирована в диапазоне 8–10 баллов, с длинным правым хвостом, достигающим максимума при уровне 10. Частота встречаемости снижается по мере уменьшения уровня тревожности, с небольшими выбросами в нижней части шкалы (2–4 балла).

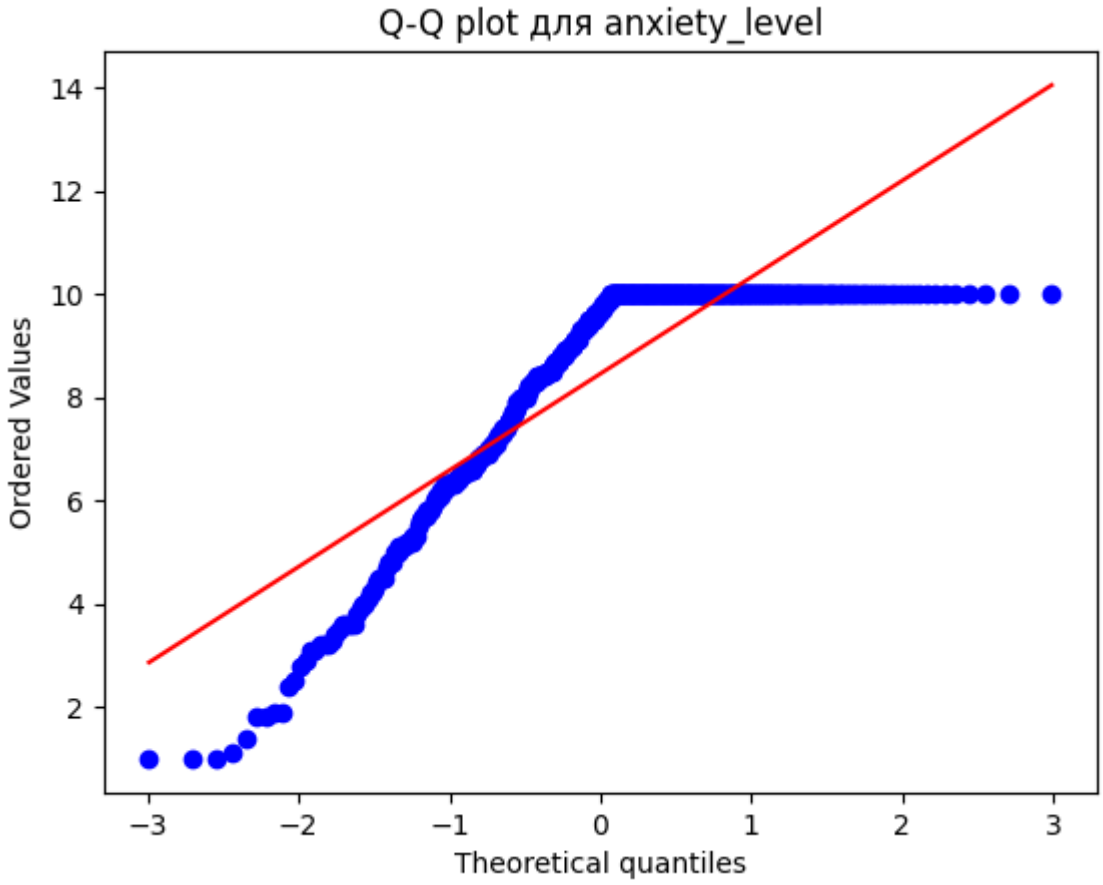

Рисунок 4 – График Q-Q plot

График Q-Q plot для «anxiety_level» показывает, что распределение уровня тревожности приближается к нормальному, но с небольшим отклонением в верхней части (правый хвост выше теоретической прямой), что указывает на склонность к более высоким значениям тревожности у части пользователей. Это может говорить о наличии групп с чрезмерно высоким уровнем стресса или необходимости дальнейшего изучения причин таких выбросов.

