ГУАП

КАФЕДРА № 41

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cтарший преподаватель |  |  |  | В. В. Боженко |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5 |
| Классификация 2024 по курсу: ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4217 |  |  |  | В. А. Милованов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**Цель работы**

Изучение алгоритмов и методов классификации на практике.

**Индивидуальное задание**

Вариант 4

Набор данных 4heart2.csv

Данные о болезнях сердца:

1. возраст: возраст пациента (лет)

2. анемия: снижение количества эритроцитов или гемоглобина (логическое значение)

3. высокое кровяное давление: если у пациента гипертония (логическое значение)

4. креатининфосфокиназа (КФК) уровень фермента КФК в крови (мкг/л)

5. диабет: если у пациента диабет (логическое значение)

6. фракция выброса: процент крови, покидающей сердце при каждом сокращении (в процентах)

7. тромбоциты: тромбоциты в крови (килотромбоциты/ мл)

8. пол: женщина или мужчина (бинарный)

9. креатинин сыворотки: уровень креатинина сыворотки в крови (мг/дл)

10. натрий сыворотки: уровень натрия сыворотки в крови (мэкв/л)

11. курение: если пациент курит или нет (логическое)

12. время: период наблюдения (дни)

13. событие смерти: если пациент умер в течение периода наблюдения (логическое значение)

**Ход работы**

Была импортирована база данных согласно варианту 4 (см. рис. 1).

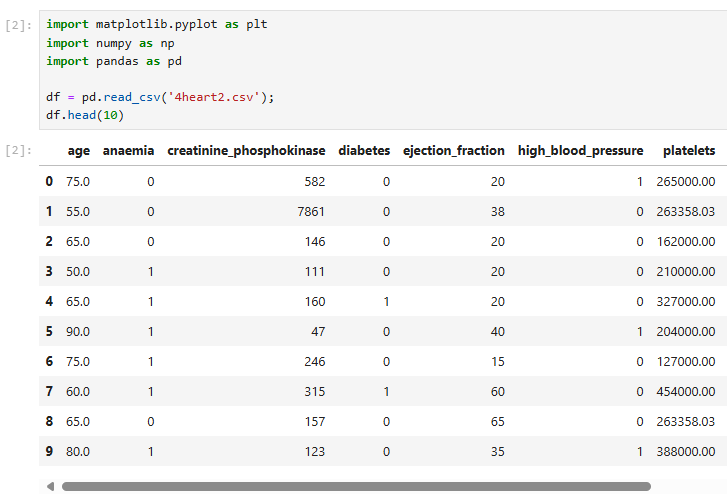


Рисунок 1 – Импорт БД

Проводится первичная обработка данных. Проверяются столбцы, типы и правильность информации. Обработка данных необходима, чтобы избежать ошибок, связанных с некорректными значениями или несоответствием типов данных (см. рис. 2).

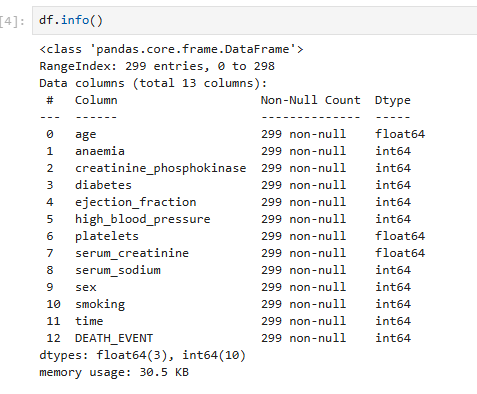


Рисунок 2 – Информация о типах данных

Так как возраст использовается в виде целых значений, было решено поменять его тип данных с float на int (см. рис. 3).

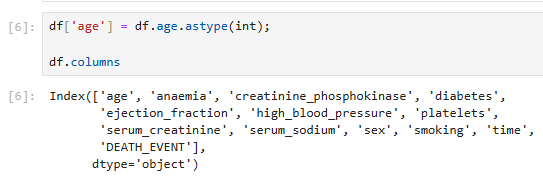


Рисунок 3 – Изменение типа данных и проверка названий

Проверка БД на пропуски (см. рис. 4).

