**Пояснительная записка.**

**Итоговая аттестация.**

Тема: «Предсказание рисков возникновения сердечно-сосудистых заболеваний»

Цель работы: Разработка и тестирование приложения для оценки рисков возникновения сердечно-сосудистых заболеваний по распространенным данным лабораторных исследований.

Автор: Файзрахманов Галим Гайсович

Датасеты:

* <https://www.kaggle.com/datasets/khyeh0719/ptb-xl-dataset>
* <https://www.kaggle.com/datasets/sandipdevre/heart1>

Ссылки на notebook

* [Расчет бинарной классификации по лабораторным исследованиям – 1 часть](https://drive.google.com/file/d/17qpv7r0LWkE7kz4fG3dy0Jgv2eSHXmdB/view?usp=sharing)
* [Расчет небинарной классификации по данным ЭКГ](https://drive.google.com/file/d/1CvDwY8AKkSYMH47fTDxNyc0NZIBeFMDt/view?usp=sharing)
* [Запуск пользовательского интерфейса](https://drive.google.com/file/d/1CskPv9PPrZ0218bl4SZ0zbb17jM2pCq3/view?usp=sharing)

Содержание

[1. Анализ предметной области и обзор существующих решений. 3](#_Toc151619212)

[2. Исследование предметной области 5](#_Toc151619213)

[3. Анализ данных лабораторных исследований. 7](#_Toc151619214)

[4. Анализ данных ЭКГ. 10](#_Toc151619215)

[5. Методика исследования и предсказания. 12](#_Toc151619216)

[6. Выбор инструментов для разработки 17](#_Toc151619217)

[7. Технология основного приложения 19](#_Toc151619218)

[8. Выводы. 20](#_Toc151619219)

[Использованная литература 21](#_Toc151619220)

[Приложение 1. Пример пользовательского интерфейса, часть 1. 22](#_Toc151619221)

[Приложение 2. Пример пользовательского интерфейса, часть 2. 23](#_Toc151619222)

[Приложение 3. Пример пользовательского интерфейса, часть 3. 24](#_Toc151619223)

1. Анализ предметной области и обзор существующих решений.

В последние десятилетия произошло резкое изменение образа жизни: изменился и продолжает меняться характер быта, труда, питания, уменьшается физическая активность, все чаще встречаются лица с избыточной массой тела; постоянно меняется среда обитания. Эти факторы не могут положительно влиять на здоровье человека. Они приводят к резкому увеличению психических, психосоматических заболеваний, уменьшению продолжительности жизни, ранней смертности. Появилась необходимость изучения и своевременного определения факторов риска заболеваний, таких как заболевания сердечно-сосудистой системы. На данный момент сформировалась концепция факторов риска, основанная на данных эпидемиологических исследований о наличии тесной связи между определенными факторами внутренней и внешней среды и развитием ишемической болезни сердца. Концепция факторов риска является основой для разработки мероприятий по первичной профилактике сердечно-сосудистых заболеваний (ССЗ).

В связи с этим возникает вопрос о своевременной, многофакторной, комплексной системе определения риска ССЗ. Для оценки риска ССЗ используют различные тесты, каждому из которых соответствует свой блок факторов. Для повышения надежности и валидности тестов при оценке риска ССЗ целесообразно осуществлять тестирование, используя несколько блоков факторов, которые доступны в данный конкретный момент времени для данного пациента, находящегося в конкретном месте. Предлагаемый способ оценки риска ССЗ учитывает максимально возможное, без излишнего увеличения времени на получение и обработку данных, количество блоков факторов. Все блоки факторов, в целях доступности предлагаемого способа, ориентированы в основном на психологическое тестирование и используют минимум средств функциональной диагностики и лабораторного анализа.

В основном текущие тесты либо ссылаются на один из параметров (показатели ЭКГ, анализ гемоглобина и тп.), либо требуют ручной обработки тестов, которые в расчете критериев используют упрощенную модель линейной регрессии.

Среди примеров есть: тест на вхождение в группу риска развития ССЗ [1]; опросник Леонгарда–Шмишека [2]; методика аутоидентификации акцентуаций характера Э.Г. Эйдемиллера [3]; клинический опросник для выявления и оценки невротических состояний К.К. Яхина, Д.М. Менделевича [4]. Эти тесты наиболее полные и достоверной клинической картины исследуемого для возможности использования в последующей оценке риска развития ССЗ. Все вопросы тестов разделены на 4 возможных ответа (совсем нет, нет, верно, совершенно верно), что также не вносит точности в предсказание рисков.

Для решения данной многокритериальной задачи было принято решение применять методы машинного обучения для анализа большого количества данных. Однако все показатели решено взять из наиболее доступных, т.к. сложные лабораторные исследования могут отпугнуть клиента и пациента.

2. Исследование предметной области

В качестве исходного датасета выбраны 2. Один из них включает результаты анализов и тестов, второй включает загрузку показателей с датчиков ЭКГ, которые проводятся в практически в каждой поликлинике.

Кратко опишем каждый датасет. Как уже было сказано, он включает в себя множество параметров и выходной результат. Этот датасет был создан путем объединения различных наборов данных, которые уже были доступны независимо друг от друга, но ранее не объединялись. В этом наборе данных 5 наборов данных о сердечных заболеваниях объединены по 11 общим признакам, что делает его самым большим набором данных о заболеваниях сердца, доступным на данный момент для исследовательских целей. Для его обработки использовались следующие пять наборов данных:

* Кливленд: 303 наблюдения
* Венгрия: 294 наблюдения
* Швейцария: 123 наблюдения
* Лонг-Бич, Вирджиния: 200 наблюдений
* Набор данных Stalog: 270 наблюдений

Исследуемые показатели:

* Возраст: возраст пациента [годы]
* Пол: пол пациента [M: Мужчина, F: Женщина]
* Тип боли в груди: тип боли в груди [TA: Типичная стенокардия, ATA: Атипичная стенокардия, NAP: Боль, не связанная со стенокардией, ASY: Бессимптомная]
* АД в покое: артериальное давление в покое [мм рт.ст.]
* Холестерин: уровень холестерина в сыворотке крови [мм/дл]
* Уровень сахара в крови натощак [1: если уровень сахара в крови натощак > 120 мг/дл, 0: в противном случае]
* ЭКГ в покое: результаты электрокардиограммы в состоянии покоя [Норма: Нормальный, ST: наличие аномалии зубца ST-T (инверсии зубца T и/или подъем или понижение ST > 0,05 мВ), ГЛЖ: наличие вероятной или определенной гипертрофии левого желудочка по критериям Эстеса]
* MaxHR: достигнутая максимальная частота сердечных сокращений [Числовое значение от 60 до 202]
* Стенокардия при физической нагрузке: стенокардия, вызванная физической нагрузкой [Y: Да, N: Нет]
* Oldpeak: oldpeak = ST [Числовое значение, измеренное при депрессии]
* ST\_Slope: наклон сегмента ST при максимальном упражнении [Вверх: наклонный, плоский: плоский, вниз: наклонный]
* Сердечная недостаточность: выходной класс [1: болезнь сердца, 0: Норма]

Пару слов о втором датасете. Электрокардиография (ЭКГ) является ключевым диагностическим инструментом для оценки состояния сердца пациента. Алгоритмы автоматической интерпретации ЭКГ в качестве систем поддержки диагностики обещают значительное облегчение работы медицинского персонала - только на основе количества ЭКГ, которые снимаются в обычном режиме. Однако разработка таких алгоритмов требует большой выборки обучающих данных и четких процедур тестирования. По моему мнению, оба аспекта не охвачены удовлетворительно существующими свободно доступными наборами данных ЭКГ. Набор данных ЭКГ PTB-XL представляет собой большой набор данных из 21837 клинических ЭКГ, измеренных в 12 учреждениях от 18885 пациентов продолжительностью 10 секунд. Необработанные данные о форме волны были прокомментированы кардиологами. В общей сложности 71 различная ЭКГ-запись соответствует стандарту SCP-ECG и охватывает диагностику, форму и ритм. Чтобы обеспечить сопоставимость алгоритмов машинного обучения, обученных на основе набора данных, мы предоставляем рекомендуемые разбивки на обучающие и тестовые.

Набор данных включает 21837 клинических записей ЭКГ в 12 отведениях продолжительностью 10 секунд от 18885 пациентов, из которых 52% составляют мужчины и 48% - женщины, возраст которых охватывает весь диапазон от 0 до 95 лет. Ценность набора данных обусловлена обширной коллекцией множества различных сопутствующих патологий, а также большой долей здоровых контрольных образцов.

Файлы формы сигнала хранятся в формате базы данных при разрешении 500 Гц (records500/). Для удобства также предоставлены уменьшенные версии данных формы сигнала с частотой дискретизации 100 Гц (records100/).

