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 и машинное обучение»

**ОПРЕДЕЛЕНИЕ СТЕПЕНИ РИСКА ЗДОРОВЬЯ СТУДЕНТА В ЗАВИСИМОСТИ ОТ ЕГО ПРИВЫЧЕК**

Разработчики проекта:

Мокрушина Мария Евгеньевна;

Силиванов Вадим Юрьевич;

Пермь, 2024

**Оглавление**

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

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

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

[Исходные данные 6](#_heading=h.2et92p0)

[Реализация проекта 7](#_heading=h.tyjcwt)

[Этап 1. Предварительная подготовка и обработка данных 7](#_heading=h.3dy6vkm)

[Этап 2. Построение модели 18](#_heading=h.1t3h5sf)

[Этап 3. Выбор оптимальной модели 19](#_heading=h.4d34og8)

[Этап 4. Обучение модели 21](#_heading=h.2s8eyo1)

[Этап 5. Оценка модели 22](#_heading=h.17dp8vu)

[Заключение 27](#_heading=h.3rdcrjn)

[Список использованных источников и литературы 28](#_heading=h.26in1rg)

[Приложения 30](#_heading=h.lnxbz9)

**ПАСПОРТ ПРОЕКТА**

**Название проекта:** Определение степени риска здоровья студента в зависимости от его привычек.

**Сведения об авторах:** Мокрушина Мария, Силиванов Вадим Юрьевич.

**Цель и задачи проекта:**

**Цель:** выполнить анализ данных о здоровье, посещаемости занятий и общем благополучии студентов с целью оценить взаимосвязь между здоровьем студентов и их успеваемостью.

**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 линейной регрессии, 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 данные по посещаемости занятий и здоровью студентов (уровень стресса, режим сна, тревожность и настроение). На основе этих данных попробовать выявить факторы, которые могут влиять на успеваемость студента и риски касаемые его здоровья.

**Конкретные ожидаемые результаты:**

Построенная регрессионная модель и рекомендации по ее использованию.

**СОДЕРЖАНИЕ ПРОЕКТА**

**Анализ проблемы исследования**

Анализ степени риска здоровья студента в зависимости от его привычек является очень важной и актуальной темой исследования в сфере педагогики.

Всё чаще студенты сталкиваются с различными проблемами, связанными со здоровьем и качеством образования. Одной из таких проблем является определение степени риска здоровья студента в зависимости от режима сна, количества занятий и их времени проведения.

Студенческий период является процессом профессионального становления молодых людей и завершающим этапом формирования психофизического здоровья.

Рассматривая статистику видно, что часто в процессе обучения создаются факторы и условия, неблагоприятные для здоровья, что по итогу приводит увеличивается количество студентов, имеющих хронические заболевания к моменту окончания вуза.

К основным причинам, которые оказывают негативное воздействие на состояние организма молодых людей, относятся чрезмерное напряжение, утомление, несбалансированное распределение учебной нагрузки, отсутствие занятия спортом и наличия каких-либо увлечений, совмещение учёбы с работой. Данные факторы, в свою очередь, способны вызывать повышение уровня тревожности, ухудшение состояния психофизиологических функций, снижение адаптационного потенциала.

Очень большое влияние на студентов оказывает режим сна. Он играет важную роль в поддержании здоровья и работоспособности студентов. Неправильный режим сна может привести к снижению успеваемости, ухудшению физического и психического здоровья. Проблемы со сном могут часто быть вызваны большим количеством учебных часов или несбалансированным расписанием уроков.

Так чрезмерная учебная нагрузка может привести к переутомлению, стрессу и снижению иммунитета. Важно найти оптимальный баланс между учебной деятельностью и отдыхом, чтобы сохранить здоровье и повысить эффективность обучения.

Так согласно исследованию Московского городского педагогического университета, 46 % опрошенных студентов испытывают общую тревожность, 32 % — трудности с самопониманием, 27 % — беспокойство при проверке знаний и аффективные расстройства.

Изучение состояния студентов позволяет осуществлять поиск эффективных способов, предупреждающих развитие негативно влияющих факторов на организм в процессе учебной деятельности.

*Цель:* выполнить анализ данных о здоровье учеников и занятиях. Построить модель зависимости рисков для здоровья студентов от их самочувствия.

1. Выполнить анализ проблемы, обосновать ее актуальность.

2. Осуществить загрузку данных и подготовку их к анализу количественными методами, включая устранение пропущенных значений если такие имеются.

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 линейной регрессии, 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**

Представленный набор данных содержит в себе следующие данные: список студентов, сведения об их занятиях (даты и время проведения и посещениях), и информацию, связанную с состоянием учащихся.

Список колонок анализируемого набора данных:

1. Student ID – уникальный идентификатор для каждого учащегося.
2. Date – дата, когда были зарегистрированы посещаемость и биометрические данные.
3. Class Time – период времени занятий в течение дня.
4. Attendance status – был ли учащийся "Присутствующим", "Отсутствующим" или "Опоздавшим".
5. Stress level (GSR) – измерение кожно-гальванической реакции (GSR), указывающее на уровень стресса.
6. Sleep Hours – количество часов, которые учащийся спал в ночь перед занятием.
7. Anxiety Level – показатель уровня тревожности учащегося.
8. Mood Score– субъективная оценка, указывающая на настроение учащегося по определенной шкале.
9. Risk Level– расчетный уровень риска (низкий, средний, высокий), основанный на посещаемости и биометрических данных, указывающий на потенциальный риск для здоровья учащегося.

Необходимо проанализировать данные из датасета и определить возможно ли выявить зависимости между представленными факторами о занятиях и здоровья учеников.

