



ПОЛИТЕХ
Санкт-Петербургский
политехнический университет
Петра Великого

Высшая инженерная школа

ТЕМА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ:

**РАЗРАБОТКА СИСТЕМЫ ПОДДЕРЖКИ
ПРИНЯТИЯ РЕШЕНИЙ (СППР) В ОБЛАСТИ
ФИЗИЧЕСКОЙ РЕАБИЛИТАЦИИ И
МЕДИЦИНЫ**

по программе профессиональной переподготовки:

«ПП. Анализ данных на языке Python»

Выполнила: Коханова Наталия Георгиевна

Руководитель: Заграновская Анна Васильевна

Обзор предметной области

Дипломная работа выполняется на базе данных сайта

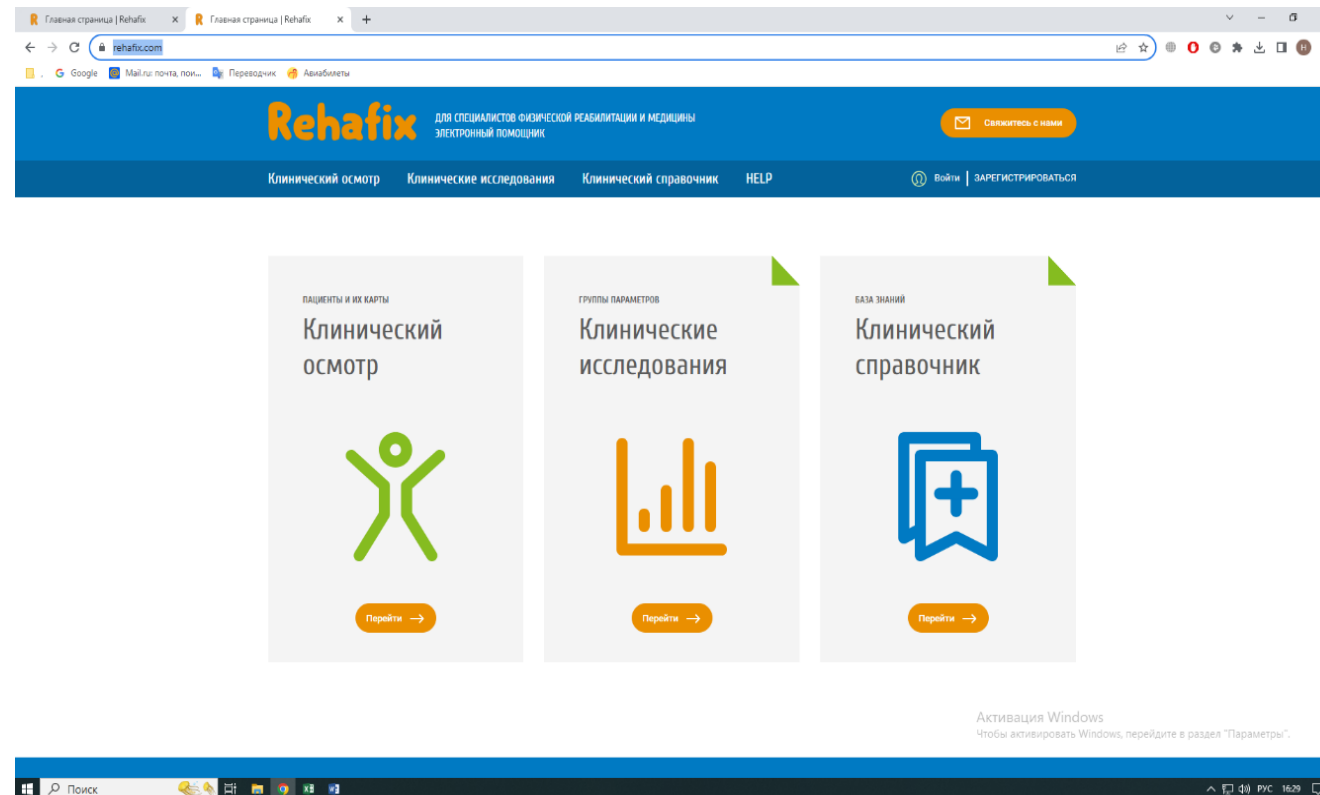
<https://www.rehafix.com>

Сайт направлен на применение компьютерных технологий
в области травматологии и ортопедии,
а именно восстановления функции организма после травм и болезней костей, суставов.

Сайт разрабатывается для специалистов.

В России на сегодняшний день аналогов нет.

СУБД – MySQL.