3. Анализ данных лабораторных исследований.

Показатели, представленные в датасете описаны выше, в датафрейме они выглядят следующим образом:

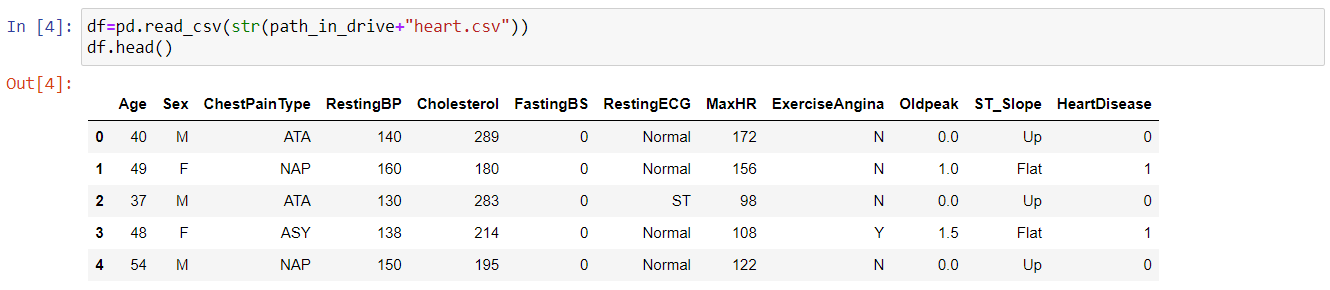


Рисунок 1. Пример таблицы с данными.

Ниже показано распределение каждого показателя:

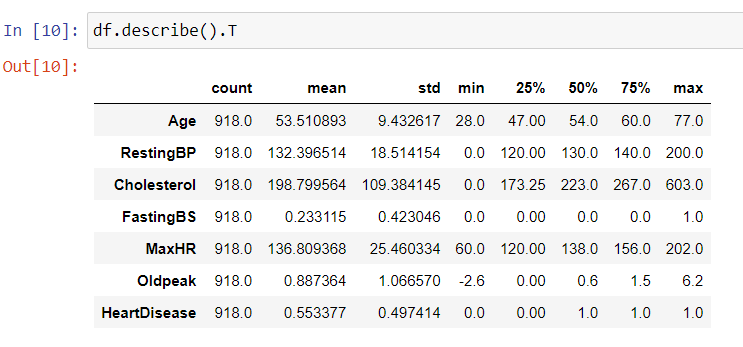


Рисунок 2. Общий анализ всех представленных данных.

По корреляционной матрице можно сделать вывод, что по, что сердечные заболевания имеют высокую отрицательную корреляцию с "MaxHR" и несколько отрицательную корреляцию с "холестерином". Положительная корреляция с "Oldpeak", "FastingBS" и "RestingBP".

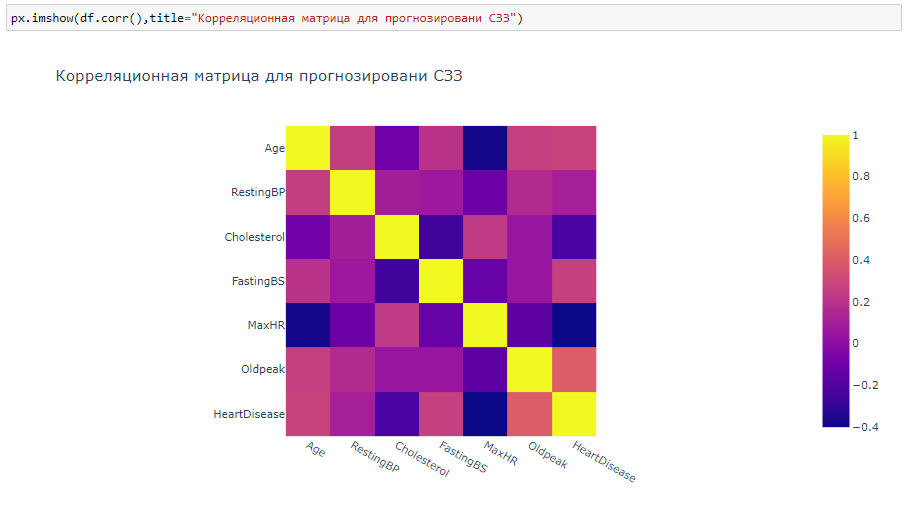


Рисунок 3. Матрица взаимокорреляции.

Также из интересных и важных замечаний. На следующей гистограмме видно, что в большей части риску заболевания подвержены мужчины.



Рисунок 4. Распределение данных по параметрам здоровья и пола.

Для того чтобы оценить взаимную корреляцию параметров и по возможности исключить один из них, оценим следующий график:



Рисунок 5. Pairplot по каждому из параметров.

Видно, что между многими параметрами есть сложная взаимная корреляция, однако отсутствуют явные линейные или другие корреляции, что говорит о том, что для дальнейшего анализа можно использовать все имеющиеся столбцы.

4. Анализ данных ЭКГ.

Показатели ЭКГ выглядит следующим образом, это образец кривой разбитой на показатели с разных датчиков:

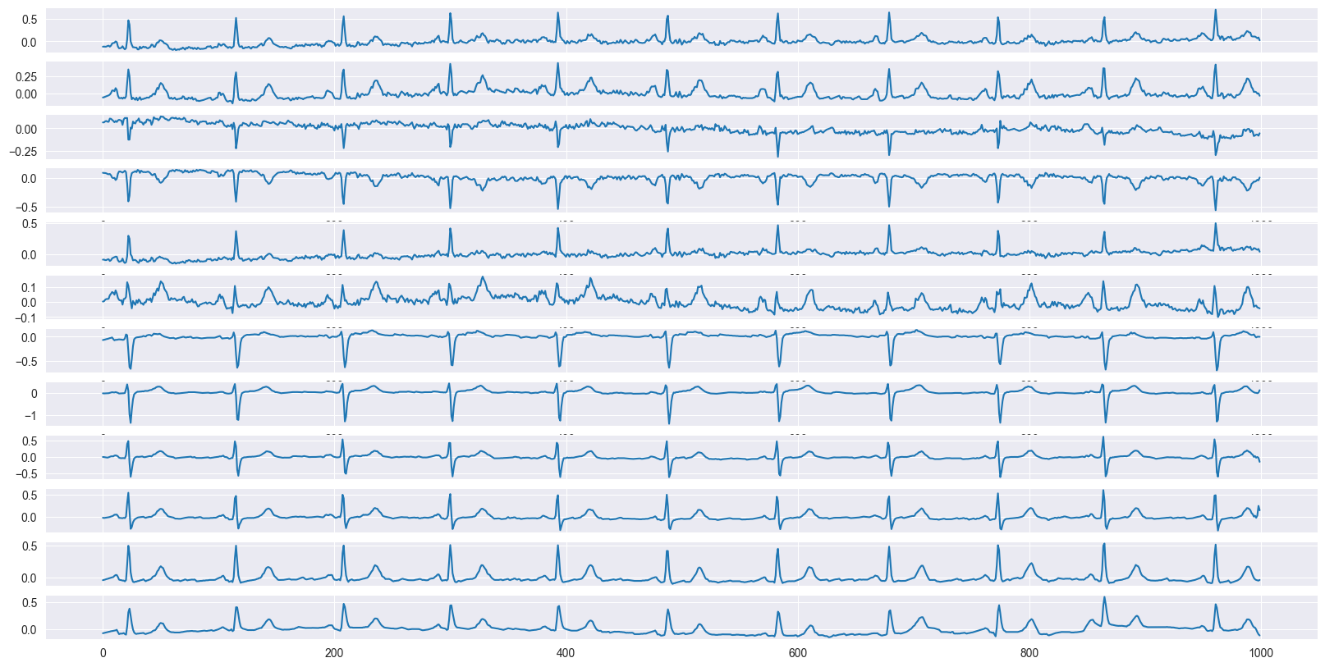
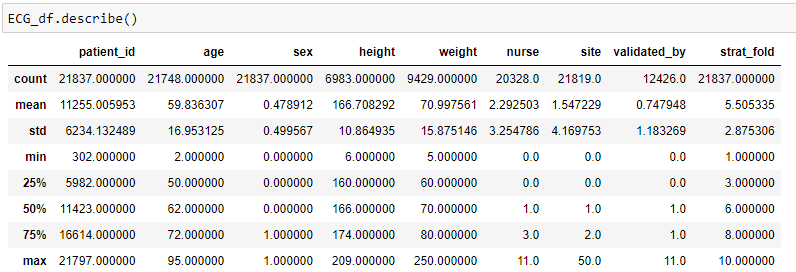
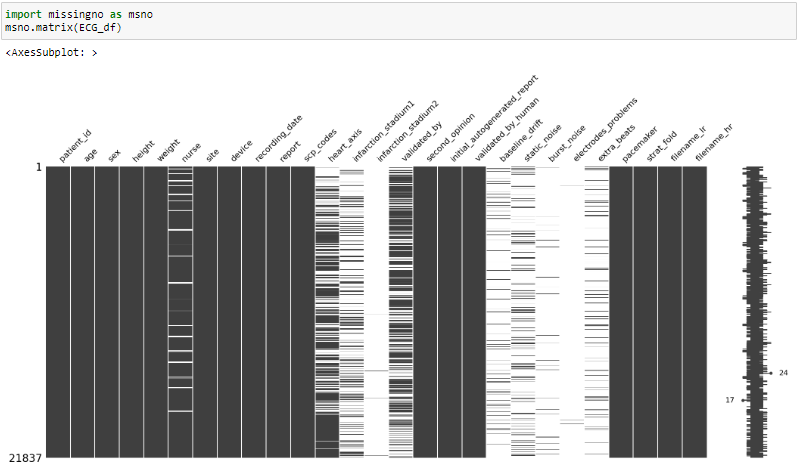


Рисунок 6. Пример кривой ЭКГ.

