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 «Анализ данных»

**ВЛИЯНИЕ ПРИВЫЧЕК СТУДЕНТОВ НА ИХ УСПЕВАЕМОСТЬ**

Разработчики проекта:

Катаев Руслан Игоревич

Важесова Валерия Михайловна

Пермь, 2023

**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:**Влияние привычек студентов на их успеваемость.

**Сведения об авторах:**Катаев Руслан Игоревич, Важесова Валерия Михайловна

**Цель:**выполнить анализ данных о стиле жизни студентов и его связи с успеваемостью, которая оценивается по среднему баллу.

**Задачи:**

1. Выполнить анализ проблемы, обосновать ее актуальность.
2. Осуществить загрузку данных и подготовку их к анализу количественными методами, включая устранение пропущенных значений.
3. Выполнить предварительный анализ данных, в том числе выявление и обработку выбросов, проверку распределения данных на нормальность, корреляционный анализ.
4. Осуществить моделирование зависимости целевого признака от факторных методом классификации., в том числе подобрать наилучшую модель, оценить ее качество и выполнить прогнозирование.
5. Выполнить интерпретацию полученных результатов и сделать выводы о достижении цели.

**Краткое описание проекта:**

Требуется проанализировать записи о ежедневных привычках студентов, связанных с учёбой, внеклассными занятиями, сном, общением, физической активностью, а так же их уровень стресса и определить, возможно ли описать зависимость успеваемости от имеющихся факторных переменных методом классификации. Дать интерпретацию полученным результатам. Сделать выводы.

**Конкретные ожидаемые результаты:**

Построенная классификационная модель и рекомендации по ее использованию.

**СОДЕРЖАНИЕ ПРОЕКТА**

**Анализ проблемы исследования**

Анализ влияния привычек студентов, их образа жизни на успеваемость играет важную роль как в образовательном процессе, так и личностном развитии. Она является основой для академических успехов и карьеры, для развития полезных навыков, таких как критическое мышление, коммуникация, способность к анализу и принятию решений.

Во-первых, этот анализ может быть полезным для образовательных учреждений в плане адаптации и корректировки учебных планов, а также для внедрения программ, направленных на улучшение и укрепление психического здоровья молодых людей.

Во-вторых, исследование может быть интересным для самих студентов для понимания того, как их образ жизни и привычки сказываются на успеваемости и как добиться прогресса.

Негативное влияние:

* Недостаток сна вызывает негативные эмоции, апатию, снижение концентрации и мотивации.
* Перегрузка учёбой сказывается негативно, вызывая стресс и усталость.
* Чрезмерная социальная активность и злоупотребление социальными сетями вызывает эмоциональную усталость, перенапряжение и снижение концентрации на поставленных целях.

Положительное влияние:

* Здоровый образ жизни (режим сна, сбалансированное питание, достаточная физическая активность и т.д.) является ключевым фактором для повышения успеваемости.
* Баланс учёбы и отдыха и, как следствие, психическая стабильность, является профилактикой снижения мотивации и концентрации.
* Анализ причин появление стресса и управление им с помощью различных методов. Волнение и переживание – чувства, которые сопровождают человека на протяжении всей жизни. В небольших количествах они могут быть полезны для самодисциплины и мотивации.

Помимо рассматриваемых факторов существует много других, например,

семейное и дружеское окружение, воспитание, финансовое положение, работа, доступ к ресурсам, жилищные условия и т.д.

Изучение влияния привычек студентов на их успеваемость — это не только

цель нашего проекта, но и важный шаг для улучшения качества жизни студентов и повышение эффективности обучения. Анализ этих факторов поможет создать более здоровую и продуктивную учебную среду, способствующую личностному и профессиональному развитию студентов. Полученные данные могут быть полезны для сферы образования, психологии и здравоохранения, что делает их важными для проектов по управлению образом жизни, прогнозированию успеваемости и повышения благополучия.

*Цель:* выполнить анализ данных об образе жизни студентов и построить модель зависимости успеваемости, которая оценивается по среднему баллу, от различных факторов с помощью метода классификации., позволяющую делать прогнозы с высокой точностью.

*Задачи:*

1. Выполнить анализ проблемы, обосновать ее актуальность.
2. Осуществить загрузку данных и подготовку их к анализу количественными методами, включая устранение пропущенных значений.
3. Выполнить предварительный анализ данных, в том числе выявление и обработку выбросов, проверку распределения данных на нормальность, корреляционный анализ.
4. Осуществить моделирование зависимости целевого признака от факторных методом классификации., в том числе подобрать наилучшую модель, оценить ее качество и выполнить прогнозирование.
5. Выполнить интерпретацию полученных результатов и сделать выводы о достижении цели.

**Исходные данные**

В настоящей работе анализируется набор данных под названием «Повседневный образ жизни и успеваемость студентов» содержит информацию о 2000 студентах, собранную с помощью опроса в Google Форме. Он включает сведения о часах учёбы, внеклассных занятиях, сне, общении, физической активности, уровне стресса и среднем балле. Данные охватывают учебный год с августа 2023 по май 2024 года и отражают образ жизни студентов в основном из Индии. Список колонок анализируемого набора данных:

1. **Study\_Hours\_Per\_Day** – среднее количество учебных часов в день.
2. **Extracurricular\_Hours\_Per\_Day** – среднее число часов, затрачиваемое на учебу в день за пределами учебного заведения.
3. **Sleep\_Hours\_Per\_Day** – среднее число часов, затрачиваемое на сон в день.
4. **Social\_Hours\_Per\_Day** – среднее число часов, затрачиваемое на общение в день.
5. **Physical\_Activity\_Hours\_Per\_Day**  – среднее число часов, затрачиваемое на физическую активность в день.
6. **GPA** – успеваемость, которая оценивается по среднему баллу.
7. **Stress\_Level Numeric** – уровень стресса.

Необходимо проанализировать данные по образу жизни студентов и определить, возможно ли описать зависимость их среднего балла от имеющихся факторных переменных методом классификации.

Выдвинем гипотезу исследования: успеваемость студента, выраженная в виде среднего балла, зависит от количества сна, уровня стресса, количества учебных часов и может быть предсказана с помощью этих показателей.