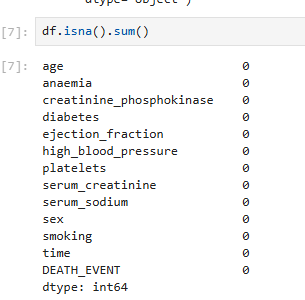


Рисунок 4 – Проверка на пропуски

Остальные характеристики БД соответствуют норме.

С помощью pairplot из библиотеки Seaborn построена матрица диаграмм рассеяния, где данные каждого класса выбранной переменной выделены разными цветами.

Признак DEATH\_EVENT был выделен как целевой, потому что он представляет собой критически важный исход в медицинском анализе данных, отражая факт смерти пациента в течение периода наблюдения (1 — смерть, 0 — выжил). Главной целью исследования является прогнозирование риска смерти на основе предоставленных медицинских показателей, таких как возраст, уровень гемоглобина, артериальное давление, креатинин и другие факторы, которые служат независимыми переменными и влияют на вероятность наступления этого события. Выделение DEATH\_EVENT ставит задачу бинарной классификации, где модели машинного обучения могут предсказывать принадлежность пациента к одной из двух групп: выживших или умерших. Решение этой задачи имеет важное практическое значение, так как позволяет выявить ключевые факторы риска, поддержать врачебные решения и провести раннюю интервенцию для пациентов с высоким риском смерти. Логическая структура данных, где DEATH\_EVENT является бинарным признаком, делает его удобным для моделирования и интерпретации результатов, что обеспечивает построение точных и значимых прогнозов на основе медицинских данных (см. рис. 5).



Рисунок 5 – Матрица рассеивания по целевому признаку

Данные были масштабированы с использованием StandardScaler, который приводит каждый признак к распределению со средним значением 0 и стандартным отклонением 1. Стандартизация данных обязательна для методов, чувствительных к масштабам признаков. Если признаки находятся в разных единицах измерения (например, метры и километры), это может привести к неверным результатам кластеризации.

Из библиотеки sklearn были импортированы методы StandardScaler для нормализации данных и train\_test\_split для разбиения данных на обучающую и тестовую выборки. Нужные столбцы для прогнозирования были выбраны и занесены в переменные X и Y. Нормализация применялась к побочным признакам X. Данные были разделены на X\_train, X\_valid, y\_train и y\_valid. Для обучения модели использовались X\_train и y\_train (см. рис. 6).

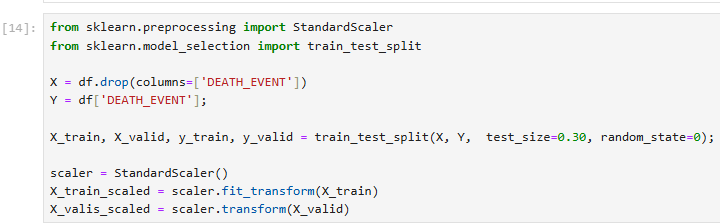


Рисунок 6 – Подготовка данных к обучению

Была обучена модель методом k-ближайших соседей. К-ближайших соседей (K-Nearest Neighbors или просто KNN) — алгоритм классификации, основанный на гипотезе компактности, которая предполагает, что расположенные близко друг к другу объекты в пространстве признаков имеют схожие значения целевой переменной или принадлежат к одному классу. Использовался импортированный метод KNeighborsClassifier c параметром n\_neighbors=5. Такое значение параметра обусловлено его эффективностью по сравнению с остальными числовыми значениями. При значениях до 5 эффективность модели последовательно увеличивается, а после начинает стремительно падать (см. рис. 7).

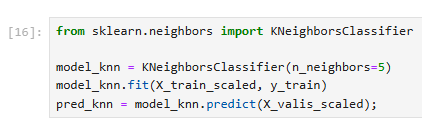


Рисунок 7 – Обучение модели методом классификации KNN

Был импортирован RandomForestClassifier из библиотеки sklearn.ensemble. Создан объект модели случайного леса с фиксированным начальным состоянием генератора случайных чисел (random\_state=42) и количеством деревьев в лесу (n\_estimators=100). Модель была обучена на тренировочных данных с использованием метода fit (см. рис. 8).