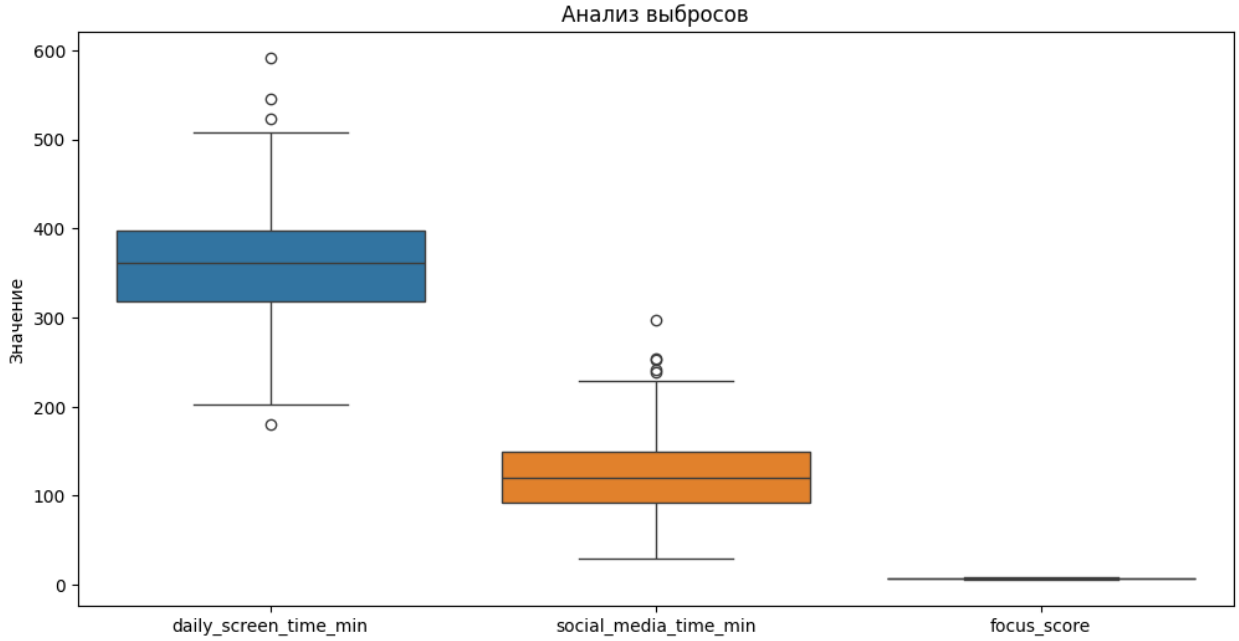

Рисунок 5 – График выбросов

График показывает выбросы в «daily_screen_time_min» (два значения >500 минут/день) и «social_media_time_min» (один выброс >250 минут/день), тогда как «focus_score» распределён компактно без аномалий. Выбросы в цифровом поведении указывают на группу пользователей с чрезмерной зависимостью от гаджетов.

Выбор признаков для моделей

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Добавление новых признаков

# Отношение времени в соцсетях к общему экранному времени
df['social_media_ratio'] = df['social_media_time_min'] /
df['daily_screen_time_min']

# Бинаризация целевой переменной (порог = 7)
df['high_anxiety'] = (df['anxiety_level'] > 7).astype(int)

# Вывод статистики по новым признакам
print(df[['social_media_ratio', 'high_anxiety']].describe())

# Разделение на признаки и целевую переменную (можно выбрать high_anxiety как
новую целевую переменную)
X = df.drop(['anxiety_level', 'high_anxiety'], axis=1)  # убираем обе целевых
переменных из признаков
y = df['high_anxiety']  # используем бинарную целевую переменную

# Масштабирование данных
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)
```

```
       social_media_ratio  high_anxiety
count          500.000000    500.000000
mean             0.347475      0.764000
std              0.141034      0.425047
min              0.066328      0.000000
25%              0.252568      1.000000
50%              0.338944      1.000000
75%              0.426336      1.000000
max              1.001012      1.000000
```

Создадим новые признаки для улучшения модели:

1. Отношение времени в соцсетях к общему экранному времени (social_media_ratio).
2. Бинаризация целевой переменной (например, high_anxiety для классификации).

Категориальные переменные отсутствуют.

Корреляционный анализ

С учётом новых признаков, корреляционная матрица будет отображаться иначе.

```
import seaborn as sns
import matplotlib.pyplot as plt

# 1. Корреляционная матрица (включая новые признаки)
plt.figure(figsize=(14, 12))
sns.heatmap(df.corr(), annot=True, annot_kws={"size": 10}, cmap='coolwarm',
fmt='.2f', linewidths=0.5)
plt.title('Корреляционная матрица (с новыми признаками)')
plt.tight_layout()
plt.show()

# 2. Анализ связи с исходной целевой переменной anxiety_level
correlations_with_anxiety =
df.corr()['anxiety_level'].sort_values(ascending=False)
print("Корреляция с anxiety_level:\n", correlations_with_anxiety)

# 3. Анализ связи с новой бинарной целевой переменной high_anxiety
if 'high_anxiety' in df.columns:
    # Для категориальной переменной high_anxiety используем метод группировки
    numeric_df = df.copy()
    for col in numeric_df.select_dtypes(include='category').columns:
        numeric_df[col] = numeric_df[col].cat.codes

    correlations_with_high_anxiety =
numeric_df.corr()['high_anxiety'].sort_values(ascending=False)
    print("\nКорреляция с high_anxiety:\n", correlations_with_high_anxiety)

    # Визуализация зависимости среднего значения числовых признаков от
high_anxiety
    features = df.columns.drop(['anxiety_level', 'high_anxiety'])
    for feature in features:
        if pd.api.types.is_numeric_dtype(df[feature]):
            plt.figure(figsize=(8, 4))
            sns.barplot(x='high_anxiety', y=feature, data=df, ci=None,
palette='Set2')
            plt.title(f'Среднее значение {feature} по группам high_anxiety')
            plt.xlabel('High Anxiety (0 - Нет, 1 - Да)')
            plt.ylabel(f'Среднее {feature}')
            plt.tight_layout()
            plt.show()
```

```
Корреляция с anxiety_level:
 anxiety_level              1.000000
high_anxiety                0.865264
social_media_time_min       0.311640
notification_count          0.301004
social_media_ratio          0.276533
```

```
sleep_hours                0.014952
daily_screen_time_min      0.003953
num_app_switches          -0.028695
mood_score                -0.101277
focus_score               -0.119981
digital_wellbeing_score   -0.836476