Выдвинем гипотезу что чем больше стресса у студентов тем выше будет уровень риска для здоровья.

**Реализация проекта**

**Этап 1.** **Предварительная подготовка и обработка данных**

В начале проекта необходимо подготовить данные, что включает в себя очистку, отбор экземпляров, нормализацию, преобразование данных, выделение признаков и отбор признаков.

Результатом предварительной обработки данных является конечный тренировочный набор.

В первую очередь подключим необходимые библиотеки, которые будут необходимы в ходе реализации проекта:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

from sklearn.preprocessing import LabelEncoder

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, f1\_score, precision\_score, recall\_score, ConfusionMatrixDisplay, roc\_auc\_score

from sklearn import model\_selection, preprocessing, feature\_selection, ensemble, linear\_model, metrics, decomposition

Загрузим наши данные из файла:

df = pd.read\_csv('student\_monnitoring\_data.csv')

df.head()

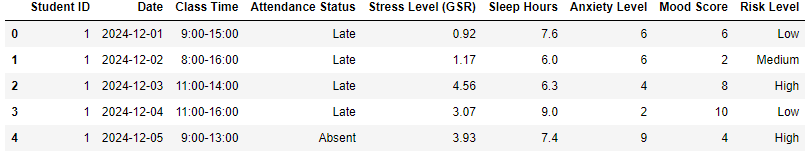


Рисунок 1. Исходный датафрейм

Удаляем признак Date и Student ID. Так как наша задача - определить зависимость уровня стресса от привычек студента, а эти признаки не дадут качественной оценки в дальнейшем анализе:

df= df.drop(['Date'], axis = 1)

df= df.drop(['Student ID'], axis = 1)

df.head()

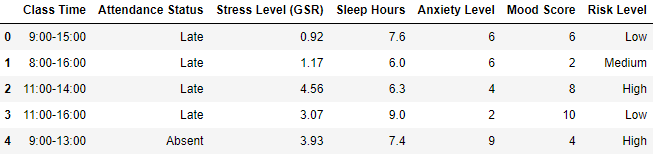


Рисунок 2. Датафрейм без признаков Date и Student ID

Используем метод describe который приводит общую статистику по конкретным полям таблицы. После чего посмотрим на общую статистику числовых данных и проанализируем нечисловые данные:

df.describe()

df.describe(include=['O'])

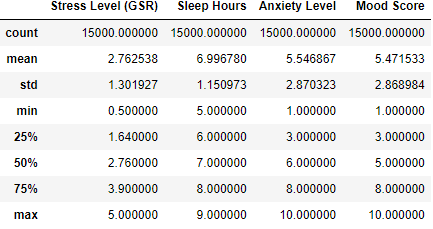


Рисунок 3. Общая статистика числовых данных

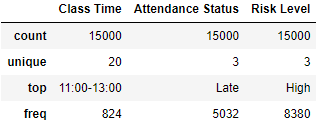


Рисунок 4. Анализ нечисловых данных

Убедимся, что все количественные столбцы имеют числовой тип. Если не так, выполним преобразования к числовому типу. Далее проверим данные на наличие недостающих значений:

df.info()

df.isnull().sum()

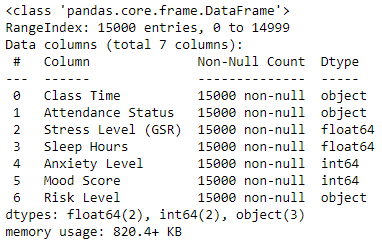


Рисунок 5. Типы данных колонок

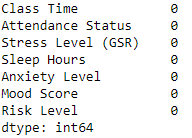


Рисунок 6. Количество пустых значений в таблице данных

Как видим, пропущенных значений нет. После чего рассмотрим столбцы, а точнее их названия (признаки).

df.columns



Целью проекта является исследования рисков для здоровья студентов, поэтому в качестве целевого признака выберем Risk Level

class\_counts = df['Risk Level'].value\_counts()

print(class\_counts)

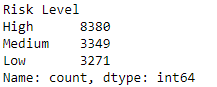


Рисунок 6. Значения целевого признака

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

Столбчатая гистограмма отображает сравнение нескольких дискретных признаков. Одна её ось показывает сравниваемые признаки, другая — измеримое значение. Иногда столбчатые гистоаграммы отображают несколько величин для каждой сравниваемой категории.

Столбчатые гистограммы обеспечивают визуальное представление категориальных данных.

Построим гистограмму значений для целевого признака.

sns.countplot(x = df['Risk Level'])

plt.title('Распределение данных целевого признака', fontsize=14)

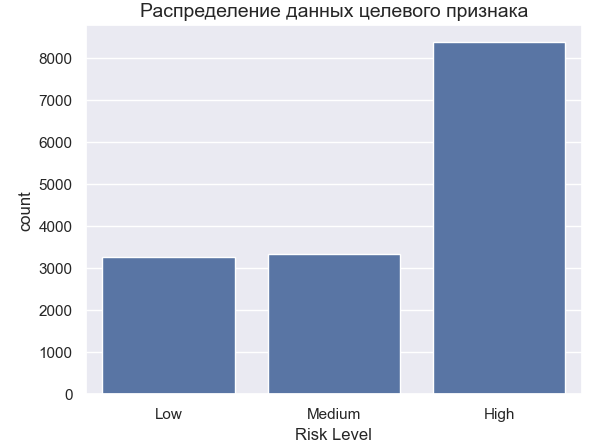


Рисунок 7. Гистограмма целевого признака

Как видим значений с высоким уровнем риска горазда больше, чем у остальных.