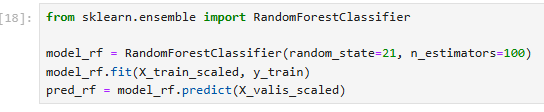


Рисунок 8 – Обучение модели методом классификации RandomForest

Логистическая регрессия — это метод классификации, который используется для предсказания вероятности принадлежности объекта к определенному классу, преобразуя линейную комбинацию признаков через сигмоидальную функцию в значения от 0 до 1. В отличие от деревьев решений, логистическая регрессия представляет собой линейную модель, где целевая переменная является вероятностью, и на основе этого производится прогноз. Этот метод особенно эффективен, когда данные линейно разделимы, и требуется быстрая и интерпретируемая модель. В Python модель логистической регрессии реализована в sklearn.linear\_model.LogisticRegression (см. рис. 9).

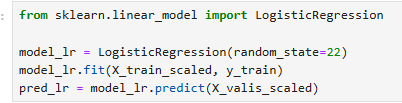


Рисунок 9 – Обучение модели методом классификации LogisticRegression

Дерево решений — это один из наиболее простых и интерпретируемых алгоритмов машинного обучения, который используется для классификации и регрессии. Алгоритм строит структуру, напоминающую дерево, где каждый узел представляет собой условие на основе одного из признаков, а листья — это предсказанные значения. В случае регрессии алгоритм вычисляет среднее значение для объектов, попавших в соответствующий кластер. При обучении дерево решений делит данные на подгруппы, максимизируя разделение между ними, основываясь на минимизации внутрикластерной дисперсии (для задачи регрессии) (см. рис. 10).

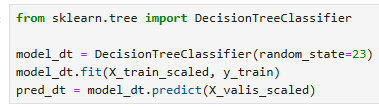


Рисунок 10 – Обучение модели методом классификации DecisionTree

Для каждой модели расчитываются следующие метрики: Accuracy, Precision, Recall, Balanced Accuracy и F1 Score. Accuracy показывает общую долю правильно предсказанных объектов, однако при несбалансированных данных может быть менее информативной, поэтому дополнительно используется Balanced Accuracy, которая усредняет точность для каждого класса. Precision измеряет точность предсказаний положительного класса, определяя, сколько из предсказанных позитивных объектов действительно являются таковыми, тогда как Recall (полнота) показывает, насколько хорошо модель находит все объекты положительного класса. F1 Score гармонически балансирует между Precision и Recall, что особенно полезно, когда важен как минимум ложных срабатываний, так и пропусков. Для вычисления этих метрик предсказания каждой модели сравниваются с истинными значениями на валидационных данных с использованием функций из библиотеки sklearn.metrics. Сравнение результатов позволяет сделать выводы об эффективности каждой модели и выбрать метод, который лучше справляется с поставленной задачей (см. рис. 11-12).



Рисунок 11 – Код реализации метрик

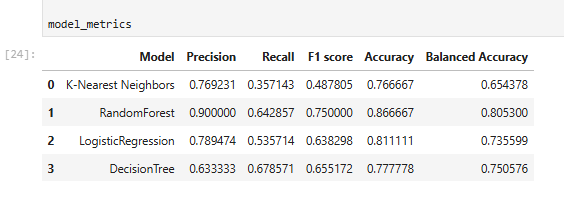


Рисунок 12 – Вывод метрик по каждой модели

Был создан DataFrame для представления результатов работы различных моделей. В столбец Model были внесены названия обученных моделей, таких как метод k-ближайших соседей, случайный лес, логистическая регрессия и дерево решений. Метрики Accuracy, Precision, Recall, Balanced Accuracy и F1 Score для каждой модели были добавлены в соответствующие столбцы, используя заранее вычисленные значения. Для формирования таблицы использовался метод pd.DataFrame, который преобразовал данные в удобный для анализа табличный формат. Итоговая таблица предназначена для сравнения моделей по их качеству. Результаты сравнения приведены в выводе (см. рис. 13-16).