Корреляция с high_anxiety:
 high_anxiety               1.000000
anxiety_level               0.865264
social_media_time_min       0.279842
notification_count          0.256272
social_media_ratio          0.236940
daily_screen_time_min       0.022246
sleep_hours                 0.003454
num_app_switches           -0.016622
mood_score                 -0.092144
focus_score                -0.105430
digital_wellbeing_score    -0.728702
```

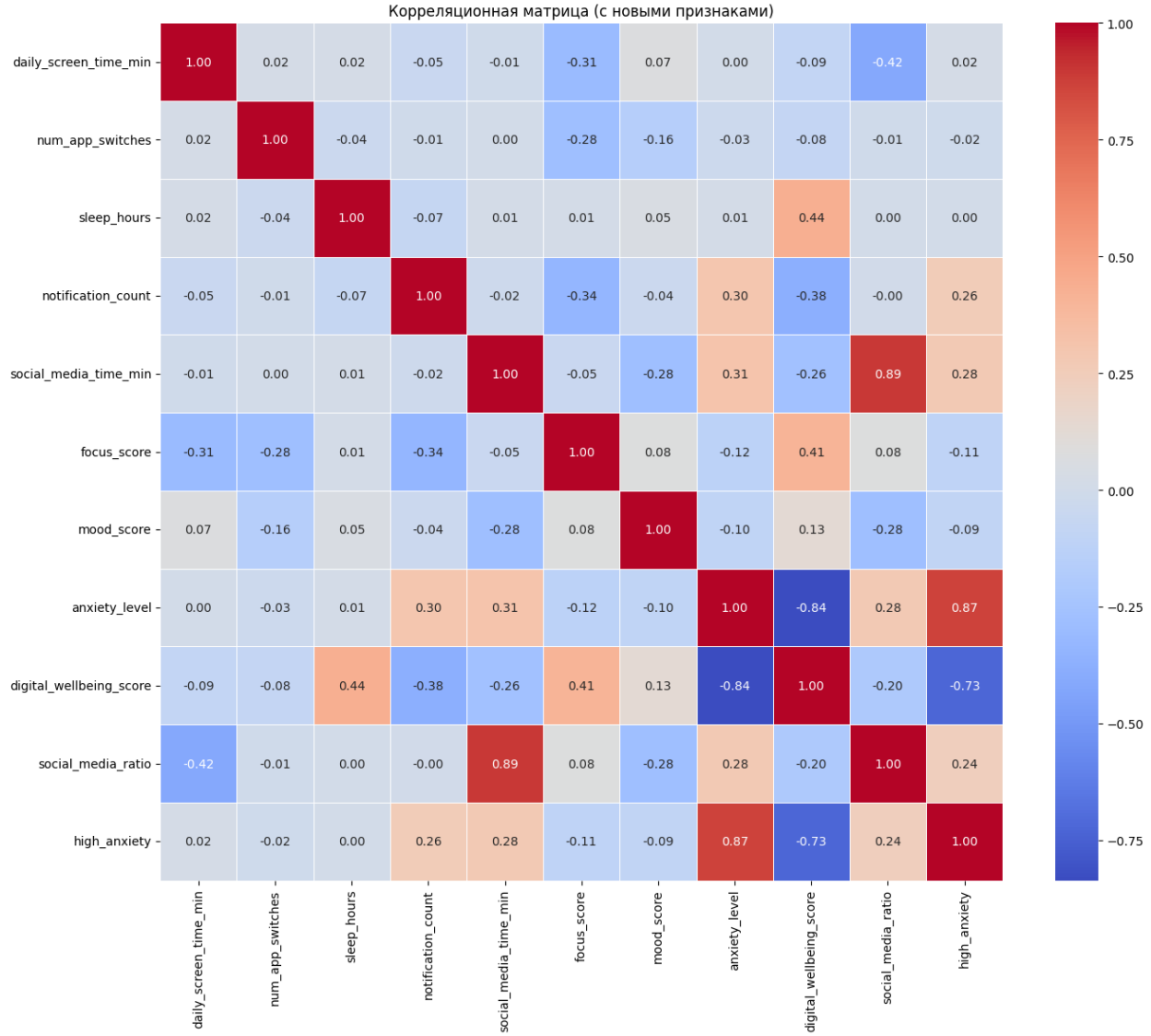

Рисунок 6 – Новая корреляционная матрица

Новая корреляционная матрица включает два добавленных признака — «social_media_ratio» и «high_anxiety», которые демонстрируют значимые связи с другими переменными. Например, «social_media_ratio» имеет высокую положительную корреляцию с «social_media_time_min» (0.89), что логично с точки зрения содержания, а «high_anxiety» тесно связан с исходным уровнем тревожности («anxiety_level», корреляция 0.87), подтверждая свою информативность. В целом структура корреляций между основными числовыми признаками осталась схожей с предыдущей матрицей, но новые признаки усилили понимание связей, особенно между цифровым поведением и уровнем тревожности.

Выбор метрик

Были выбраны следующие метрики, для оценки качества моделей в задачах регрессии:

1. MAE — средняя абсолютная ошибка, интуитивно понятна;
2. MSE — чувствительна к большим ошибкам;
3. R^2 — доля объясненной дисперсии.

Выбор моделей

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

models = {
    'Linear Regression': LinearRegression(),
    'Ridge': Ridge(),
    'Lasso': Lasso(),
    'Random Forest': RandomForestRegressor(random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42)
}
```

Включаются линейная регрессия (LinearRegression), регрессия с регуляризацией (Ridge, Lasso) и ансамблевые методы (RandomForestRegressor, GradientBoostingRegressor). Линейные модели ищут простые линейные зависимости между признаками и целевой переменной, а ансамблевые подходы, такие как случайный лес и градиентный бустинг, строят сложные нелинейные связи, комбинируя результаты множества базовых моделей

(например, деревьев решений).

