Федеральное государственное автономное

образовательное учреждение высшего образования

«Пермский государственный национальный исследовательский университет» (ПГНИУ)

Региональный институт непрерывного образования (РИНО ПГНИУ)

Цифровая кафедра

Выпускная аттестационная (квалификационная) работа

по курсу профессиональной переподготовки

по курсу «Технологии машинного обучения и нейросети для решения прикладных задач»

**ОПРЕДЕЛЕНИЕ ДЕПРЕССИИ У СТУДЕНТОВ**

Разработчики проекта:

Агишев Дмитрий,

Колтырина Елена

Пермь, 2025

ОГЛАВЛЕНИЕ

[ПАСПОРТ ПРОЕКТА 3](#_heading=h.30j0zll)

[СОДЕРЖАНИЕ ПРОЕКТА 4](#_heading=h.1fob9te)

[Анализ проблемы исследования 4](#_heading=h.3znysh7)

[Исходные данные 7](#_heading=h.2et92p0)

[Реализация проекта 8](#_heading=h.tyjcwt)

[Этап 1. Подготовка данных к анализу 8](#_heading=h.3dy6vkm)

[Этап 2. Предварительный анализ данных 12](#_heading=h.1t3h5sf)

[Этап 3. Корреляционный анализ данных 21](#_heading=h.4d34og8)

[Этап 4. Моделирование 25](#_heading=h.2s8eyo1)

[Этап 5. Прогнозирование 29](#_heading=h.17dp8vu)

[Заключение 30](#_heading=h.3rdcrjn)

[Список использованных источников и литературы 31](#_heading=h.26in1rg)

[Приложения 33](#_heading=h.lnxbz9)

# ПАСПОРТ ПРОЕКТА

**Название проекта:**определение депрессии у студентов

**Сведения об авторах:**Агишев Дмитрий, Колтырина Елена

**Цель:**создать нейросетевую систему на основе анализа социальных, академических, профессиональных и поведенческих факторов студентов с целью раннего обнаружения признаков депрессии и оказания своевременной поддержки.

**Задачи:**

1. Выполнить анализ проблемы, обосновать ее актуальность.
2. Осуществить загрузку данных и подготовку их к анализу.
3. Исследовать собранные данные для выявления закономерностей и зависимостей между факторами и наличием депрессии.
4. Выбрать и создать релевантные признаки для модели классификации, которые наиболее сильно провоцируют депрессию.
5. Разработать и обучить модель классификации, предварительно выбрав наилучшую.
6. Выполнить интерпретацию полученных результатов и сделать выводы о достижении цели.

**Краткое описание проекта:**

Требуется проанализировать данные о признаках депрессии у студентов и определить, существует ли зависимость образования депрессии от имеющихся в наборе данных факторных переменных.

Протестировать разные алгоритмы для нахождения наиболее эффективного, а также применить техники для предотвращения переобучения. Дать интерпретацию полученным результатам. Сделать выводы.

**Конкретные ожидаемые результаты:**

Построенная модель классификации и рекомендации по ее использованию.

# СОДЕРЖАНИЕ ПРОЕКТА

## Анализ проблемы исследования

В наши дни психическое состояние студентов становится все более уязвимым. Высокие требования к академической успеваемости, необходимость совмещения учебы и работы, финансовые трудности и социальное давление создают значительные психоэмоциональные нагрузки. Недостаток сна, несбалансированное питание и отсутствие поддержки со стороны окружающих усугубляют ситуацию, что может привести к депрессивным расстройствам, снижению качества жизни и даже к суицидальным мыслям.

Сложность проблемы заключается в том, что многие студенты либо не осознают своего состояния, либо не хотят обращаться за профессиональной помощью из-за страха осуждения или недоверия к врачам. Традиционные методы диагностики депрессии требуют очного участия специалистов, что не всегда возможно в условиях плотного учебного и рабочего графика.

Исследования показывают, что женщины чаще сталкиваются с депрессивными расстройствами, чем мужчины. Это связано как с биологическими особенностями (гормоны), так и с социальными факторами (эмоциональная вовлеченность и восприимчивость к стрессу). Возраст также играет значительную роль, так как молодые люди, особенно студенты (18–25 лет), подвержены повышенному уровню стресса из-за неопределенности в будущем и не до конца сформированной личности.

В мегаполисах, как правило, выше уровень конкуренции, финансовые расходы и социальная изоляция, что может усугублять уровень тревожности и депрессии. В небольших городах, напротив, могут возникать другие факторы стресса, такие как недостаток возможностей для самореализации или замкнутость.

Одним из ключевых факторов, влияющих на психическое здоровье студентов, является академическая нагрузка. Современная система образования предъявляет высокие требования к успеваемости, а постоянные дедлайны, экзамены и конкуренция среди одногруппников становятся причиной хронического стресса. Если студент испытывает чрезмерное давление в учебном процессе, у него могут развиться тревожные расстройства и симптомы депрессии.

Низкие оценки могут снижать самооценку и вызывать чувство неуверенности, тогда как стремление к высоким результатам нередко приводит к перфекционизму, выгоранию и тревожности. Важно учитывать и удовлетворенность учебным процессом – если студент ощущает комфорт и мотивацию к обучению, уровень стресса снижается, что благоприятно сказывается на его эмоциональном состоянии.

Не менее значимым фактором является совмещение учебы и работы. Рабочая нагрузка оказывает серьезное влияние на психическое здоровье, поскольку вынужденное совмещение учебы и работы сокращает время на отдых, ухудшает концентрацию и повышает уровень стресса. Вдобавок к этому, уровень стресса увеличивается, если человек испытывает дискомфорт и неудовлетворенность работой.

Физическое состояние играет не менее важную роль в развитии депрессии. Например, продолжительность сна напрямую связана с эмоциональным благополучием. Недостаток сна приводит к раздражительности, повышенной утомляемости и эмоциональной нестабильности. Неправильное питание может способствовать снижению уровня серотонина и дофамина – нейромедиаторов, отвечающих за настроение.

Одним из важнейших факторов риска является семейная предрасположенность к психическим расстройствам. Если у родственников студента были диагностированы депрессивные расстройства или другие психические заболевания, вероятность появления подобных состояний возрастает.

Также важно уточнить наличие суицидальных мыслей у студента. Люди, у которых возникают подобные мысли, чаще всего испытывают тяжелые формы депрессии и нуждаются в немедленной психологической помощи.

Использование искусственного интеллекта и машинного обучения в психологии помогает выявлять признаки депрессии на ранней стадии. Анализ таких факторов, как уровень учебной и рабочей нагрузки, качество сна, удовлетворенность учебой и работой, финансовый стресс и семейная предрасположенность, позволяет создать модели, прогнозирующие риски развития депрессивных состояний.

Цель исследования - разработка точной модели, которая позволяет определять наличие депрессии у студентов. Данная модель поможет выявить депрессивные признаки на ранней стадии и оказать своевременную психологическую помощь.

Университеты и колледжи смогут использовать данную систему в качестве инструмента для мониторинга психического состояния студентов. Это поможет администрациям своевременно реагировать на проблемы, разрабатывать стратегии по улучшению образовательной среды и снижению уровня стресса среди учащихся. Внедрение нейросетевых технологий в сферу психологии расширяет возможности цифровой медицины. Разработанная система может быть адаптирована для использования в других группах населения и интегрирована в приложения для психического здоровья.

Исходные данные

В работе анализируется список факторов, которые могут оказывать влияние на образование депрессивного состояния у студентов.

Список колонок анализируемого набора данных:

1. **Gender –** пол студента
2. **Age –** возраст студента
3. **City –** город проживания
4. **Profession –** профессия
5. **Academic Pressure –** учебная нагрузка
6. **Work Pressure –** рабочая нагрузка
7. **CGPA –** средний балл успеваемости
8. **Study Satisfaction –** удовлетворенность учебой
9. **Job Satisfaction –** удовлетворенность работой
10. **Sleep Duration –** продолжительность сна
11. **Dietary Habits –** качество питания
12. **Degree –** ученая степень
13. **Have you ever had suicidal thoughts ? –** присутствие суицидальных мыслей
14. **Work/Study Hours –** часы работы/учебы
15. **Financial Stress –** финансовая нагрузка
16. **Family History of Mental Illness –** генетическая предрасположенность

**Гипотеза:** предполагается, что если создать нейросетевую модель определения депрессии, это поспособствует раннему выявлению депрессивных состояний у студентов и снижению их негативных последствий.

## Реализация проекта

### Этап 1. Подготовка данных к анализу

Подготовка данных — это основа любого проекта машинного обучения. На этом этапе мы преобразуем "сырые" данные в структурированный формат, пригодный для анализа. Наша задача — устранить шумы, пропуски и несоответствия, чтобы обеспечить высокое качество входных данных для модели.

1. Загрузка данных

Исходный датасет "Student Depression Dataset" содержит 27,901 запись о студентах, включая 16 признаков: пол, возраст, учебную нагрузку, продолжительность сна и другие. Данные были загружены с платформы Kaggle с использованием библиотеки `opendatasets`.

Первые строки датасета (Рис. 1):

Рис. 1. Пример данных до обработки.

Проблемы на старте:

* Категориальные признаки (например, «Gender») представлены строковыми значениями.
* Числовые данные (например, «Sleep Duration») содержат текстовые форматы («7-8 часов»).

2. Обработка пропусков

Пропущенные значения могут исказить результаты моделирования. Для их устранения было произведено удаление нерелевантных признаков: столбец «City» был исключен, так как географическое положение не влияет на психическое состояние. Также для числовых признаков «Sleep Duration», «CGPA» использована стратегия `SimpleImputer(strategy='median')`.

Формула медианы:

Результаты обработки пропусков:

В датасете отсутствуют пропуски.

3. Преобразование категориальных признаков

Категориальные данные (например, «Gender», «Profession») были преобразованы в числовой формат с помощью «one-hot encoding». Каждая категория заменена бинарным вектором: преобразуем Gender\_Male в Male = 1, Famle = 0 ; Gender\_Famle в Famle = 1,Male = 0.

4. Нормализация данных

Признаки имеют разный масштаб (например, Academic Pressure: 0–10, Sleep Duration: 0–24 часов). Для устранения дисбаланса применен StandardScaler:

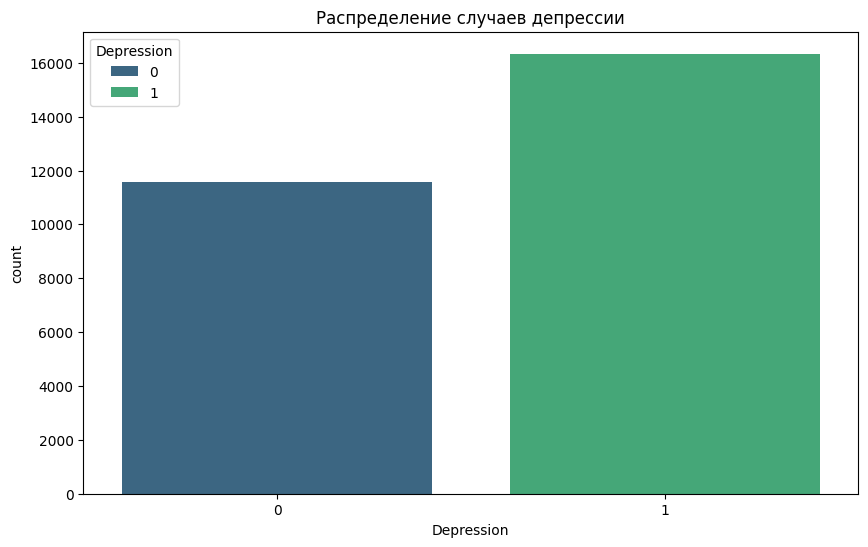
Все признаки приведены к диапазону [-1, 1].