Чтобы лучше оценить влияние других признаков на целевой, построим ещё несколько гистограмм. После чего проанализируем полученные данные:

sns.countplot(data=df, x="Risk Level", hue='Attendance Status');

sns.countplot(data=df, x="Anxiety Level", hue='Risk Level');

sns.countplot(data=df, x="Mood Score", hue='Risk Level');

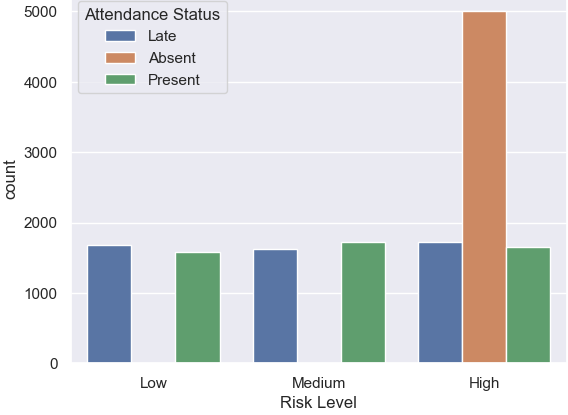


Рисунок 8. Гистограмма для пары признаков (Risk Level, 'Attendance Status)

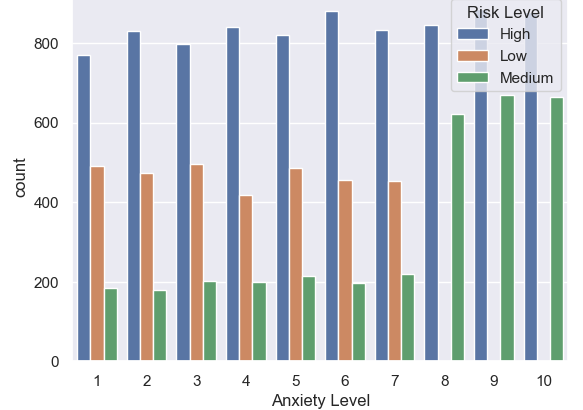


Рисунок 9. Гистограмма для пары признаков (Risk Level, Anxiety Level)

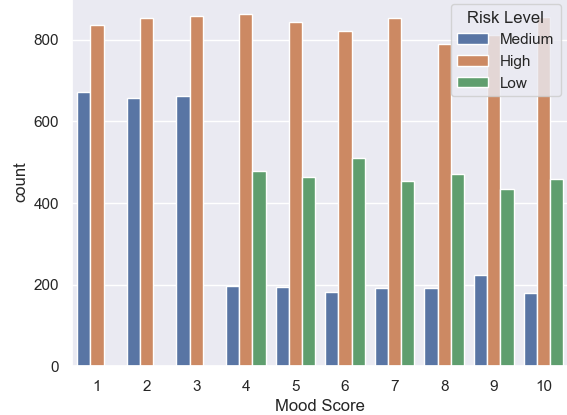


Рисунок 10. Гистограмма для пары признаков (Risk Level, Mood Score)

Визуально видно, что есть зависимость у всех трёх пар с целевым признаком. При высоких уровнях тревожности и низком настроении студентов повышаются риски для здоровья.

Для более глубокого анализа построим скрипичные диаграммы.

Гистограмма с накоплением (Скрипичная диаграмма) — это метод визуализации распределения числовых данных по нескольким переменным. Она очень похожа на ящичковою диаграмму, но с немного другим графиком, что даёт больше информации о плотности распределения по оси Y. Плотность отображается зеркально и переворачивается, а полученная форма заполняется, создавая изображение, напоминающее скрипку. Преимущество такой гистограммы с накоплением в том, что она может показать нюансы распределения, которые не видны на ящичковой диаграмме.

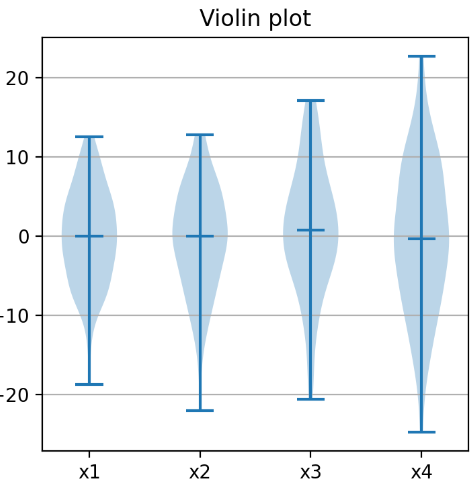


Рисунок 11. Пример скрипичной гистограммы

Построим скрипичные графики для двух других признаков:

sns.violinplot(y='Risk Level', x='Stress Level (GSR)', data=df, inner='quartile')

plt.show()

sns.violinplot(y='Risk Level', x='Sleep Hours', data=df, inner='quartile')

plt.show()