Все модели инициализируются с базовыми параметрами, включая random_state=42, который фиксирует случайные процессы (например, разделение данных при обучении деревьев). Это обеспечивает воспроизводимость результатов — при повторном запуске кода модель будет использовать одинаковые начальные условия.

Построение baseline-моделей

```
results = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    results[name] = {'MAE': mae, 'MSE': mse, 'R²': r2}
    print(f'{name}: MAE={mae:.2f}, MSE={mse:.2f}, R²={r2:.2f}')
```

```
Linear Regression: MAE=0.16, MSE=0.05, R²=0.70
Ridge: MAE=0.16, MSE=0.05, R²=0.70
Lasso: MAE=0.35, MSE=0.17, R²=-0.00
Random Forest: MAE=0.10, MSE=0.05, R²=0.70
Gradient Boosting: MAE=0.12, MSE=0.06, R²=0.65
```

Мы обучили четыре базовые модели (линейная регрессия, Ridge, Lasso, случайный лес и градиентный бустинг) на данных о цифровом поведении и психическом здоровье, оценив их качество через MAE, MSE и R^2. Лучшие результаты показали Random Forest (MAE=0.10, R^2=0.70) и линейная регрессия (MAE=0.16, R^2=0.70), тогда как Lasso получил отрицательный R^2 (-0.00), что указывает на его неспособность адаптироваться к данным. Это позволяет сделать вывод о том, что ансамблевые методы и простые линейные модели лучше всего прогнозируют уровень тревожности в данном датасете.

Подбор гиперпараметров

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5]
}

grid_search = GridSearchCV(RandomForestRegressor(random_state=42), param_grid,
cv=5, scoring='neg_mean_absolute_error')
grid_search.fit(X_train, y_train)

best_rf = grid_search.best_estimator_
print('Лучшие параметры:', grid_search.best_params_)
```

```
Лучшие параметры: {'max_depth': None, 'min_samples_split': 2, 'n_estimators':
100}
```

Модель случайного леса показала наилучшие результаты при использовании 100 деревьев, отсутствии ограничения на глубину и минимальном количестве выборок для разбиения узла, равном двум. Эти параметры обеспечивают достаточную сложность модели для захвата закономерностей в данных, при этом сохраняя её обобщающую способность благодаря кросс-валидации. Полученные значения гиперпараметров свидетельствуют о том, что модель не склонна к сильному переобучению и эффективно минимизирует ошибку на проверочных данных.

```
from sklearn.model_selection import GridSearchCV

# Параметры для подбора
param_grid_gb = {
    'n_estimators': [100, 200],
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 5]
}

# Поиск лучших параметров
grid_search_gb = GridSearchCV(GradientBoostingRegressor(random_state=42),
                              param_grid_gb,
                              cv=5,
                              scoring='neg_mean_absolute_error')
grid_search_gb.fit(X_train, y_train)
```

```
# Вывод результатов
print('Лучшие параметры GB:', grid_search_gb.best_params_)
print('MAE на валидации:', -grid_search_gb.best_score_)
```

```
Лучшие параметры GB: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}
MAE на валидации: 0.08572060706194322
```

Результаты показывают, что лучшие параметры для модели — это n_estimators=100, learning_rate=0.1 и max_depth=5, которые обеспечивают наилучшее качество прогнозов с MAE равным 0.08572 на валидационных данных.

Оценка оптимизированных моделей

```
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.dummy import DummyRegressor

# --- 1. Обучение baseline-моделей (по умолчанию) ---
rf_baseline = RandomForestRegressor(random_state=42)
gb_baseline = GradientBoostingRegressor(random_state=42)

rf_baseline.fit(X_train, y_train)
gb_baseline.fit(X_train, y_train)

# --- 2. Обучение моделей с найденными оптимальными гиперпараметрами ---
rf_best = RandomForestRegressor(n_estimators=100, max_depth=None,
min_samples_split=2, random_state=42)
gb_best = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,
max_depth=5, random_state=42)

rf_best.fit(X_train, y_train)
gb_best.fit(X_train, y_train)

# --- 3. Предсказания на тестовой выборке ---
y_pred_rf_base = rf_baseline.predict(X_test)
y_pred_gb_base = gb_baseline.predict(X_test)
y_pred_rf_best = rf_best.predict(X_test)
y_pred_gb_best = gb_best.predict(X_test)

# --- 4. Оценка качества моделей ---
def evaluate_model(y_true, y_pred, model_name):
    mae = mean_absolute_error(y_true, y_pred)
    rmse = mean_squared_error(y_true, y_pred, squared=False)
    r2 = r2_score(y_true, y_pred)
    print(f"{model_name}: MAE={mae:.5f}, RMSE={rmse:.5f}, R²={r2:.5f}")
```