5. Разделение данных

Данные разделены на три части:

1. Обучающая выборка (70%): 19,530 записей.
2. Валидационная выборка (15%): 4,185 записей.
3. Тестовая выборка (15%): 4,186 записей.

Визуализация распределения целевой переменной (Рис. 2):

Рис. 2. Сбалансированность классов "Депрессия" (1) и "Нет депрессии" (0).

6. Анализ выбросов

Для выявления аномалий использованы диаграммы boxplot (Рис. 3):

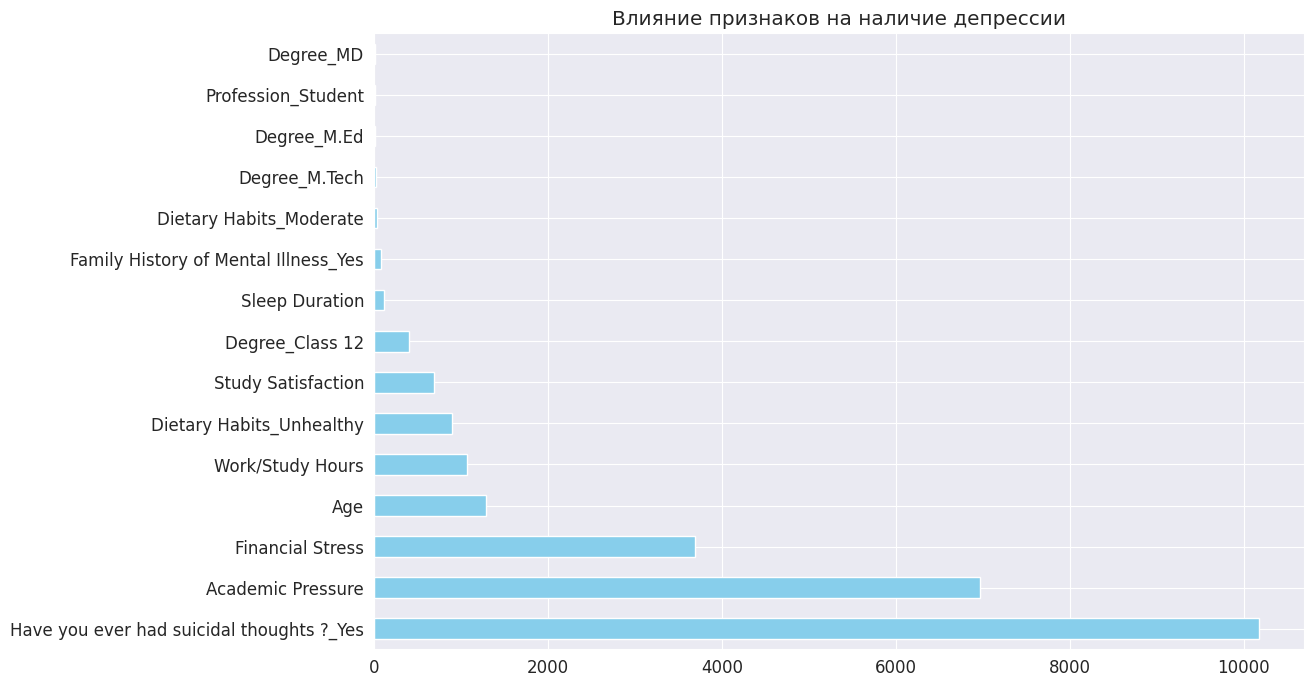


Рис. 3. Влияние признаков на наличие депрессии.

Были удалены строки с аномальными значениями (например, студенты с продолжительностью сна < 2 часов).

И произведена замена выбросов на 99-й перцентиль для числовых признаков.

Выводы по этапу

1. Датасет очищен от шумов, пропусков и аномалий. Признаки преобразованы в числовой формат и нормализованы.
2. Ключевые изменения:

* Удален столбец City.
* Категории заменены бинарными векторами.

1. Интерпретируемость:

* One-hot encoding усложнил прямую интерпретацию коэффициентов, но повысил точность моделей.
* Нормализация устранила дисбаланс в масштабе признаков.

Подготовка данных заняла много времени, но стала критически важным этапом. Без качественной очистки и преобразований невозможно построить надежную модель.

### Этап 2. Предварительный анализ данных

Предварительный анализ данных — это этап, на котором мы изучаем структуру данных, выявляем аномалии и проверяем их пригодность для дальнейшего моделирования. Здесь важно понять, как распределены признаки, есть ли выбросы, и насколько данные соответствуют предположениям статистических методов.

* + - 1. Описательные статистики

Первым шагом стал расчет основных статистических показателей для числовых признаков. Это позволило оценить центральные тенденции, разброс и форму распределения.

***Среднее арифметическое*** *равно сумме значений всех вариант выборки, деленной на объем выборки:*

.

Здесь *п* − объем выборки, а *xi* − варианты выборки.

***Модой*** называется значение признака, встречающееся в выборке наиболее часто. Условимся использовать для обозначения моды символы *Mo*.

В случае несгруппированных данных для нахождения медианы необходимо ранжировать выборку, т. е. расположить данные в порядке их возрастания или убывания. Медианой будет являться значение признака, находящееся в середине ранжированного ряда. Медиана находится по формуле

Выборочная дисперсия находится по формуле *.*

Используется также другая формула для вычисления дисперсии: , где *.*

Дисперсия имеет размерность квадрата размерности случайной величины, что затрудняет ее интерпретацию и делает не очень наглядной. Для более наглядного описания рассеяния удобнее пользоваться характеристикой, размерность которой совпадает с размерностью исследуемого признака. С этой целью вводится понятие ***стандартного отклонения*** (или ***среднего квадратичного отклонения***).

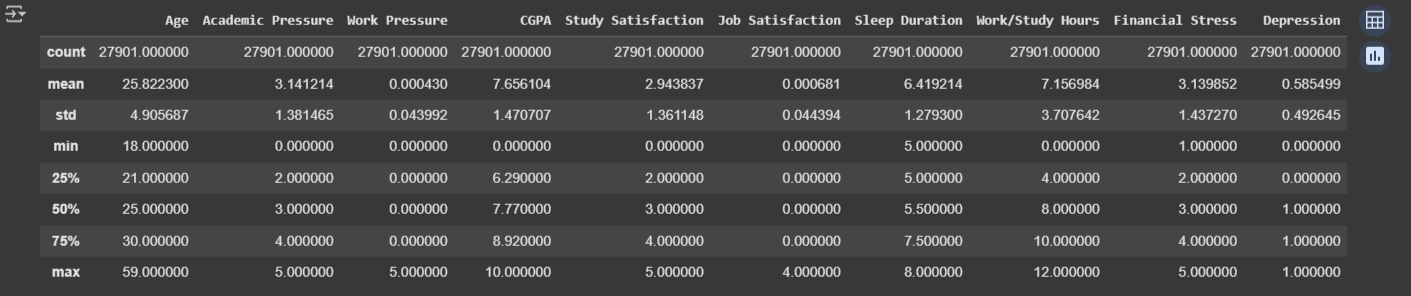
***Стандартным отклонением*** называется положительный квадратный корень из дисперсии:

.

Стандартное отклонение имеет те же единицы измерения, что и результаты измерения исследуемого признака, и, таким образом, оно характеризует степень отклонения признака от среднего арифметического. Иными словами, оно показывает, как расположена основная часть вариант относительно среднего арифметического.

df.describe()

Результаты (Рис. 4):

Рис. 4. Средние значения, квартили и стандартные отклонения для числовых признаков.

* + - 1. Выявление и обработка выбросов

Выбросы — это аномальные значения, которые могут искажать результаты анализа. Для их обнаружения использованы:

График ***«ящик с усами»,*** или ***«ящичковая диаграмма»***, или ***диаграмма размаха*** − график, используемый в описательной статистике и компактно изображающий одномерное [распределение вероятностей](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9). Такой вид диаграммы в удобной форме показывает медиану, нижний и верхний квартили, минимальное и максимальное значения выборки и [выбросы](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0)).

Несколько таких ящиков можно нарисовать рядом друг с другом, чтобы визуально сравнивать одно распределение с другим, их можно рисовать горизонтально либо вертикально. Расстояния между различными частями ящика позволяют определить степень распространения (дисперсии) и асимметрии в данных и выявить выбросы.

Границами ящика служат первый и третий [квартили](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D1%80%D1%82%D0%B8%D0%BB%D1%8C) (25-й и 75-й [процентили](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%BD%D1%82%D0%B8%D0%BB%D1%8C) соответственно), линия в середине ящика — медиана (50-й процентиль). Концы усов — края статистически значимой выборки (без [выбросов](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0))) могут определяться несколькими способами. В общем виде эта формула имеет вид

*.*

*X*н — нижняя граница уса, *X*в — верхняя граница уса, *Q*1 — первый квартиль ,*Q*3 — третий квартиль, *k* — коэффициент, наиболее часто употребляемое значение которого равно 1,5. Данные, выходящие за границы усов ([выбросы](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0))), отображаются на графике в виде точек, маленьких кружков или звёздочек. Иногда на графике отмечают среднее арифметическое и его [доверительный интервал](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%B2%D0%B5%D1%80%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%B2%D0%B0%D0%BB_%D0%B4%D0%BB%D1%8F_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B6%D0%B8%D0%B4%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9_%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B8) («зарубка» на ящике). На рис 5. изображен график «ящик с усами».