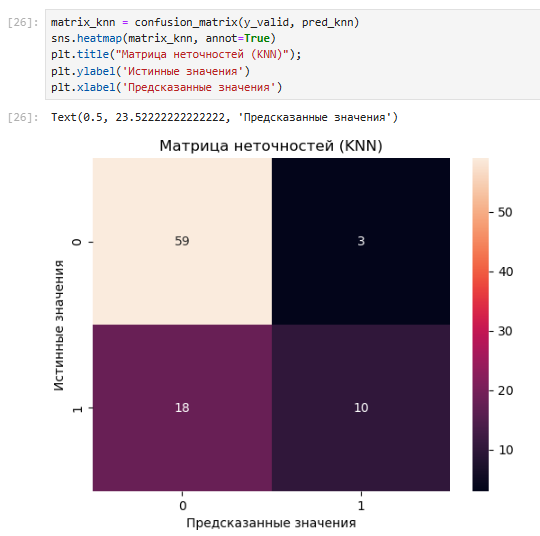


Рисунок 13 – Матрица неточностей (KNN)

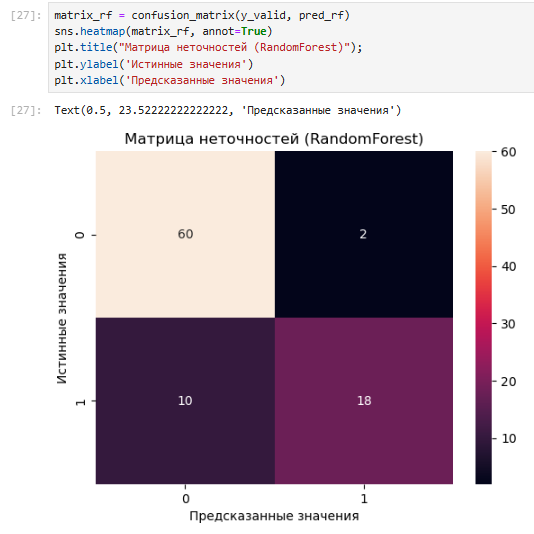


Рисунок 14 – Матрица неточностей (RandomForest)

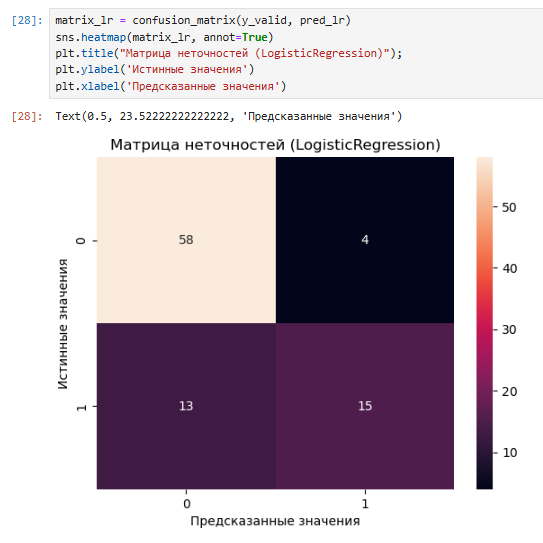


Рисунок 15 – Матрица неточностей (LogisticRegression)

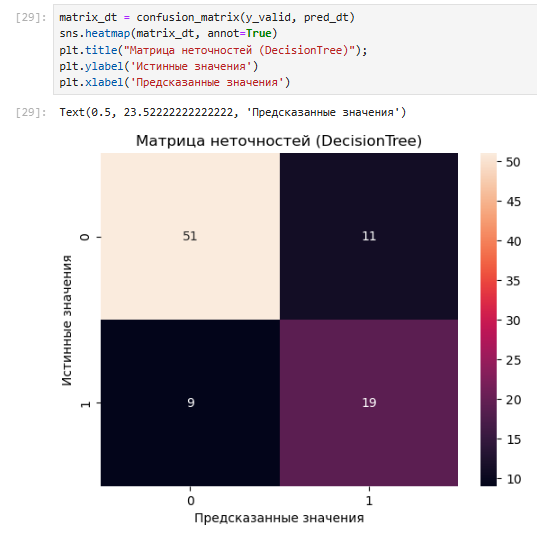


Рисунок 16 – Матрица неточностей (DecisionTree)

