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»

**ПРОВЕРКА НАЛИЧИЯ ДИАБЕТА МЕТОДАМИ ИИ**

Разработчики проекта:

Сизов Артём Александрович,

Сурков Дмитрий Сергеевич,

Хаюмов Глеб Олегович

Пермь, 2024

**Оглавление**

[ПАСПОРТ ПРОЕКТА 3](#_Toc185632396)

[СОДЕРЖАНИЕ ПРОЕКТА 5](#_Toc185632397)

[Анализ проблемы исследования 5](#_Toc185632398)

[Реализация проекта 6](#_Toc185632399)

[Этап 1. Анализ данных 6](#_Toc185632400)

[Этап 2. Моделирование 15](#_Toc185632401)

[Этап 3. Прогнозирование 16](#_Toc185632402)

[Этап 4. Балансировка 21](#_Toc185632403)

[Заключение 22](#_Toc185632404)

[Список использованных источников и литературы 23](#_Toc185632405)

[Приложения 25](#_Toc185632406)

# **ПАСПОРТ ПРОЕКТА**

**Название проекта:**Проверка наличия диабета методами ИИ.

**Сведения об авторах:**Сурков Дмитрий Сергеевич, Сизов Артём Александрович, Хаюмов Глеб Олегович

**Цель** выполнить анализ медицинских данных о пациентках и построить модель бинарной классификации для прогнозирования наличия диабета на основе различных факторов, связанных со здоровьем, с использованием методов машинного обучения, позволяющую делать точные прогнозы и помогать в диагностике заболевания

**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 медицинские данные о пациентках и определить, возможно ли построить модель бинарной классификации для прогнозирования наличия диабета на основе имеющихся факторов, таких как количество беременностей, уровень глюкозы, артериальное давление, толщина кожной складки, уровень инсулина, ИМТ, функция родословной по диабету и возраст. Дать интерпретацию полученным результатам и оценить точность модели. Сделать выводы о влиянии различных факторов на риск развития диабета.

**Конкретные ожидаемые результаты:**

Построенная модель бинарной классификации с использованием ансамблевых методов и оптимизированных гиперпараметров для прогнозирования наличия диабета.

# **СОДЕРЖАНИЕ ПРОЕКТА**

## **Анализ проблемы исследования**

**Исходные данные**

Набор данных содержит подробные медицинские диагностические показатели, которые были собраны для прогнозирования развития диабета на основе нескольких факторов, связанных со здоровьем. Он состоит из 768 записей о пациентках, каждая из которых характеризуется 8 атрибутами, связанными со здоровьем. Переменная «Результат» указывает, есть ли у пациентки диабет (1) или нет (0). Этот набор данных можно использовать для обучения и тестирования моделей машинного обучения для задач классификации, связанных с прогнозированием диабета.

1. (Pregnancies) Беременности: Количество раз, когда пациентка была беременна.
2. (Glucose) Глюкоза: концентрация глюкозы в плазме крови после 2-часового перорального теста на толерантность к глюкозе.
3. (BloodPressure) Артериальное давление: Диастолическое артериальное давление (мм рт. ст.).
4. (SkinThickness) Толщина кожи: Толщина кожной складки на трицепсе (мм).
5. (Insulin) Инсулин: 2-часовой уровень инсулина в сыворотке крови (мкЕд/мл).
6. (BMI) ИМТ: индекс массы тела (вес в кг/(рост в м)^2).
7. (DiabetesPedigreeFunction) Функция DiabetesPedigree: функция, которая представляет родословную пациента по диабету (т. е. вероятность развития диабета на основе семейной истории).
8. (Age) Возраст: Возраст пациента (годы).
9. (Outcome) Результат: двоичный результат (0 или 1), где 1 означает наличие диабета, а 0 — его отсутствие.

## **Реализация проекта**

### Этап 1. Анализ данных

Загрузим данные в датафрейм и подключим необходимые библиотеки:

*#Импорт бтблиотек*

**from** sklearn.model\_selection **import** GridSearchCV, StratifiedKFold, train\_test\_split

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.svm **import** SVC

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.ensemble **import** RandomForestClassifier, GradientBoostingClassifier

**from** sklearn.naive\_bayes **import** GaussianNB

**import** pandas **as** pd

**from** sklearn.model\_selection **import** GridSearchCV, cross\_val\_score

**from** sklearn.svm **import** SVC

**import** matplotlib.pyplot **as** plt

**from** xgboost **import** XGBClassifier

**from** sklearn.preprocessing **import** RobustScaler

**from** sklearn.preprocessing **import** StandardScaler, MinMaxScaler

**import** warnings

**from** imblearn.over\_sampling **import** SMOTE

**import** numpy **as** np

**import** seaborn **as** sns

warnings**.**filterwarnings("ignore")

**from** imblearn.pipeline **import** Pipeline

**from** sklearn.ensemble **import** StackingClassifier

**from** sklearn.metrics **import** classification\_report, accuracy\_score, roc\_auc\_score

Скачаем данные Kaggle и загрузим:

df **=** pd**.**read\_csv("diabetes\_dataset.csv")

df



Рисунок 1. Исходный датафрейм

Убедимся, что все количественные столбцы имеют числовой тип. Если это не так, выполним преобразование типа столбца к числовому.

df.info()

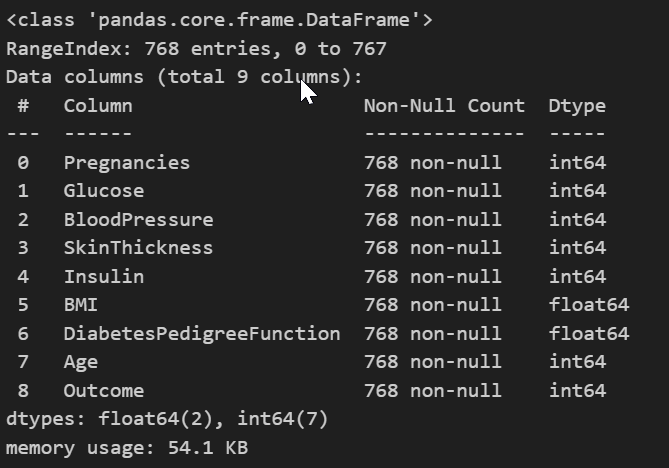


Рисунок 2. Типы данных колонок

Самой элементарной оценкой центрального положения является среднее значение, или среднее арифметическое. Среднее — это сумма всех значений, деленная на число значений. Формула среднего значения для ряда из n значений , следующая:



А самыми известными оценками вариабельности являются дисперсия и стандартное отклонение, которые основаны на квадратических отклонениях. Дисперсия — это среднее квадратических отклонений, а стандартное отклонение — квадратный ко рень из дисперсии. [1]