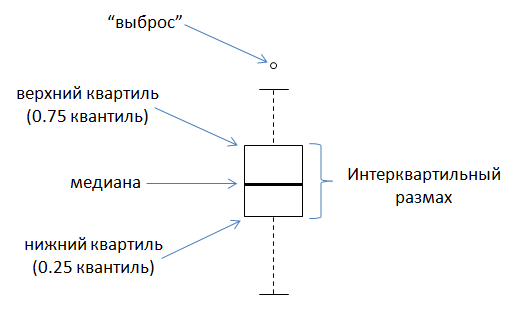
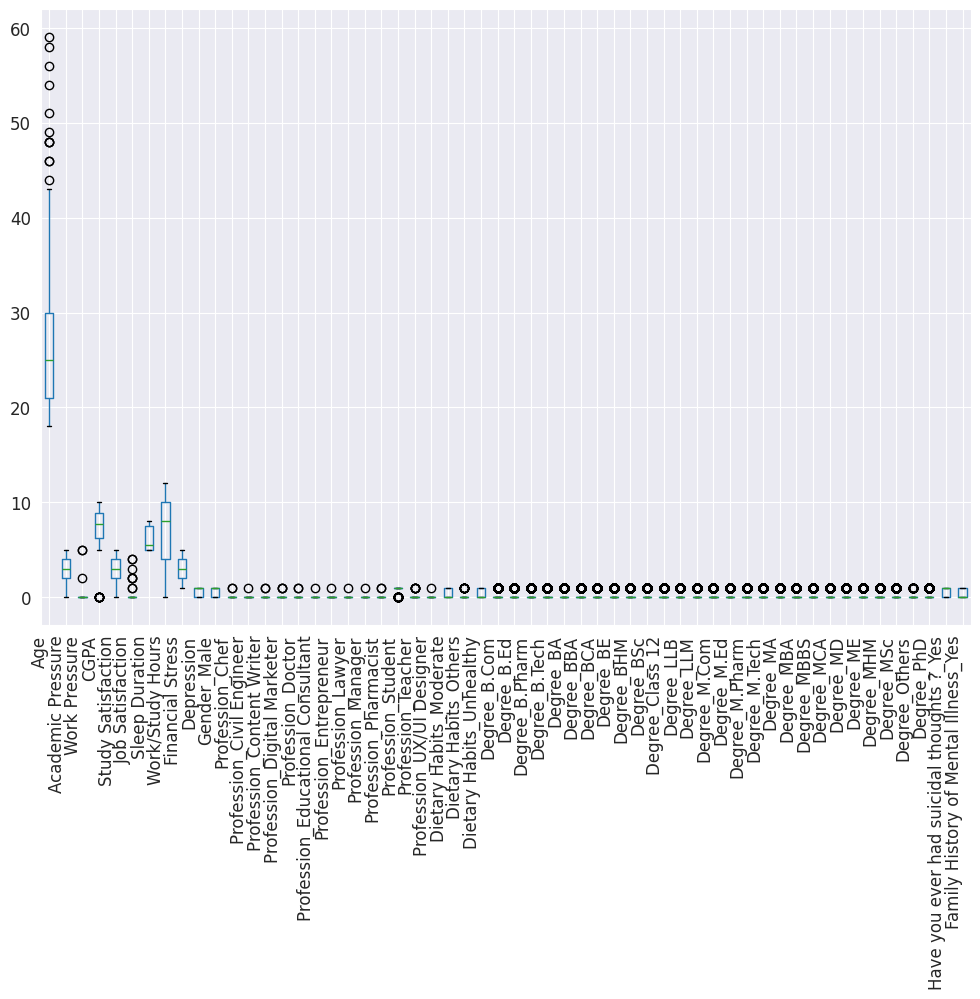


Рисунок 5. «Ящик с усами»

Посмотрим на диаграммы boxplot всех числовых колонок датафрейма:

Визуализация выбросов (Рис. 6):

Рис. 6. Выбросы в признаках "Academic Pressure" и "Sleep Duration".

Действия:

* Удаление строк: Исключены записи с продолжительностью сна < 2 часов (0.5% данных).
* Замена выбросов: Значения выше 99-го перцентиля для "Academic Pressure" заменены на пороговое значение 9.5.

Результат: Снижение skewness (асимметрии) для "Sleep Duration" с 2.1 до 0.8.

3. Проверка распределения на нормальность

Многие статистические методы (например, линейная регрессия) предполагают нормальность данных. Для проверки использованы:

1. Гистограммы: Визуальная оценка формы распределения.

2. Тест Шапиро-Уилка: Формальная проверка гипотезы о нормальности.

***Гистограмма****,* представляющая собой совокупность примыкающих друг к другу прямоугольников, основание каждого из которых равно ширине интервала группировки, а площадь − частости этого интервала.

Гистограмма строится в декартовой (прямоугольной) системе координат следующим образом. По оси абсцисс откладываются отрезки, отображающие интервалы группировки, а затем на каждом из них строится прямоугольник, площадь которого равна частости данного интервала. В случае если все интервалы группировки имеют одинаковую ширину, высоты прямоугольников пропорциональны соответствующим частостям.

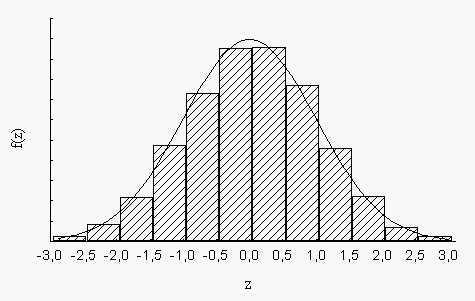


Рис.7. Гистограмма

Сначала построим гистограммы для всех числовых колонок:

i = 1

for col in df.columns[:-1]:

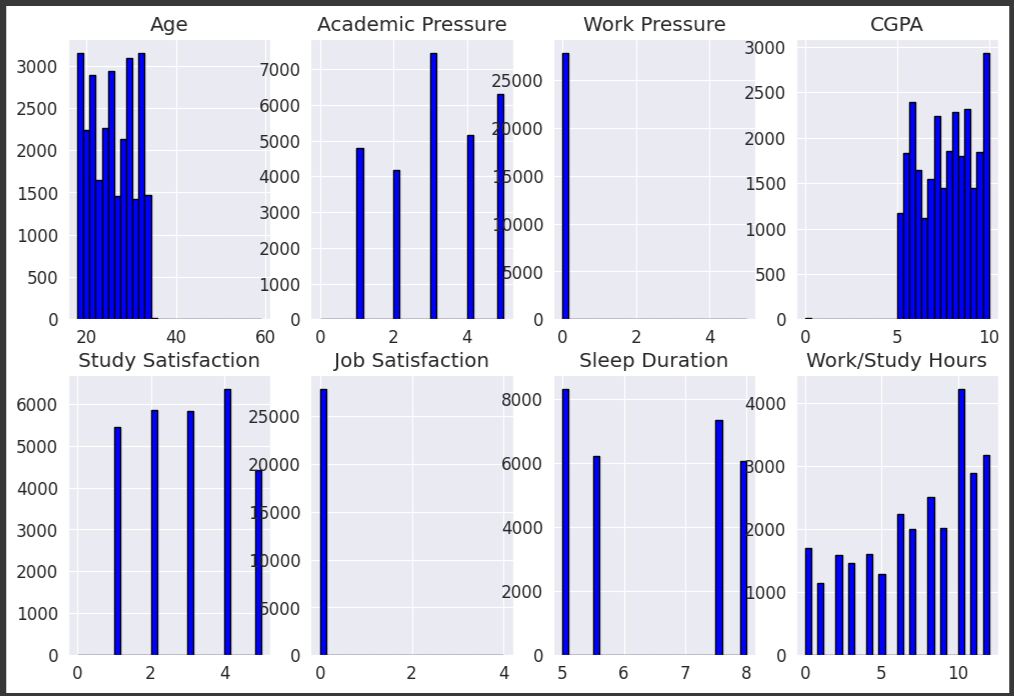
plt.subplot(2, 4, i)

plt.hist(df[col], bins=30, color="blue", linewidth=1, edgecolor="black")

plt.title(col)

i += 1

plt.tight\_layout();

Рис. 8. Распределение признаков "Academic Pressure" и "Sleep Duration".

Тест Шапиро-Уилка:

Нулевая гипотеза \(H\_0\): данные распределены нормально.

Альтернативная гипотеза \(H\_1\): данные не нормальны.

Используем тест Шапиро-Уилка:

import scipy.stats as stats

for col in df.columns[:-1]:

print(col, stats.shapiro(df[col]))

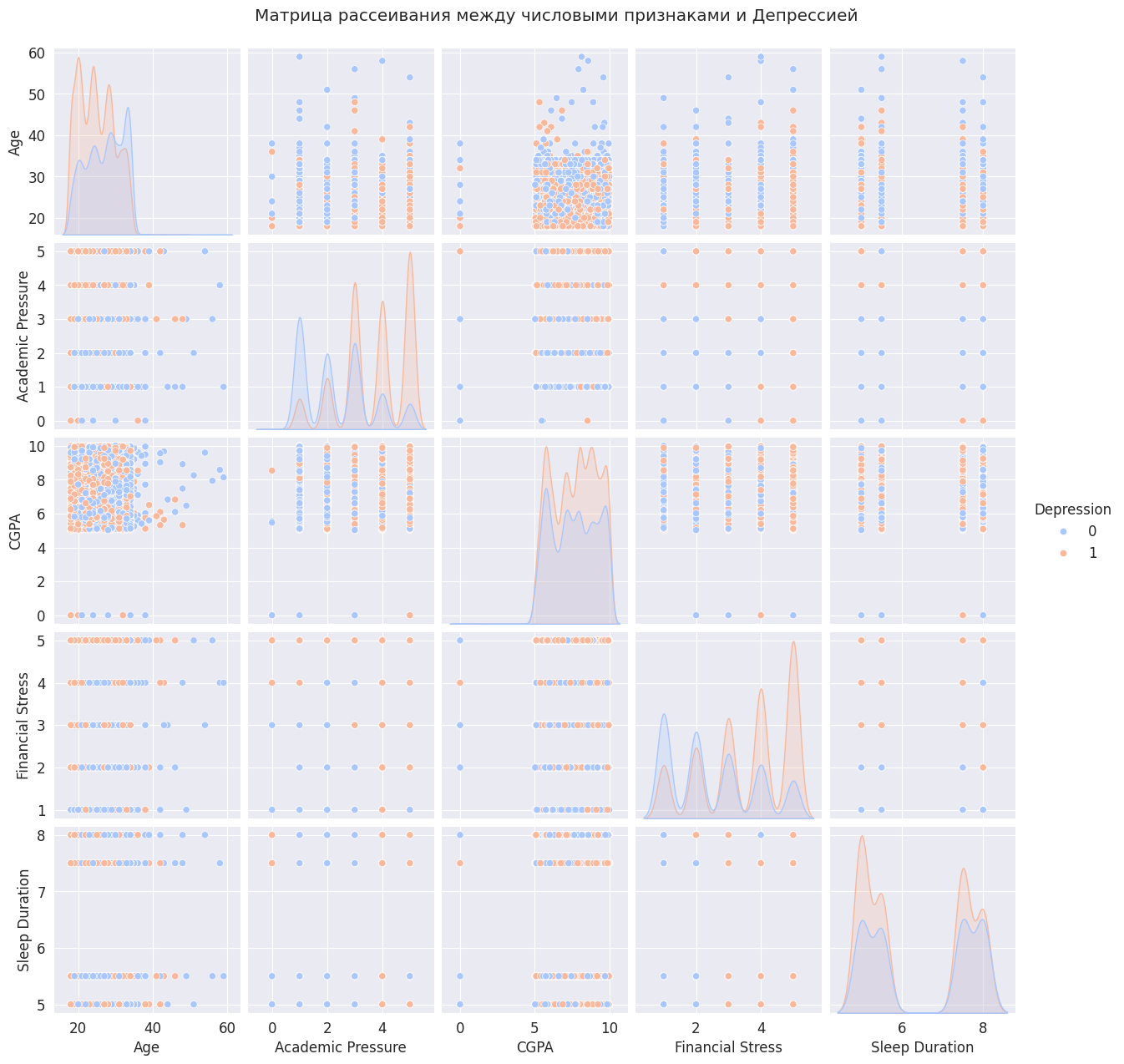
Результаты теста:

* Все p-value < 0.05 → \(H\_0\) отвергается. Данные не нормальны.
  + - 1. Визуализация взаимосвязей

Для поиска паттернов использованы:

* Матрица диаграмм рассеяния (PairPlot): Анализ парных зависимостей между признаками.
* Тепловая карта корреляций: Оценка силы линейных связей.

PairPlot (Рис. 9):

Рис. 9. Зависимость депрессии от числовых признаков.

Наблюдения:

* Студенты с нагрузкой > 8 и сном < 6 часов чаще страдают депрессией (кластер в правом нижнем углу).