```python
print("Baseline модели:")
evaluate_model(y_test, y_pred_rf_base, "Random Forest (baseline)")
evaluate_model(y_test, y_pred_gb_base, "Gradient Boosting (baseline)")

print("\nМодели с оптимальными гиперпараметрами:")
evaluate_model(y_test, y_pred_rf_best, "Random Forest (настроенный)")
evaluate_model(y_test, y_pred_gb_best, "Gradient Boosting (настроенный)")
```

```
Baseline модели:
Random Forest (baseline): MAE=0.10310, RMSE=0.22645, R²=0.70117
Gradient Boosting (baseline): MAE=0.12158, RMSE=0.24565, R²=0.64835

Модели с оптимальными гиперпараметрами:
Random Forest (настроенный): MAE=0.10310, RMSE=0.22645, R²=0.70117
Gradient Boosting (настроенный): MAE=0.07898, RMSE=0.21539, R²=0.72964
```

Результаты показывают, что настроенная модель градиентного бустинга (Gradient Boosting) с оптимальными гиперпараметрами демонстрирует значительное улучшение по сравнению с базовой версией модели. В частности, её MAE снизился на 35% (с 0.12158 до 0.07898), RMSE также уменьшился, а коэффициент детерминации R^2 увеличился с 0.64835 до 0.72964. С другой стороны, модель случайного леса (Random Forest) не изменила свои метрики качества после настройки гиперпараметров, что может указывать на то, что параметры по умолчанию уже были достаточно эффективными для этой модели.

Формирование выводов

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import learning_curve

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# 1. Сбор метрик
metrics = {
    "Random Forest (baseline)": {
        "MAE": mean_absolute_error(y_test, y_pred_rf_base),
        "RMSE": np.sqrt(mean_squared_error(y_test, y_pred_rf_base)),
        "R2": r2_score(y_test, y_pred_rf_base)
    },
    "Gradient Boosting (baseline)": {
        "MAE": mean_absolute_error(y_test, y_pred_gb_base),
        "RMSE": np.sqrt(mean_squared_error(y_test, y_pred_gb_base)),
        "R2": r2_score(y_test, y_pred_gb_base)
    },
    "Random Forest (настроенный)": {
        "MAE": mean_absolute_error(y_test, y_pred_rf_best),
        "RMSE": np.sqrt(mean_squared_error(y_test, y_pred_rf_best)),
        "R2": r2_score(y_test, y_pred_rf_best)
    },
    "Gradient Boosting (настроенный)": {
        "MAE": mean_absolute_error(y_test, y_pred_gb_best),
        "RMSE": np.sqrt(mean_squared_error(y_test, y_pred_gb_best)),
        "R2": r2_score(y_test, y_pred_gb_best)
    }
}

# 2. Визуализация метрик
plt.figure(figsize=(14, 5))

# MAE
plt.subplot(1, 3, 1)
sns.barplot(x=[m['MAE'] for m in metrics.values()], y=list(metrics.keys()),
palette="Blues_d")
plt.title("Средняя абсолютная ошибка (MAE)")
plt.xlabel("MAE")

# RMSE
plt.subplot(1, 3, 2)
sns.barplot(x=[m['RMSE'] for m in metrics.values()], y=list(metrics.keys()),
palette="Greens_d")
plt.title("Корень из средней квадратичной ошибки (RMSE)")
plt.xlabel("RMSE")
```

```
# R²
plt.subplot(1, 3, 3)
sns.barplot(x=[m['R2'] for m in metrics.values()], y=list(metrics.keys()),
palette="Purples_d")
plt.title("Коэффициент детерминации (R²)")
plt.xlabel("R²")

plt.tight_layout()
plt.show()

# 3. Важность признаков для настроенных моделей
def plot_feature_importances(model, name, feature_names):
    importances = model.feature_importances_
    indices = np.argsort(importances)[::-1]
    plt.figure(figsize=(10, 6))
    plt.title(f"Важность признаков ({name})")
    plt.bar(range(len(importances)), importances[indices], align="center",
color='skyblue')
    plt.xticks(range(len(importances)), [feature_names[i] for i in indices],
rotation=90)
    plt.tight_layout()
    plt.show()

plot_feature_importances(rf_best, "Random Forest (настроенный)", X.columns)
plot_feature_importances(gb_best, "Gradient Boosting (настроенный)", X.columns)

# 4. Кривые обучения для Gradient Boosting
def plot_learning_curve(estimator, title, X, y):
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=5, scoring='neg_mean_squared_error',
        train_sizes=np.linspace(0.1, 1.0, 10), n_jobs=-1
    )
    train_scores_mean = -train_scores.mean(axis=1)
    test_scores_mean = -test_scores.mean(axis=1)

    plt.figure(figsize=(8, 5))
    plt.plot(train_sizes, train_scores_mean, label="Ошибка на обучении",
color="r")
    plt.plot(train_sizes, test_scores_mean, label="Ошибка на валидации",
color="g")
    plt.title(title)
    plt.xlabel("Размер обучающей выборки")
    plt.ylabel("MSE")
    plt.legend()
    plt.grid(True)
    plt.show()

plot_learning_curve(gb_best, "Кривая обучения: Gradient Boosting", X_train,
y_train)
```