Дисперсия:

Стандартное отклонение:

Для полноты картины выведем еще квантили. Квантилью уровня *p* функции распределения *F(x)* СВ *X* называется минимальное значение , при котором функция распределения *F(x)* не меньше значения *p,* гдет.е

На рис. 3 указаны квантили уровней некоторой функции распределения *F(x)*

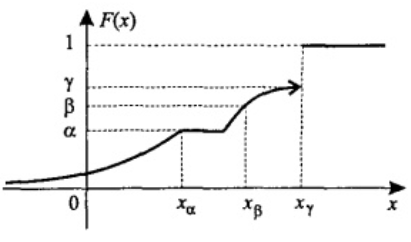


Рисунок 3

Если функция распределения строго монотонна и непрерывна, то квантиль является единственным решением уравнения F(=p. Квантиль уровня p=1/2 называется медианой. [2]

df.describe()

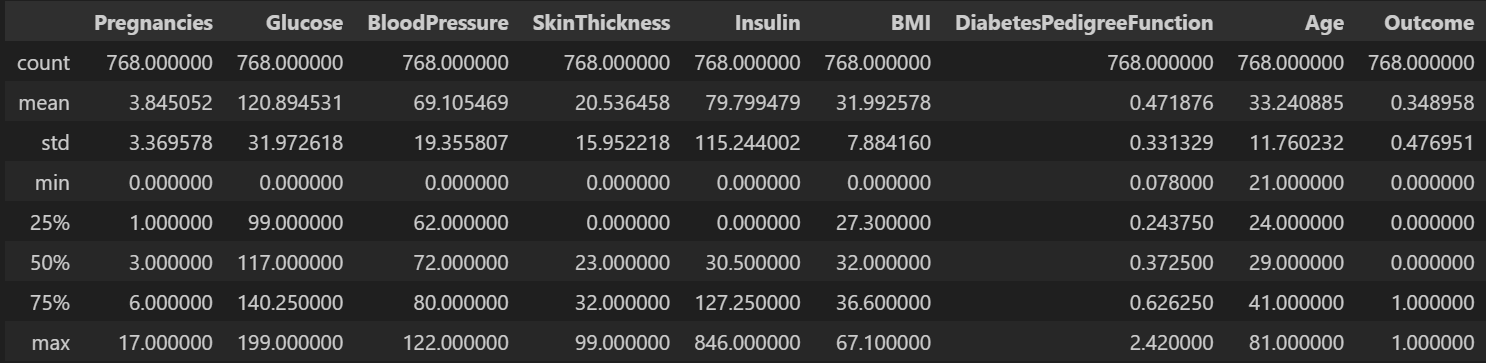


Рисунок 5. Описательные статистики по колонкам

Нарисуем столбчатую диаграмму для отображения распределения значений в целевой переменной Outcome.

count\_classes = pd.value\_counts(df['Outcome'], sort = True)

count\_classes.plot(kind = 'bar', rot=0, color="g")

plt.xticks(range(2), ['Нет диабета','Есть диабет'])

plt.ylabel("Частота")

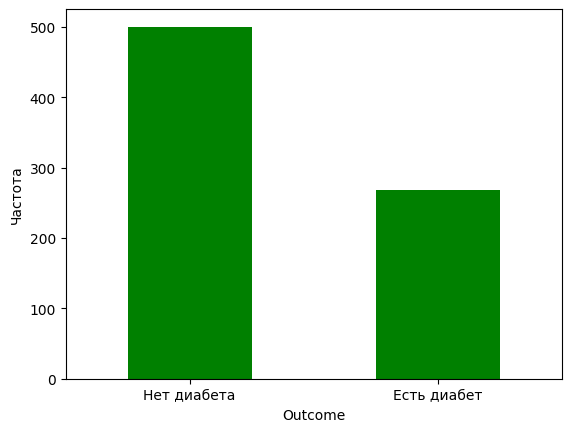


Рисунок 6. Столбчатая диаграмма для визуализации частоты каждого класса

Изображение наглядно демонстрирует, что классы в датасете несбалансированные, так как один из классов значительно преобладает над другим. Это может быть проблемой для обучения.

В задаче классификации данные называются **несбалансированными (Imbalanced Data),** если в обучающей выборке доли объектов разных классов существенно различаются, также говорят, что **«классы не сбалансированы» [3].**

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

Выполним корреляционный анализ данных с помощью матрицы корреляции.

**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.

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

def CorrMatr(dff):

f, ax = plt.subplots(1, 1, figsize=(20, 24))

correlation\_matrix = dff.corr()

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

plt.show()

CorrMatr(df)