**Реализация проекта**

**Этап 1. Подготовка данных к анализу**

Загрузим данные в датафрейм и подключим необходимые библиотеки:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix, accuracy\_score, classification\_report

from sklearn.model\_selection import train\_test\_split

Загрузим данные:

import kagglehub

df = pd.read\_csv(kagglehub.dataset\_download("steve1215rogg/student-lifestyle-dataset", path='student\_lifestyle\_dataset.csv'), delimiter=',')

df

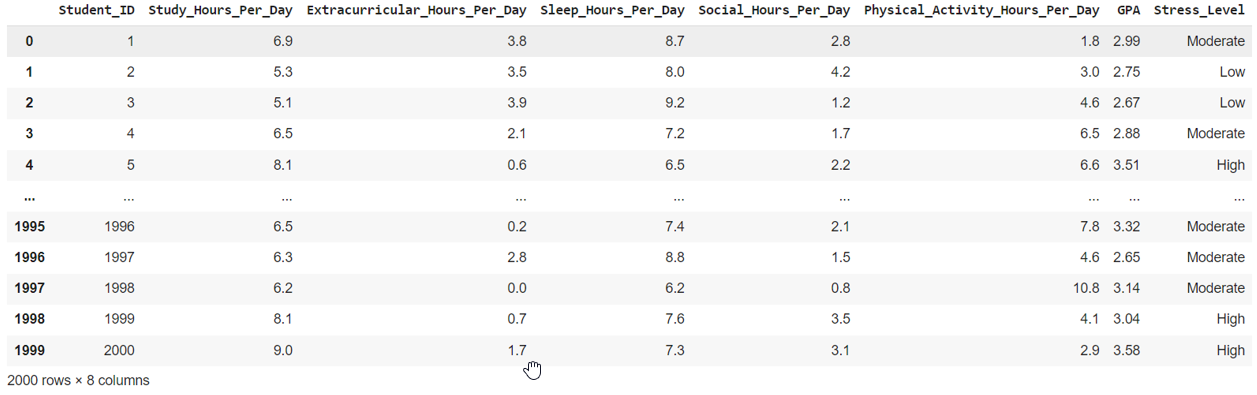


Рисунок 1. Исходный датафрейм

Убедимся, что все количественные столбцы имеют числовой тип. Если это не так, выполним преобразование типа столбца к числовому. Для применения количественных методов анализа все столбцы должны быть числовыми

Удалим столбец, который не влияет на целевую переменную.

df.drop('Student\_ID', axis=1, inplace=True)

df.head()

Посмотрим, какой тип имеют данные

df.info()

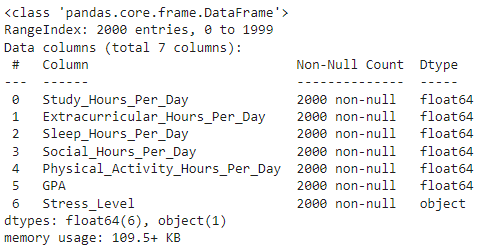


Рисунок 2. Типы данных колонок

Видно, что строковым типом обладает только уровень стресса, а все остальные колонки по смыслу числовые.

Проверим данные на пропуски.

df.isnull().sum()

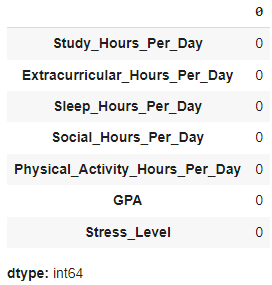


Рисунок 3. Пропуски в колонках

Проверим, какие категории есть в параметре уровня стресса.

df['Stress\_Level'].unique()



Рисунок 4. Категории стресса

Можно видеть, что уровень стресса, который испытывают студенты, в данном датасете был поделен на три категории.

Преобразуем категории стресса из строкового типа данных в целочисленный. Будем считать, что низкому уровню стресса будет соответствовать нуль. Среднему уровню стресса соответствует единица. Высокому уровню стресса соответствует двойка.

category\_to\_numeric = {

    'Low': 0,

    'Moderate': 1,

    'High': 2

}

df['Stress\_Level'].replace(category\_to\_numeric, inplace=True)

df.head()

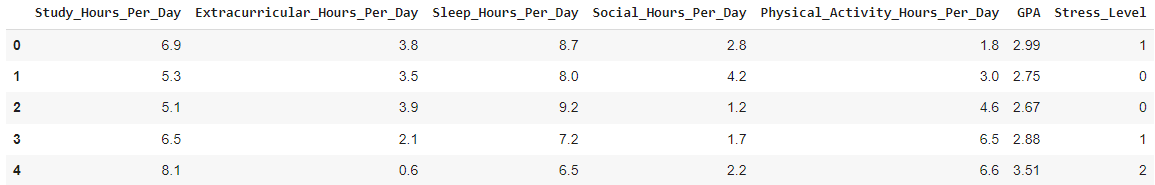


Рисунок 5. Преобразованный датасет

Проверим тип данных по столбцам.

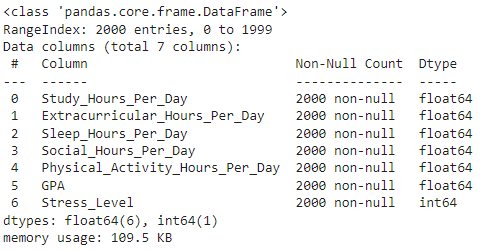


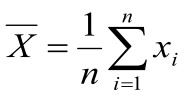
Рисунок 5. Преобразованный тип данных

Итак, результат первого этапа – это готовый к анализу набор данных в виде датафрейма.

**Этап 2. Предварительный анализ данных**

Вычислим описательные статистики по колонкам (среднее, моду, медиану, стандартное отклонение, квартили).

***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*.

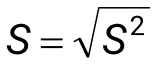
В случае несгруппированных данных для нахождения медианы необходимо ранжировать выборку, т. е. расположить данные в порядке их возрастания или убывания. Медианой будет являться значение признака, находящееся в середине ранжированного ряда. Медиана находится по формуле

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()

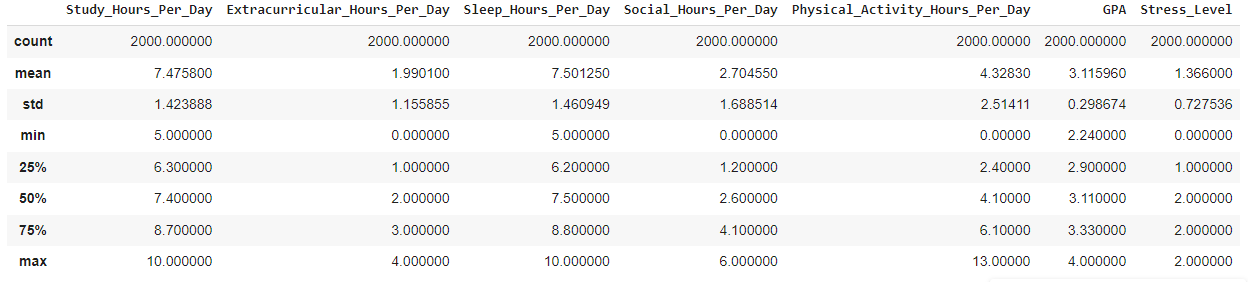


Рисунок 6. Описательные статистики по колонкам

***Аномальными наблюдениями*** (*выбросами*, англ. *Outliers, Extreme values*) называют такие значения уровня временного ряда, которые значительно отличаются от остальных. При выявлении подобных «выбросов» возникают серьезные вопросы: являются ли отклоняющиеся данные действительно ошибками (например, регистрации) или это реальные значения и как получить адекватные оценки для параметров изучаемой совокупности.

