УПРАВЛЕНИЕ ОБРАЗОВАНИЯ ГОРОДА ПЕНЗЫ

МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«ГИМНАЗИЯ № 53» г. ПЕНЗЫ

(МБОУ «Гимназия № 53» г. Пензы)

ул. Попова, 14, г. Пенза, 440046

телефон (8-412) 54-32-03, 54-30-32 E-mail: school53@guoedu.ru

ОКПО 24020409, ОГРН 1025801443568

ИНН/КПП 5837009907/583701001

«ВЫСШИЙ ПИЛОТАЖ - ПЕНЗА» 2024
Использование методов машинного обучения в медицине

Выполнила:
Чикмазова Алиса Ивановна,
учащаяся 10 класса
Научный руководитель:
Артюхина Елена Владимировна, старший преподаватель кафедры «Компьютерные технологии» ПГУ

Пенза, 2024

Введение

Сегодня уже трудно представить себе мир без современных информационных технологий. Они так глубоко вошли в нашу жизнь, что мы уже и не замечаем, какой стремительный рост их возможностей. В последние годы стремительно развивается область искусственного интеллекта (ИИ), достижения в области машинного обучения и робототехники произвели революцию во многих областях.

Актуальность:

Предприятия широко внедряют искусственный интеллект и машинное обучение в различных отраслях. Медицинский сектор ничем не является исключением. Многие исследовательские центры и организации здравоохранения осознали потенциал машинного обучения и активно улучшают свои услуги.

Неоспоримым является тот факт, что диагностика играет важную роль в медицине. Своевременно и точно поставленный диагноз, замеченная на ранних стадиях болезнь, существенно повышает вероятность выздоровления больного. И при этом сам процесс лечения будет менее дорогостоящим и быстрым. На начальных этапах клиническая картина может быть весьма размытой, различным стадиям развития болезни может соответствовать определенная и весьма сложная комбинация изменений наблюдаемых переменных, которая может быть с легкостью обнаружена и распознана, например, нейросетевыми технологиями [2].

Объектом изучения являются методы машинного обучения.

Целью работы является исследование возможностей методов машинного обучения для решения задачи постановки медицинского диагноза на примере определения диабета.

Задачи:

1. Изучить основные методы машинного обучения.
2. Подготовить и загрузить данные для анализа.
3. Выбрать наиболее подходящие для реализации методы решения поставленной задачи.
4. Провести экспериментальные исследования для выбора лучшего метода машинного обучения для определения диабета.

Ожидаемые результаты:

Реализованные алгоритмы позволят эффективно, т.е. с высокой точностью определять диабет.

Практическая значимость:

Полученные в работе результаты могут быть использованы при создании медицинских систем с применением методов машинного обучения.

Теоретическая часть

В настоящее время много говорят об искусственном интеллекте. ИИ - это комплекс технологических решений, позволяющий имитировать когнитивные функции человека и получать при выполнении конкретных практически значимых задач результаты, сопоставимые, как минимум, с результатами интеллектуальной деятельности человека.

В искусственном интеллекте приято выделять сильный ИИ — интеллектуальный алгоритм, способный решать широкий спектр интеллектуальных задач как минимум наравне с человеческим разумом" и слабый ИИ, прикладной ИИ — интеллектуальный алгоритм, имитирующий человеческий разум в решении конкретных узкоспециализированных задач. Сильный ИИ в настоящее время — это, скорее цель для науки. В области прикладного ИИ достигнуты огромные результаты: распознавание лиц, общение на естественном языке, поиск информации и т.п.).

Структура современного прикладного ИИ показана на рис. 1. Основным направление современного искусственного интеллекта является ***машинное обучение***— технологии автоматического обучения алгоритмов ИИ на примерах, в результате чего качество работы алгоритмов повышается. Важнейшей (но не единственной) частью современного машинного обучения являются *искусственные нейронные сети*— математические модели, состоящие из слоёв "нейронов", построенных по принципу организации и функционирования биологических нейронных сетей. Новым направлением нейронных сетей являются глубокие нейронные сети — сети, содержание большое число слоев.