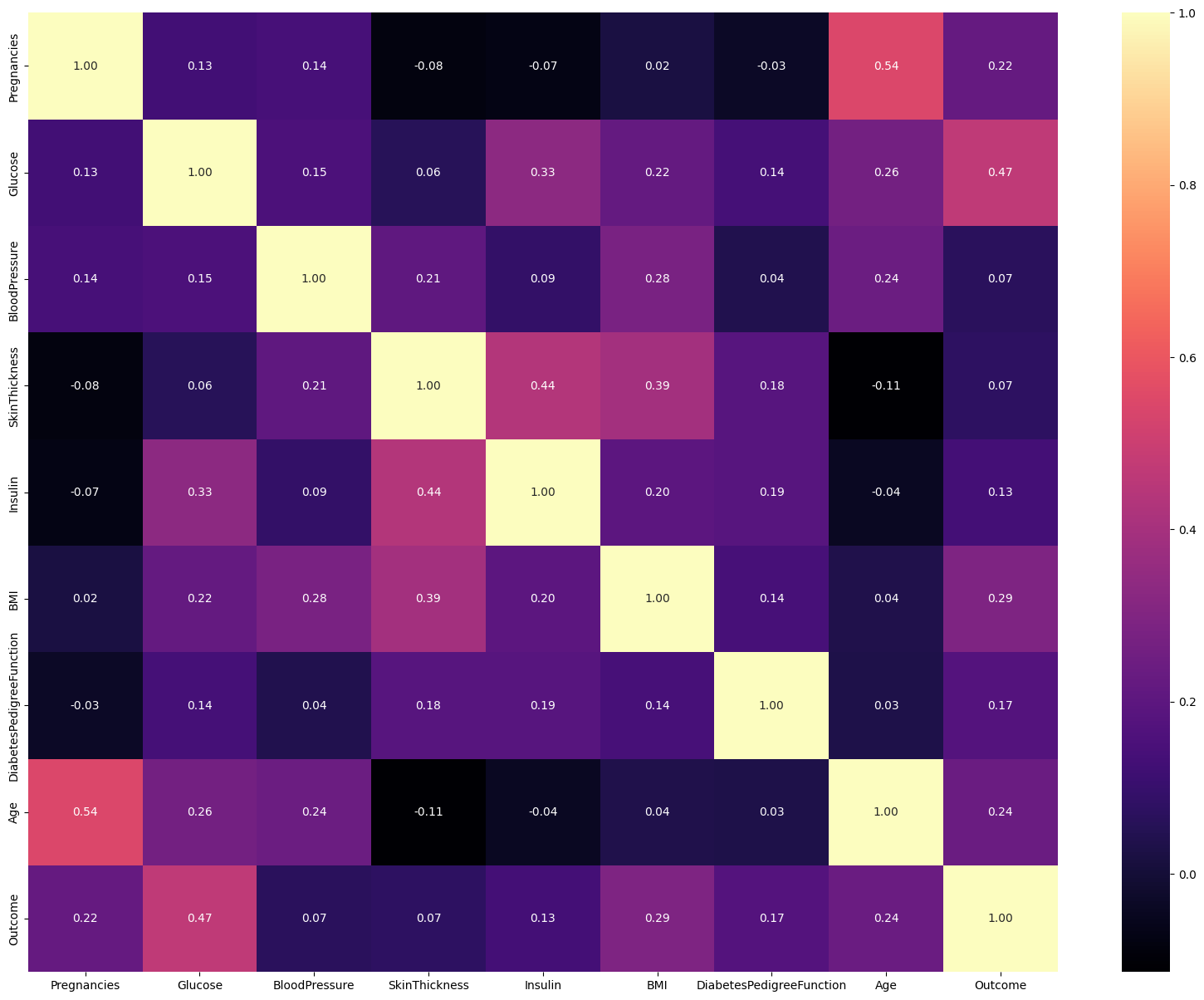


Рисунок 7. Тепловая карта матрицы корреляции

По корреляционной матрице можно сделать следующие выводы: уровень глюкозы является одним из наиболее важных факторов, связанных с диабетом; индекс массы тела и толщина кожной складки также имеют значительную связь с диабетом; генетическая предрасположенность играет роль в риске диабета, хотя и в меньшей степени; артериальное давление и уровень инсулина имеют слабую связь с исходом диабета.

Поскольку значения в корреляционной матрице являются небольшими, целесообразно построить матрицу диаграмм рассеивания для более детального анализа зависимостей между признаками.

plt.figure(figsize=(6, 4))

sns.pairplot(df, hue="Outcome", palette="husl")

plt.show()

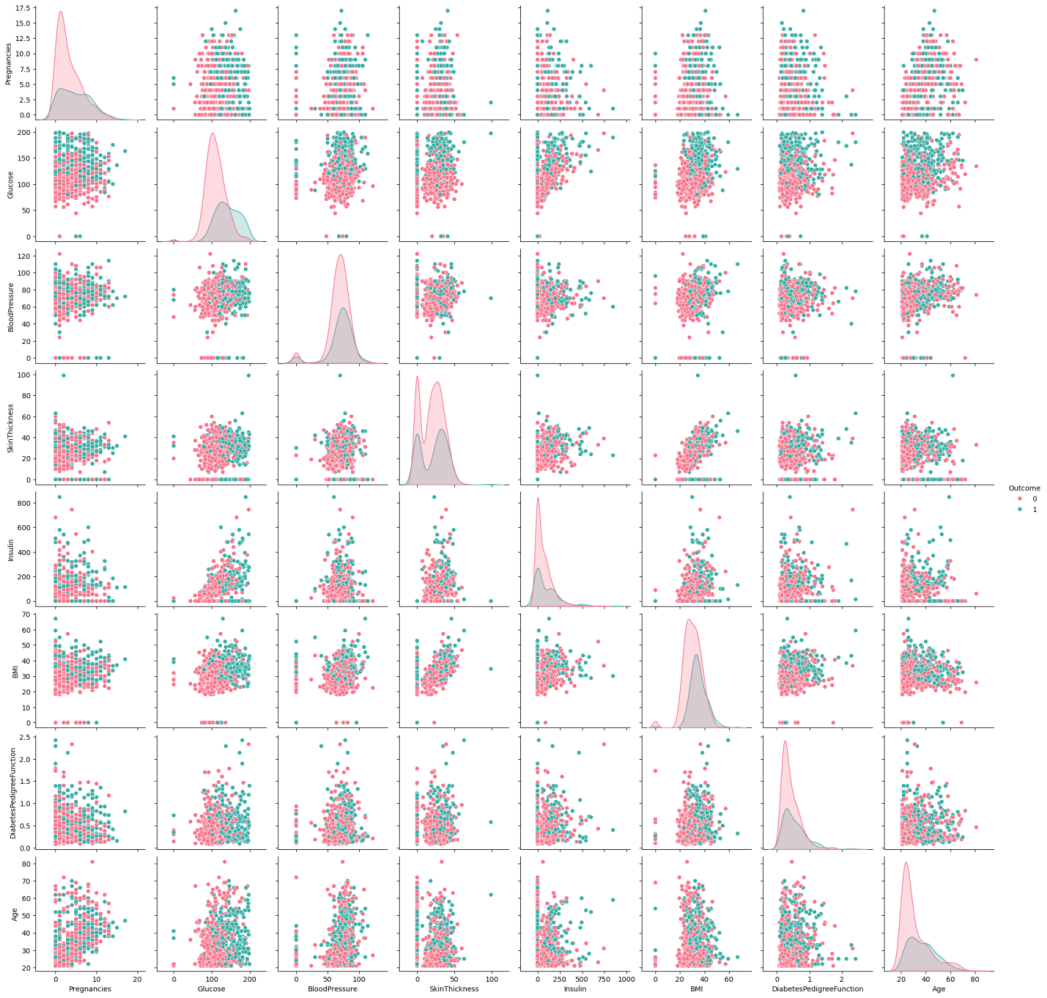


Рисунок 8. Матрица диаграмм рассеивания

На диаграммах рассеивания можно выявить слабые признаки разделения на классы или кластеры, а также определить наличие незначительного количества выбросов.

Чтобы уточнить количество выбросов можно вывести ящики с усами.