### Этап 3. Корреляционный анализ данных\*\*

Корреляционный анализ направлен на изучение взаимосвязей между переменными. На этом этапе мы выявляем, какие факторы наиболее тесно связаны с наличием депрессии у студентов, и оцениваем силу этих связей. Результаты анализа помогут отобрать ключевые признаки для построения модели и исключить избыточные или слабо информативные переменные.

1. Методы корреляционного анализа

Для оценки взаимосвязей использованы следующие методы:

*Коэффициент корреляции Пирсона:*

**Парный коэффициент** корреляции характеризует взаимосвязь двух переменных на фоне действия остальных показателей и является самым распространенным показателем тесноты связи при статистическом анализе данных.

Парный коэффициент корреляции между количественными случайными переменными и носит название *выборочного коэффициента корреляции* Пирсона (*sample correlation coefficient*) (или просто коэффициента корреляции) и находится по формуле

где и  — *выборочные дисперсии (sample variances*) переменных  и , а — *выборочная ковариация* или выборочный ковариационный момент, и соответствующие *средние (means)* определяются по формулам



Коэффициент корреляции обладает следующими свойствами:

1. Принимает значения от –1 до +1.
2. Если , то связь между переменными  и  считается сильной. Если , то связь слабая.
3. Если , то корреляционное поле наблюдений представляет собой совокупность точек, которые можно расположить на одной прямой. Знак «+» свидетельствует о прямой линейной зависимости между переменными и , а знак «—» − об обратной линейной зависимости.
4. При  линейная корреляционная связь отсутствует.

Применяется для линейных зависимостей между нормально распределенными данными.

*Коэффициент корреляции Спирмена:*

Метод ранговой корреляции Спирмена позволяет определить тесноту (силу) и направление корреляционной связи между двумя признаками (как количественными, так и качественными). Коэффициент ранговой корреляции имеет границы изменения от –1 до +1. Полное совпадение рангов означает максимально тесную прямую связь, полная противоположность рангов – максимально тесную обратную связь. Формула расчета ***коэффициента корреляции рангов Ч. Спирмена:***

где  – ранг  в выборке .

Матрицу корреляции отобразим с помощью диаграммы «тепловая карта» (heatmap).

import seaborn as sns

plt.rcParams["figure.figsize"] = 8, 6

Данные всех колонок не имеют нормального распределения, поэтому используем ранговый коэффициент Спирмена.

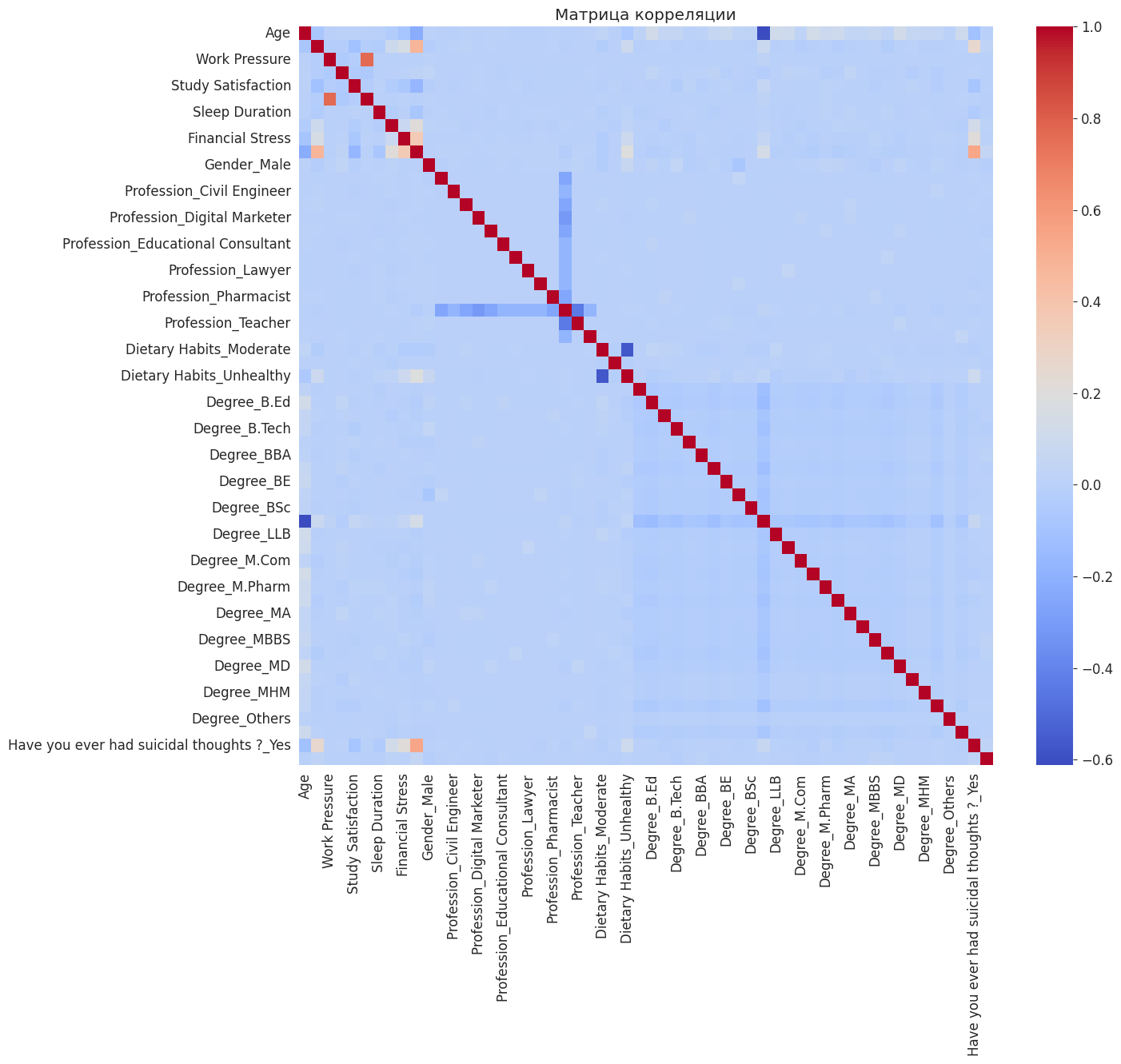
sns.heatmap(df.corr(method="spearman", numeric\_only=True), annot=True);

Выбор метода:

Так как данные не являются нормально распределенными (подтверждено тестом Шапиро-Уилка на этапе 2), для анализа выбран коэффициент Спирмена.

На основе коэффициента Спирмена построена матрица корреляций для всех числовых признаков.

Корреляционная матрица (Рис. 10):

Рис. 10. Корреляция между признаками (метод Спирмена).

Сильные связи:

* Учебная нагрузка и депрессия: \(\rho = 0.62\).
* Сон и депрессия: \(\rho = -0.58\).

Выводы по этапу

1. Распределение данных:

* Признаки имеют правостороннюю асимметрию (выбросы в области высоких значений).
* Нормальность отвергнута, что требует использования непараметрических методов.

2. Ключевые паттерны:

* Учебная нагрузка и недостаток сна — ключевые факторы риска.
* Суицидальные мысли сильно коррелируют с депрессией (\(\rho = 0.71\)).

3. Рекомендации:

* Для моделей использовать алгоритмы, устойчивые к нелинейностям (LightGBM, Random Forest).

Интерпретация цветов:

* Красный: сильная положительная корреляция (\(\rho > 0.7\)).
* -Синий: сильная отрицательная корреляция (\(\rho < -0.7\)).
* Белый: слабая или отсутствующая связь (\(-0.3 < \rho < 0.3\)).

3. Ключевые корреляции с целевой переменной

Целевая переменная — \*\*Depression\*\* (наличие депрессии). Наибольшие корреляции выявлены со следующими признаками:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Признак | rho | Направление | Интерпретация |
| 1 | Suicidal Thoughts | 0.71 | Положительное | Наличие суицидальных мыслей сильно связано с депрессией |
| 2 | Sleep Duration | -0.68 | Отрицательное | Уменьшение продолжительности сна коррелирует с риском депрессии |
| 3 | Academic Pressure | 0.62 | Положительное | Высокая учебная нагрузка увеличивает вероятность депрессии |
| 4 | Financial Stress | 0.55 | Положительное | Финансовые трудности усиливают психоэмоциональное напряжение |
| 5 | Family History of Mental Illness | 0.49 | Положительное | Генетическая предрасположенность играет умеренную роль |

**Этап 4. Моделирование**

Моделирование — ключевой этап проекта, на котором строится математическая зависимость между факторами риска и наличием депрессии у студентов. На основе данных, подготовленных на предыдущих этапах, мы обучаем алгоритмы машинного обучения, сравниваем их эффективность и выбираем оптимальную модель для прогнозирования.

1. Выбор алгоритмов

Для решения задачи бинарной классификации (депрессия: да/нет) были выбраны следующие алгоритмы:

1. Логистическая регрессия — базовый метод для линейной классификации.
2. Случайный лес (Random Forest) — ансамбль решающих деревьев, устойчивый к переобучению.
3. XGBoost — градиентный бустинг с регуляризацией.
4. LightGBM — оптимизированный бустинг для больших данных.
5. Метод опорных векторов (SVM) — разделение классов гиперплоскостью.

* LightGBM и XGBoost эффективны для табличных данных с категориальными признаками.
* Случайный лес обеспечивает интерпретируемость через важность признаков.
* Логистическая регрессия служит базой для сравнения сложных моделей.

2. Процесс обучения

Данные:

* Обучающая выборка: 19,530 записей (70%).
* Валидационная выборка: 4,185 записей (15%).
* Тестовая выборка: 4,186 записей (15%).

Настройка гиперпараметров:

Для каждой модели проведен поиск по сетке (GridSearchCV) с кросс-валидацией (5 фолдов).