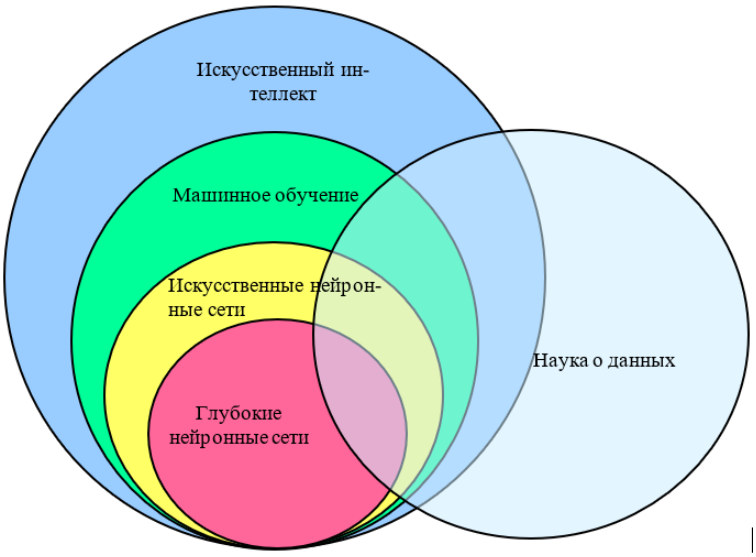

Рис. 1. Структура прикладного искусственного интеллекта

С искусственным интеллектом пересекается наука о данных (Data Science) — раздел информатики, изучающий проблемы анализа, обработки и представления данных в цифровой форме.

Машины не учатся как люди. Машинное обучение заключается в поиске математической модели, применение которой к набору входных данных (обучающему набору) дает желаемые результаты. Эта модель должна также обладать обобщающей способностью — формировать правильные выходные данные для входных данных, отличающихся от обучающих данных, при условии, что входные данные имеют такое же или близкое статистическое распределение, что и обучающие данные.

Машинное обучение делится на следующие основные виды:

Обучение с учителем или обучением на размеченных данных. При обучении с учителем имеется набор обучающих (тренировочных) примеров. Обучающие примеры — это наборы признаков, факторов, описывающих отдельные объекты, например, показатели, описывающие состояние больного. Каждому примеру соответствует известный ответ (метка, отклик, реакция, целевое значение). Задача обучения — так настроить на имеющихся примерах модель, чтобы на тех данных, на которых модель не обучалась, она выдавала достаточно точный ответ.

Обучение без учителя— это обучение на неразмеченных данных, когда целевые значения неизвестны. Типичная задача обучения без учителя — это кластеризация, когда необходимо разделить объекты на заранее неизвестные группы (кластеры) в зависимости от близости их признаков.

Обучение с частичным привлечением учителя использует обучающий набор данных, включающий как размеченные, так и не размеченные данные (таких данных, обычно, намного больше, чем размеченных). Привлечение к обучению неразмеченных данных вносит больше разнообразия в обучающие данные и повышает точность модели.

В обучении с подкреплением модель "живет" в некотором окружении и выполняет некоторые действия. Разные действия приносят разные вознаграждения. Цель алгоритма обучения с подкреплением — построить поведение, приносящее максимальные вознаграждения.

Практическая часть

Лучший способ научиться машинному обучению — проектировать и завершать свои небольшие проекты. Проект по машинному обучению обычно имеет несколько выраженных этапов: постановка задачи, загрузка данных, реализация алгоритмов и оценка качества их работы, оптимизация результата, презентация результата.

Постановка задачи

При постановке диагноза большую помощь врачу-диагносту может оказать компьютерная система, способная проводить самостоятельную обработку медицинских данных и давать какие-либо заключения. В данной работе исследуется вопрос о возможности применения нейронных сетей для диагностики диабета. Исходные данные, указанные в каждом задании, берутся из Репозитория данных для машинного обучения [3].

Выборка содержит 768 записей со следующими полями:

- Pregnancies Число случаев беременности;
- Glucose Концентрация глюкозы;
- BloodPressure Артериальное диастолическое давление, мм. рт. ст.;
- SkinThickness Толщина кожной складки трехглавой мышцы, мм. ;
- Insulin 2-х часовой сывороточный инсулин;
- BMI Индекс массы тела;
- DiabetesPedigreeFunction Числовой параметр наследственности диабета;
- Age Возраст, лет ;
- Outcome Зависимая переменная (1 – наличие заболевания, 0 – отсутствие).

Распределение зависимой переменной следующее: 500 случаев отсутствия заболевания, 268 – его наличие. Данная задача решалась с помощью инструмента логистическая регрессия в работе [4], что позволит провести сравнение результатов.

Загрузка данных и их визуализация

Программа начинается с импорта библиотек. Pandas необходима для обработки и анализа данных, seaborn и matplotlib для рисования графиков, tensorflow и sklearn – для машинного обучения.

После этого в программу загружаются и обрабатываются данные из файла

```
# считавыние данных из файла
ds = pd.read_csv('diabetes.csv')
```

Для получения представления о размерности исходных данных применяем метод shape. Чтобы увидеть первые 15 строк данных, выведем срез

```
print('Размерности исходных данных')
print(ds.shape)
```

```
#срез данных
print(ds.head(15))
```

Видим, что наш датасет содержит 768 строк и 9 столбцов.

Для получения некоторых описательных характеристик данных можно использовать метод

```
print(ds.describe())
```

```
Описательные статистика получаем с помощбю метода  describe
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

Рис. 3 Описательные статистики

Мы видим (рис. 3) количество данных, среднее значение, стандартное отклонение, минимальные и максимальные значения признаков, квартили. Для получения информации о распределении данных по выходному значению Outcome (наличие или отсутствие заболевания).