В остальном, датасет включает 21837 строк данных. Краткое описание выглядит следующим образом:



(а)



(б)

Рисунок 7. Статистика по переменным (а) – общая статистика, (б) – статистика пропусков данных.

По статистике из Рисунка 7 видно, что многие параметры имеют большое количество пропусков, и вряд ли имеют значительное влияние на здоровье пациента (напр. имя врача).

Однако сложность при работе с этим датасетом возникает при анализе пропусков. Видно, что по многим показателям пропусков значительное количество. Также многие показатели являются заполняемыми людьми, такие данные будем стараться исключить из итогового анализа.

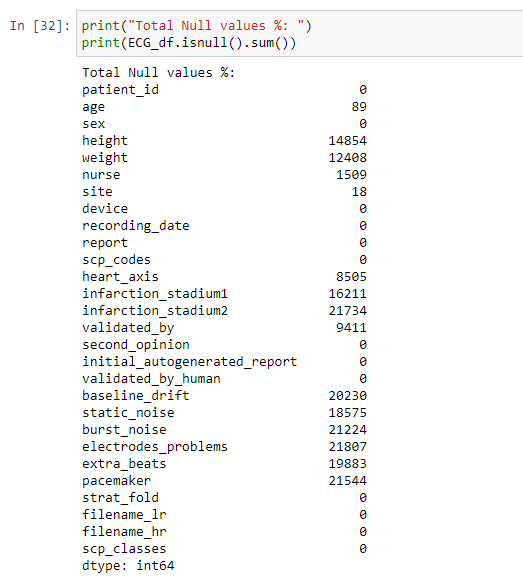


Рисунок 8. Статистика по пропускам значений.

Использовать из всех предлагаемых столбцов будем age – возраст, sex – пол, height- высота пациента, weight – вес пациента, infarction\_stadium1 – стадия инфаркта, infarction\_stadium2 – стадия инфаркта 2 (последние 2 пункта имеются только если у пациента ранее был зафиксирован инфаркт), pacemaker – наличие кардистимулятора.

5. Методика исследования и предсказания.

Для того чтобы оценить риск возникновения СЗЗ по результатам лабораторных исследований была решена задача бинарной классификации (0-1 здоров-не здоров). Т.к. мы имеем достаточно большой датасет, то мы имеем возможность использовать разделение на тестовую и тренировочную выборку, а также использовать метод разделения выборки на фолды для независимой оценки с разных сторон.

Предварительная обработка данных включала в себя LabelEncoder и выгрузка этого трансформера в файл, для дальнейшей работы с ним с новыми данными.

Пример кода применения энкодера и выгрузки его в отдельный файл для дальнейшего использования:

*# энкодер для использования в древовидными алгоритмами*

*# создаем отдельный фрейм данных для данного типа алгоритма*

df\_tree = df.copy()

*# определяем энкодер (стандартные настройки)*

encoder = LabelEncoder()

*# т.к. потребуется пройти по каждой колонке, сохраним их*

categorical\_features = df\_tree.columns.tolist()

for each in categorical\_features:

*# «обучаем» энкодер*

encoder.fit(df\_tree[each])

*# сохраняем энкодер для конкретного столбца в нужную папку*

joblib.dump(encoder, path\_in\_drive+f'{each}\_lblenc.pkl')

*# для дополнительной самопроверки – тут же загрузим его*

encoder = joblib.load(path\_in\_drive+f'{each}\_lblenc.pkl')

*# применим его для нужного столбца*

df\_tree[each] = encoder.transform(df[each])

*# выведем новый фрейм данных*

df\_tree.head()

Т.к. в анализе также были представлены не только древовидные алгоритмы, для них применялся one-hot encoding:

*# Создание one hot encoded столбцы для работы с алгоритмами*

df\_nontree=pd.get\_dummies(df,columns=string\_col,drop\_first=False)

*# Целевой столбец добавим в конце*

*target="HeartDisease"*

*# перевод в массив данных numpy*

y=df\_nontree[target].values

df\_nontree.drop("HeartDisease",axis=1,inplace=True)

df\_nontree=pd.concat([df\_nontree,df[target]],axis=1)

df\_nontree.head()

Таким образом получаем большое количество колонок необходимых для алгоритмов основанных на дереве решений

Рассмотрим пример кода для алгоритма не основанного на дереве решений:

*# импорт всех необходимых библиотек не загруженных ранее*

from sklearn import model\_selection

from sklearn.linear\_model import LogisticRegression

*# создание списка показателей точности модели*

acc\_mod=recll\_sc = []

recll\_sc = []

*# применим кросс-валидацию по 5 частям*

kf=model\_selection.StratifiedKFold(n\_splits=5)

for fold, (trn\_,val\_) in enumerate(kf.split(X=df\_nontree,y=y)):

*# разделим на тестовую, валидационную и тренировочную*

X\_train=df\_nontree.loc[trn\_,feature\_col\_nontree]

y\_train=df\_nontree.loc[trn\_,target]

X\_valid=df\_nontree.loc[val\_,feature\_col\_nontree]

y\_valid=df\_nontree.loc[val\_,target]

*# проведем масштабирование данных*

ro\_scaler=MinMaxScaler()

X\_train=ro\_scaler.fit\_transform(X\_train)

X\_valid=ro\_scaler.transform(X\_valid)

*# выбор метода (в данном случае логистическая регрессия)*

clf=LogisticRegression()

*# обучение метода*

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

*# предсказание на валидационной выборке и вывод результатов*

y\_pred=clf.predict(X\_valid)

print(f"The fold is : {fold} : ")

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

acc=roc\_auc\_score(y\_valid,y\_pred)

acc\_mod.append(acc)

recll\_sc.append(recall\_score(y\_valid,y\_pred))

print(f"The accuracy for Fold {fold+1} : {acc}")

pass

*# зафиксируем показатели алгоритма в таблицу для сравнения*

dict\_res\_al.loc["LogisticRegression", ['accuracy','recall']] =

[ np.array(acc\_mod).mean(),np.array(recll\_sc).mean() ]

Пример кода для метода CatBoost выглядит следующим образом:

*# применим кросс-валидацию по 5 частям*

splits = 5

skf = model\_selection.StratifiedKFold(n\_splits=splits, shuffle=True, random\_state=42)

*# делим на фичи и параметр*

X=df\_tree.loc[:,feature\_col\_tree]

y=df\_tree.loc[:,target]

oof\_preds = np.zeros((X.shape[0],))

preds = np.zeros((X.shape[0],len(np.unique(y))))

model\_fi = total\_mean\_acc = 0

*# создание списка показателей точности модели*

acc\_mod = recll\_sc = []

for num, (train\_idx, valid\_idx) in enumerate(skf.split(X, y)):

X\_train, X\_valid = X.loc[train\_idx], X.loc[valid\_idx]

y\_train, y\_valid = y.loc[train\_idx], y.loc[valid\_idx]

*# выберем модель (стандартные настройки)*

model = CatBoostClassifier()

*# обучим модель на наших данных*

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

verbose=False,

eval\_set=(X\_valid, y\_valid) )

*# предскажем дополнительно вероятность*

preds += model.predict\_proba(X) / splits

oof\_preds[valid\_idx] = model.predict(X\_valid).flatten()