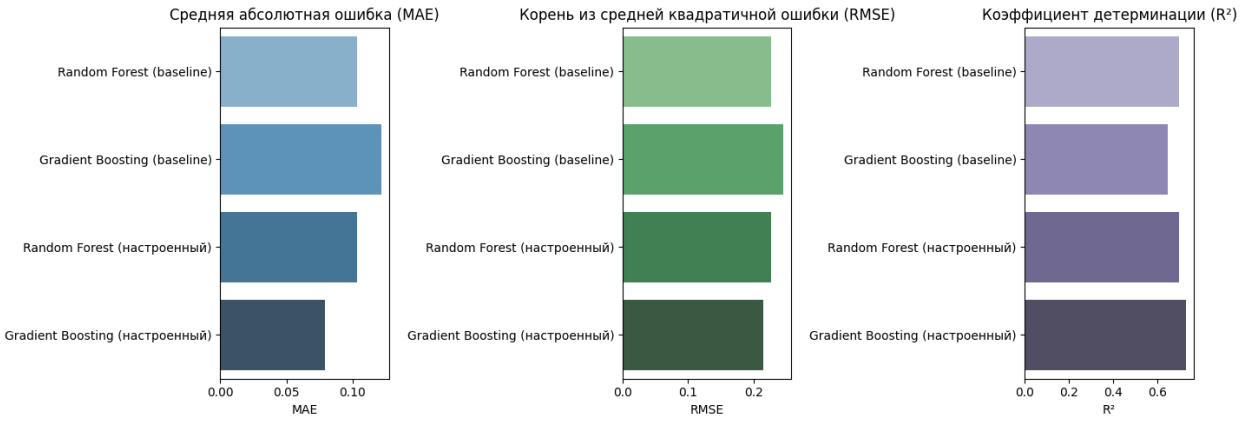

Рисунок 7 – Графики сравнения качества моделей

На графиках представлено сравнение качества моделей Random Forest и Gradient Boosting с базовыми и настроенными гиперпараметрами. Модель Gradient Boosting с оптимальными параметрами демонстрирует существенное улучшение по всем метрикам — снижение MAE и RMSE, а также увеличение R^2 до уровня 0.7, что свидетельствует о её высокой предсказательной способности. В то же время, модель Random Forest не показывает значимых изменений в метриках после настройки, сохраняя стабильное качество как с оптимальными, так и с базовыми параметрами. Таким образом, Gradient Boosting с подобранными гиперпараметрами становится наиболее эффективным решением для данной задачи регрессии.