Проверим данные на наличие выбросов, для этого можно использовать диаграмму «ящик с усами» (boxplot). Если выбросов мало, то следует их сгладить.

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. изображен график «ящик с усами».

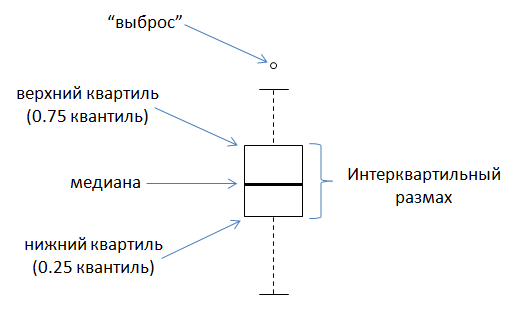


Рисунок 7. «Ящик с усами»

Посмотрим на диаграммы boxplot всех колонок датафрейма:

for i in df.columns:

plt.figure()

sns.boxplot(data=df[i])

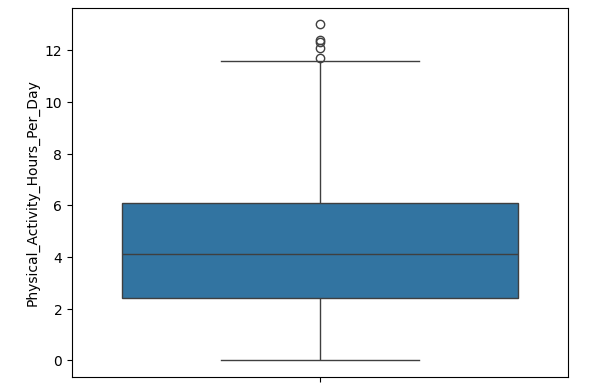


Рисунок 8. Ящик с усами для переменной “Физическая  
активность (среднее количество часов в день)”

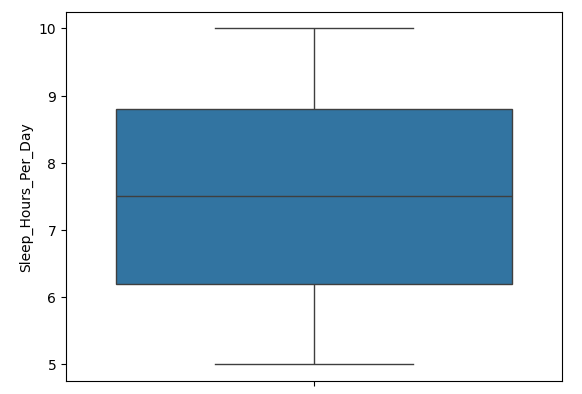
****

Рисунок 9. Ящик с усами для переменной “Сон

(среднее количество часов в день)”

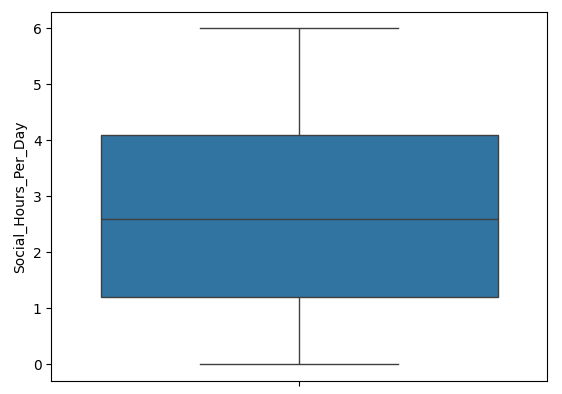
****

Рисунок 10. Ящик с усами для переменной “Общение

(среднее количество часов в день)”

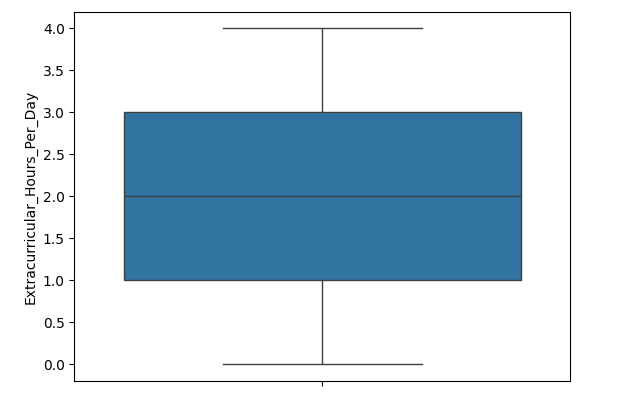
****

Рисунок 11. Ящик с усами для переменной “Дополнительная  
 активность (среднее количество часов в день)”

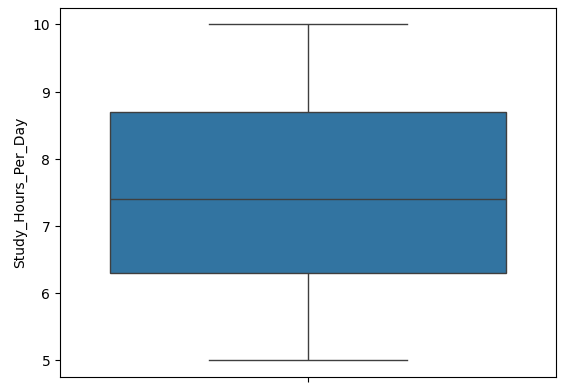
****

Рисунок 12. Ящик с усами для переменной  
 “Учебные часы (среднее количество часов в день)”

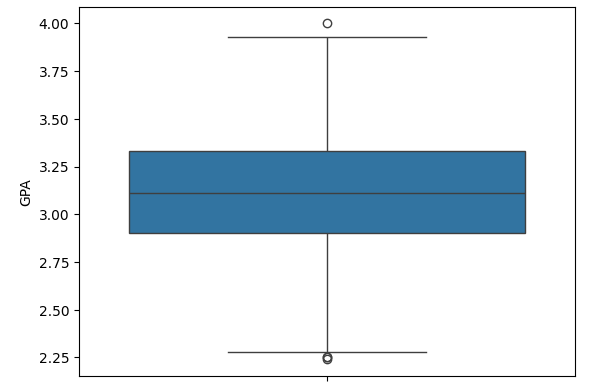
****

Рисунок 13. Ящик с усами для переменной   
“Успеваемость по среднему баллу”

Данных, которые считаются выбросами не много, поэтому мы заменим их на 1 и 99 квартили.