*# получим метрики качества и выведем их*

fold\_acc = accuracy\_score(y\_valid, oof\_preds[valid\_idx])

fold\_rec = recall\_score(y\_valid, oof\_preds[valid\_idx])

print(f"Fold {num} accuracy: {fold\_acc}, recall: {fold\_rec}")

print(classification\_report(y\_valid,oof\_preds[valid\_idx]))

acc=roc\_auc\_score(y\_valid,oof\_preds[valid\_idx])

acc\_mod.append(acc)

total\_mean\_acc += fold\_acc / splits

recll\_sc.append(recall\_score(y\_valid,oof\_preds[valid\_idx]))

*# зафиксируем показатели алгоритма в таблицу для сравнения*

dict\_res\_al.loc["CatBoostClassifier", ['accuracy','recall']] =

[ np.array(acc\_mod).mean(),np.array(recll\_sc).mean() ]

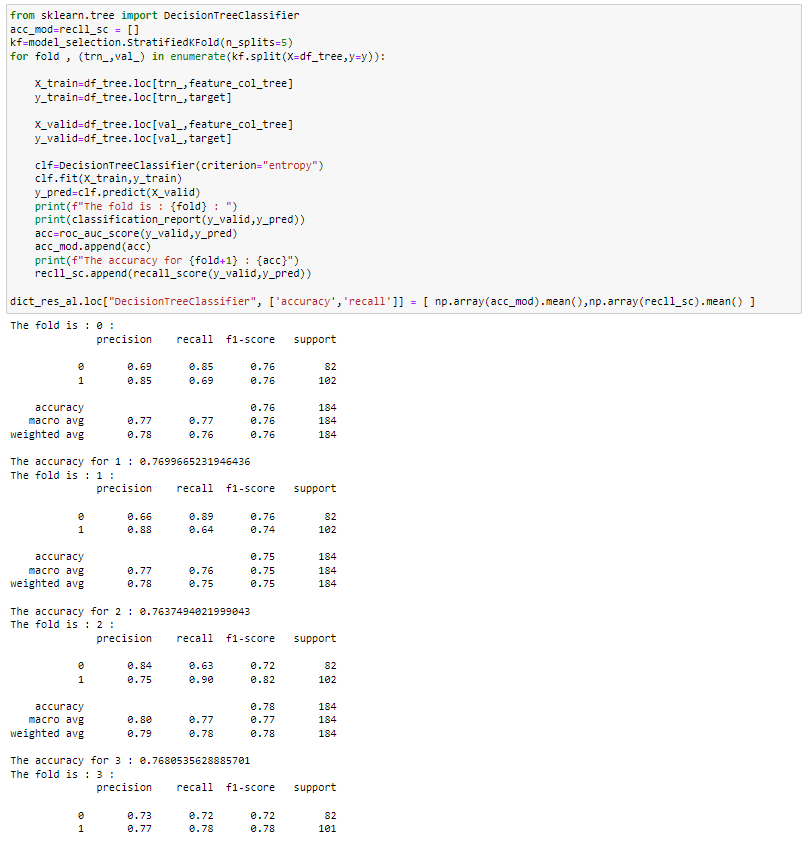


Рисунок 9. Пример кода применения метода классификации.

Если кратко описать методику данной части кода:

1. Разделение датасета на целевую колонку и фичи;
2. Разделение на фолды (части) всей выборки;
3. Применение метода CatBoost для каждой фичи в отдельности;
4. Осреднение показателей по каждому фолду.

Кратко опишем применяемые алгоритмы:

1. LogisticRegression - основная идея логистической регрессии заключается в том, что пространство исходных значений может быть разделено линейной границей (т.е. прямой) на две соответствующих классам области.

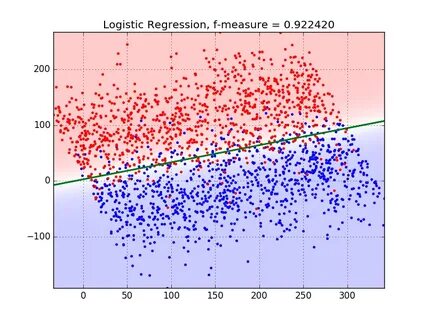


Рисунок 9. Визуализация основы работы метода логистической регрессии.

1. GaussianNB - наивные методы Байеса — это набор алгоритмов контролируемого обучения, основанных на применении теоремы Байеса с «наивным» предположением об условной независимости между каждой парой характеристик при заданном значении переменной класса. Теорема Байеса утверждает следующее отношение, учитывая переменную классаy и зависимый вектор
2. SVC (SVM) - это алгоритм машинного обучения, используемый для решения задач классификации, который строит гиперплоскость в n-мерном пространстве для разделения объектов двух или более классов. Гиперплоскость выбирается таким образом, чтобы максимизировать расстояние между гиперплоскостью и ближайшими объектами разных классов (зазор). Объекты, которые расположены ближе всего к гиперплоскости, называются опорными векторами.

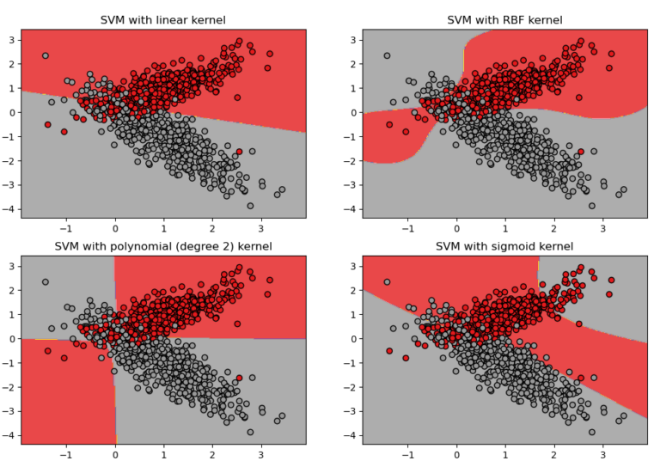


Рисунок 10. Визуализация метода опорных векторов при разных поверхностях.

1. KNeighborsClassifier - Алгоритм k ближайших соседей основан на принципе близости объектов в пространстве признаков. Он состоит в следующем: для каждого объекта из тестовой выборки находим k ближайших соседей из обучающей выборки, и классифицируем объект на основе классов его соседей. Класс, который наиболее часто встречается среди соседей, и будет классом, к которому относится исходный объект.

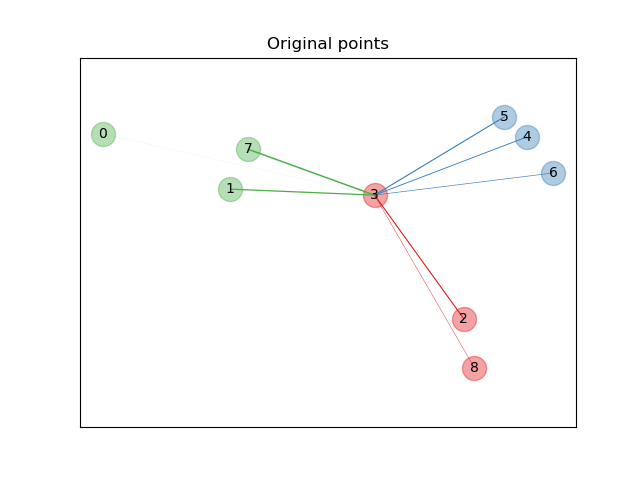


Рисунок 11. Визуализация метода ближайших соседей.

1. DecisionTreeClassifier - это непараметрический контролируемый метод обучения, используемый для классификации и регрессии . Цель состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной, изучая простые правила принятия решений, выведенные из характеристик данных. Дерево можно рассматривать как кусочно-постоянное приближение.

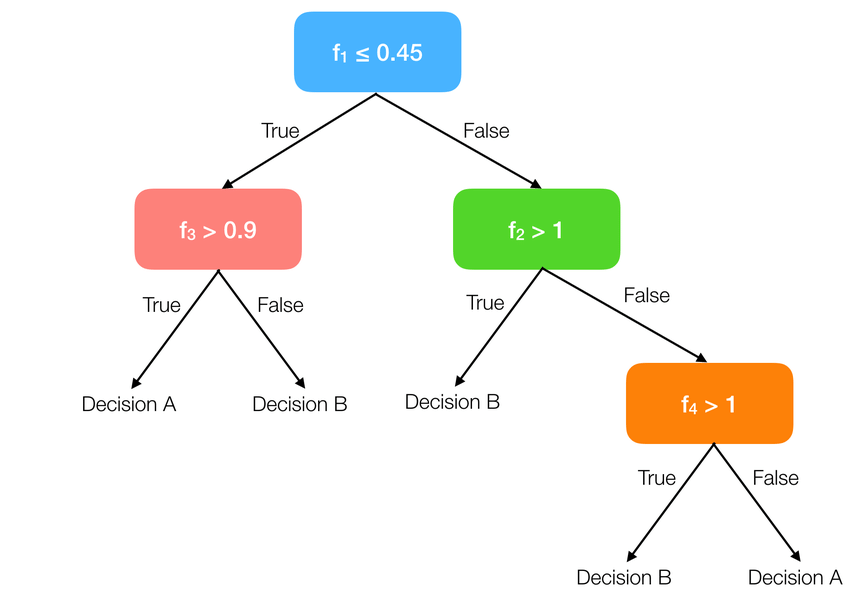


Рисунок 12. Визуализация метода ближайших соседей.