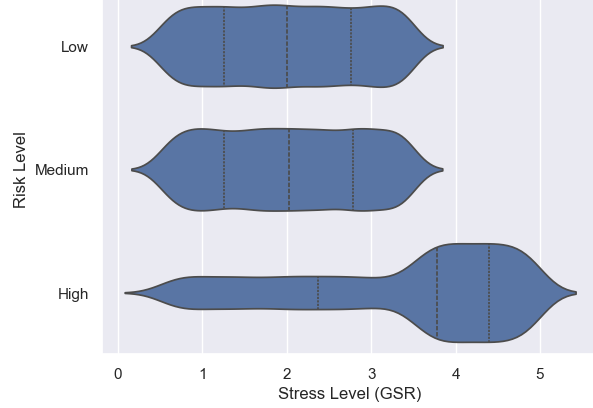


Рисунок 12. Скрипичный график для признаков Risk Level и Stress Level

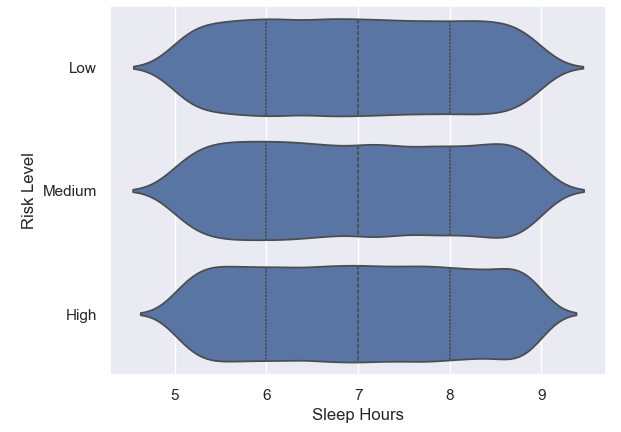


Рисунок 13. Скрипичный график для признаков Risk Level и Sleep Hours

Перейдём к категориальным признакам и их кодированию:

df['Attendance Status'] = df['Attendance Status'].map({'Present': 1, 'Absent': 0, 'Late': 2})

df['Risk Level'] = df['Risk Level'].map({'Low': 0, 'Medium': 1, 'High': 2})

Для кодирования Class Time признака используем LabelEncoder так как много уникальных значений.

LabelEncoder — это метод в библиотеке Scikit-Learn. Он используется для преобразования категориальных меток данных в числовые значения. Он помогает алгоритмам машинного обучения обрабатывать данные, а именно избавляться от большого количества уникальных значений. Суть LabelEncoder заключается в том, что он присваивает уникальный номер каждой уникальной категории в наборе данных.

df['Class Time'] = LabelEncoder().fit\_transform(df['Class Time'])

df.head()

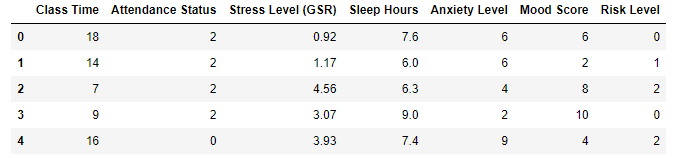


Рисунок 14. Датасет с кодированными данными

Проведём нормализацию, т.е приведём данные к единому масштабу для улучшения работы моделей:

from sklearn.preprocessing import MinMaxScaler # StandardScaler

scaler\_mm = MinMaxScaler()

df[['Stress Level (GSR)']] = scaler\_mm.fit\_transform(df[['Stress Level (GSR)']])

df[['Sleep Hours']] = scaler\_mm.fit\_transform(df[['Sleep Hours']])

df[['Anxiety Level']] = scaler\_mm.fit\_transform(df[['Anxiety Level']])

df[['Mood Score']] = scaler\_mm.fit\_transform(df[['Mood Score']])

df.head()

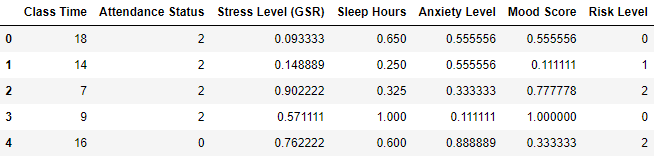


Рисунок 15. Нормализация

Построим тепловую карту для наших данных, чтобы проверить есть ли корреляция между признаками (dataset.corr() используется для поиска попарной корреляции всех столбцов в кадре данных.):

plt.figure(figsize=(10,8))

sns.heatmap(df.corr(numeric\_only = True), annot=True)

plt.show()

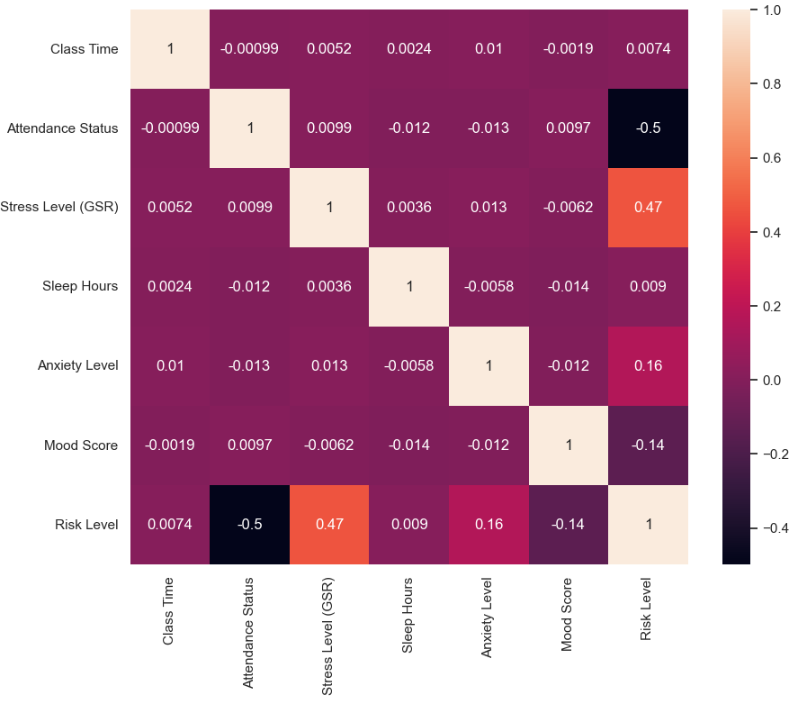


Рисунок 16. Тепловая карта

Как видим уровень риска коррелирует с уровнем стресса (stress level) и обратно коррелирует с посещаемостью (attendance status), то есть по мере увеличения одной переменной другая переменная имеет тенденцию к уменьшению, и наоборот. У остальных признаков корреляции между собой не наблюдается. Значит наша гипотеза о наличии прямой зависимости между степенью риска и уровнем стресса оказалась верной.