На матрицах неточностей для K-Nearest Neighbors (KNN), Random Forest, Logistic Regression и Decision Tree показаны различия в классификации моделей. Для KNN модель правильно предсказала 59 объектов класса 0 и 10 объектов класса 1, но 18 объектов класса 1 были неверно классифицированы как 0, что объясняет более низкую метрику Recall. В случае Random Forest модель имеет наилучшие результаты: правильно определены 60 объектов класса 0 и 18 объектов класса 1, при этом только 10 ошибок для класса 1, что демонстрирует высокую Precision и Recall. Для Logistic Regression модель показала чуть худшие результаты, чем Random Forest: правильно классифицированы 58 объектов класса 0 и 15 объектов класса 1, с 13 ошибками для класса 1, что подтверждает её сбалансированную производительность. Наконец, для Decision Tree модель допустила больше ошибок: 51 объект класса 0 и 19 объектов класса 1 классифицированы правильно, но 11 объектов класса 0 и 9 объектов класса 1 неверно предсказаны, что снижает Precision и Recall по сравнению с остальными моделями. Таким образом, Random Forest демонстрирует наилучшие результаты среди представленных методов, обеспечивая минимальные ошибки и оптимальный баланс между Precision и Recall.

Построение ROC-кривых и AUC для различных моделей (см. рис. 17-18).



Рисунок 17 – Код ROC-кривых и AUC для различных моделей

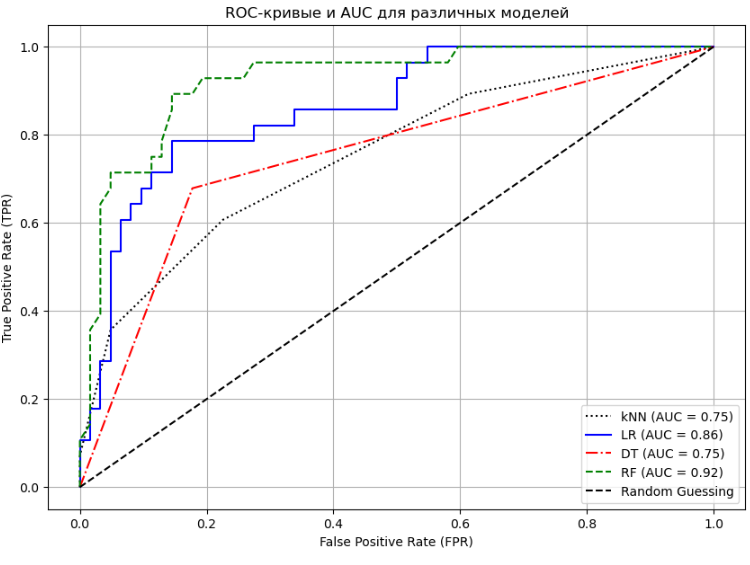


Рисунок 18 – Вывод ROC-кривых и AUC для различных моделей

ROC-кривая (Receiver Operating Characteristic) — это инструмент для оценки качества бинарного классификатора на всех возможных порогах принятия решений. Она строится на основе двух основных метрик: True Positive Rate (TPR) и False Positive Rate (FPR). По оси X на графике откладывается FPR, который показывает долю ложно положительных предсказаний, а по оси Y — TPR, демонстрирующий долю истинно положительных предсказаний. TPR (True Positive Rate), также называемый Recall или Sensitivity, показывает, какая часть объектов положительного класса была правильно предсказана. Чтобы построить ROC-кривую, необходимо рассчитать TPR и FPR для различных порогов классификации. Для этого была импортирована функция roc\_curve, которая принимает на вход вектор истинных меток y\_valid и вектор предсказанных вероятностей probabilities

Значения AUC для различных моделей классификации позволяют оценить их способность разделять классы. Чем выше значение AUC (Area Under the Curve), тем лучше модель справляется с задачей. Вот интерпретация представленных значений:

KNN (AUC = 0.75): K-Nearest Neighbors показывает умеренную способность к классификации. Значение 0.75 указывает, что модель в 75% случаев правильно отличает положительные примеры от отрицательных. Однако из-за ограниченной гибкости KNN модель чувствительна к структуре данных и масштабированию признаков.