1. RandomForestClassifier - это метод (основанный на подходе «разделяй и властвуй»), использующий ансамбль деревьев решений, созданных на случайно разделенном датасете. Набор таких деревьев-классификаторов образует лес. Каждое отдельное дерево решений генерируется с использованием метрик отбора показателей, таких как критерий прироста информации, отношение прироста и индекс Джини для каждого признака.

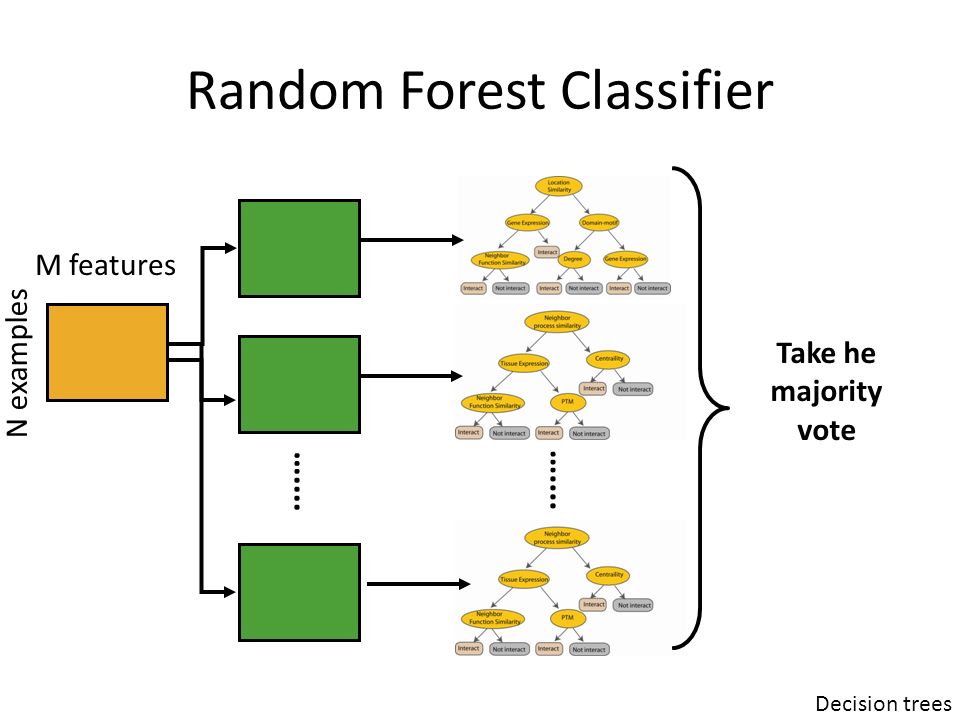


Рисунок 13. Визуализация метода ближайших соседей.

1. XGBClassifier - это усовершенствованная и настроенная версия системы дерева решений с градиентным повышением, созданная с учетом производительности и скорости. XGBoost на самом деле означает “Экстремальный градиентный бустинг”, и это относится к тому факту, что алгоритмы и методы были настроены так, чтобы раздвинуть предел возможного для алгоритмов градиентного бустинга.

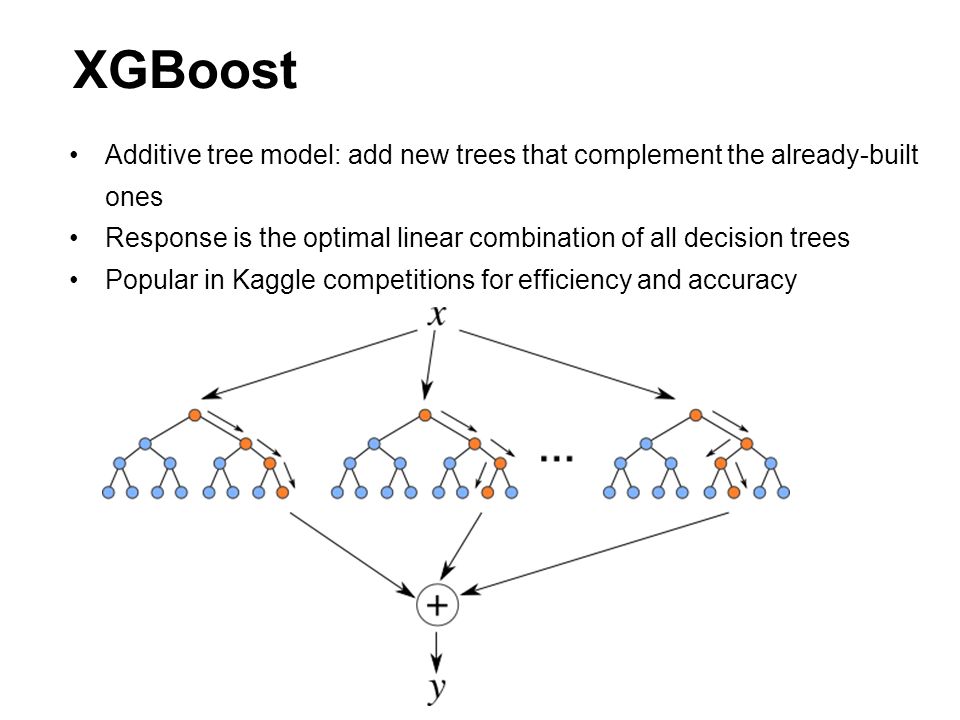


Рисунок 14. Визуализация метода ближайших соседей.

1. CatBoostClassifier - библиотека, которая была разработана Яндексом в 2017 году, представляет разновидность семейства алгоритмов Boosting и является усовершенствованной реализацией Gradient Boosting Decision Trees (GBDT).

Таблица 1. Пример показателей точности по 4 фолду

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **0** | 0.93 | 0.76 | 0.83 | 82 |
| **1** | 0.83 | 0.95 | 0.88 | 101 |
| **accuracy** |  |  | 0.86 | 183 |
| **macro avg** | 0.88 | 0.85 | 0.86 | 183 |
| **weighted avg** | 0.87 | 0.86 | 0.86 | 183 |

Вручную требуется оценить насколько сильно различаются метрики качества в каждом фолде. Если разница некритичная – методика настроена верно.

Таким образом были оценены следующие методы, которые дали следующие результаты:

Таблица 2. Пример показателей точности по 4 фолду

|  |  |  |
| --- | --- | --- |
| **Методика** | **accuracy** | **recall** |
| **LogisticRegression** | 0.829 | 0.850 |
| **GaussianNB** | 0.843 | 0.843 |
| **SVC(kernel=linear)** | 0.832 | 0.832 |
| **SVC(kernel=sigmoid)** | 0.793 | 0.793 |
| **SVC(kernel=rbf)** | 0.842 | 0.842 |
| **SVC(kernel=poly)** | 0.832 | 0.832 |
| **KNeighborsClassifier** | 0.855 | 0.855 |
| **DecisionTreeClassifier** | 0.728 | 0.728 |
| **RandomForestClassifier** | 0.846 | 0.846 |
| **XGBClassifier** | 0.825 | 0.825 |
| **CatBoostClassifier** | 0.895 | 0.895 |

Таким образом выбираем метод CatBoostClassifier, который значительно дает результаты, относительно остальных методик.

Выгрузим данную модель в папку, для того чтобы использовать его в дальнейшем при работе с интерфейсом. В отличие от обычной регрессии, в методе логистической регрессии не производится предсказание значения числовой переменной исходя из выборки исходных значений. Вместо этого, значением функции является вероятность того, что данное исходное значение принадлежит к определенному классу.

model.save\_model(path\_in\_drive+"/cb\_model\_pretrained.cbm",

format="cbm",

export\_parameters=None,

pool=None)

Работа с данными ЭКГ включала в себя 2 подэтапа. Первый – работа с статичными параметрами (возраст, пол и тд.) и второй – работа с рядами – ЭКГ. Решив первую задачу – сформировали базовый прогноз, с которым сравниваем точность модели при подключении данных с ЭКГ. Нейронная сеть решает задачу небинарной классификации, разделяя целевую колонку на 5 суперклассов (нормальная работа сердца, риск инфаркта миокарда, нарушенный ритм, гипертрофия, нарушенная отношение амплитуды работы сердца).

Этот код создает две модели нейронных сетей: model01 и model02.