Обзор предметной области

В некоторых странах созданы СППР, которые уже сейчас помогают врачам и пациентам выбрать оптимальные упражнения, оборудование и инструменты для восстановления после травмы или операции на основе их клинических данных и индивидуальных характеристик.

Основные из них:

1. "PhysioAdvisor"

Сайт: <https://physioadvisor.com.au/>

2. "Rehabilitation Robotics Lab"

Сайт: <https://www.med.upenn.edu/rehabilitation-robotics-lab/>

3. "PhysioTools "

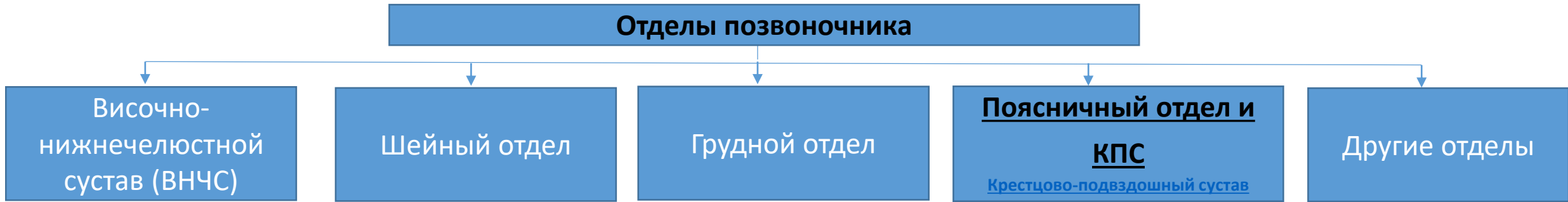
Сайт: <https://www.physiotools.com/>

Обзор предметной области

Разделы сайта <https://www.rehafix.com/>

- 1) Клинический осмотр (Пациенты и их карты) – закрытая информация, только для Администратора
- 2) Клинические исследования – частично закрытая информация, доступна для зарегистрированных специалистов
- 3) Клинический справочник - открытая информация

Структура раздела «Клинический осмотр»



Тестирование пациента (более 1000 параметров)



Заключение (результативные признаки)



Цель:

Разработать **программный прототип** для системы рекомендаций физической терапии в области заболеваний поясничного отдела с помощью анализа данных на языке Python.

Задачи:

- 1) подготовить данные (очистить , преобразовать),
- 2) сгруппировать данные,
- 3) проанализировать данные о заболеваниях поясничного отдела (по категориям, параметрам),
- 4) выбрать модель классификации.

Проблемы на текущий момент:

1. Сайт находится в разработке, тестируется.
2. База данных не заполнена, заполнение в части поясничного отдела планируется в марте 2024 года.
3. Проблема с выгрузкой данных.
 - При выгрузке в файл export отсутствуют наименования столбцов.
 - Функционал выгрузки в дальнейшем будет переделан в рамках следующих доработок в марте 2024 .
4. Слишком большое количество параметров (более 1 000).
 - Для дальнейшего анализа будут отбираться наиболее значимые (не более 20 параметров).
 - Планируется использование метода рекурсивного исключения признаков (RFE) .
 - Незначительное количество наблюдений – 18 пациентов.
5. Для корректной классификации данных необходимы честные ответы пациентов, реально отражающие самочувствие.
6. Прототип будет дорабатываться, учитывая экспертное мнение специалистов.

Подготовка данных.

Добавление наименований столбцов и удаление конфиденциальной информации.

1) Т.к. содержание базы данных в процессе формирования может меняться, наименования столбцов загружаем из отдельного файла.

```
# Загрузка таблицы Excel с наименованиями столбцов
df_names =
pd.read_excel('/content/gdrive/MyDrive/Colab_Data/Наименования_столбцов.xls
x', header=None)
df_names
# Присваиваем заголовки столбцам таблицы df
df.columns = df_names.iloc[0]
df
```

2) Удаляем столбцы с личными данными и преобразуем значения NaN в нулевые

```
df_new = df.drop(columns=['Email', 'Телефон']) #удалем столбцы с личными
данными
df_new.fillna(0, inplace=True) # Преобразуем значения NaN в нулевые
df_new.head(10)
```

Предварительная подготовка данных.

```
# Выводим общую информацию о наборе данных
df_new.info()

# Проверяем есть ли пропущенные значения
df_new.isnull().sum()

# Выводим описательные статистики
df_new.describe()
```

Добавление результирующего признака

```
# Т.к. база данных выгружается некорректно и не заполнена в целях тестирования кода  
добавляем последний столбец с Упражнениями  
# случайно присваиваем значения от 0 до 4. Номер последнего столбца -1217
```

```
last_column = np.random.randint(0, 4, size=len(df_new))  
df_new.insert(1217, 'Упражнения', last_column)  
df_new
```