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

sns.boxplot(data=df, palette='viridis')

plt.title('Графики с усами для признаков')

plt.xlabel('Признаки')

plt.ylabel('Значения')

plt.show()

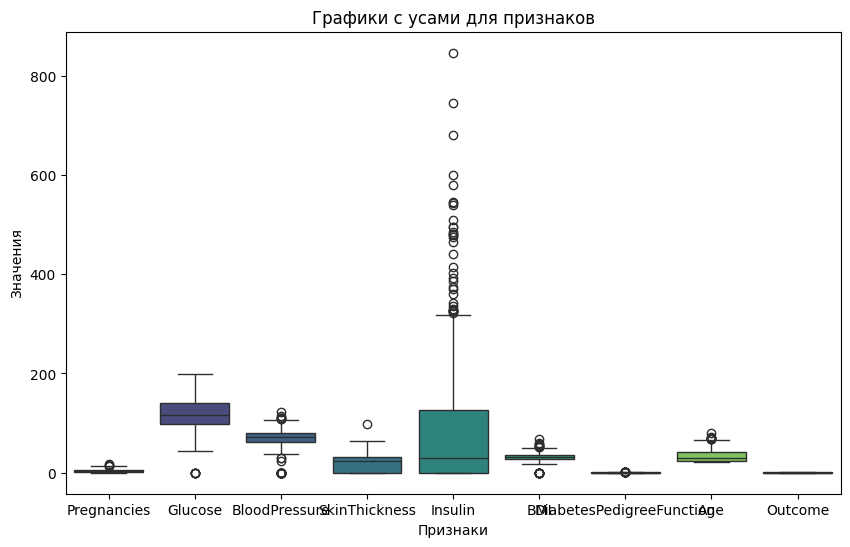


Рисунок 9. Ящики с усами

По графику видно нулевые значения в столбцах Glucose, BloodPressure, BMI и других являются выбросами и могут указывать на ошибки в данных или пропущенные значения. Высокие значения в столбцах Pregnancies, Insulin, BMI и DiabetesPedigreeFunction также могут быть выбросами и требуют дальнейшего анализа

Для еще более детального анализа признаков рисуются графики плотности для каждого числового признака в зависимости от категории Outcome, по которым можно определить предсказательную силу признака (код см. приложение №1). График плотности является одним из самых мощных инструментов для анализа взаимосвязи между двумя непрерывными переменными. На таких графиках показана предполагаемая совместная функция плотности вероятности двух переменных, которая показывает не только расположение точек данных, но и их относительную концентрацию. [5]

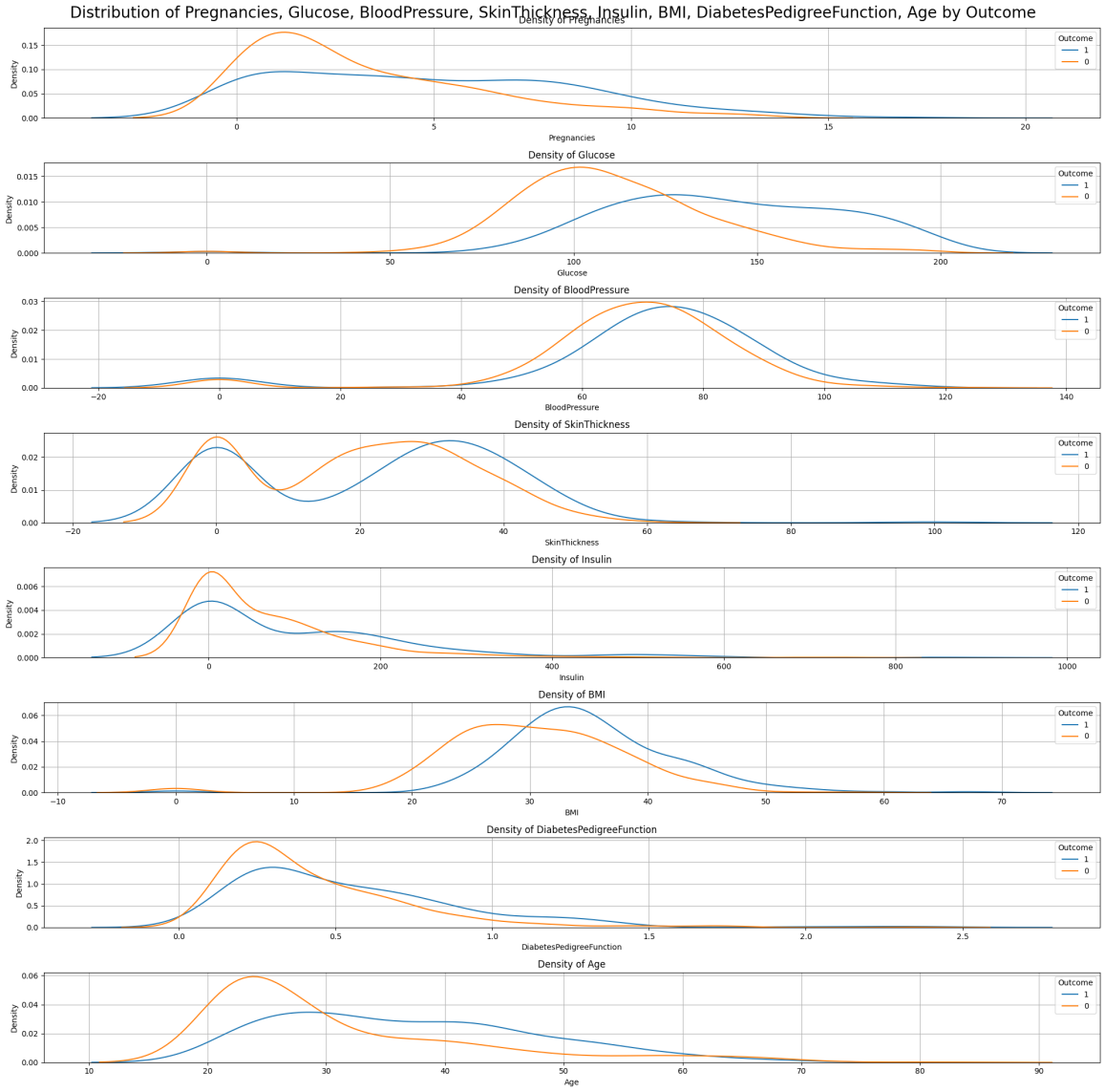


Рисунок 10. Графики плотности

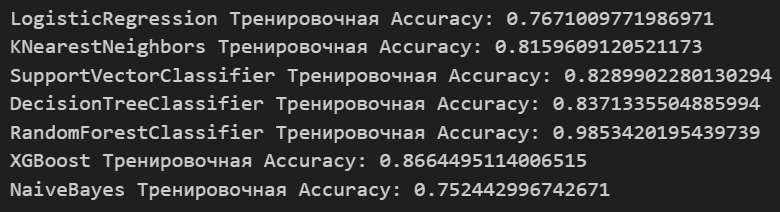
Анализируя графики можно сделалть заключения, что наиболее предсказательные признаки это: Glucose, DiabetesPedigreeFunction, BMI, Age, Pregnancies. Признаки с низкой предсказательной силой: BloodPressure, Insulin, SkinThickness.