Функция create\_X\_model определяет архитектуру модели create\_X\_model, которая обрабатывает данные X (статичные параметры). Эта модель состоит из нескольких слоев:

1. Полносвязный слой с 64 нейронами и функцией активации ReLU;
2. Слой Dropout с коэффициентом отсечения 0.3;
3. Полносвязный слой с 64 нейронами и функцией активации ReLU;
4. Слой Dropout с коэффициентом отсечения 0.5.

Модель model01 принимает на вход данные X (функцию create\_X\_model) и выдает на выходе данные Z, которые являются предсказаниями модели для задачи классификации. Модель model01 состоит из нескольких слоев:

1. Входной слой X\_inputs, который принимает на вход данные X;
2. Слой create\_X\_model, который обрабатывает данные X;
3. Полносвязный слой с 64 нейронами и функцией активации ReLU;
4. Полносвязный слой с 64 нейронами и функцией активации ReLU;
5. Слой Dropout с коэффициентом отсечения 0.5;
6. Выходной слой Z\_outputs с функцией активации sigmoid, который выдает предсказания модели для задачи классификации.

Функция create\_Y\_model определяет архитектуру модели create\_Y\_model, которая обрабатывает данные Y. Эта модель состоит из нескольких слоев:

1. Слой Conv1D с фильтрами f1, размером ядра k1 и шагом s1, который применяет свертку к данным Y;
2. Слой BatchNormalization, который нормализует данные после свертки;
3. Слой ReLU, который применяет нелинейную функцию активации к данным после нормализации;
4. Слой MaxPool1D, который уменьшает размерность данных Y;
5. Слой Conv1D с фильтрами f2, размером ядра k2 и шагом s2, который применяет свертку к данным Y;
6. Слой BatchNormalization, который нормализует данные после свертки;
7. Слой ReLU, который применяет нелинейную функцию активации к данным после нормализации;
8. Слой MaxPool1D, который уменьшает размерность данных Y;
9. Слой Conv1D с фильтрами f3, размером ядра k3 и шагом s3, который применяет свертку к данным Y;
10. Слой BatchNormalization, который нормализует данные после свертки;
11. Слой ReLU, который применяет нелинейную функцию активации к данным после нормализации;
12. Слой GlobalAveragePooling1D, который усредняет данные по временной оси;
13. Слой Dropout с коэффициентом отсечения 0.5.

Модель model02 принимает на вход два входных тензора X и Y (а также функции create\_X\_model и create\_Y\_model) и выдает на выходе данные Z, которые являются предсказаниями модели для задачи классификации. Модель model02 состоит из нескольких слоев:

1. Входные слои X\_inputs и Y\_inputs, которые принимают на вход данные X и Y соответственно;

2. Слой Concatenate, который объединяет выходы из двух моделей: create\_X\_model для данных X и create\_Y\_model для данных Y;

3. Полносвязный слой с 64 нейронами и функцией активации ReLU;

4. Полносвязный слой с 64 нейронами и функцией активации ReLU;

5. Слой Dropout с коэффициентом отсечения 0.5;

6. Выходной слой Z\_outputs с функцией активации sigmoid, который выдает предсказания модели для задачи классификации.

Обе нейронные сети настроены на остановку расчета при достижении прироста точность в соседних эпохах. Полная архитектура полученного алгоритма представлена в Приложении 4.

Свёрточная нейронная сеть (ConvNet/CNN) — это алгоритм глубокого обучения, который может принимать входное изображение, присваивать важность (изучаемые веса и смещения) аспектам или объектам изображении и отличать одно от другого. Архитектура CNN аналогична структуре связей нейронов в мозгу человека, учёные черпали вдохновение в организации зрительной коры головного мозга. Отдельные нейроны реагируют на стимулы только в некоторой области поля зрения, также известного как перцептивное поле. Множество перцептивных полей перекрывается, полностью покрывая поле зрения CNN.

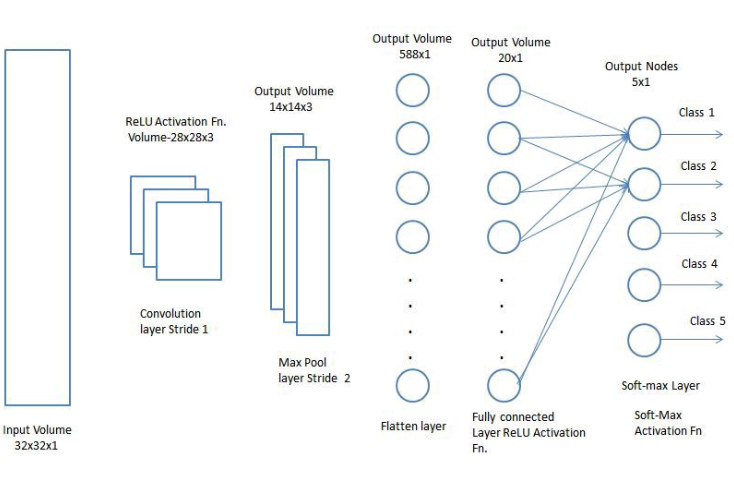


Рисунок 15. Визуализация метода ближайших соседей.

По результату расчета получаем следующие показатели.

Таблица 3. Пример показателей точности.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **NORM** | 0.83 | 0.9 | 0.86 | 964 |
| **MI** | 0.81 | 0.75 | 0.78 | 553 |
| **STTC** | 0.77 | 0.75 | 0.76 | 523 |
| **CD** | 0.88 | 0.6 | 0.71 | 498 |
| **HYP** | 0.72 | 0.44 | 0.55 | 263 |
| **micro avg** | 0.81 | 0.75 | 0.78 | 2801 |
| **macro avg** | 0.8 | 0.69 | 0.73 | 2801 |
| **weighted avg** | 0.81 | 0.75 | 0.77 | 2801 |
| **samples avg** | 0.79 | 0.76 | 0.76 | 2801 |

Лучшую модель выгружаем в файл для дальнейшего использования в пользовательском интерфейсе.

6. Выбор инструментов для разработки

Для создание пользовательского интерфейса воспользовались Dash plotly. Пример кода объекта и ответной команды приведены ниже.

app.layout = html.Div([

html.H4('Часть 1. Проверим показатели сердца по параметрам ЭКГ'),

html.Div([

"Возраст пациента [годы]",

dcc.Dropdown(

id="dropdown\_age",

options=np.array(list(range(10,100))),

value=np.array(list(range(10,100)))[30],

clearable=False,

searchable=True,

placeholder="Введите возраст")]),

…

@app.callback(

Output('tab\_1', 'data'),

[Input('button\_calc\_1', 'n\_clicks')])

def update\_graph(n\_clicks):

if n\_clicks==None:

df\_res\_cl = pd.DataFrame()

df\_res\_cl.loc[0, ['есть риск', 'нет риска']] = ['-', '-']

return df\_res\_cl.to\_dict('records')

else:

df\_res\_cl = pd.DataFrame()

df\_res\_cl.loc[0, ['есть риск', 'нет риска']] = [1, 0]

df\_check\_1 = df\_cl.rename(columns={'dropdown\_age\_1':'Age', 'dropdown\_Sex\_1':'Sex', 'dropdown\_ChestPainType\_1':'ChestPainType', 'dropdown\_RestingBP\_1':'RestingBP', dropdown\_Cholesterol\_1':'Cholesterol', 'dropdown\_FastingBS\_1':'FastingBS', 'dropdown\_MaxHR\_1':'MaxHR', 'dropdown\_ExerciseAngina\_1':'ExerciseAngina', 'dropdown\_Oldpeak\_1':'Oldpeak', dropdown\_RestingECG\_1':'RestingECG', 'dropdown\_ST\_Slope\_1':'ST\_Slope'})

df\_res\_cl = predct\_cb(df\_check\_1, path\_in\_drive, saved\_model\_cl)