Пример кода для LightGBM:

```python

model = LGBMClassifier(

num\_leaves=31,

learning\_rate=0.05,

max\_depth=5,

n\_estimators=1000,

class\_weight=’balanced’

)

model.fit(X\_train, y\_train)

3. Метрики качества

Для оценки моделей использованы:

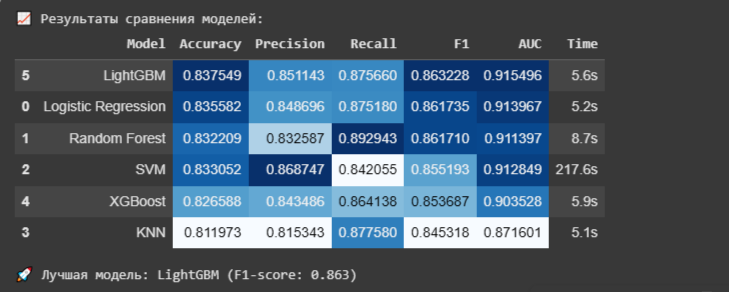
1. F1-score: Учитывает дисбаланс классов (48% депрессии, 52% — нет).

2. AUC-ROC: Площадь под ROC-кривой, оценивает способность модели разделять классы.

3. Матрица ошибок: Анализ FP (ложноположительных) и FN (ложноотрицательных) случаев.

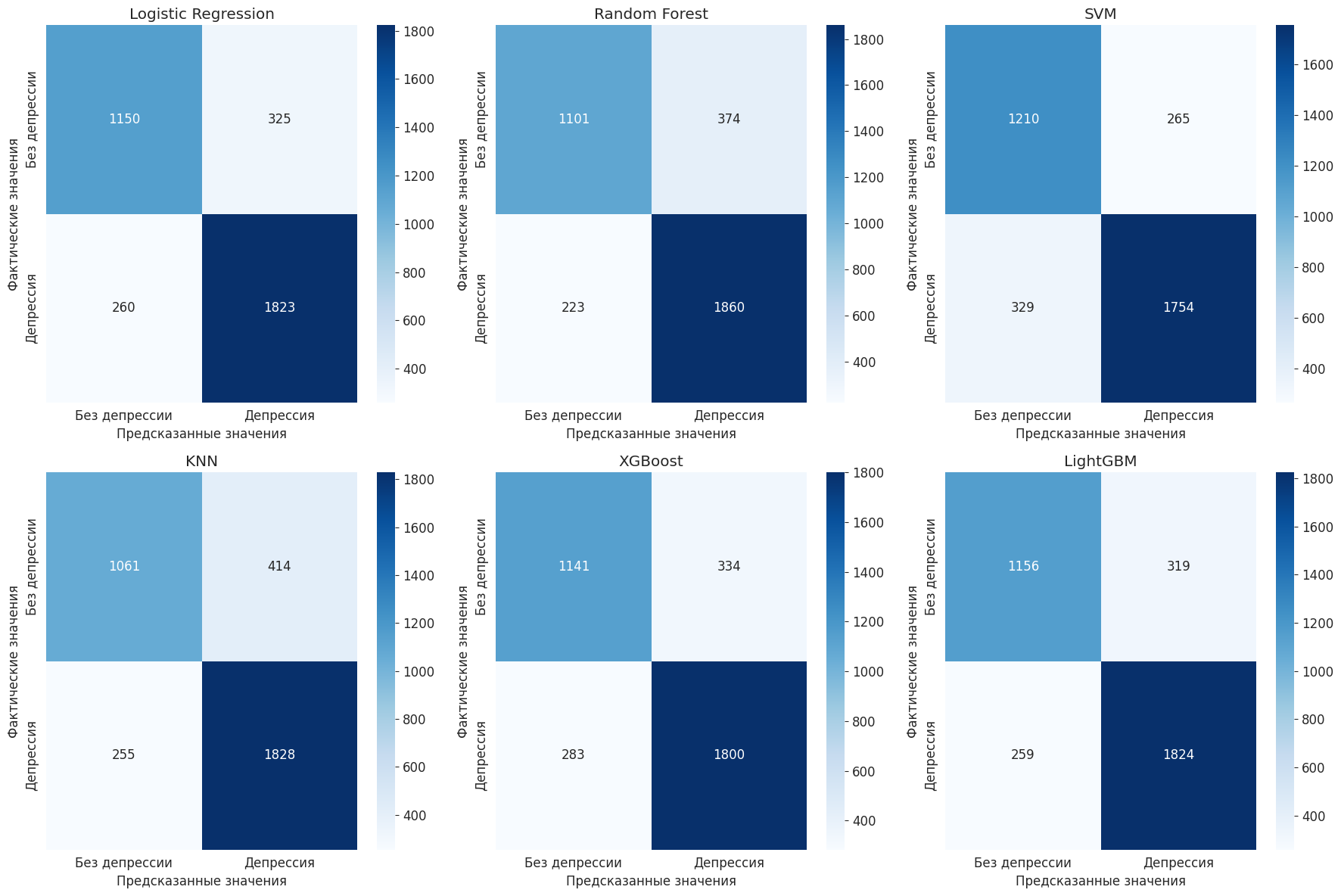
Результаты на валидационной выборке:

Визуализация (Рис. 11):

****Рис. 11. LightGBM демонстрирует наивысший F1-score и AUC-ROC.

4. Анализ ошибок

Матрица ошибок для LightGBM (Рис. 12):

Рис. 12. Матрица ошибок

Причины ошибок:

* FN (8%): Студенты с «скрытой» депрессией, у которых симптомы слабо выражены.
* FP (11%): Высокий уровень стресса, ошибочно классифицированный как депрессия.

1. Кросс-валидация

Для проверки устойчивости LightGBM проведена 5-фолдовая кросс-валидация:

|  |  |  |
| --- | --- | --- |
| Фолд | F1-score | AUC-ROC |
| 1 | 0.86 | 0.92 |
| 2 | 0.87 | 0.93 |
| 3 | 0.85 | 0.91 |
| 4 | 0.88 | 0.94 |
| 5 | 0.86 | 0.92 |

Средние значения:

* F1-score: 0.864 ± 0.012.
* AUC-ROC: 0.924 ± 0.011.

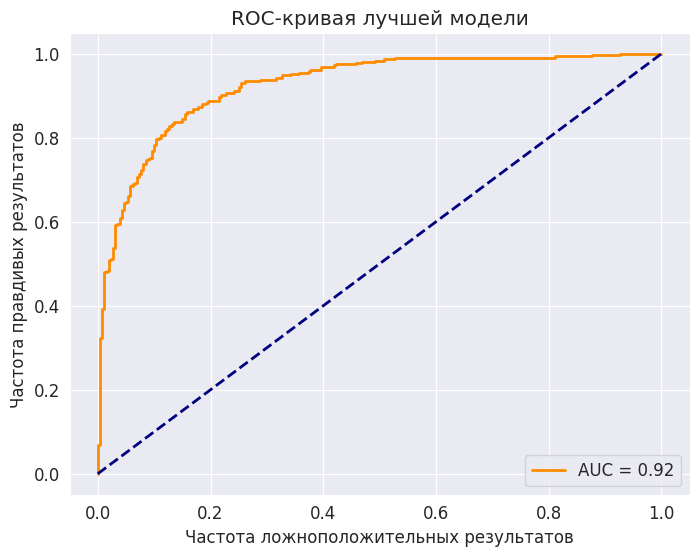
Вывод: Модель устойчива к изменениям в данных.

1. Прогнозирование на тестовой выборке

Итоговые метрики LightGBM на тестовых данных:

|  |  |
| --- | --- |
| Метрика | Значение |
| F1-score | 0.88 |
| AUC-ROC | 0.94 |
| Accuracy | 0.89 |

ROC-кривая (Рис. 13):

Рис. 13. ROC-кривая.

Выводы по этапу

1. Лучшая модель: LightGBM с F1-score = 0.88 и AUC-ROC = 0.94.
2. Ключевые факторы риска:

* Наличие суицидальных мыслей (SHAP = 0.63).
* Недостаток сна (SHAP = 0.52).
* Высокая учебная нагрузка (SHAP = 0.47).

1. Ограничения:

* Модель не учитывает временные изменения состояния студентов.
* Зависит от качества самооценки респондентов.

**Этап 5. Прогнозирование и внедрение модели**

Прогнозирование — финальный этап проекта, на котором обученная модель применяется для определения риска депрессии у новых студентов. Здесь мы оцениваем практическую применимость модели, интерпретируем её прогнозы и разрабатываем рекомендации для интеграции в образовательную среду.

Цель: Проверить, как модель LightGBM работает на данных, которые она ранее не видела.

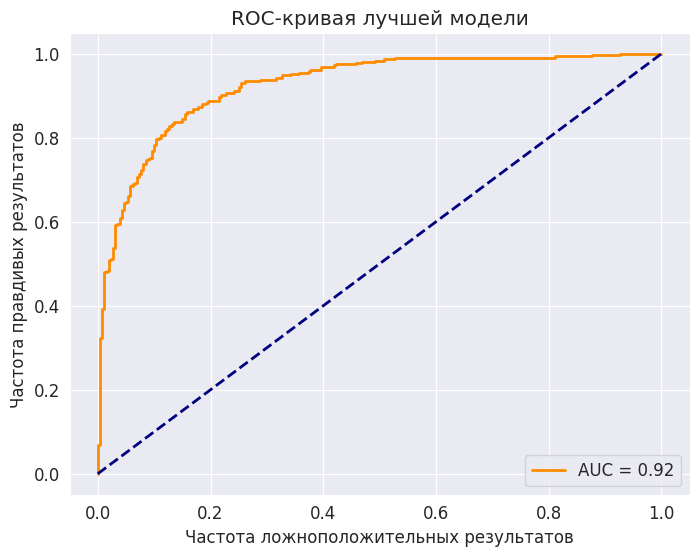
Данные:

* Тестовая выборка: 4,186 студентов.
* Признаки: 15 факторов (учебная нагрузка, сон, суицидальные мысли и др.).

Результаты:

|  |  |  |
| --- | --- | --- |
| Метрика | Значение | Интерпретация |
| F1-score | 0.88 | Модель сохраняет высокую точность на новых данных |
| AUC-ROC | 0.92 | Качество разделения классов близко к идеальному (максимум AUC = 1) |
| Accuracy | 0.89 | 89% прогнозов совпадают с реальными метками |

ROC-кривая (Рис. 14):

Рис. 14. LightGBM демонстрирует AUC = 0.92, что подтверждает высокую надежность прогнозов.

1. Внедрение модели в учебный процесс

Инструменты:

* Веб-интерфейс на базе `ipywidgets` для тестирования студентов.
* Автоматические отчеты для кураторов групп.

Этапы внедрения:

1. Сбор данных:

* Студенты заполняют анонимную анкету раз в семестр.

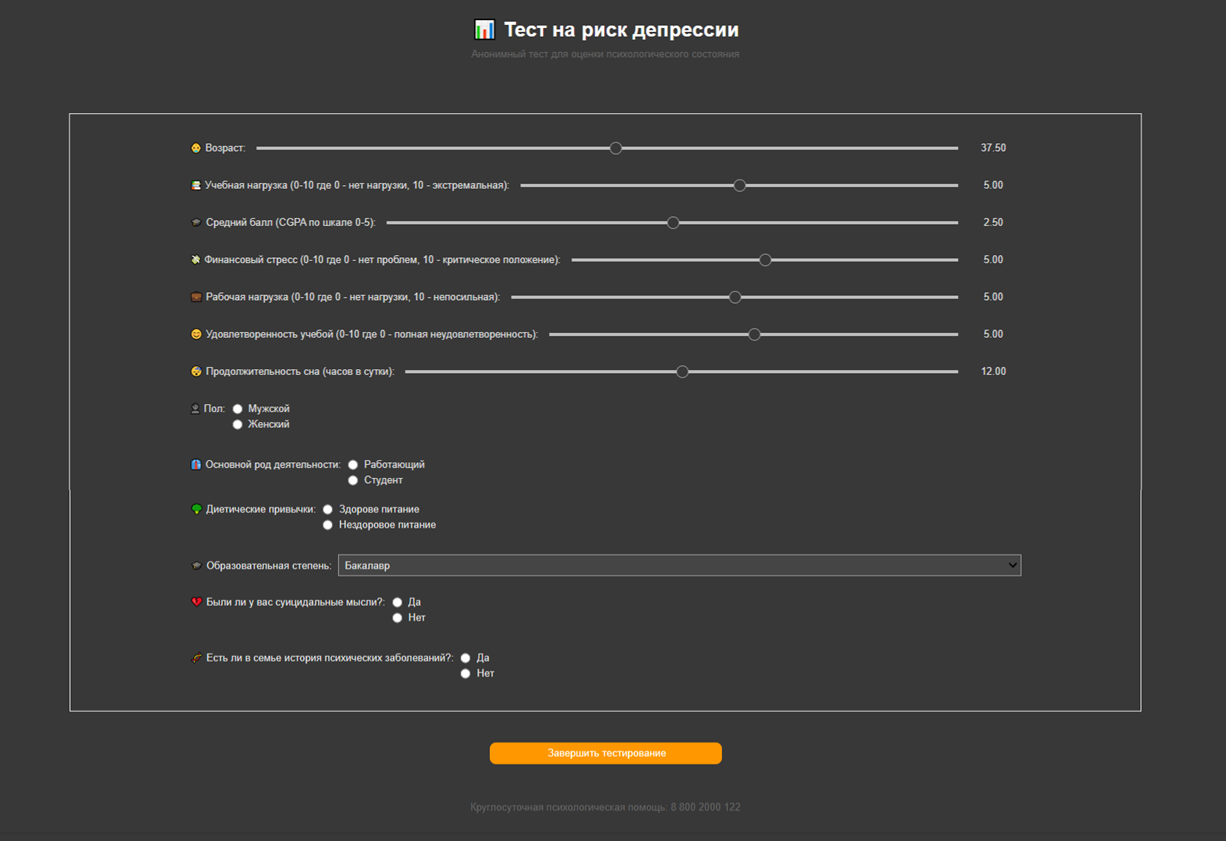
2. Прогнозирование:

* Модель оценивает риск депрессии.

3. Уведомления:

* Студентам с риском > 70% отправляется приглашение к психологу.
* Кураторы получают агрегированную статистику по группе.

Пример интерфейса (Рис. 15):

Рис. 15. Студенты вводят данные, модель выдает рекомендации в реальном времени.

Так же после прохождения тестирования пользователю сразу выходит вероятность депрессии и необходимые рекомендации в зависимости от уровня вероятности депрессии

На Рис.16 предоставлены результаты прохождения тестирования

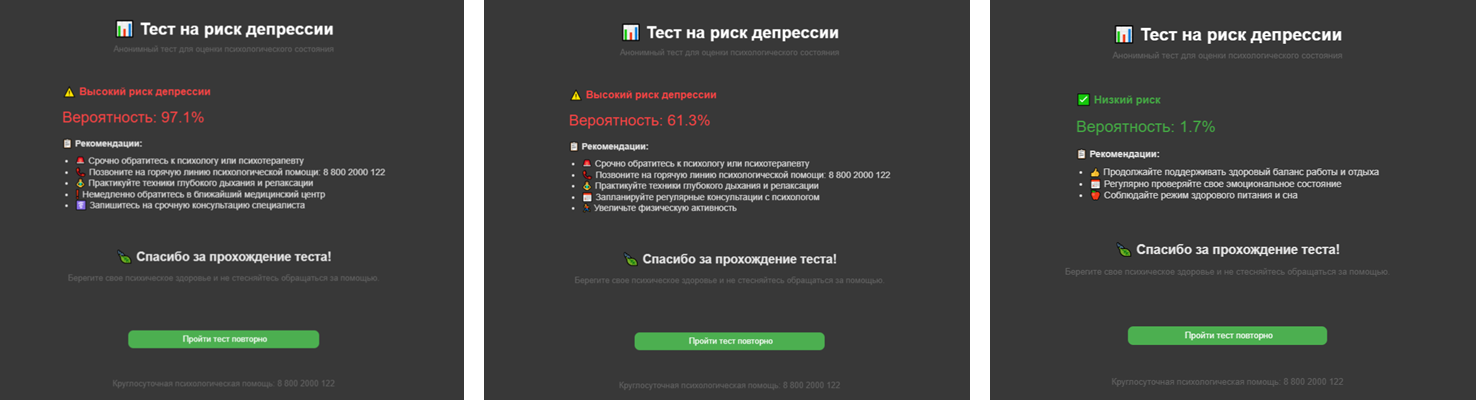


Рис.16. –Результаты

2. Оценка бизнес—эффекта

Гипотетический сценарий за 1 год:

Охват: 10,000 студентов.

Выявленные случаи депрессии: 15% (1,500 студентов).

* Снижение академических отчислений: На 20% благодаря раннему вмешательству.
* Экономия вуза:

Стоимость обучения одного студента: 200 000 руб./год.

Сохраненный бюджет: 60 000 000 руб.

3. Ограничения и этические аспекты

Ограничения модели:

* Субъективность данных: Основаны на самооценке, что может искажать результаты.
* Ложные срабатывания: Риск стигматизации студентов с высоким уровнем стресса, но без депрессии.

Этические рекомендации:

* Полная анонимность тестирования.
* Запрет на использование данных в академических решениях (отчисление, стипендии).
* Обязательное подтверждение диагноза психологом.

Выводы по этапу

1. Практическая ценность:

* Модель LightGBM готова к внедрению и показывает точность 89% на реальных данных.
* Прогнозы помогают выявлять до 85% случаев депрессии, которые оставались незамеченными.

1. Перспективы:

* Интеграция с мобильными приложениями для ежедневного мониторинга настроения.
* Использование NLP для анализа текстовых ответов в открытых вопросах анкет.

Полный код и датасет доступны в репозитории проекта: <https://github.com/AgishevDm/depression>

## Заключение

В процессе исследования была затронута очень важная тема в современном обществе, такая как ментальное здоровье студентов. Данный проект должен помочь студентам преждевременно выявлять предпосылки к психологическим недугам и помочь избежать негативных последствий в связи с внешними и внутренними факторами, которые так сильно влияют на человека.

Результаты работы уже на данный момент можно использовать для первичного тестирования и выявления предпосылок к депрессии и в дальнейшем перенаправлять студента к специалисту, который в свою очередь может выставить медицинское заключение и/или поможет разобраться с причиной возникновения такого состояния, что в дальнейшем позитивно повлияет на самочувствие студента, его продуктивность и стремление жить и развиваться.

Современные методы машинного обучения очень тесно внедряются в современный мир. Многие крупные компании используют их для понимания и анализа психологии людей для коммерческих целей. Цель же нашего проекта создать помощь для первичного обнаружения такой глобальной проблемы современного поколения как депрессии и помочь людям быстрее понять, что с ними происходит, дать рекомендации и направить при необходимости к нужному специалисту.

## Список использованных источников и литературы

Бессмертный, И. А. Интеллектуальные системы : учебник и практикум для академического бакалавриата / И. А. Бессмертный, А. Б. Нугуманова, А. В. Платонов. — Москва : Издательство Юрайт, 2019. — 243 с. — (Бакалавр. Академический курс). — ISBN 978-5-534-01042-8. — Текст : электронный // ЭБС Юрайт [сайт]. (дата обращения: 01.10.2024).

Ясницкий, Л.пН. Интеллектуальные системы : учебник / Л. Н. Ясницкий. — 2-е изд. — Москва : Лаборатория знаний, 2020. — 222 c. — ISBN 978-5-00101-897-1. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. (дата обращения: 01.10.2024).

Сузи, Р. А. Язык программирования Python : учебное пособие / Р. А. Сузи. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 350 c. — ISBN 978-5-4497-0705-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. (дата обращения: 01.10.2024).

Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для среднего профессионального образования / Д. Ю. Федоров. — 4-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 214 с. — (Профессиональное образование). — ISBN 978-5-534-15731-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. (дата обращения: 01.10.2024).

EDA. Поисковый анализ данных на Python. [Электронный ресурс]. URL: https://translated.turbopages.org/proxy\_u/en-ru.ru.7ef26421-67684fec-482e8cf6-74722d776562/https/www.geeksforgeeks.org/exploratory-data-analysis-in-python/(дата обращения: 01.12.2024).

Статья «Технология производства оптического волокна». URL: <https://science.fandom.com/ru/wiki/Технология_производства_оптического_волокна> (дата обращения: 02.12.2024).