### Этап 2. Моделирование

Разобьём данные на обучающую и тестовую выборки, используя соотношение 8 к 2:

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.2, random\_state**=**42)

Сделаем словарь models, который содержит настройки для различных моделей машинного обучения (код см. приложение №2). В него входят следующую модели:

* LogisticRegression (Логистическая регрессия)
* KNearestNeighbors (k-ближайших соседей)
* SupportVectorClassifier (Метод опорных векторов)
* DecisionTreeClassifier (Дерево решений)
* RandomForestClassifier (Случайный лес)
* XGBoost (Градиентный бустинг)
* NaiveBayes (Наивный Байес)
* Выполним поиск гиперпараметров для каждой модели из словаря с использованием кросс-валидации. После поиска лучшие модели сохраняются в список best\_models (код см. приложение №3).
* Результат работы поиска приведен на рисунке 10.
* 
* Рис. 10.

После просмотра точностей на тренировочных данных, и анализа моделей выбрано оставить следующие модели:

* Случайный лес
* k-ближайших соседей
* Градиентный бустинг
* Метод опорных векторов

И отброшены следующие модели:

* Логистическая регрессия — строит линейные границы решений, что ограничивает её способность моделировать сложные, нелинейные зависимости в данных. [6]
* Дерево решений — случайный лес уже включает в себя деревья решений, но в большом количестве и с дополнительными механизмами (например, бэггинг).
* Нативный баесовский метод — основывается на предположении о независимости признаков при заданном классе. В датасете если признаки имеют нелинейные зависимости и NBC не сможет корректно моделировать эти взаимосвязи, что приводит к снижению точности.[8]

После фильтрации моделей создадим ансамблевую модель с использованием метода stacking, где финальной моделью будет логистическая регрессия.

stacking\_model **=** StackingClassifier(

estimators**=**best\_models,

final\_estimator**=**LogisticRegression(),

cv**=**5

)

stacking\_model**.**fit(X\_train, y\_train)

**Этап 3. Прогнозирование**

print("Точность ансамблевой модели на тренировочных данных:", stacking\_model**.**score(X\_train, y\_train))

Точность ансамблевой модели составляет 0.8729641693811075, что больше чем у всех моделей, за исключением случайного леса, который демонстрирует очень высокую точность на тренировочных данных (0.985), что значительно выше, чем у других моделей, таких как KNearestNeighbors (0.816), SupportVectorClassifier (0.829) и XGBoost (0.866). В статье [9] пишется "В целом, увеличение размера дерева или количества деревьев приводит к более точному прогнозированию, но это может привести к переобучению на обучающих данных." ). Это явный признак того, что RandomForestClassifier переобучился на тренировочных данных. Также в статье пишется: "Если вы объедините прогнозы группы регрессионных или классификационных моделей, вы часто будете получать более точные прогнозы, чем с помощью лучшей отдельной модели", что подтверждает факт того, что ансамблевая модель, которая показывает меньшую точность на тренировочных данных, чем RandomForestClassifier, может быть более устойчивой к переобучению и лучше обобщаться на тестовых данных.

Выполним прогнозирование целевой переменной на тестовой выборке.

best\_model **=** grid\_search**.**best\_estimator\_

y\_pred **=** stacking\_model**.**predict(X\_test)

print("\nClassification Report:")

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

print("Test Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Test ROC-AUC:", roc\_auc\_score(y\_test, stacking\_model**.**predict\_proba(X\_test)[:, 1]))

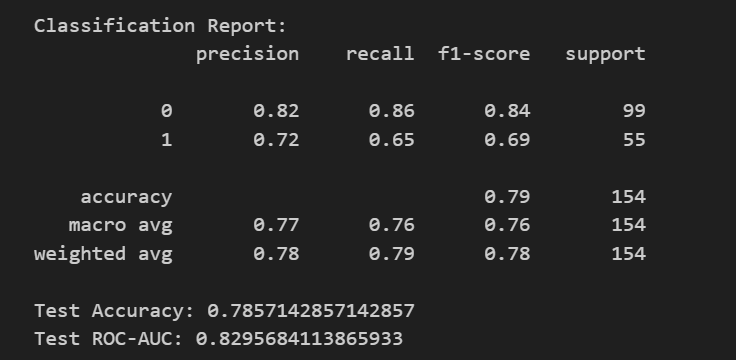
Результат: 

Рисунок 11. Результат прогнозирования переменной на тестовой выборке.

Модель показывает результаты с точностью 0.79, ROC-AUC 0.83, хотя класс 1 (диабет) имеет более низкие метрики (precision 0.72, recall 0.65), что указывает на некоторую склонность к ложноотрицательным ошибкам.

Выполним кросс-валидацию с использованием стратифицированного метода (StratifiedKFold) с 10 фолдами, чтобы оценить производительность модели и построим ROC-кривые для каждого фолда, вычислим площадь под кривой для каждой кривой и затем построим среднюю ROC-кривую на основе всех фолдов. Это позволяет визуализировать стабильность и качество модели на разных подмножествах данных. (код см. приложение №4)

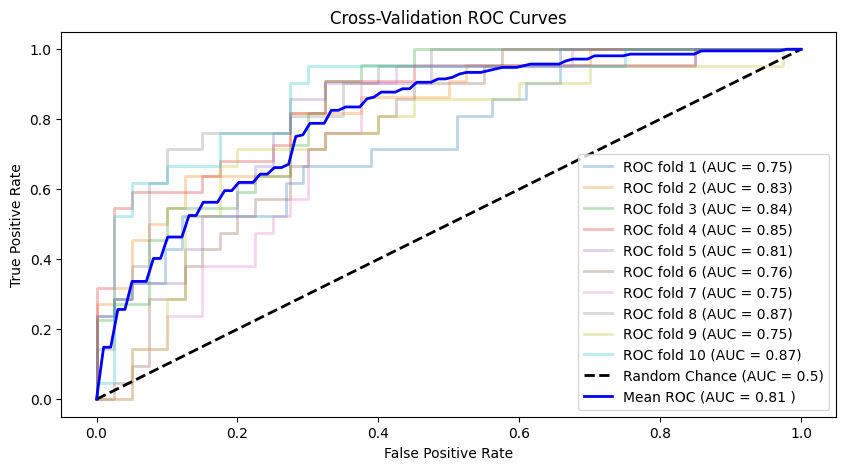
****

Рисунок 12. Рок кривые

График ROC-кривых показывает, что модель демонстрирует стабильные результаты с средним AUC 0.81, хотя есть вариации между фолдами (от 0.75 до 0.87), что свидетельствует о хорошей способности модели различать классы, но с некоторым разбросом в производительности.