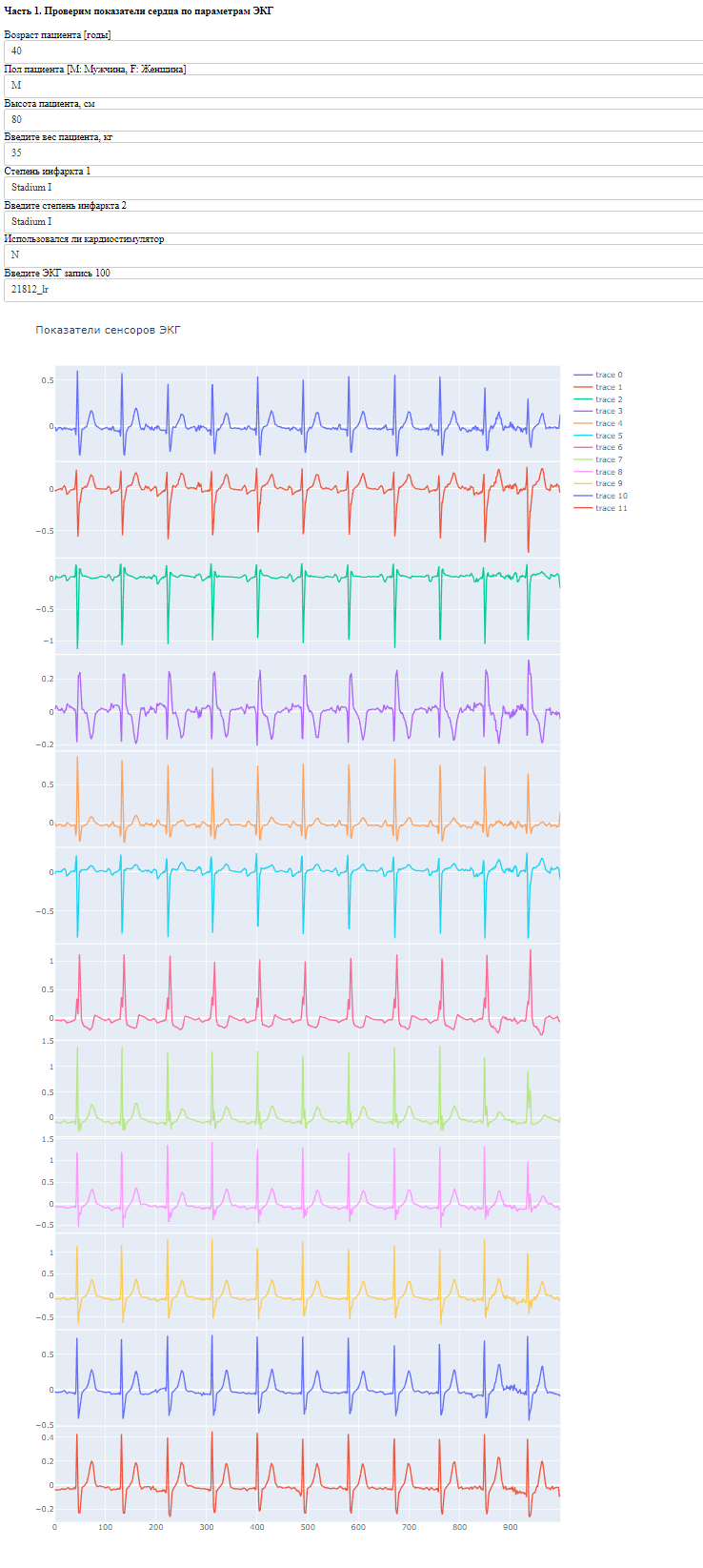
df\_res\_cl = df\_res\_cl\*100

df\_res\_cl = df\_res\_cl.round(2)

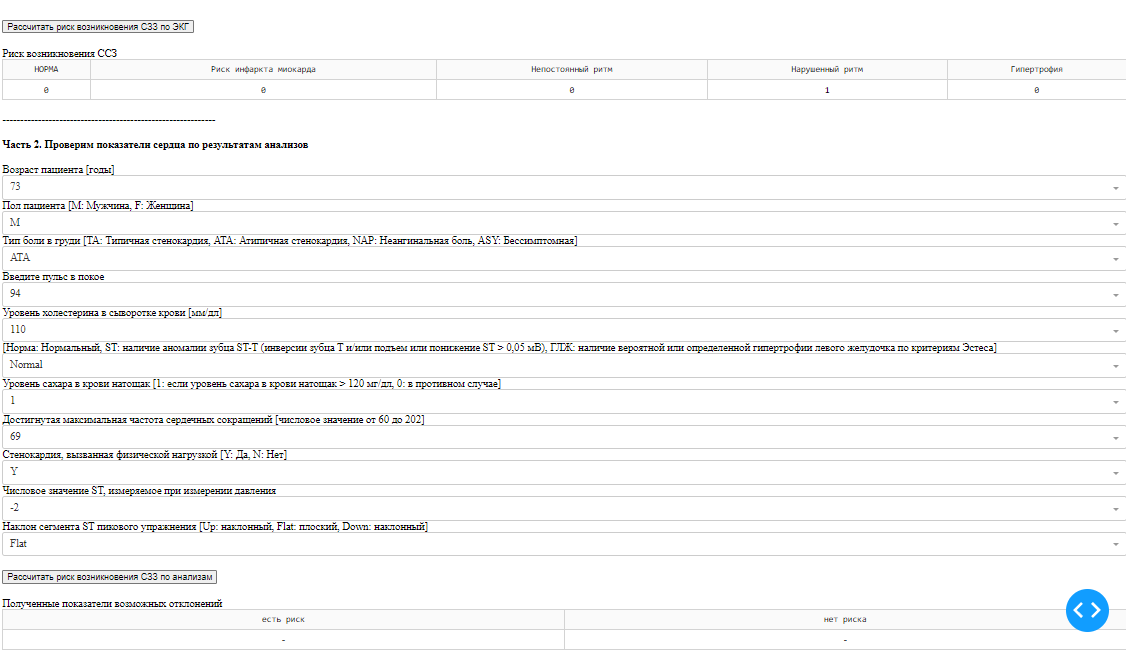
df\_res\_cl = df\_res\_cl.astype(str) + "%"

return df\_res\_cl.to\_dict('records')

Скриншот интерфейса выглядит следующим образом.



(а)



(б)

Рисунок 15. Пример интерфейса по 1 и 2 части работы (а) и (б).

Пример ответных команд (callbacks), применяемых в интерфейсе

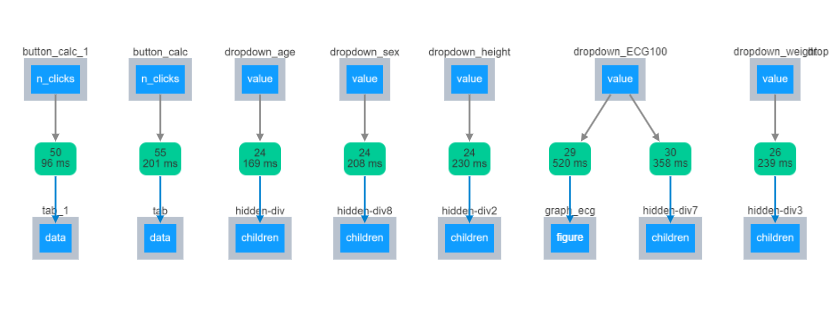


Рисунок 16. Пример структуры ответных команд.

Структура кода интерфейса состоит из html объектов: 8 разворачивающихся списков (dropdown), 1 кнопки (button), 1 таблицы (dashtable) с результатами и 1 графика (graph), вторая часть состоит из 11 разворачивающихся списков, 1 кнопки и 1 таблицы с результатами.

По нажатию кнопки вызывается команда, которая собирает данные из разворачивающихся списков и предсказывает вероятность СЗЗ.

Таким образом, пользователю следует:

1. Загрузить кривые в нужном формате в папку с примерами;
2. Обновить страницу сервера;
3. Ввести исходную информацию с помощью выпадающих списков;
4. Нажать кнопку расчета риска возникновения СЗЗ.

После этого интерфейс выведет показатели.

Возможные ошибки в расчете могут быть связаны с некачественным форматом предоставления ЭКГ, неточным вводом данных в выпадающие списки, сложности перевода показателей на русский язык (часть параметров не удалось среди русскоязычных источников).

7. Технология основного приложения

Для работы использовался python 3.7

Библиотеки и версии:

* wfdb 4.1.2
* numpy 1.23.4
* pandas 1.5.1
* seaborn 0.12.2
* joblib 1.3.1
* catboost 1.2.2
* sklearn 1.3.0
* plotly 5.11.0
* tensorflow 2.10.0

Процессор Intel(R) Xeon(R) CPU E5-2687W v3 @ 3.10GHz.Однако дополнительно при написании использовались мощности предоставляемые google collab.

Расчет первой части занимает около 3 минут, расчет второй части занимает около 45 минут на ЦП, пользовательский интерфейс загружается за несколько секунд.

Таким образом можно сделать вывод, что методика имеет перспективу к усложнению и уточнению, т.к. мощности могут позволить.

8. Выводы.

В дальнейшем можно написать приложение, которое не будет зависеть от сервера, для того чтобы использовать его без сети.

В целом система стабильна, т.к. интерфейс сведен к минимуму и ошибок быть не должно, т.к. данные пользователь практически не вводит самостоятельно.

Далее планируется улучшать алгоритм, хотя и текущие показатели удовлетворительные.