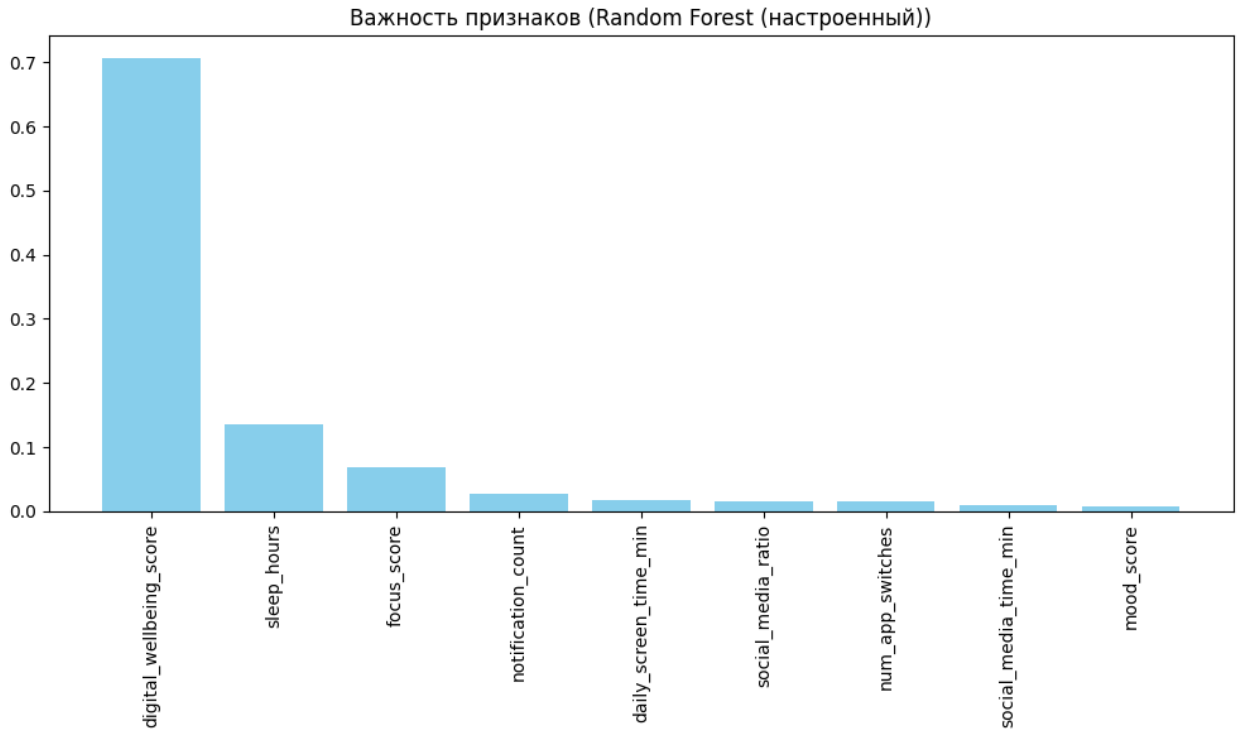

Рисунок 8– График важности признаков для настроенного Random Forest

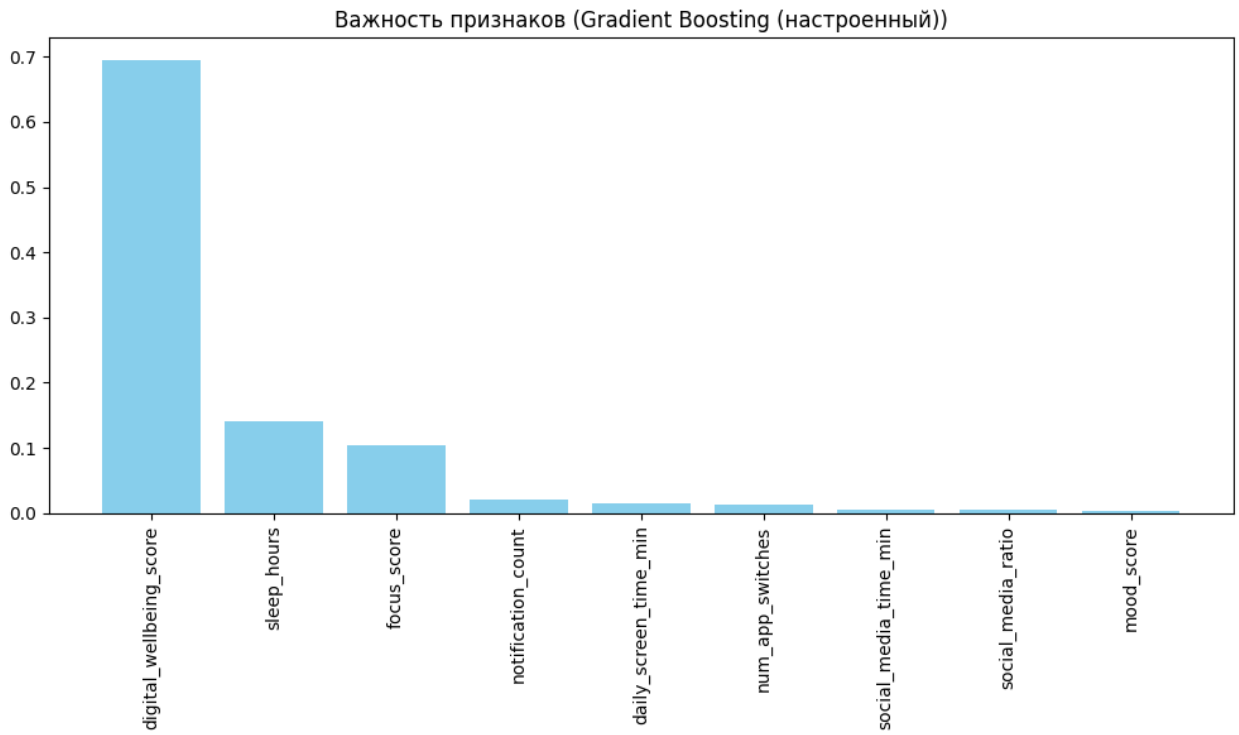

Рисунок 9 – График важности признаков для настроенного Gradient Boosting

На рисунках 8 и 9 можем заметить, что модели Random Forest и Gradient Boosting с настроенными гиперпараметрами выявили одинаковую структуру важности признаков. Обе модели считают «digital_wellbeing_score» ключевым фактором, а остальные признаки имеют незначительное или отсутствующее влияние на качество прогнозов. Это указывает на то, что «digital_wellbeing_score» играет доминирующую роль в объяснении целевой переменной «anxiety_level», которую мы предсказываем с помощью моделей.