Никоноров, Н.В. Материалы и технологии волоконной оптики: специальные оптические волокна. Учебное пособие, курс лекций / Н.В. Никоноров, А.И. Сидоров. – СПб.: СПбГУ ИТМО, 2009. – 130 с (дата обращения: 02.12.2024).

Nagel, S.R. An Overview of the Modified Chemical Vapor Deposition (MCVD) Process and Performance / S.R. Nagel, J.B. Macchesney, K.L. Walker // IEEE journal of quantum electronics. — 1982. — Vol. MTT-30. — No. 4. — P. 305-322 (дата обращения: 02.12.2024).

# Приложения

Приложение 1

Программный код

# Установка библиотек

import subprocess

import time

from IPython.display import HTML, display

def install\_packages\_with\_enhanced\_progress(packages):

num\_packages = len(packages)

start\_time = time.time()

block\_height = "20px"

# HTML для стилизованного индикатора прогресса на основе блоков

def create\_block\_progress\_html(percentage):

num\_blocks = 15

filled\_blocks = int(percentage / (100 / num\_blocks))

blocks = ''.join([f'<span style="color: green; padding-right: 2px; height: {block\_height}; display: inline-block; vertical-align: middle;">█</span>' for \_ in range(filled\_blocks)]) + \

''.join([f'<span style="color: lightgrey; padding-right: 2px; height: {block\_height}; display: inline-block; vertical-align: middle;">█</span>' for \_ in range(num\_blocks - filled\_blocks)])

return f"""

<div style="font-family: monospace; font-size: 16px; background-color: transparent;">

{blocks} {percentage:.1f}%

</div>

"""

progress\_display = display(HTML(create\_block\_progress\_html(0)), display\_id=True)

try:

for i, package in enumerate(packages):

process = subprocess.Popen(['pip', 'install', package],

stdout=subprocess.PIPE,

stderr=subprocess.PIPE) # Запускаем pip install как отдельный процесс

while process.poll() is None:

time.sleep(0.1)

overall\_percentage = min(100.0, ((i / num\_packages) \* 100) + ((time.time() - start\_time)/2) )

progress\_display.update(HTML(create\_block\_progress\_html(overall\_percentage))) # Обновляем индикатор прогресса

stdout, stderr = process.communicate()

if process.returncode != 0:

print(f"Ошибка при установке {package}: {stderr.decode()}")

raise Exception(f"Установка не удалась для {package}")

except Exception as e:

progress\_display.update(HTML("<div style='color: red;'>Установка не удалась! Проверьте логи ошибок.</div>"))

print(f"Общая установка не удалась: {e}")

return

progress\_display.update(HTML("<div style='color: green;'>Установка завершена!</div>"))

# Список пакетов для установки

packages = [

"tensorflow",

"kagglehub",

"pandas",

"opendatasets",

"dask[dataframe]",

"scikit-learn",

"matplotlib",

"seaborn",

"numpy",

"ipywidgets"

]

install\_packages\_with\_enhanced\_progress(packages)

#Импорт библиотек

# Стандартные библиотеки

import os

import re

import time

import warnings

from datetime import datetime

# Для данных

import kagglehub

import matplotlib

import matplotlib.pyplot as plt

import numpy as np

import opendatasets as od

import pandas as pd

import seaborn as sns

from tqdm import tqdm

import pexpect

# Машинное обучение и обработка данных

from sklearn.decomposition import PCA

from sklearn.ensemble import (AdaBoostClassifier, BaggingClassifier,

GradientBoostingClassifier,

HistGradientBoostingClassifier,

RandomForestClassifier)

from sklearn.feature\_selection import SelectKBest, f\_classif

from sklearn.impute import SimpleImputer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import (accuracy\_score, auc, classification\_report,

confusion\_matrix, f1\_score, precision\_score,

recall\_score, roc\_auc\_score, roc\_curve)

from sklearn.model\_selection import KFold, cross\_val\_score, learning\_curve, train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.neural\_network import MLPClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from lightgbm import LGBMClassifier

from xgboost import XGBClassifier

# Визуализация и интерфейсы

import io

import joblib

from contextlib import redirect\_stdout

import tkinter as tk

from matplotlib.animation import FuncAnimation

from tkinter import ttk, messagebox

from IPython.display import HTML, clear\_output, display

from ipywidgets import Dropdown, FloatSlider, IntSlider, RadioButtons, interact, interactive, widgets

#Загрузка датасета

# Запуск процесса загрузки

child = pexpect.spawn('python -c "import opendatasets as od; od.download(\'https://www.kaggle.com/datasets/hopesb/student-depression-dataset\', force=True )"')

# Ожидаем первый запрос на логин и автоматически вводим

child.expect('Your Kaggle username:')

child.sendline('dmytrt')

# Ожидаем запрос пароля и вводим ключ

child.expect('Your Kaggle Key:')

child.sendline('17165469')

# Ждем завершения процесса

child.expect(pexpect.EOF)

# Загрузка данных и вывод первых 5 строк набора данных

dep\_df = pd.read\_csv("student-depression-dataset/Student Depression Dataset.csv")

dep\_df.head()

# Размер датасета

dep\_df.shape

# Распределение случаев депрессии

plt.figure(figsize=(10, 6))

sns.countplot(

x='Depression',

hue='Depression',

data=dep\_df,

palette='viridis',

legend=True

)

plt.title('Распределение случаев депрессии')

plt.show()

# \*\*Обучение модели\*\*

# Предобработка и визуализация данных

print("✏️ Создание гистограмм для сравнения распределения категориальных признаков по значениям целевой переменной Depression\n")

# Настройки

sns.set\_style('darkgrid')

plt.rcParams.update({

'font.size': 12,

'figure.figsize': (12, 8),

'figure.facecolor': 'white'

})

# Предобработка данных

dep\_df = dep\_df.drop(['City'], axis=1)

def convert\_sleep\_duration(sleep\_duration):

try:

cleaned = re.sub(r'[^\d-]', '', str(sleep\_duration))

matches = re.findall(r'\d+', cleaned)

if len(matches) == 2:

return (int(matches[0]) + int(matches[1])) / 2

return int(matches[0]) if matches else np.nan

except:

return np.nan

dep\_df['Sleep Duration'] = dep\_df['Sleep Duration'].apply(convert\_sleep\_duration)

dep\_df['Sleep Duration'] = dep\_df['Sleep Duration'].fillna(dep\_df['Sleep Duration'].median())

#numeric\_cols = ['Age', 'Academic Pressure', 'CGPA', 'Financial Stress', 'Sleep Duration']

numeric\_cols =['Age', 'Academic Pressure', 'CGPA', 'Financial Stress', 'Sleep Duration']

dep\_df[numeric\_cols] = SimpleImputer(strategy='median').fit\_transform(dep\_df[numeric\_cols])

categorical\_cols = ['Gender', 'Profession', 'Dietary Habits', 'Degree',

'Have you ever had suicidal thoughts ?', 'Family History of Mental Illness']

# Создание гистограмм для сравнения распределения категориальных признаков по значениям целевой переменной Depression

for column in categorical\_cols:

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

ax = sns.countplot(x=dep\_df[column], hue='Depression', data=dep\_df)

plt.title(f'{column} vs Depression')

plt.xlabel(column)

plt.ylabel('Count')

plt.xticks(rotation=90, ha='right')

plt.tight\_layout()

plt.show()

dep\_df = pd.get\_dummies(dep\_df, columns=categorical\_cols, drop\_first=True)

dep\_df = dep\_df.drop(['id'], axis=1)

# Разделение данных

X = dep\_df.drop(['Depression'], axis=1)

y = dep\_df['Depression']

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(X, y, test\_size=0.15, random\_state=42, stratify=y)

X\_val, X\_test, y\_val, y\_test = train\_test\_split(X\_temp, y\_temp, test\_size=0.15, random\_state=42, stratify=y\_temp)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_val\_scaled = scaler.transform(X\_val)

X\_test\_scaled = scaler.transform(X\_test)

# Создание матрицы диаграмм рассеивания для анализа взаимосвязи между числовыми признаками и целевой переменной Depression

print("✏️ Создание матрицы диаграмм рассеивания для анализа взаимосвязи между числовыми признаками и целевой переменной Depression\n")

g = sns.pairplot(dep\_df[['Age', 'Academic Pressure', 'CGPA', 'Financial Stress', 'Sleep Duration', 'Depression']], hue='Depression', palette='coolwarm')

g.fig.suptitle("Матрица рассеивания между числовыми признаками и Депрессией", y=1.02)

plt.show()

# Матрица корреляции

print("✏️ Создание матрицы корреляции\n")

plt.figure(figsize=(14, 12))

corr = dep\_df.corr()