Теперь построим попарные графики для всех признаков:

sns.pairplot(df, kind="scatter", hue="Risk Level");

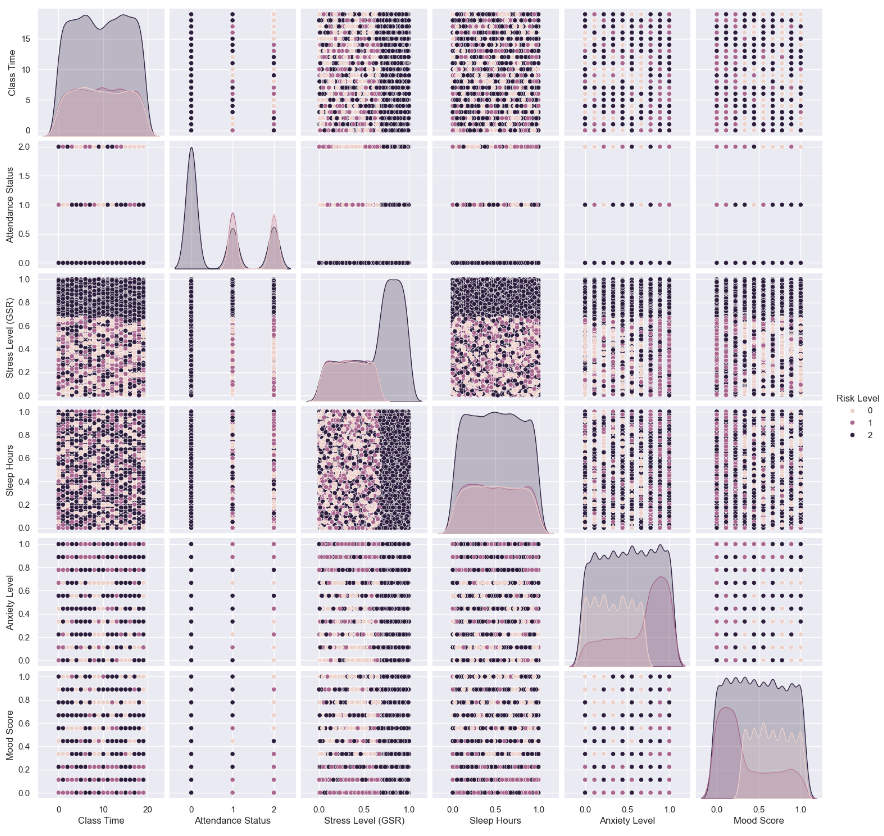


Рисунок 17. Попарные графики

**Этап 2.** **Построение модели**

Разделим наши данные. Пусть Y - целевая переменная, которую предсказываем. Мы ищем зависимость этой переменной от соответствующих признаков. В нашем случае это столбец Risk Level. В X складываем весь набор данных без целевой переменной.

X = df.drop(['Risk Level'], axis=1)

y = df['Risk Level']

print(f'X shape: {X.shape} | y shape: {y.shape} ')

X shape: (15000, 6) | y shape: (15000,)

Разделяем наш набор данных на обучающий (train) и тестовый (test) с помощью train\_test\_split(), мы берем 80% данных для обучения нашей модели и 20% оставляем в качестве проверочного набора данных. Такое деление общепринятое, оно считается оптимальным для большинства задач машинного обучения и позволяет оценить производительность модели без сильного влияния на точность. Чтобы использовать эту функцию, необходимо импортировать модуль sklearn.model\_selection:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=10)

**Этап 3. Выбор оптимальной модели**

Чтобы выбрать оптимальную модель попробуем следующие алгоритмы:

* Logistic Regression (LR) — это метод бинарной классификации, использующий сигмоидальную функцию для преобразования линейной комбинации.
* Linear Discriminant Analysis (LDA) — это метод классификации, основанный на разложении Фишера (минимизирует ошибку).
* K-Nearest Neighbors (KNN) — это метод, который определяет класс нового объекта на основе его сходства с обучающей выборкой.
* Gradient Boosting Classifier (GBC) — он строит предсказание в виде ансамбля слабых предсказывающих моделей, которыми в основном являются деревья решений.
* Gaussian Naive Bayes (NB) — это метод, основанный на использовании теоремы Байеса для определения вероятности принадлежности объекта к классу.
* Support Vector Machines (SVM) — это метод классификации, строящий гиперплоскость, разделяющую объекты разных классов.

Создаем лист для тех моделей, которые будем изучать:

from sklearn.linear\_model import LogisticRegression

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.model\_selection import StratifiedKFold

from sklearn.model\_selection import cross\_val\_score

models = []

models.append(('LR', LogisticRegression(solver='lbfgs', max\_iter=1000)))

models.append(('LDA', LinearDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVC', SVC(gamma='auto')))

models.append(('GBC', GradientBoostingClassifier()))

После чего оцениваем их метрики:

results = []

model\_names = []

for name, model in models:

kfold = StratifiedKFold(n\_splits=10, random\_state=1, shuffle=True)

cv\_results = cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring='accuracy')

results.append(cv\_results)

model\_names.append(name)

print('%s: %f (%f)' % (name, cv\_results.mean(), cv\_results.std()))

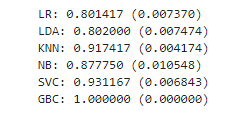


Рисунок 18. Оценка метрик

Как видим, из оценок что Gradient Boosting Classifier (GBC) показал лучшую метрику среди всех алгоритмов, его мы и будем использовать.