Выведем матрицу ошибок. (код см. приложение №5)

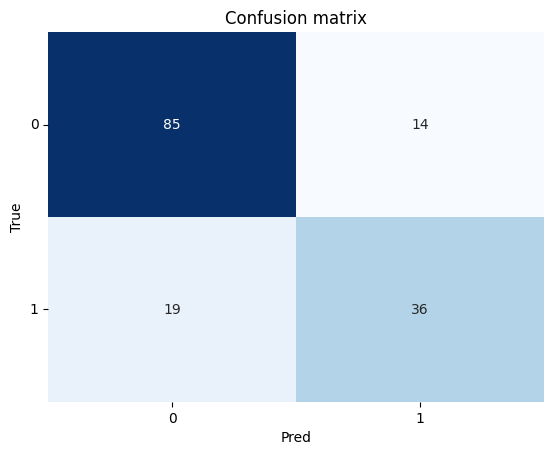


Рисунок 13. Матрица ошибок

Матрица ошибок показывает, что модель правильно классифицировала 85 случаев отсутствия диабета (True Negative) и 14 случаев наличия диабета (False Negative), но допустила 14 ошибок (False Positive) при классификации отсутствия диабета и 36 ошибок (False Negative) при классификации наличия диабета. Вывод: модель склонна к ложноотрицательным ошибкам.

Выведем еще два графика для оценки бинарной классификации (код см. приложение №6)

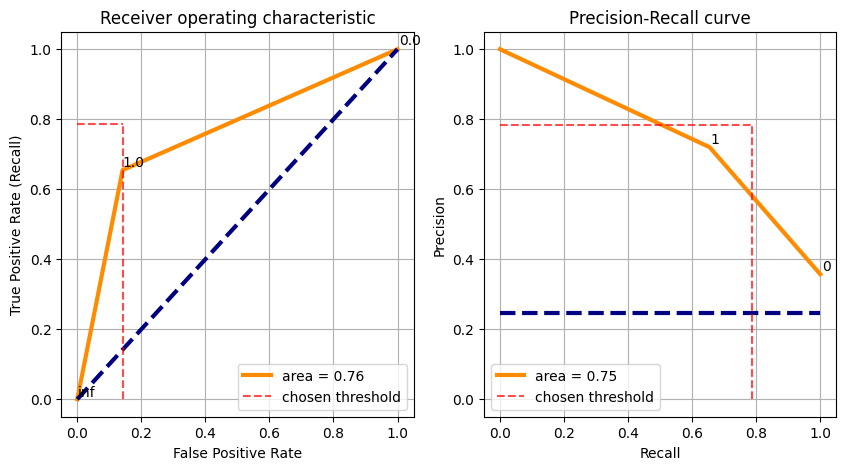


Рисунок 14. Матрица ошибок

Модель демонстрирует удовлетворительное качество классификации с площадью под ROC-кривой (AUC) равной 0.76, что указывает на среднюю способность различать классы. Кривая Precision-Recall также показывает компромисс между точностью (Precision) и полнотой (Recall), что важно для задач с несбалансированными классами. Модель работает достаточно хорошо, но требует дальнейшего анализа и улучшений для достижения более высокой точности и стабильности.

**Этап 4. Балансировка**

На предыдущих этапах анализа было установлено, что классы целевой переменной являются несбалансированными, и существуют предпосылки считать, что причиной склонности модели к ложноотрицательным ошибкам является несбалансированность классов.

Для оценки возможных улучшений качества модели проведены сравнения различных методов балансировки данных (включая отсутствие балансировки). Для каждого метода будет выполнена оценка модели с использованием кросс-валидации и тестовых данных. В качестве метрик оценки будут рассчитаны точность (accuracy), отчет о классификации (classification report) и площадь под кривой ROC-AUC (ROC-AUC). tМетрики приведены в таблице 1:

Таблица 1 – метрики методов балансировки

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Метод | Cross-Validation Accuracy | Test Accuracy | Test ROC-AUC | Precision (0) | Precision (1) |
| NoSampling | 0.7851 +/- 0.0363 | 0.7792 | 0.8288 | 0.81 | 0.71 |
| RandomUnderSampler | 0.7443 +/- 0.0171 | 0.7143 | 0.8173 | 0.84 | 0.58 |
| NearMiss | 0.6955 +/- 0.0337 | 0.6818 | 0.8323 | 0.89 | 0.53 |
| RandomOverSampler | 0.7605 +/- 0.0318 | 0.7468 | 0.8077 | 0.81 | 0.64 |
| SMOTE | 0.7687 +/- 0.0184 | 0.7208 | 0.8134 | 0.82 | 0.59 |
| ADASYN | 0.7589 +/- 0.0260 | 0.7078 | 0.8029 | 0.82 | 0.57 |

# **Заключение**

В ходе исследования был проведен анализ набора данных, содержащего медицинские показатели для прогнозирования диабета. Анализ показал, что данные несбалансированны и есть незначительные выбросы, что может повлиять на качество модели.

На этапе анализа данных были выявлены ключевые признаки, такие как уровень глюкозы, индекс массы тела (ИМТ), возраст и генетическая предрасположенность, которые имеют наибольшую связь с диагнозом диабета. Слабые признаки, такие как артериальное давление и уровень инсулина, были также идентифицированы.

Для построения моделей были протестированы различные алгоритмы машинного обучения, включая логистическую регрессию, k-ближайших соседей, метод опорных векторов, случайный лес, градиентный бустинг и другие. После поиска гиперпараметров и анализа результатов были отобраны модели: случайный лес, k-ближайших соседей, градиентный бустинг и метод опорных векторов. Ансамблевая модель, построенная с использованием метода stacking, показала точность 0.87 на обучающих данных и 0.79 на тестовых данных.

Оценка модели на тестовых данных показала, что модель демонстрирует удовлетворительную точность (0.79) и ROC-AUC (0.83), однако склонна к ложноотрицательным ошибкам, что особенно важно в задачах медицинской диагностики. Кросс-валидация подтвердила стабильность модели с AUC 0.81, хотя наблюдались некоторые вариации между фолдами.

Для улучшения качества модели были протестированы различные методы балансировки данных, включая RandomUnderSampler, NearMiss, RandomOverSampler, SMOTE и ADASYN. Но наилучшие результаты показала модель без балансировки.