**for i in ['Physical\_Activity\_Hours\_Per\_Day', 'GPA']:**

**Q1 = np.percentile(df[i], 25)**

**Q3 = np.percentile(df[i], 75)**

**IQR = Q3 - Q1**

**limit\_max = Q3 + 1.5 \* IQR**

**limit\_min = Q1 - 1.5 \* IQR**

**df.loc[df[i] > limit\_max, i] = limit\_max**

**df.loc[df[i] < limit\_min, i] = limit\_min**

Проверим, что получилось.

**for i in ['Physical\_Activity\_Hours\_Per\_Day', 'GPA']:**

**plt.figure()**

**sns.boxplot(data=df[i])**

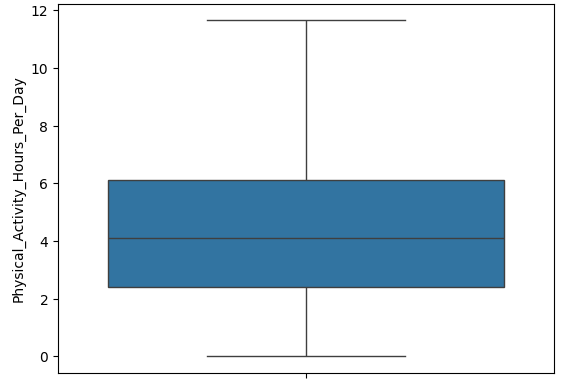
****

Рисунок 14. Ящик с усами для переменной “Физическая  
 активность (часы в день)” без выбросов

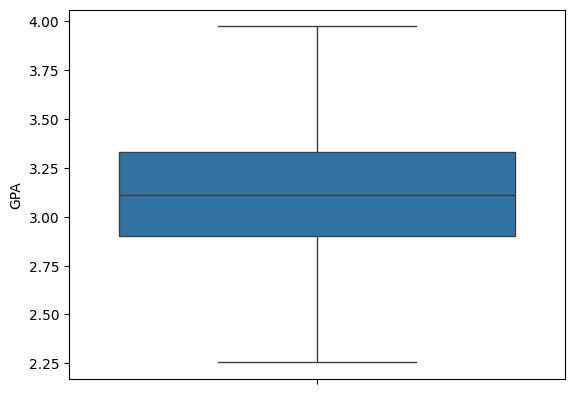
****

Рисунок 15. Ящик с усами для переменной  
 “Успеваемость” без выбросов

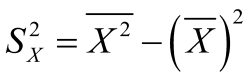
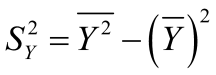
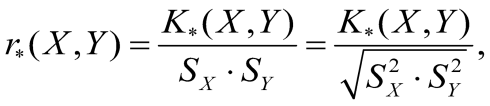
**Этап 3. Корреляционный анализ данных**

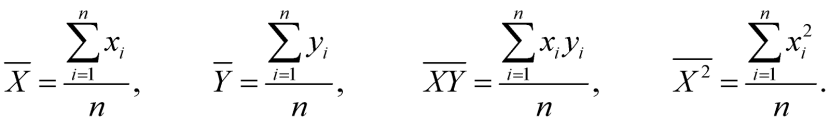
***Корреляционный анализ*** – это совокупность методов оценивания степени тесноты статистической связи между анализируемыми переменными.

Выполним корреляционный анализ данных с помощью матрицы корреляции. В случае нормальности всех данных следует использовать коэффициент корреляции Пирсона, в противном случае – ранговые коэффициенты корреляции Спирмена.

**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 значение <= 0,7 , то связь между переменными  и  считается сильной. Если значение < 0,3 , то связь слабая.
3. При  линейная корреляционная связь отсутствует.

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

correlation\_matrix = df.corr(method='pearson')

# Визуализация матрицы корреляции с помощью тепловой карты

plt.figure(figsize=(7, 5))

sns.heatmap(correlation\_matrix, annot=True, fmt=".2f", cmap='coolwarm')

plt.show()

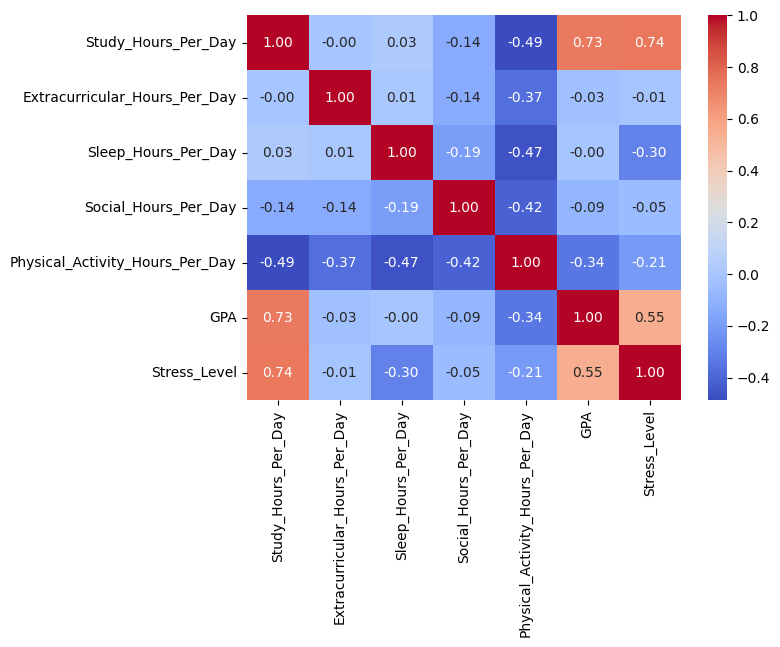


Рисунок 16. Тепловая карта матрицы корреляции.

Таким образом, на третьем этапе мы видим, что есть есть связи между целевой переменной “Успеваемость по среднему баллу” и несколькими переменными (Количеством учебных часов и уровнем стресса, а также обратная связь с переменной физическая активность).

На данном этапе выдвигается гипотеза о том, что успеваемость зависит от уровня стресса, учебной деятельностью и физической активностью. Возможно также то, что сон и другие переменные косвенно влияют на данную зависимость. Применим классификацию для проверки гипотезы с помощью логистической регрессии.