Алгоритм работы Gradient Boosting можно описать следующим образом:

1. Инициализация базовой модели с некоторыми начальными предсказаниями.
2. Расчёт остатков между предсказаниями базовой модели и значениями, которые мы получили на самом деле. Остатки становятся целевыми для следующей модели.
3. Обучение новой модели на остатках, предполагается что это улучшит предсказания новой модели.
4. Обновление предсказаний
5. Повторение всех вышеперечисленных шагов.
6. Финальный результат, созданный с помощью комбинации всех моделей.

Создадим модель:

model = ensemble.GradientBoostingClassifier()

**Этап 4. Обучение модели**

Следуем всем вышеперечисленным этапам. Для начала определим нашу модель Gradient Boosting Classifier, после чего подгоним/обучим нашу модель на X\_train и Y\_train с помощью метода .fit(), и в конце произведёт прогноз:

moddel = GradientBoostingClassifier()

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

predicted\_prob = model.predict\_proba(X\_test)

predicted = model.predict(X\_test).astype('int32')

По итогу мы получим вектор вероятности отнесения этого объекта к классам для каждого из объекта:

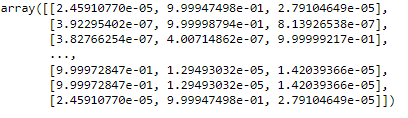


Рисунок 19. Вектор вероятности для каждого объекта

**Этап 5. Оценка модели**

В задачах машинного обучения для оценки качества модели используются метрики и графики. В нашей задаче, а именно задаче на классификацию будет использоваться кривая ошибок или ROC- кривая, для оценки производительности полученной модели. Так же будет использоваться F1-score метрика. Они позволяют оценить точность модели с точки зрения способность правильно определять положительные примеры и гармоническое среднее. Важно понимать, что для каждой задачи необходимо использовать подходящие под условия задачи метрики, часто требуется экспериментировать с различными подходами чтобы получить лучший результат.

Так же найдём Accuracy которая является одной из самых простых метрик, она показывается количество правильно проставленных меток класса. Она считается следующим образом:

Где TP - истинно положительный исход, TN – истинно отрицательный, FP – ложно положительный, FN – ложно отрицательный

Recall – это метрика определяющая количество истинно положительных среди всех меток класса, которые были определены как положительные, данная метрика вычисляется следующим образом:

В начале оценим точность классификации и выведем основные метрики для оценки:

accuracy = metrics.accuracy\_score(y\_test, predicted)

auc = metrics.roc\_auc\_score(y\_test, predicted\_prob, multi\_class='ovo',average= "weighted") #

print("Accuracy (overall correct predictions):", round(accuracy,2))

print("Auc:", round(auc,2))

## Precision e Recall

recall = metrics.recall\_score(y\_test, predicted, average="weighted")

precision = metrics.precision\_score(y\_test, predicted, average="weighted")

print("Recall (all 1s predicted right):", round(recall,2))

print("Precision:", round(precision,2))

f1 = f1\_score(y\_test, predicted, average='weighted')

print(f'F1 Score: {f1:.2f}')

print("Detail:")

print(metrics.classification\_report(y\_test, predicted, target\_names=[str(i) for i in np.unique(y\_test)]))

По итогу получим следующие значения для метрик:

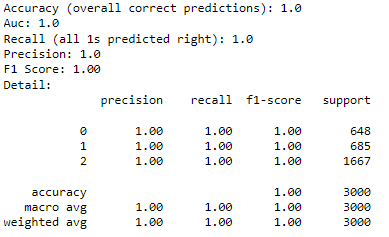


Рисунок 20. Оценка метрик и точности модели

Далее построим ROC- кривую. Это график, который показывает зависимость верно классифицируемых объектов положительного класса от ложно положительного. Чтобы построить кривую воспользуемся iloc, он позволяет получить определенное значение, принадлежащее к строке и столбцу,

используя указанные значения индекса

X\_train — это весь экземпляр с атрибутами,

y\_train — это метка каждого экземпляра.

train\_selected\_test - разделяет массивы / матрицы на случайные тестовые подмножества:

cv = model\_selection.StratifiedKFold(n\_splits=10, shuffle=True)

tprs, aucs = [], []

mean\_fpr = np.linspace(0,1,100)

fig = plt.figure(figsize=(10,5),dpi=100)

i = 1

for train, test in cv.split(X\_train, y\_train):

X\_train\_selected = X\_train.iloc[train]

y\_train\_selected = y\_train.iloc[train]

X\_train\_selected\_test = X\_train.iloc[test]

y\_train\_selected\_test = y\_train.iloc[test]

prediction = model.fit(X\_train\_selected, y\_train\_selected).predict\_proba(X\_train\_selected\_test)

fpr, tpr, t = metrics.roc\_curve(y\_train\_selected\_test, prediction[:, 1], pos\_label=1)

tprs.append(np.interp(mean\_fpr, fpr, tpr))

roc\_auc = metrics.auc(fpr, tpr)

aucs.append(roc\_auc)

plt.plot(fpr, tpr, lw=2, alpha=0.3, label='ROC fold %d (AUC = %0.2f)' % (i, roc\_auc))

i = i+1

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

mean\_tpr = np.mean(tprs, axis=0)

mean\_auc = metrics.auc(mean\_fpr, mean\_tpr)

plt.plot(mean\_fpr, mean\_tpr, color='blue', label=r'Mean ROC (AUC = %0.2f )' % (mean\_auc), lw=2, alpha=1)

plt.xlabel('False Positive Rate')