# **Список использованных источников и литературы**

1. Брюс, П. Практическая статистика для специалистов Data Science: Пер. с англ. / П. Брюс, Э. Брюс. — СПб.: БХВ-Петербург, 2018. — 304 с.: ил. ISBN 978-5-9775-3974-6
2. Теория вероятностей и математическая статистика. базовый курс с примерами и задачами / А.И.Кибзун, Е.Р.Горяинова, А.В.Наумов, А.Н.Сиротин; — ФИЗМАТЛИТ : 2002. – 224 с
3. Дисбаланс классов [электронный ресурс]  — <https://alexanderdyakonov.wordpress.com/2021/05/27/imbalance/>
4. Сальникова, К. В. Анализ массива данных с помощью инструмента визуализации «Ящик с усами». — ФГБОУ ВО «Ижевский государственный технический университет имени М.Т. Калашникова», 2021. — 7 с.: DOI - 10.32743/UniLaw.2021.81.6.11778
5. Глубокое погружение в 2D-графики плотности для исследовательского анализа данных на Python: взгляд ИИ/МО [электронный ресурс] — <https://www.33rdsquare.com/fundamentals-of-exploratory-data-analysis/> (дата последнего обращения: 12.09.21)
6. Преимущества и недостатки логистической регрессии [электронный ресурс] — <https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/> (дата последнего обращения: 14.09.21)
7. Комплексное обучение с использованием SVM и деревьев принятия решений [электронный ресурс] — <https://www.geeksforgeeks.org/ensemble-learning-with-svm-and-decision-trees/> (дата последнего обращения: 17.09.21)
8. Наивный Байесовский классификатор Простой Байесовский классификатор и реализация на Python [электронный ресурс] — <https://blog.csdn.net/weixin_42555080/article/details/90739046> (дата последнего обращения: 18.09.21)
9. Модели ансамбля [электронный ресурс] — <https://github.com/mikecinnamon/MLearning/blob/main/%5BML-13%5D%20Ensemble%20models.md> (дата последнего обращения: 18.09.21)

**Приложения**

Приложение 1

Программный код для графиков плотности

import matplotlib.pyplot as plt

import seaborn as sns

def plot\_distribution\_for\_columns(df, cat, num\_columns, figsize=(20, 20)):

fig, ax = plt.subplots(nrows=len(num\_columns), ncols=1, sharex=False, sharey=False, figsize=figsize)# распаковываем кортеж в переменные figи ax.

fig.suptitle(f"Distribution of {', '.join(num\_columns)} by {cat}", fontsize=20)# получаем объект рисунка и массив подграфиков,где первое число - количество строк,второе - столбцов, sharex, sharey задают нужно ли разделять ось между подграфиками

# Проверяем, если передан только один столбец, то ax будет не массивом, а одним объектом

if len(num\_columns)==1:

ax=[ax]

# графики плотности для каждого столбца

for i, num in enumerate(num\_columns):

ax[i].title.set\_text(f'Density of {num}')

for category in df[cat].unique():

sns.kdeplot(df[df[cat] == category][num], label=category, ax=ax[i])

ax[i].grid(True)

ax[i].legend(title=cat)

plt.tight\_layout()

plt.show()

cat = "Outcome"

num\_columns =[

'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age'] # Список столбцов для построения графиков

plot\_distribution\_for\_columns(df, cat, num\_columns)

Приложение 2

Словарь для перебора гиперпараметров

models **=** {

"LogisticRegression": { *# Логистическая регрессия*

"model": LogisticRegression(),

"params": {

"C": [0.01, 0.1, 1, 10], *# обратное значение силы регуляризации. меньшие значения указывают на более сильную регуляризацию*

"penalty": ["l1", "l2"], *# параметр регуляризации*

"solver": ["lbfgs", "liblinear"] *#оптимизация*

}

},

"KNearestNeighbors": {

"model": KNeighborsClassifier(),

"params": {

"n\_neighbors": [3, 5, 7], *#количество ближайших соседей*

"weights": ["uniform", "distance"], *#параметр, который определяет, как вес должен распределяться между значениями соседей*

"p": [1, 2] *#метрика для вычисления расстояний*

}

},

"SupportVectorClassifier": {

"model": SVC(probability**=True**),

"params": {

"C": [1, 10],

"kernel": ["linear", "rbf"], *#ядро*

"gamma": ["scale"] *# параметр, определяющий степень влияния между обучающими образцами. Чем ниже его значение, тем больше радиус влияния образцов на формирование решающей границы, что делает её более гладкой*

}

},

"DecisionTreeClassifier": {

"model": DecisionTreeClassifier(),

"params": {

"criterion": ["gini"], *# rритерий разделения узлов дерева (коэффициент Джини)*

"max\_depth": [5, 7], *# максимальная глубина дерева*

"min\_samples\_split": [2, 4], *# минимальное количество образцов, необходимое для разделения узла*

"min\_samples\_leaf": [1, 2] *#минимальное количество образцов, необходимое для того, чтобы узел стал листом*

}

},

"RandomForestClassifier": {

"model": RandomForestClassifier(),

"params": {

"n\_estimators": [100, 200], *# количество деревьев в лесу*

"max\_depth": [5, **None**],

"min\_samples\_split": [2, 4],

"min\_samples\_leaf": [1, 2]

}

},

"XGBoost": {

"model": XGBClassifier(),

"params": {

"n\_estimators": [100, 200], *#количество деревьев в ансамбле*

"max\_depth": [3, 5], *# максимальная глубина деревьев*

"learning\_rate": [0.01, 0.1],

"subsample": [0.8, 1.0], *# доля выборки, используемая для обучения каждого дерева*

"colsample\_bytree": [0.8, 1.0] *# доля признаков, используемых для обучения каждого дерева*

}

},

"GradientBoostingClassifier": {

"model": GradientBoostingClassifier(),

"params": {

"learning\_rate": [0.05, 0.1],

"n\_estimators": [100, 200],

"max\_depth": [3, 5],

"min\_samples\_split": [2, 4],

"min\_samples\_leaf": [1, 2],

"max\_features": [2, 3, 4], *# количество признаков, рассматриваемых при поиске лучшего разделения*

"subsample": [0.8, 1.0] *# доля выборки, используемая для обучения каждого дерева*

}

},

"NaiveBayes": {

"model": GaussianNB(),

"params": {

"var\_smoothing": [1e-9, 1e-10, 1e-11] *# параметр сглаживания дисперсии для стабилизации вычислений*

}

}

}

Приложение 3

Поиск гиперпараметров моделей

best\_models **=** []

**for** name, model\_info **in** models**.**items():

pipeline **=** Pipeline([

('scaler', RobustScaler()),*#нормализация данных*

('model', model\_info["model"])*#модельь*

])

params **=** {f'model\_\_{key}': value **for** key, value **in** model\_info["params"]**.**items()}

grid\_search **=** GridSearchCV(

pipeline,

params,

cv**=**StratifiedKFold(n\_splits**=**5),

scoring**=**'accuracy',

n\_jobs**=-**1

)

grid\_search**.**fit(X\_train, y\_train)

best\_models**.**append((name, grid\_search**.**best\_estimator\_))

print(f"{name} Тренировочная Accuracy: {grid\_search**.**best\_estimator\_**.**score(X\_train, y\_train)}")