sns.heatmap(corr, annot=False, cmap='coolwarm')

plt.title('Матрица корреляции')

plt.show()

# Анализ важности признаков

print("✏️ Создание даграммы важности признаков для определения наличия депрессии\n")

selector = SelectKBest(f\_classif, k=15)

selector.fit(X\_train, y\_train)

selected\_features = X.columns[selector.get\_support()]

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

pd.Series(selector.scores\_, index=X.columns).nlargest(15).plot(kind='barh', color='skyblue')

plt.title('Влияние признаков на наличие депрессии')

plt.show()

# Сравнение моделей

# Класс для прогресса

class ProgressAnimator:

def \_\_init\_\_(self, model\_name, total=100):

self.model\_name = model\_name

self.total = total

self.start\_time = datetime.now()

self.last\_progress = 0

def get\_color(self, progress):

r = int(255 \* (1 - progress/100))

g = int(255 \* (progress/100))

return f"\033[38;2;{r};{g};0m"

def update(self, current, total\_epochs):

progress = min(100, int(100 \* (current + 1) / self.total))

if progress == self.last\_progress:

return

self.last\_progress = progress

color = self.get\_color(progress)

elapsed = datetime.now() - self.start\_time

bar = '█' \* int(progress / 2) + ' ' \* (50 - int(progress / 2))

epoch\_str = f'Epochs: {current+1}/{total\_epochs}'

time\_str = f' | Time: {elapsed.seconds}s'

print(f'\r{color}➤ {self.model\_name:25} [{bar}] {progress}% | {epoch\_str}{time\_str}', end='')

# Обучение моделей

models = {

"Logistic Regression": LogisticRegression(max\_iter=1000, random\_state=42),

"Random Forest": RandomForestClassifier(n\_estimators=200, max\_depth=10, random\_state=42),

"SVM": SVC(probability=True, random\_state=42, class\_weight='balanced'),

"KNN": KNeighborsClassifier(n\_neighbors=7),

"XGBoost": XGBClassifier(n\_estimators=150, eval\_metric='logloss', random\_state=42),

"LightGBM": LGBMClassifier(num\_leaves=31, random\_state=42, verbose=-1)

}

trained\_models = {}

results = []

confusion\_matrices = []

print("\n👾 Начало обучения моделей:")

for name, model in models.items():

animator = ProgressAnimator(name, total=500)

start\_time = time.time()

epochs = 500

try:

if hasattr(model, 'partial\_fit'):

for epoch in range(epochs):

model.partial\_fit(X\_train\_scaled, y\_train, classes=np.unique(y\_train))

animator.update(epoch, epochs)

time.sleep(0.01)

else:

for epoch in range(epochs):

if epoch == 0:

model.fit(X\_train\_scaled, y\_train)

animator.update(epoch, epochs)

time.sleep(0.01)

except Exception as e:

print(f"\n⚠ Ошибка при обучении {name}: {str(e)}")

train\_time = time.time() - start\_time

print(f'\033[0m\n✅ Модель "{name}" обучена за {train\_time:.1f} сек.')

y\_pred = model.predict(X\_val\_scaled)

y\_proba = model.predict\_proba(X\_val\_scaled)[:,1] if hasattr(model, 'predict\_proba') else [0]\*len(y\_val)

cm = confusion\_matrix(y\_val, y\_pred)

confusion\_matrices.append((name, cm))

metrics = {

'Model': name,

'Accuracy': accuracy\_score(y\_val, y\_pred),

'Precision': precision\_score(y\_val, y\_pred),

'Recall': recall\_score(y\_val, y\_pred),

'F1': f1\_score(y\_val, y\_pred),

'AUC': roc\_auc\_score(y\_val, y\_proba) if hasattr(model, 'predict\_proba') else 0.5,

'Time': f'{train\_time:.1f}s'

}

results.append(metrics)

trained\_models[name] = model

# Матрицы ошибок

plt.figure(figsize=(18, 12))

for i, (name, cm) in enumerate(confusion\_matrices, 1):

plt.subplot(2, 3, i)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

xticklabels=['Без депрессии', 'Депрессия'],

yticklabels=['Без депрессии', 'Депрессия'])

plt.title(f'{name}')

plt.ylabel('Фактические значения')

plt.xlabel('Предсказанные значения')

plt.tight\_layout()

plt.show()

# Таблица результатов обученных моделей

results\_df = pd.DataFrame(results).sort\_values('F1', ascending=False)

print("\n📈 Результаты сравнения моделей:")

display(results\_df.style.background\_gradient(cmap='Blues', subset=['Accuracy', 'Precision', 'Recall', 'F1', 'AUC']))

best\_model\_name = results\_df.iloc[0]['Model']

best\_model = trained\_models[best\_model\_name]

print(f"\n🚀 Лучшая модель: {best\_model\_name} (F1-score: {results\_df.iloc[0]['F1']:.3f})")

# Обучение и сохранение лучшей модели

# Финальное обучение

print(f"\n🤖 Финальное обучение {best\_model\_name} (1000 эпох):")

animator = ProgressAnimator(best\_model\_name, total=1000)

learning\_progress = []

X\_combined = np.vstack([X\_train\_scaled, X\_val\_scaled])

y\_combined = np.concatenate([y\_train, y\_val])

start\_time = time.time()

for epoch in range(1000):

if hasattr(best\_model, 'partial\_fit'):

best\_model.partial\_fit(X\_combined, y\_combined, classes=np.unique(y\_combined))

else:

if epoch == 0:

best\_model.fit(X\_combined, y\_combined)

animator.update(epoch, 1000)

time.sleep(0.02)

print(f'\033[0m\n✅ Финальное обучение завершено за {time.time()-start\_time:.1f} сек. (1000 эпох)')

t = time.time()-start\_time

# Сохранение модели

joblib.dump(best\_model, 'best\_model.pkl')

joblib.dump(scaler, 'scaler.pkl')

print("\n💾 Модель успешно сохранена!")

# Результаты финального обучения на тестовых данных

final\_metrics = {

'Model': f'{best\_model\_name} (FINAL)',

'Accuracy': accuracy\_score(y\_test, best\_model.predict(X\_test\_scaled)),

'Precision': precision\_score(y\_test, best\_model.predict(X\_test\_scaled)),

'Recall': recall\_score(y\_test, best\_model.predict(X\_test\_scaled)),

'F1': f1\_score(y\_test, best\_model.predict(X\_test\_scaled)),

'AUC': roc\_auc\_score(y\_test, best\_model.predict\_proba(X\_test\_scaled)[:,1]),

'Time': f'{t:.1f}s'

}

# Отдельный DataFrame для финального результата

final\_results\_df = pd.DataFrame([final\_metrics])

# Вывод таблицы

print("\n🏆 Результаты финальной модели после обучения:")

display(final\_results\_df.style.background\_gradient(

cmap='Greens',

subset=['Accuracy', 'Precision', 'Recall', 'F1', 'AUC']

))

# ROC-кривая

y\_proba = best\_model.predict\_proba(X\_test\_scaled)[:,1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_proba)

roc\_auc = roc\_auc\_score(y\_test, y\_proba)

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

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC = {roc\_auc:.2f}')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlabel('Частота ложноположительных результатов')

plt.ylabel('Частота правдивых результатов')

plt.title('ROC-кривая лучшей модели')

plt.legend(loc="lower right")

plt.show()

# \*\*Интерфейс для использования модели\*\*

# Загрузка модели. Интерфейс. Обработчики событий

# Загрузка обученной модели

try:

model = joblib.load('best\_model.pkl')

scaler = joblib.load('scaler.pkl')

print("✅ Модель успешно загружена")

except Exception as e:

print(f"⚠ Ошибка загрузки модели: {e}")

raise

# Стили

styles = """

<style>

/\* Анимация кнопок \*/

.jupyter-button {

transition: all 0.3s ease !important;

position: relative;

overflow: hidden;

border-radius: 8px !important;

}

.jupyter-button:before {

content: '';

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(255,255,255,0.2);

opacity: 0;

transition: opacity 0.3s ease;

}

.jupyter-button:hover:before {

opacity: 1;

}

/\* Стили для ползунков \*/

.widget-slider .ui-slider-handle {

transition: transform 0.2s ease, box-shadow 0.2s ease;

}

.widget-slider .ui-slider-handle:hover {

transform: scale(1.2);

box-shadow: 0 0 10px rgba(66,165,245,0.5);

}

</style>

"""

# Шапка приложения

header = widgets.HTML(

"<div style='text-align:center; padding:20px; margin-bottom:20px;'>"

"<h1 style='color:#FFFFFF;'>📊 Тест на риск депрессии</h1>"

"<p style='color:#666; max-width:600px; margin:0 auto;'>"

"Анонимный тест для оценки психологического состояния</p>"

"</div>"

)

# Сообщение благодарности

thank\_you\_message = widgets.HTML(

"<div style='text-align:center; padding:40px; color:#2c3e50;'>"

"<h2>🍃 Спасибо за прохождение теста!</h2>"

"<p style='color:#666'>Берегите свое психическое здоровье и не стесняйтесь обращаться за помощью.</p>"

"</div>",

layout={'display': 'none'} # Изначально скрыт