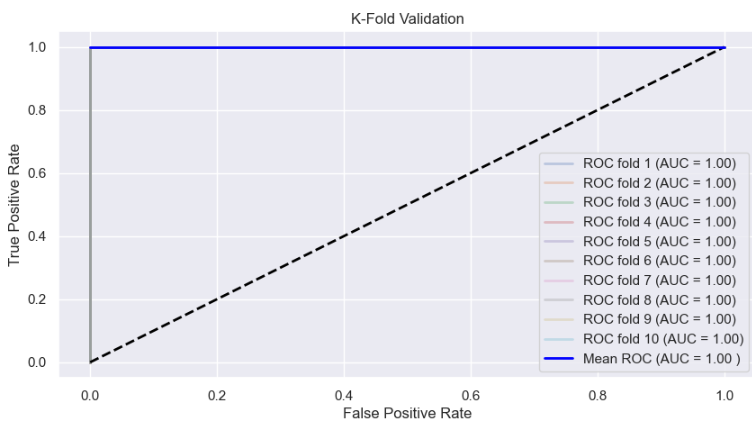
plt.ylabel('True Positive Rate')

plt.title('K-Fold Validation')

plt.legend(loc="lower right")

plt.show()

Получим результат:

Рисунок 21. ROC кривые

Матрица ошибок – это способ визуализации, чтобы оценить качество классификаторов. Чаще всего её используют в машинном обучении с учителем. Она представляет собой таблицу, состоящую из соответствующих фактических и предсказанных классов.

Теперь построим её:

classes = np.unique(y\_test)

fig, ax = plt.subplots()

cm = metrics.confusion\_matrix(y\_test, sk\_cart\_pred\_res, labels=classes)

sns.heatmap(cm, annot=True, fmt='d', cmap=plt.cm.Blues, cbar=False)

ax.set(xlabel="Pred", ylabel="True", title="Confusion matrix")

ax.set\_yticklabels(labels=classes, rotation=0)

plt.show()

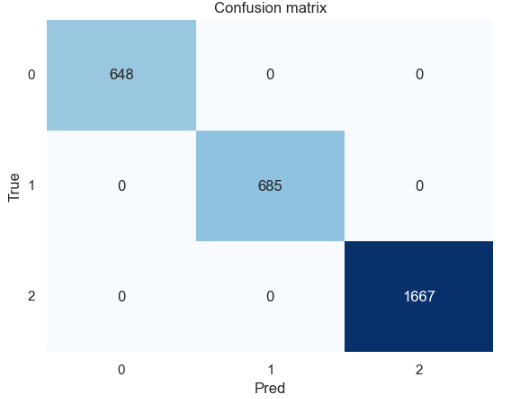


Рисунок 19. Матрица ошибок

**Заключение**

На основе проведённого анализа можно сделать вывод, что выдвинутая в начале гипотеза подтвердилась. Действительно уровень стресса и самочувствие студентов напрямую влияет на уровень риска их здоровья. Проведя данное исследования, мы теперь можем предсказать уровень риска для студента основываясь на его самочувствии и привычках.

Таким образом поставленная цель была достигнута. Был выполнен анализ данных о проведения занятий и состоянии студентов. Построенная модель зависимости привычек на уровень риска студента.

Для достижения этой цели были решены следующие задачи:

* Изучена актуальность темы.
* Осуществлена загрузка и подготовка данных к анализу.
* Создание модели для прогнозирования рисков.
* Оценка полученной модели с помощью метрик.

**Список использованных источников и литературы**

1. Барский, А. Б. Введение в нейронные сети : учебное пособие / А. Б. Барский. — 4-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2024. — 357 c. — ISBN 978-5-4497-2381-9. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт].
2. 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].
3. Ясницкий, Л. Н. 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].
4. Кудрявцев, В. Б. Интеллектуальные системы : учебник и практикум для вузов / В. Б. Кудрявцев, Э. Э. Гасанов, А. С. Подколзин. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 165 с. — (Высшее образование). — ISBN 978-5-534-07779-7. — Текст : электронный // ЭБС Юрайт [сайт].
5. Станкевич, Л. А. Интеллектуальные системы и технологии : учебник и практикум для бакалавриата и магистратуры / Л. А. Станкевич. — Москва : Издательство Юрайт, 2019. — 397 с. — (Бакалавр и магистр. Академический курс). — ISBN 978-5-534-02126-4. — Текст : электронный // ЭБС Юрайт [сайт].
6. Вандер П. Python для сложных задач: наука о данных и машинное обучение. — СПб.: Питер, 2018. — 576 с.
7. Н.А. Климович, М.С. Бобанко. Влияние качества сна на здоровье студентов – Журнал: Международный студенческий научный вестник. 2022 [статья].
8. Васильев, А.Н. Программирование на Python в примерах и задачах / А,Н. Васильев. — Москва : Эксмо, 2021. — 616 с.