**Этап 4. Моделирование и прогнозирование**

*Классификация* — это задача машинного обучения, в которой модель обучается на основе имеющихся данных для предсказания категории или класса, к которому относится объект.

*Логистическая регрессия* — это статистический метод, используемый для двоичной классификации. Он прост и интерпретируем, но неэффективен для сложных нелинейных зависимостей без дополнительной обработки данных,

*Решающее дерево* — это модель, которая делит данные на основе правил, сформированных на каждом этапе. Она также проста в интерпретации и может учитывать нелинейные зависимости, но чувствительна к шуму в данных.

*Случайный лес* — это ансамбль решающих деревьев, который объединяет их прогнозы для повышения точности, где окончательный прогноз основывается на голосовании деревьев (для классификации). Отличается высокой точностью и устойчивостью к шуму, способна работать с большими наборами данных, но требует больше ресурсов для обучения.

* + 1. Разделим данные на обучающую (80%) и тестовую (20%) выборки, где признаки (X) — все столбцы, кроме GPA, а целевая переменная (y) — столбец GPA.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df.drop('GPA', axis=1), df['GPA'],test\_size=0.20, random\_state=15)

Данные разбиваются на 4 части: признаки и целевая переменная для

обучения и тестирования. Это позволяет обучить модель на одной части данных и проверить её качество на другой, чтобы оценить её производительность.

* + 1. Проверим распределение целевой переменной в выборках:

count = df['GPA'].value\_counts()

Результат, показывающий, что классы разделены равномерно:

0: для обучения - 0.7981; 1: для обучения - 0.8034

0: для тестирования - 0.2019; 1: для тестирования - 0.1966

* + 1. Рассмотрим три модели: логистическая регрессия, решающее дерево и случайный лес:

models = {

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

"Decision Tree": DecisionTreeClassifier(random\_state=42),

"Random Forest": RandomForestClassifier(random\_state=42)

}

* + 1. Функция «train\_and\_evaluate\_models»обучает несколько моделей машинного обучения, оценивает их производительность и визуализирует результаты.

def train\_and\_evaluate\_models(models, X\_train, y\_train, X\_test, y\_test):

* + 1. Модель обучается на тренировочных данных с помощью метода:

«.fit(X\_train, y\_train).» и делает предсказания на тестовых данных с помощью «.predict(X\_test).».

* + 1. Сравниваются прогнозы модели preds с реальными значениями y\_test, чтобы вычислить долю правильных ответов (точность) с помощью «accuracy\_score».
    2. С помощью «confusion\_matrix(y\_test, preds)» вычисляется матрица ошибок и отображается в виде тепловой карты с помощью библиотеки «seaborn»:

conf\_matrix = confusion\_matrix(y\_test, preds)

        plt.figure(figsize=(8, 6))

        sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='PuBuGn',

                    xticklabels=model.classes\_, yticklabels=model.classes\_)

        plt.title(f'Confusion Matrix for {name}')

        plt.xlabel('Predicted')

        plt.ylabel('Actual')

        plt.show()

* + 1. Выводится подробный отчет по каждой модели с метриками Precision, Recall, F1-score и поддержкой (support) через «classification\_report».
    2. Формируется DataFrame, где записываются значения Accuracy для каждой модели.

Результат:

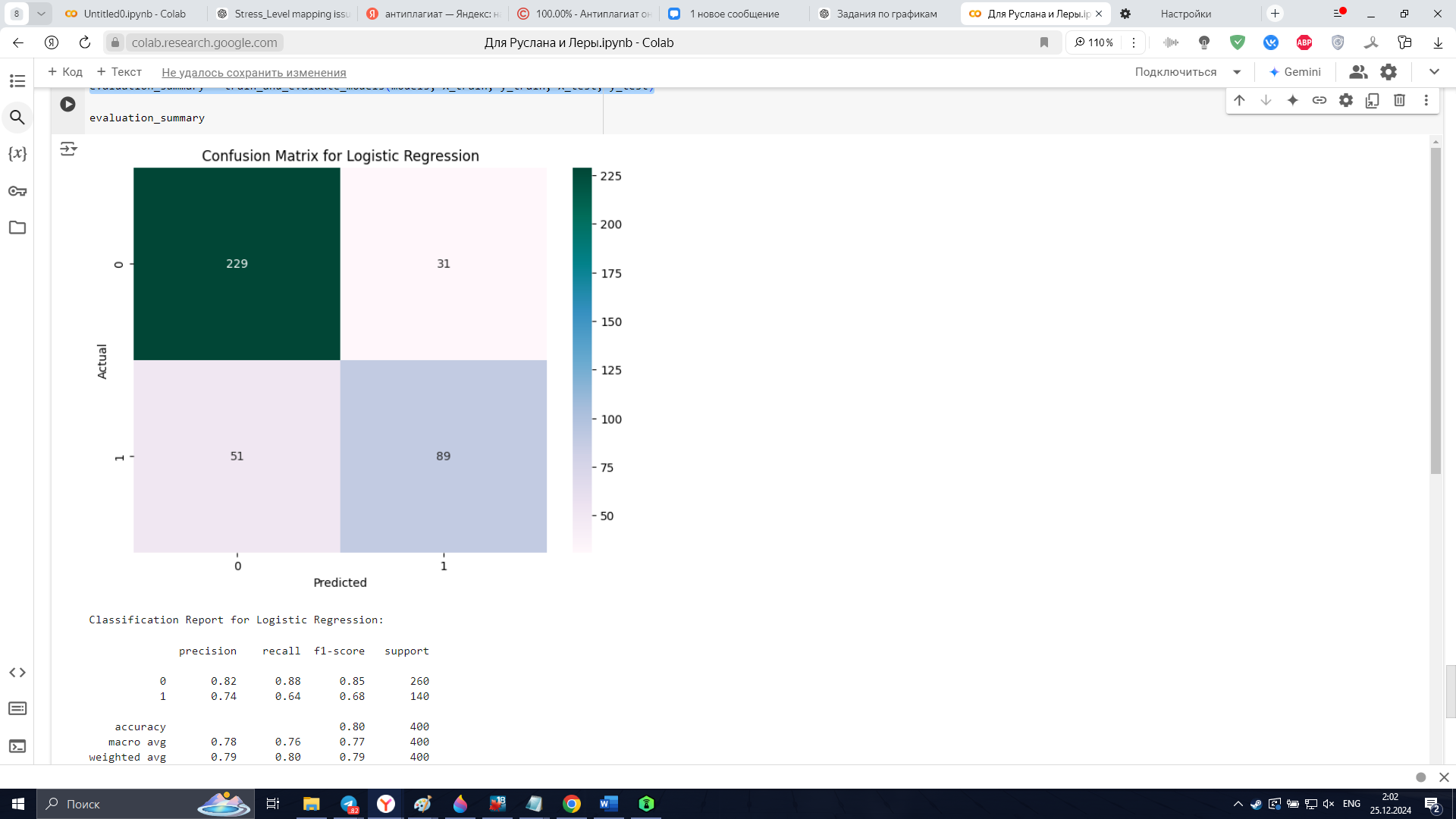


Рис. 17. Матрица ошибок для модели «Логистическая регресия»

Изображение выглядит как снимок экрана, текст, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рис. 18. Метрики для модели «Логистическая регресия»

Правильные предсказания.