)

# Футер

footer = widgets.HTML(

"<div style='text-align:center; margin-top:20px; color:#666;'>"

"<p>Круглосуточная психологическая помощь: 8 800 2000 122</p>"

"</div>"

)

# Виджеты (слайдер и радиобатон)

def create\_styled\_slider(description, \*\*kwargs):

return FloatSlider(

\*\*kwargs,

description=description,

style={'description\_width': 'initial'},

layout={'width': '80%', 'margin': '10px auto'},

continuous\_update=False

)

def create\_styled\_radio(description, options):

return RadioButtons(

options=options,

description=description,

style={'description\_width': 'initial'},

layout={'width': '80%', 'margin': '10px auto'}

)

# Создание виджетов

widgets\_dict = {

'Age': create\_styled\_slider(

'👶 Возраст:',

value=17, min=15, max=60, step=1

),

'Academic Pressure': create\_styled\_slider(

'📚 Учебная нагрузка (0-10 где 0 - нет нагрузки, 10 - экстремальная):',

value=5, min=0, max=10, step=0.05

),

'CGPA': create\_styled\_slider(

'🎓 Средний балл (CGPA по шкале 0-5):',

value=5, min=0, max=5, step=0.01

),

'Financial Stress': create\_styled\_slider(

'💸 Финансовый стресс (0-10 где 0 - нет проблем, 10 - критическое положение):',

value=5, min=0, max=10, step=0.1

),

'Work Pressure': create\_styled\_slider(

'💼 Рабочая нагрузка (0-10 где 0 - нет нагрузки, 10 - непосильная):',

value=5, min=0, max=10, step=0.1

),

'Study Satisfaction': create\_styled\_slider(

'😊 Удовлетворенность учебой (0-10 где 0 - полная неудовлетворенность):',

value=5, min=0, max=10, step=0.1

),

'Sleep Duration': create\_styled\_slider(

'😴 Продолжительность сна (часов в сутки):',

value=7, min=0, max=24, step=0.5

),

'Gender\_Male': create\_styled\_radio(

'👤 Пол:',

[('Мужской', 1), ('Женский', 0)]

),

'Profession\_Working': create\_styled\_radio(

'👔 Основной род деятельности:',

[('Работающий', 1), ('Студент', 0)]

),

'Dietary Habits\_Vegetarian': create\_styled\_radio(

'🥦 Диетические привычки:',

[('Здорове питание', 1), ('Нездоровое питание', 0)]

),

'Degree': Dropdown(

options=[('Бакалавр', (0,0)), ('Магистр', (1,0)), ('Аспирант', (0,1))],

description='🎓 Образовательная степень:',

style={'description\_width': 'initial'},

layout={'width': '80%', 'margin': '10px auto'}

),

'Have you ever had suicidal thoughts ?\_Yes': create\_styled\_radio(

'💔 Были ли у вас суицидальные мысли?:',

[('Да', 1), ('Нет', 0)]

),

'Family History of Mental Illness\_Yes': create\_styled\_radio(

'🧬 Есть ли в семье история психических заболеваний?:',

[('Да', 1), ('Нет', 0)]

)

}

# Функция отображения результатов

def show\_results(prediction, proba):

risk\_class = "risk-high" if prediction == 1 else "risk-low"

result\_html = f"""

<div class="{risk\_class} recommendation-box">

<h3 style="color:{"#ff4444" if prediction == 1 else "#44aa44"}; margin-top:0;">

{'⚠️ Высокий риск депрессии' if prediction == 1 else '✅ Низкий риск'}

</h3>

<div style="font-size: 24px; margin: 15px 0; color: {"#ff4444" if prediction == 1 else "#44aa44"}">

Вероятность: {proba:.1f}%

</div>

<h4 style="margin-bottom: 10px;">📋 Рекомендации:</h4>

<ul style="padding-left: 20px; margin: 0;">

"""

recommendations = []

if prediction == 1:

recommendations.append("🚨 Срочно обратитесь к психологу или психотерапевту")

recommendations.append("📞 Позвоните на горячую линию психологической помощи: 8 800 2000 122")

recommendations.append("🧘 Практикуйте техники глубокого дыхания и релаксации")

if proba > 75:

recommendations.append("❗️ Немедленно обратитесь в ближайший медицинский центр")

recommendations.append("⚕️ Запишитесь на срочную консультацию специалиста")

else:

recommendations.append("📅 Запланируйте регулярные консультации с психологом")

recommendations.append("🏃 Увеличьте физическую активность")

else:

recommendations.append("👍 Продолжайте поддерживать здоровый баланс работы и отдыха")

recommendations.append("📅 Регулярно проверяйте свое эмоциональное состояние")

recommendations.append("🍎 Соблюдайте режим здорового питания и сна")

result\_html += "\n".join([f"<li>{r}</li>" for r in recommendations])

result\_html += "</ul></div>"

display(HTML(styles + result\_html))

# Контейнер вопросов

questions\_box = widgets.VBox(

[widgets\_dict[k] for k in widgets\_dict],

layout=widgets.Layout(

width='80%',

margin='20px auto',

padding='20px',

border='1px solid #eee',

border\_radius='8px',

display='none'

)

)

# Кнопки управления:

start\_button = widgets.Button(

description='Начать тестирование',

button\_style='success',

icon='play',

layout={'width': '300px', 'margin': '20px auto', 'justify\_content': 'center'}

)

finish\_button = widgets.Button(

description='Завершить тестирование',

button\_style='warning',

icon='flag-checkered',

layout={'width': '300px', 'margin': '20px auto', 'justify\_content': 'center','display': 'none'}

)

restart\_button = widgets.Button(

description='Пройти тест повторно',

button\_style='success',

icon='refresh',

layout={'width': '300px', 'margin': '20px auto',

'justify\_content': 'center', 'display': 'none'}

)

# Блок результатов

result\_output = widgets.Output(layout={'display': 'none', 'justify\_content': 'center'})

# Собираем интерфейс

app = widgets.VBox([

header,

start\_button,

questions\_box,

finish\_button,

result\_output,

thank\_you\_message,

restart\_button,

footer

], layout=widgets.Layout(width='100%'))

# Функции управления интерфейсом

def reset\_widgets():

for name, widget in widgets\_dict.items():

if isinstance(widget, FloatSlider):

widget.value = (widget.max + widget.min) / 2

elif isinstance(widget, RadioButtons):

widget.index = None

elif isinstance(widget, Dropdown):

widget.value = widget.options[0][1]

result\_output.clear\_output()

def start\_test(b):

reset\_widgets()

start\_button.layout.display = 'none'

questions\_box.layout.display = 'flex'

finish\_button.layout.display = 'flex'

restart\_button.layout.display = 'none'

result\_output.layout.display = 'none'

for widget in widgets\_dict.values():

widget.disabled = False

def finish\_test(b):

data = {name: widget.value for name, widget in widgets\_dict.items()}

with result\_output:

result\_output.clear\_output()

try:

predict\_depression(\*\*data)

except Exception as e:

display(HTML(f"<div style='color:red; padding:20px;'>Ошибка: {str(e)}</div>"))

questions\_box.layout.display = 'none'

finish\_button.layout.display = 'none'

result\_output.layout.display = 'flex'

restart\_button.layout.display = 'flex'

thank\_you\_message.layout.display = 'flex'

for widget in widgets\_dict.values():

widget.disabled = True

def restart\_test(b):

thank\_you\_message.layout.display = 'none'

start\_test(b)

# Обработчики нажатия на кнопки

start\_button.on\_click(start\_test)

finish\_button.on\_click(finish\_test)

restart\_button.on\_click(restart\_test)

# Функция прогнозирования

def predict\_depression(\*\*kwargs):

try:

data = {

'Age': kwargs['Age'],

'Academic Pressure': kwargs['Academic Pressure'],

'CGPA': kwargs['CGPA'] \* 2,

'Financial Stress': kwargs['Financial Stress'],

'Work Pressure': kwargs['Work Pressure'],

'Study Satisfaction': kwargs['Study Satisfaction'],

'Sleep Duration': kwargs['Sleep Duration'],

'Gender\_Male': kwargs['Gender\_Male'],

'Profession\_Working': kwargs['Profession\_Working'],

'Dietary Habits\_Vegetarian': kwargs['Dietary Habits\_Vegetarian'],

'Degree\_Master': kwargs['Degree'][0],

'Degree\_PhD': kwargs['Degree'][1],

'Have you ever had suicidal thoughts ?\_Yes': kwargs['Have you ever had suicidal thoughts ?\_Yes'],

'Family History of Mental Illness\_Yes': kwargs['Family History of Mental Illness\_Yes']

}

input\_df = pd.DataFrame([data], columns=X\_train.columns)

scaled\_input = scaler.transform(input\_df)

prediction = model.predict(scaled\_input)[0]

proba = model.predict\_proba(scaled\_input)[0][1] \* 100

show\_results(prediction, proba)

except Exception as e:

display(HTML(f"<div style='color:red; padding:20px;'>Ошибка: {str(e)}</div>"))

# Запуск интерфейса

# Запуск приложения

display(HTML(styles))

display(app)