Logistic Regression (AUC = 0.86): Логистическая регрессия демонстрирует высокую эффективность (AUC = 0.86), что говорит о хорошем разделении классов. Модель работает лучше KNN и достаточно уверенно определяет вероятность принадлежности объектов к классу 1.

Decision Tree (AUC = 0.75): Дерево решений имеет такое же значение AUC (0.75), как и KNN. Это указывает на умеренное качество классификации, но модель может быть менее устойчивой из-за склонности к переобучению.

Random Forest (AUC = 0.92): Случайный лес показывает наилучший результат среди всех моделей с AUC = 0.92. Это указывает на отличное качество классификации и способность модели эффективно обобщать данные. Random Forest минимизирует переобучение за счет использования ансамбля деревьев и случайности в обучении.

Ссылка на файл:

<https://github.com/Vvvvv55531/LR5.git>

**Вывод**

K-Nearest Neighbors (KNN) Precision: 0.769 (76.9%) — неплохая точность предсказаний положительного класса. Recall: 0.357 (35.7%) — очень низкая полнота, модель пропускает значительное количество объектов положительного класса. F1 score: 0.487 — гармоническое среднее между Precision и Recall показывает низкую сбалансированность. Accuracy: 76.7% — общая точность модели на всех классах. Balanced Accuracy: 65.4% — модель имеет неудовлетворительную производительность на несбалансированных данных. Модель KNN показывает неплохую точность (Precision), но низкую полноту (Recall), что говорит о слабой способности находить объекты положительного класса. Общая и сбалансированная точность ниже, чем у других моделей.

Random Forest Precision: 0.900 (90.0%) — высокая точность предсказаний, модель редко ошибается при определении положительного класса. Recall: 0.643 (64.3%) — хорошая полнота, но есть пропущенные положительные объекты. F1 score: 0.750 — сбалансированный показатель Precision и Recall демонстрирует стабильную производительность. Accuracy: 86.7% — наивысшая общая точность среди всех моделей. Balanced Accuracy: 80.5% — высокая сбалансированная точность на всех классах. Random Forest является лучшей моделью среди представленных. Она демонстрирует высокие показатели точности, сбалансированную полноту и общее качество, что делает её наиболее эффективной для данной задачи.

Logistic Regression Precision: 0.789 (78.9%) — средняя точность, модель достаточно хорошо предсказывает положительный класс. Recall: 0.536 (53.6%) — средняя полнота, модель пропускает около половины положительных объектов. F1 score: 0.638 — показывает умеренный баланс между Precision и Recall. Accuracy: 81.1% — хорошая общая точность. Balanced Accuracy: 73.5% — сбалансированная точность на среднем уровне. Logistic Regression показывает стабильные результаты, уступая Random Forest по всем метрикам. Модель является неплохой, но менее точной и сбалансированной.

Decision Tree Precision: 0.633 (63.3%) — самая низкая точность среди всех моделей, модель часто ошибается при определении положительного класса. Recall: 0.679 (67.9%) — хорошая полнота, модель успешно находит положительные объекты. F1 score: 0.655 — сбалансированный показатель, но ниже, чем у Random Forest. Accuracy: 77.8% — общая точность выше, чем у KNN, но ниже, чем у Logistic Regression и Random Forest. Balanced Accuracy: 75.0% — умеренное значение, модель работает лучше на несбалансированных данных, чем KNN. Decision Tree показывает хорошую полноту (Recall), но низкую точность (Precision), что делает её менее надёжной. Производительность находится на среднем уровне.

Сравнение моделей показывает, что Random Forest является наилучшей моделью, так как она демонстрирует наивысшие показатели Precision, F1 score, Accuracy и Balanced Accuracy. Logistic Regression занимает второе место, показывая стабильные, но менее высокие результаты. Decision Tree имеет хорошую полноту, но низкую точность, что снижает её эффективность. Модель K-Nearest Neighbors (KNN) показывает худшую производительность, особенно по Recall, что указывает на её слабую способность находить объекты положительного класса.