Класс 0: 229; Класс 1: 89.

Ошибочные предсказания:

Класс 0 предсказан как 1: 31;

Класс 1 предсказан как 0: 51.

Accuracy (точность): Общий процент правильных прогнозов. В нашем случае accuracy составляет 0.80 (80%), что говорит о том, что модель правильно предсказала 80% всех случаев, что является приемлемым результатом.

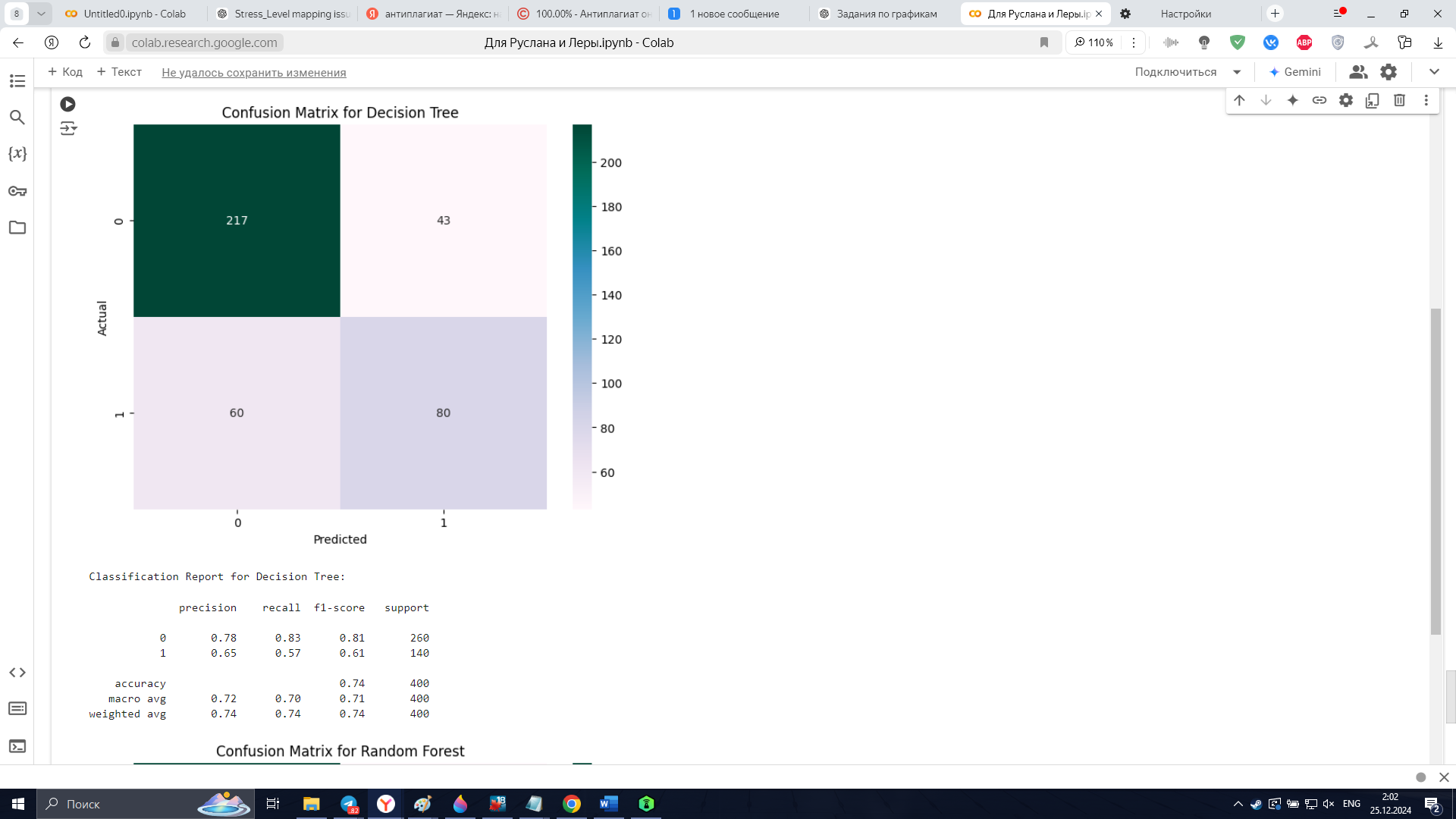


Рис. 19. Матрица ошибок для модели «Решающее дерево»

Изображение выглядит как снимок экрана, текст, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рис. 20. Метрики для модели «Решающее дерево»

Правильные предсказания.

Класс 0: 217; Класс 1: 80.

Ошибочные предсказания:

Класс 0 предсказан как 1: 43;

Класс 1 предсказан как 0: 60.

Accuracy (точность): Общий процент правильных прогнозов. В нашем случае accuracy составляет 0.74 (74%), что говорит о том, что модель правильно предсказала 74% всех случаев, что является приемлемым результатом.