**Приложения**

Приложение 1

Программный код

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

from sklearn.preprocessing import LabelEncoder

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, f1\_score, precision\_score, recall\_score, ConfusionMatrixDisplay, roc\_auc\_score, plot\_roc\_curve

from sklearn import model\_selection, preprocessing, feature\_selection, ensemble, linear\_model, metrics, decomposition

from sklearn.metrics import confusion\_matrix

df = pd.read\_csv('student\_monnitoring\_data.csv')

df.head()

df= df.drop(['Date'], axis = 1)

df= df.drop(['Student ID'], axis = 1)

df.head()

df.describe()

df.describe(include=['O'])

df.info()

df.isnull().sum()

df.columns

class\_counts = df['Risk Level'].value\_counts()

print(class\_counts)

sns.countplot(x = df['Risk Level'])

plt.title('Распределение данных целевого признака', fontsize=14)

sns.countplot(data=df, x="Risk Level", hue='Attendance Status');

sns.countplot(data=df, x="Anxiety Level", hue='Risk Level');

sns.countplot(data=df, x="Mood Score", hue='Risk Level');

sns.violinplot(y='Risk Level', x='Stress Level (GSR)', data=df, inner='quartile')

plt.show()

sns.violinplot(y='Risk Level', x='Sleep Hours', data=df, inner='quartile')

plt.show()

df['Attendance Status'] = df['Attendance Status'].map({'Present': 1, 'Absent': 0, 'Late': 2})

df['Risk Level'] = df['Risk Level'].map({'Low': 0, 'Medium': 1, 'High': 2})

df['Class Time'] = LabelEncoder().fit\_transform(df['Class Time'])

df.head()

from sklearn.preprocessing import MinMaxScaler # StandardScaler

scaler\_mm = MinMaxScaler()

df[['Stress Level (GSR)']] = scaler\_mm.fit\_transform(df[['Stress Level (GSR)']])

df[['Sleep Hours']] = scaler\_mm.fit\_transform(df[['Sleep Hours']])

df[['Anxiety Level']] = scaler\_mm.fit\_transform(df[['Anxiety Level']])

df[['Mood Score']] = scaler\_mm.fit\_transform(df[['Mood Score']])

df.head()

plt.figure(figsize=(10,8))

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

plt.show()

sns.pairplot(df, kind="scatter", hue="Risk Level");

X = df.drop(['Risk Level'], axis=1)

y = df['Risk Level']

print(f'X shape: {X.shape} | y shape: {y.shape} ')

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=10)

from sklearn.linear\_model import LogisticRegression

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.model\_selection import StratifiedKFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import GradientBoostingClassifier

models = []

models.append(('LR', LogisticRegression(solver='lbfgs', max\_iter=1000)))

models.append(('LDA', LinearDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVC', SVC(gamma='auto')))

models.append(('GBC', GradientBoostingClassifier()))

results = []

model\_names = []

for name, model in models:

kfold = StratifiedKFold(n\_splits=10, random\_state=1, shuffle=True)

cv\_results = cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring='accuracy')

results.append(cv\_results)

model\_names.append(name)

print('%s: %f (%f)' % (name, cv\_results.mean(), cv\_results.std()))

model = ensemble.GradientBoostingClassifier()

accuracy = metrics.accuracy\_score(y\_test, predicted) #Оценим точность классификации.

auc = metrics.roc\_auc\_score(y\_test, predicted\_prob, multi\_class='ovo',average= "weighted") #возвращаем долю правильно классифицированных выборок.

print("Accuracy (overall correct predictions):", round(accuracy,2))

print("Auc:", round(auc,2))

## Precision e Recall

recall = metrics.recall\_score(y\_test, predicted, average="weighted")

precision = metrics.precision\_score(y\_test, predicted, average="weighted")

print("Recall (all 1s predicted right):", round(recall,2))

print("Precision:", round(precision,2))

#Выведем Основные метрики для оценки

f1 = f1\_score(y\_test, predicted, average='weighted')

print(f'F1 Score: {f1:.2f}')

print("Detail:")

print(metrics.classification\_report(y\_test, predicted, target\_names=[str(i) for i in np.unique(y\_test)]))

cv = model\_selection.StratifiedKFold(n\_splits=10, shuffle=True)

tprs, aucs = [], []

mean\_fpr = np.linspace(0,1,100)

fig = plt.figure(figsize=(10,5),dpi=100)

i = 1

for train, test in cv.split(X\_train, y\_train):

X\_train\_selected = X\_train.iloc[train]

y\_train\_selected = y\_train.iloc[train]

X\_train\_selected\_test = X\_train.iloc[test]

y\_train\_selected\_test = y\_train.iloc[test]

prediction = model.fit(X\_train\_selected, y\_train\_selected).predict\_proba(X\_train\_selected\_test)

fpr, tpr, t = metrics.roc\_curve(y\_train\_selected\_test, prediction[:, 1], pos\_label=1)

tprs.append(np.interp(mean\_fpr, fpr, tpr))

roc\_auc = metrics.auc(fpr, tpr)

aucs.append(roc\_auc)

plt.plot(fpr, tpr, lw=2, alpha=0.3, label='ROC fold %d (AUC = %0.2f)' % (i, roc\_auc))

i = i+1

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

mean\_tpr = np.mean(tprs, axis=0)

mean\_auc = metrics.auc(mean\_fpr, mean\_tpr)

plt.plot(mean\_fpr, mean\_tpr, color='blue', label=r'Mean ROC (AUC = %0.2f )' % (mean\_auc), lw=2, alpha=1)

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('K-Fold Validation')

plt.legend(loc="lower right")

plt.show()

classes = np.unique(y\_test)

fig, ax = plt.subplots()

cm = metrics.confusion\_matrix(y\_test, predicted, labels=classes)

sns.heatmap(cm, annot=True, fmt='d', cmap=plt.cm.Blues, cbar=False)

ax.set(xlabel="Pred", ylabel="True", title="Confusion matrix")

ax.set\_yticklabels(labels=classes, rotation=0)

plt.show()