Отбор и выделение признаков

Применяем метод RFE для отбора факторных признаков в задачах классификации

Создаем объект `rfe` с помощью функции `RFE()` из модуля `sklearn.feature_selection`, предназначенный для отбора заданного в параметре `n_features_to_select` количества признаков на основе заданной в параметре `estimator` модели. Пусть будет отобрано 20 признаков на основе дерева классификации.

```
rfe = RFE(estimator=DecisionTreeClassifier(), n_features_to_select=20)
```

Задаем модель классификации, в которую поступят отобранные признаки- дерево классификации, которое задается с помощью функции `DecisionTreeClassifier()` из модуля `sklearn.tree`. Результат записываем в переменную

```
model.model = DecisionTreeClassifier()
```

Проводим кодирование факторных признаков обучающего и тестового набора на основе функции `OrdinalEncoder()`

```
ordinal_encoder = OrdinalEncoder(handle_unknown='use_encoded_value',  
unknown_value=-1)  
ordinal_encoder.fit(X_train)  
X_train = ordinal_encoder.transform(X_train)  
X_test = ordinal_encoder.transform(X_test)
```

С существующими данными проводим группировку по признакам.

Определяем количество мужчин и женщин.

```
# Определяем количество мужчин и женщин
grouped = df_new.groupby('Пол') # группируем
данные по столбцу "Пол"
# подсчитываем количество значений и определяем
долю мужчин и женщин
count = grouped.size()
male_percent = count['Мужской'] / count.sum() *
100
female_percent = count['Женский'] / count.sum() *
100
# строим круговую диаграмму с долями
plt.pie([male_percent, female_percent],
labels=['Мужчины', 'Женщины'], autopct='%1.1f%%',
startangle=90)
plt.title('Доля мужчин и женщин')
plt.show()
```



В целях анализа определяем возрастные группы пациентов по ВОЗ.

```
# Определяем долю по возрастным группам
grouped = df_new.groupby('Возрастная группа') #
группируем данные по столбцу "Возрастная группа"
# подсчитываем количество значений и определяем долю
count = grouped.size()
children_percent = count['<19'] / count.sum() * 100
middle_percent = count['20-59'] / count.sum() * 100
senior_percent = count['60-79'] / count.sum() * 100
elderly_percent = count['80+'] / count.sum() * 100
# строим круговую диаграмму с долями
plt.pie([children_percent, middle_percent,
senior_percent, elderly_percent ], labels=['<19', '20-
59', '60-79', '80+'], autopct='%1.1f%%', startangle=90)
plt.title('Доля по возрастным группам')
plt.show()
```



Генерация выборки для машинного обучения

Т.к. база данных не готова, генерируем данные, аналогичные нашим параметрам для дальнейшего тестирования и моделирования

Генерируем выборку для задачи многоклассовой классификации, воспользовавшись функцией `make_classification()` из модуля `sklearn.datasets`.
Объем выборки 100 наблюдений, количество признаков - 30, из них информативных – 20, классов – 10.
Результат записываем в переменные X, y.

```
X, y = make_classification(n_samples= 100, n_features=30,  
n_informative=20, n_classes=10)
```

Выбор модели машинного обучения для классификации данных

Обучаем и выбираем подходящую модель машинного обучения для классификации данных. .

Метод	Средняя доля правильных ответов mean()	Стандартное отклонение полученных оценок std()
Логистическая регрессия	0.32	0.05
Метод опорных векторов (SVM). Метод 'linear'. Линейный	0.38	0.07
Метод опорных векторов (SVM). Метод 'rbf'. Нелинейный	0.53	0.09
Линейный дискриминантный анализ	0.97	0.04
Случайный лес Random Forest	0.48	0.07
Градиентный бустинг (Gradient Boosting Machines)	0.38	0.05
Модель решающего дерева Bagged Decision Trees	0.48	0.06

Из представленных моделей машинного обучения для классификации данных лучше всего себя показала модель **линейного дискриминантного анализа**, имеющая наивысшее значение средней доли правильных ответов (0.97) и наименьшее стандартное отклонение полученных оценок (0.04).

При окончательном выборе модели буду учитывать специфику задачи и особенности данных, на

Выбор модели машинного обучения для классификации данных

Используя

```
from sklearn.metrics import classification_report
```

Получаем отчеты о качестве работы каждой модели и выбираем наиболее подходящий вариант, основываясь на метриках точности, полноты и F-меры для каждого класса.

Спасибо за внимание!