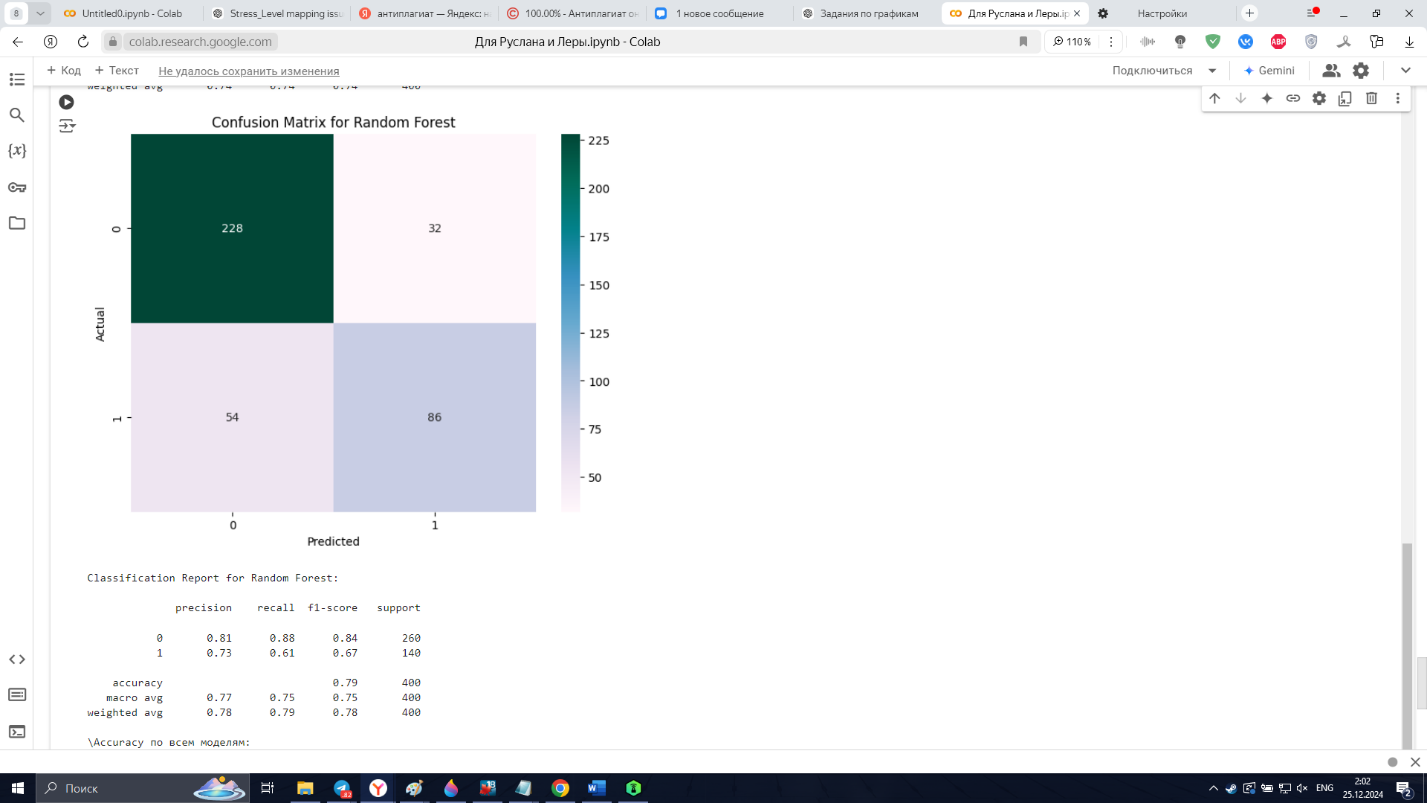


Рис. 21. Матрица ошибок для модели «Случайный лес»

Изображение выглядит как снимок экрана, текст, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рис. 22. Метрики для модели «Случайный лес»

Правильные предсказания.

Класс 0: 228; Класс 1: 86.

Ошибочные предсказания:

Класс 0 предсказан как 1: 32;

Класс 1 предсказан как 0: 54.

Accuracy (точность): Общий процент правильных прогнозов. В нашем случае accuracy составляет 0.79 (79%), что говорит о том, что модель правильно предсказала 79% всех случаев, что является приемлемым результатом.



Рис. 23. Значение Accuracy для каждой модели

Основываясь только на точности, можно сделать вывод, что логистическая регрессия показала лучший результат среди трёх рассмотренных моделей.

**Заключение**

На основе анализа классификационных моделей можно сделать вывод, что цель достигнута: разработаны и протестированы классификационные модели, позволяющие с высокой точностью прогнозировать целевой признак на основе предоставленных данных.

Очевидно, что логистическая регрессия (80%) показала самую высокую точность среди рассмотренных моделей, что подтверждает ее пригодность для решения данной задачи. Благодаря своей высокой интерпретируемости и вычислительной эффективности, эта модель является предпочтительным выбором. Случайный лес (точность 79%) продемонстрировал сопоставимую точность, но более высокую устойчивость к шуму и нелинейным зависимостям. Дерево решений (74%) уступает другим моделям в точности, но может быть полезно благодаря своей объяснимости и визуализации процесса классификации.

Для достижения поставленной цели были решены следующие задачи:

Проведен анализ проблемы и обоснована ее актуальность.

Данные загружены и предварительно обработаны, включая устранение пропущенных значений и масштабирование.

Был проведен предварительный анализ данных: проверка распределений, идентификация выбросов, анализ корреляций между признаками.

Строятся и оцениваются три модели классификации: логистическая регрессия, дерево решений и случайный лес.

Модели сравниваются на основе показателя точности, а также их применимости к задаче.

Проводится интерпретация полученных результатов.

Построенные модели классификации могут быть использованы для прогнозирования класса объектов, что полезно для задач анализа данных и автоматизации процессов. Логистическая регрессия является лучшим выбором в текущем эксперименте, однако для более сложных задач можно рекомендовать случайный лес.

**Список использованных источников и литературы**

1. Амоа К.А. Разработка программных пакетов на языке Python: учебное пособие / К.А. Амоа, Н.А. Рындин, Ю.С. Скворцов. – Воронеж: Воронежский государственный технический университет, ЭБС АСВ, 2020. – 61 c. // Электронно-библиотечная система IPR BOOKS: [сайт].
2. Криволапов С.Я. Математика на Python : учебник / С.Я. Криволапов, М.Б. Хрипунова. — Москва: КНОРУС, 2022. — 456 с.
3. Сузи Р.А. Язык программирования Python: учебное пособие / Р.А. Сузи. – 3-е изд. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. – 350 c. // Электронно-библиотечная система IPR BOOKS: [сайт].
4. Федоров Д.Ю. Программирование на языке высокого уровня Python: учебное пособие / Д.Ю. Федоров. – 2-е изд.– М.: Юрайт, 2020. – 161 с.