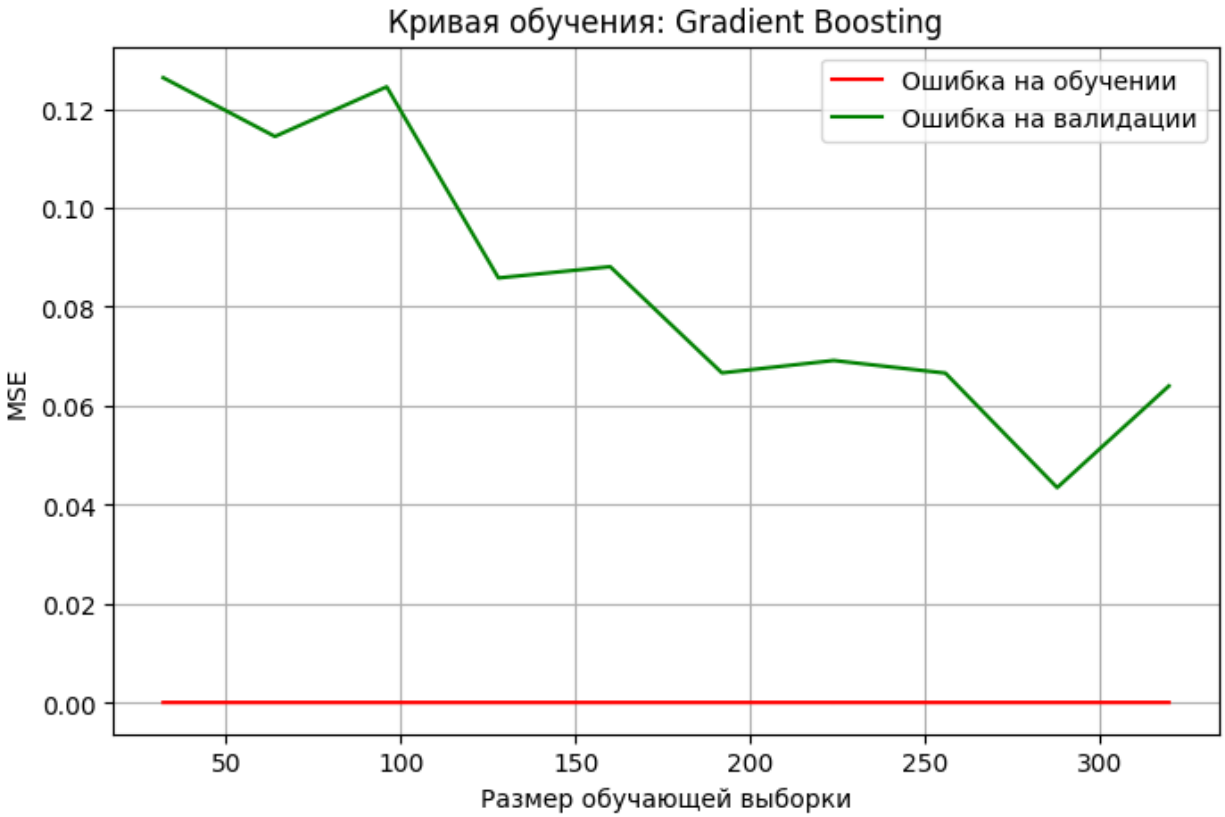

Рисунок 10 – График кривой обучения для Gradient Boosting

Рисунок 10 показывает, что ошибка на обучающей выборке стремится к нулю с увеличением размера обучающей выборки, в то время как ошибка на валидационной выборке снижается до определённого уровня (около 0.4–0.6) и затем начинает расти. Это указывает на переобучение модели: модель слишком хорошо подстраивается под обучающую выборку, но её способность обобщать на новые данные ухудшается при больших размерах обучающей выборки. Для достижения оптимального баланса между обучением и обобщающей способностью рекомендуется использовать размер обучающей выборки, где MSE на валидации достигает своего минимума.

Заключение

В ходе исследования были построены и сравнены модели машинного обучения для прогнозирования уровня тревожности на основе данных о цифровом поведении и образе жизни. Были реализованы baseline-модели случайного леса и градиентного бустинга, после чего проведена настройка гиперпараметров с использованием GridSearchCV. Анализ показал, что модель градиентного бустинга с оптимальными параметрами (`n_estimators=100`, `learning_rate=0.1`, `max_depth=5`) демонстрирует наилучшие результаты: значение MAE уменьшилось примерно на 35%, а коэффициент детерминации R^2 достиг уровня 0.73, что говорит о высокой точности предсказаний. Модель случайного леса не изменила свои метрики после настройки, что свидетельствует о её стабильности и эффективности уже с базовыми параметрами.

Анализ важности признаков выявил, что ключевым фактором в прогнозировании уровня тревожности стал «digital_wellbeing_score» — он оказался значительно важнее других признаков. Остальные переменные оказали незначительное влияние на предсказания моделей. Кривые обучения указали на наличие переобучения у градиентного бустинга при больших объёмах данных, поэтому рекомендуется использовать размер выборки, при котором ошибка на валидации минимальна.

Полученные модели и результаты могут быть использованы в практических задачах психологии и здравоохранения, например в мобильных приложениях для мониторинга психического состояния пользователей или для создания персонализированных рекомендаций по улучшению благополучия на основе индивидуальных данных об использовании гаджетов и образе жизни.

Таким образом, исследование позволило построить эффективную модель прогнозирования уровня тревожности и выделить ключевые факторы, которые могут быть использованы в дальнейшей аналитике и разработке решений для поддержания психоэмоционального здоровья.

Список используемых источников

1. Библиотека Numpy. Полезные инструменты / [Электронный ресурс] // Devpractice : [сайт]. — URL: https://devpractice.ru/ (дата обращения: 19.05.2025).
2. Абдрахманов М. И. Pandas. Работа с данными / Абдрахманов М. И. — 2-е изд. — Москва: Devpractive Team, 2020.
3. Devpractice Team. Python. Визуализация данных. Matplotlib. Seaborn. Mayavi. - devpractice.ru, 2020 - 412 с.
4. Pyplot tutorial / [Электронный ресурс] // Matplotlib. — URL: https://matplotlib.org/stable/tutorials/pyplot.html (дата обращения: 20.05.2025).
5. How to Combine Pandas, NumPy, and Scikit-learn Seamlessly / [Электронный ресурс] // Machine Learning Mastery. — URL: https://machinelearningmastery.com/10-underrated-books-for-mastering-machine-learning/ (дата обращения: 20.05.2025).
6. Визуализация данных / [Электронный ресурс] // Hexlet. — URL: https://ru.hexlet.io/courses/python-pandas/lessons/visual-tips/theory_unit (дата обращения: 20.05.2025).