Приложение 4

Строительство ROC-кривых

**from** sklearn **import** model\_selection, metrics

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

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\_idx, test\_idx **in** cv**.**split(X\_train, y\_train):

X\_train\_cv **=** X\_train**.**iloc[train\_idx]

y\_train\_cv **=** y\_train**.**iloc[train\_idx]

X\_test\_cv **=** X\_train**.**iloc[test\_idx]

y\_test\_cv **=** y\_train**.**iloc[test\_idx]

best\_model**.**fit(X\_train\_cv, y\_train\_cv)

prediction **=** best\_model**.**predict\_proba(X\_test\_cv)

fpr, tpr, \_ **=** metrics**.**roc\_curve(y\_test\_cv, prediction[:, 1], pos\_label**=**1)

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

tprs[**-**1][0] **=** 0.0

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 **+=** 1

plt**.**plot([0, 1], [0, 1], linestyle**=**'--', lw**=**2, color**=**'black', label**=**'Random Chance (AUC = 0.5)')

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

mean\_tpr[**-**1] **=** 1.0

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('Cross-Validation ROC Curves')

plt**.**legend(loc**=**"lower right")

plt**.**show()

Приложение 5

Матрица ошибок

classes **=** np**.**unique(y\_test)

fig, ax **=** plt**.**subplots()

cm **=** metrics**.**confusion\_matrix(y\_test, y\_pred, 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()

Приложение 6

Визуализации пороговых значени

classes **=** np**.**unique(y\_test)

fig, ax **=** plt**.**subplots(nrows**=**1, ncols**=**2,figsize**=**(10,5),dpi**=**100)

fpr, tpr, thresholds **=** metrics**.**roc\_curve(y\_test, y\_pred, pos\_label**=**1)

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

ax[0]**.**plot(fpr, tpr, color**=**'darkorange', lw**=**3, label**=**'area = %0.2f' **%** roc\_auc)

ax[0]**.**plot([0,1], [0,1], color**=**'navy', lw**=**3, linestyle**=**'--')

ax[0]**.**hlines(y**=**recall, xmin**=**0, xmax**=**1**-**cm[0,0]**/**(cm[0,0]**+**cm[0,1]), color**=**'red', linestyle**=**'--', alpha**=**0.7, label**=**"chosen threshold")

ax[0]**.**vlines(x**=**1**-**cm[0,0]**/**(cm[0,0]**+**cm[0,1]), ymin**=**0, ymax**=**recall, color**=**'red', linestyle**=**'--', alpha**=**0.7)

ax[0]**.**set(xlabel**=**'False Positive Rate', ylabel**=**"True Positive Rate (Recall)", title**=**"Receiver operating characteristic")

ax[0]**.**legend(loc**=**"lower right")

ax[0]**.**grid(**True**)

thres\_in\_plot **=** []

**for** i,t **in** enumerate(thresholds):

t **=** np**.**round(t,1)

**if** t **not** **in** thres\_in\_plot:

ax[0]**.**annotate(t, xy**=**(fpr[i],tpr[i]), xytext**=**(fpr[i],tpr[i]),

textcoords**=**'offset points', ha**=**'left', va**=**'bottom')

thres\_in\_plot**.**append(t)

**else**:

next

precisions, recalls, thresholds **=** metrics**.**precision\_recall\_curve(y\_test, y\_pred, pos\_label**=**1)

roc\_auc **=** metrics**.**auc(recalls, precisions)

ax[1]**.**plot(recalls, precisions, color**=**'darkorange', lw**=**3, label**=**'area = %0.2f' **%** roc\_auc)

ax[1]**.**plot([0,1], [(cm[1,0]**+**cm[1,0])**/**len(y\_test), (cm[1,0]**+**cm[1,0])**/**len(y\_test)], linestyle**=**'--', color**=**'navy', lw**=**3)

ax[1]**.**hlines(y**=**precision, xmin**=**0, xmax**=**recall, color**=**'red', linestyle**=**'--', alpha**=**0.7, label**=**"chosen threshold")

ax[1]**.**vlines(x**=**recall, ymin**=**0, ymax**=**precision, color**=**'red', linestyle**=**'--', alpha**=**0.7)

ax[1]**.**set(xlabel**=**'Recall', ylabel**=**"Precision", title**=**"Precision-Recall curve")

ax[1]**.**legend(loc**=**"lower left")

ax[1]**.**grid(**True**)

thres\_in\_plot **=** []

**for** i,t **in** enumerate(thresholds):

t **=** np**.**round(t,1)

**if** t **not** **in** thres\_in\_plot:

ax[1]**.**annotate(np**.**round(t,1), xy**=**(recalls[i],precisions[i]),

xytext**=**(recalls[i],precisions[i]),

textcoords**=**'offset points', ha**=**'left', va**=**'bottom')

thres\_in\_plot**.**append(t)

**else**:

next

plt**.**show()

Приложение 7

Кросс-валидация методов балансировки

from imblearn.over\_sampling import SMOTE, ADASYN, RandomOverSampler

from imblearn.under\_sampling import RandomUnderSampler, NearMiss

# cписок методов балансировки данных

sampling\_methods = [

('NoSampling', None),

('RandomUnderSampler', RandomUnderSampler()),

('NearMiss', NearMiss()),

('RandomOverSampler', RandomOverSampler()),

('SMOTE', SMOTE()),

('ADASYN', ADASYN())

]

def evaluate\_model(model, X, y, cv=5):

scores = cross\_val\_score(model, X, y, cv=cv, scoring='accuracy', n\_jobs=-1)

print(f"Cross-Validation Accuracy: {scores.mean():.4f} +/- {scores.std():.4f}")

model.fit(X, y)

y\_pred = model.predict(X\_test)

print("\nClassification Report:")

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

print("Test Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Test ROC-AUC:", roc\_auc\_score(y\_test, model.predict\_proba(X\_test)[:, 1]))

# прогон через все методы балансировки

for method\_name, sampler in sampling\_methods:

print(f"\nEvaluating with {method\_name}:")

if sampler is not None:

pipeline = ImbPipeline([

('scaler', RobustScaler()),

('sampler', sampler),

('model', stacking\_model)

])

else:

pipeline = Pipeline([

('scaler', RobustScaler()),

('model', stacking\_model)

])

# Оцениваем модель

evaluate\_model(pipeline, X\_train, y\_train)