**Приложения**

Приложение 1

Программный код

# Подключаем необходимые библиотеки

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix, accuracy\_score, classification\_report

from sklearn.model\_selection import train\_test\_split

import kagglehub

df = pd.read\_csv(kagglehub.dataset\_download("steve1215rogg/student-lifestyle-dataset", path='student\_lifestyle\_dataset.csv'), delimiter=',')

df

# Student\_ID нам не зачем, удалим его

df.drop('Student\_ID', axis=1, inplace=True)

df.head()

# Посмотрим какой тип имеют данные

df.info()

# Проверим данные на пропуски

df.isnull().sum()

# Проверим какие категории есть в признаке Stress\_Level

df['Stress\_Level'].unique()

# Преобразуем категории из строк в числа

category\_to\_numeric = {

    'Low': 0,

    'Moderate': 1,

    'High': 2

}

df['Stress\_Level'].replace(category\_to\_numeric, inplace=True)

df.head()

# Снова проверяем тип данные по столбцам

df.info()

# Выводим статистические данные

df.describe()

# Проверим на выбросы с помощью графика "бочка с усами"

for i in df.columns:

  plt.figure()

  sns.boxplot(data=df[i])

# Данных, которые считаются выбросами не много, поэтому мы заменим их на 1 и 99 квартили

for i in ['Physical\_Activity\_Hours\_Per\_Day', 'GPA']:

  # Расчет квартилей и границы

  Q1 = np.percentile(df[i], 25)

  Q3 = np.percentile(df[i], 75)

  IQR = Q3 - Q1

  limit\_max = Q3 + 1.5 \* IQR

  limit\_min = Q1 - 1.5 \* IQR

  # Замена выбросов

  df.loc[df[i] > limit\_max, i] = limit\_max

  df.loc[df[i] < limit\_min, i] = limit\_min

# Проверим, что получилось

for i in ['Physical\_Activity\_Hours\_Per\_Day', 'GPA']:

  plt.figure()

  sns.boxplot(data=df[i])

# Построим матрицу корреляций

correlation\_matrix = df.corr(method='spearman')

# Визуализация матрицы корреляции с помощью тепловой карты

plt.figure(figsize=(7, 5))

sns.heatmap(correlation\_matrix, annot=True, fmt=".2f", cmap='coolwarm')

plt.show()

# Проверим целевой признак на упорядоченность

from scipy.stats import spearmanr

# Тест Спирмена

spearman\_corr, spearman\_p = spearmanr(df['GPA'].index, df['GPA'])

print(f"Spearman correlation: {spearman\_corr}, p-value: {spearman\_p}")

# Уровень значимости

alpha = 0.05

if spearman\_p < alpha:

    print("Целевой признак упорядочен. Перемешиваем данные.")

    df = df.sample(frac=1).reset\_index(drop=True)

else:

    print("Целевой признак не упорядочен. Перемешивание не требуется.")

# Переберём всевозможные комбинации признаков

from itertools import combinations

def combo():

  y = df['GPA']

  dict\_pred = {} # Словарь для хранения списка признаков, на которых модели удовлетворяют условию

  columns = df.drop('GPA', axis=1).columns.tolist()

  # Перебираем все возможные комбинации столбцов

  for i in range(1, len(columns) + 1):  # от 1 до числа столбцов

      for combo in combinations(columns, i):

          X = df[list(combo)]

          X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=15)

          model = LinearRegression()

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

          r2 = model.score(X\_test, y\_test)

          if r2 > 0.5:

            dict\_pred[combo] = [r2, model] # Сохраняем коэффициент детерминации модели на обучающей и тестовой выборках

            print('Модель подходит')

          del X, X\_train, X\_test, y\_train, y\_test, model

  return dict\_pred

dict\_pred = combo()

dict\_pred

# Определяем набор признаков с наилучшим показателем на тестовой выборке

best\_columns = max(dict\_pred, key=lambda k: dict\_pred[k][0])

best\_model = dict\_pred[best\_columns][1]

print(f"Набор признаков: {best\_columns}")

print(f"Коэффициент детерминации на тестовой выборке: {dict\_pred[best\_columns][0]}")

# Коэффециенты модели

s = ""

model\_zip = list(zip(best\_columns, best\_model.coef\_))

for i in range(len(model\_zip)):

  s += f"{model\_zip[i][1]:0.4f}\*{model\_zip[i][0]} + "

s += f"{best\_model.intercept\_:0.4f}"

print(s)

# График распределения GPA

X = df['Study\_Hours\_Per\_Day'].values

# target data is array of shape (n,)

y = df['Social\_Hours\_Per\_Day'].values

plt.scatter(X, y,color='g')

plt.plot(X, best\_model.predict(df[['Study\_Hours\_Per\_Day', 'Social\_Hours\_Per\_Day']]),color='k')

plt.xlabel('Study\_Hours\_Per\_Day', size=12)

plt.ylabel('Social\_Hours\_Per\_Day', size=12)

plt.show()

df.head()

# Функция для категоризации

def categorize\_grades(grade):

    if 3.0 <= grade <= 5:

        return 0 # Пусть учится

    else:

        return 1 # Отчислить

df['GPA'] = df['GPA'].apply(categorize\_grades)

df.head()

# Проверит распределение целевой переменной в выборках

count = df['GPA'].value\_counts()

print(f"0: для обучения     - {y\_train.value\_counts()[0] / count[0] : 0.4f};  1: для обучения     - {y\_train.value\_counts()[1] / count[1] : 0.4f}")

print(f"0: для тестирования - {y\_test.value\_counts()[0] / count[0] : 0.4f};  1: для тестирования - {y\_test.value\_counts()[1] / count[1] : 0.4f}")

# Классы распределены равномерно, продолжаем

models = {

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

    "Decision Tree": DecisionTreeClassifier(random\_state=42),

    "Random Forest": RandomForestClassifier(random\_state=42)

}

print('models:', list(models.keys()))

def train\_and\_evaluate\_models(models, X\_train, y\_train, X\_test, y\_test):

    results = {}

    for name, model in models.items():

        # Обучение

        model.fit(X\_train, y\_train)

        # Предсказание на тестовых данных

        preds = model.predict(X\_test)

        # Вычисляем Accuracy

        accuracy = accuracy\_score(y\_test, preds)

        results[name] = {

            'Accuracy': accuracy,

            'Predictions': preds

        }

        # Отображаем confusion matrix для тестовых данных

        conf\_matrix = confusion\_matrix(y\_test, preds)

        plt.figure(figsize=(8, 6))

        sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='PuBuGn',

                    xticklabels=model.classes\_, yticklabels=model.classes\_)

        plt.title(f'Confusion Matrix for {name}')

        plt.xlabel('Predicted')

        plt.ylabel('Actual')

        plt.show()

        # Отображаем classification report для тестовых данных

        print(f"\nClassification Report for {name}:\n")

        print(classification\_report(y\_test, preds))

    results\_df = pd.DataFrame({

        name: {"Accuracy": results[name]["Accuracy"]}

        for name in results

    }).T

    # Accuracy по всем моделям

    print("\Accuracy по всем моделям:\n")

    return results\_df

evaluation\_summary = train\_and\_evaluate\_models(models, X\_train, y\_train, X\_test, y\_test)

evaluation\_summary