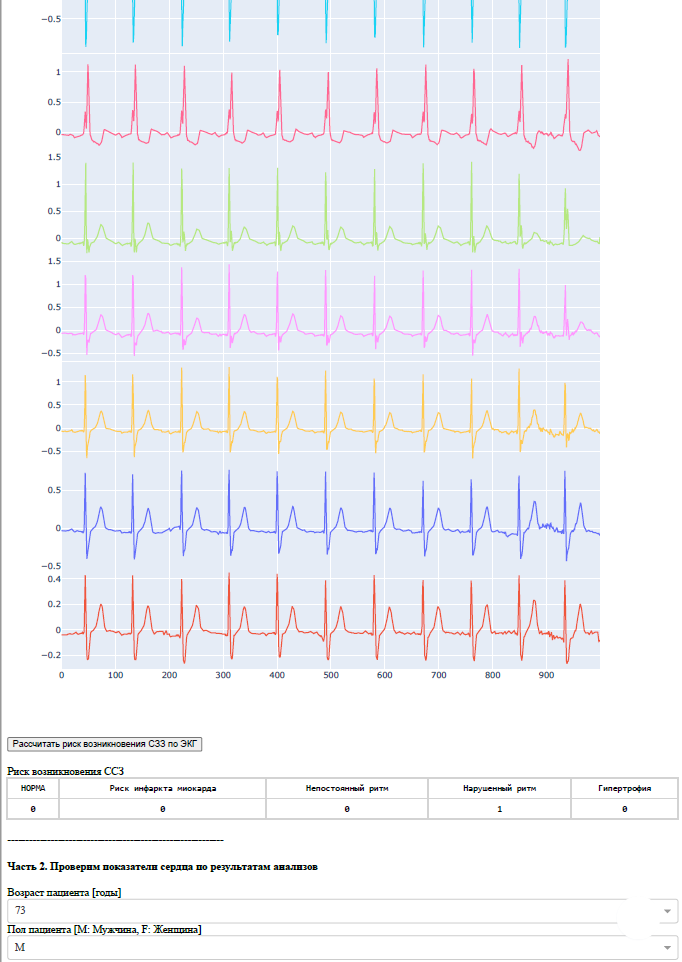
Использованная литература

1. Баевский Р.М. Оценка адаптационных возможностей организма и риск развития заболеваний / Р.М. Баевский, А.П. Берсенева. – М.: Медицина, 1997. – 265 c.
2. Елисеев О.П. Практикум по психологии личности / О.П. Елисеев. – СПб.: Питер, 2001.– 560 с. (Практикум по психологии).
3. Рогов Е.И. Настольная книга практического психолога/ Рогов Е.И. –М.: Владос, 2004. –384 с.
4. Яхин К.К. Клинический опросник для выявления и оценки невротических состояний Яхин К.К., Менделевич Д.М. // Клиническая и медицинская психология: Учебное пособие / ред. Менделевич В.Д. – М.: МЕДпресс-информ, 2005. – С. 375-423.
5. Цекин В.П. Типы отношения к болезни у больных с сердечно-сосудистыми заболеваниями. Поиск закономерностей развития / Цекин В.П. // Независимый психиатрический журнал. – 2004. – №2. – С. 80-82.
6. Елисеев О.П. Конструктивная типология и психодиагностика личности / Елисеев О.П.; под ред. Панферова В.Н. – Псков: Изд-во Псковского обл. Ин-та совершенствования учителей, 1994. – 280 с.
7. Бурлачук Л.Ф. Акцентуации личности: что диагностируем? / Л.Ф. Бурлачук, В.Н. Духневич // Вопросы психологии. – 1998. – № 2. – C. 136-144.
8. <https://www.kaggle.com/datasets/khyeh0719/ptb-xl-dataset>
9. <https://www.kaggle.com/code/sandipdevre/heart-disease-prediction-dataset>

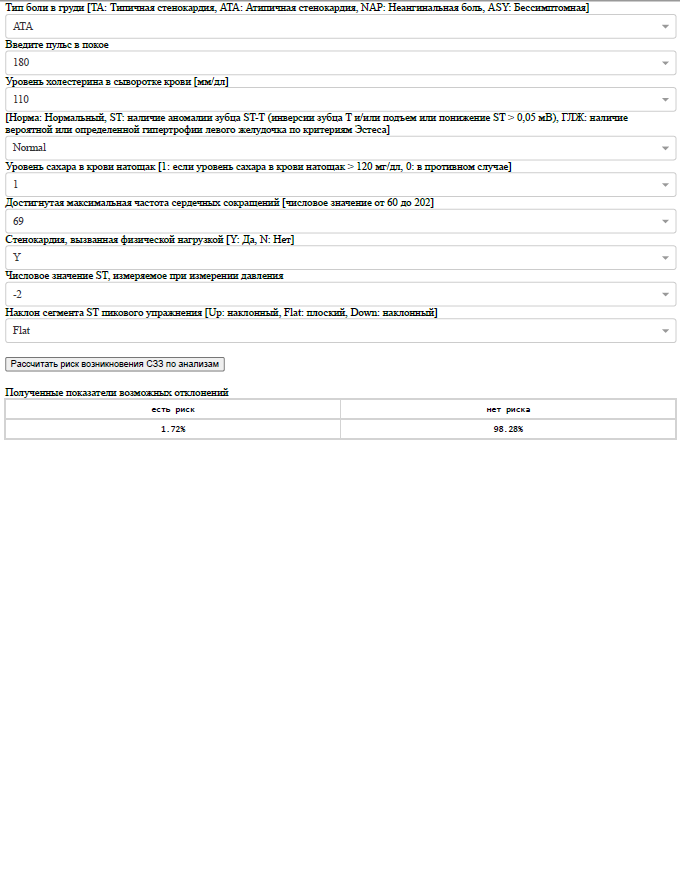
Приложение 1. Пример пользовательского интерфейса, часть 1.



Приложение 2. Пример пользовательского интерфейса, часть 2.



Приложение 3. Пример пользовательского интерфейса, часть 3.



Приложение 4. Архитектура модели сверточной нейронной сети.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

=================================================================================================

Y\_inputs (InputLayer) [(None, 1000, 12)] 0 []

Y\_conv\_1 (Conv1D) (None, 1000, 64) 5440 ['Y\_inputs[0][0]']

Y\_norm\_1 (BatchNormalization) (None, 1000, 64) 256 ['Y\_conv\_1[0][0]']

Y\_relu\_1 (ReLU) (None, 1000, 64) 0 ['Y\_norm\_1[0][0]']

Y\_pool\_1 (MaxPooling1D) (None, 500, 64) 0 ['Y\_relu\_1[0][0]']

Y\_conv\_2 (Conv1D) (None, 500, 128) 24704 ['Y\_pool\_1[0][0]']

Y\_norm\_2 (BatchNormalization) (None, 500, 128) 512 ['Y\_conv\_2[0][0]']

Y\_relu\_2 (ReLU) (None, 500, 128) 0 ['Y\_norm\_2[0][0]']

Y\_pool\_2 (MaxPooling1D) (None, 250, 128) 0 ['Y\_relu\_2[0][0]']

X\_inputs (InputLayer) [(None, 7)] 0 []

Y\_conv\_3 (Conv1D) (None, 250, 256) 98560 ['Y\_pool\_2[0][0]']

X\_dense\_1 (Dense) (None, 32) 256 ['X\_inputs[0][0]']

Y\_norm\_3 (BatchNormalization) (None, 250, 256) 1024 ['Y\_conv\_3[0][0]']

X\_drop\_1 (Dropout) (None, 32) 0 ['X\_dense\_1[0][0]']

Y\_relu\_3 (ReLU) (None, 250, 256) 0 ['Y\_norm\_3[0][0]']

X\_dense\_2 (Dense) (None, 32) 1056 ['X\_drop\_1[0][0]']

Y\_aver (GlobalAveragePooling1D (None, 256) 0 ['Y\_relu\_3[0][0]']

)

X\_drop\_2 (Dropout) (None, 32) 0 ['X\_dense\_2[0][0]']

Y\_drop (Dropout) (None, 256) 0 ['Y\_aver[0][0]']

Z\_concat (Concatenate) (None, 288) 0 ['X\_drop\_2[0][0]',

'Y\_drop[0][0]']

Z\_dense\_1 (Dense) (None, 64) 18496 ['Z\_concat[0][0]']

Z\_dense\_2 (Dense) (None, 64) 4160 ['Z\_dense\_1[0][0]']

Z\_drop\_1 (Dropout) (None, 64) 0 ['Z\_dense\_2[0][0]']

Z\_outputs (Dense) (None, 5) 325 ['Z\_drop\_1[0][0]']

================================================================================================

Total params: 154,789

Trainable params: 153,893

Non-trainable params: 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_