```
print(ds.groupby('Outcome').size())
```

```
Распределение данных по выходной переменной
Outcome
0    500
1    268
dtype: int64
```

Рис. 4 Результат `groupby`

Какими бы не были информативными данные результаты они остаются “плохочитаемыми”. Решением данной проблемы является построение различи гистограмм.

```
import matplotlib.pyplot as plt
sns.catplot(x='Outcome', kind="count", palette="Greens_r", data=ds)
plt.title('Распределение данных по выходной переменной')
```

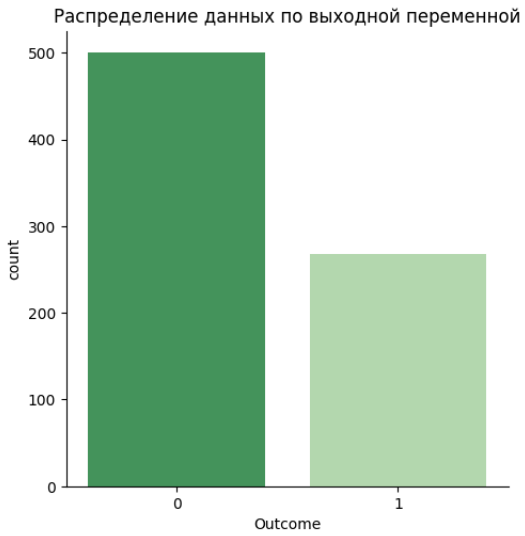

Рис. 5. Диаграмма распределения данных по выходному признаку

Или такую объемную диаграмму взаимодействия признаков:

```
plt.show()
# Диаграмма взаимодействия
sns.pairplot(ds)
```

На главной диагонали (рис 6) мы видим гистограммы распределения значений каждой переменной в отдельности. В других, недиагональных ячейках расположены диаграммы рассеяния признаков. Следует обратить внимание на диагональный характер некоторых из них, что говорит о наличии линейной связи между признаками.

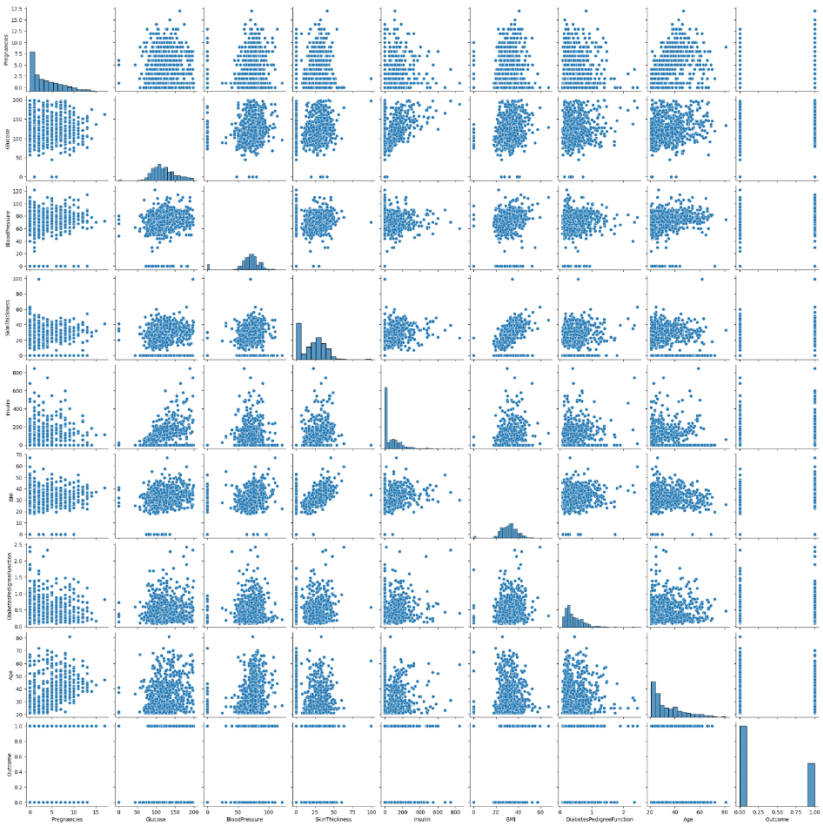

Рис. 6. Объемная диаграмма взаимодействия признаков

Возникает вопрос, как каждый признак в отдельности влияет на результат, для этого мы можем построить такие гистограммы.

Например, для выяснения изменяется число больных диабетом от возраста и индекса массы тела

```
plt.show()
sns.histplot(data=ds, x="Age", binwidth=0.5, hue="Outcome")
plt.show()
sns.histplot(data=ds, x="BMI", binwidth=0.5, hue="Outcome")
```

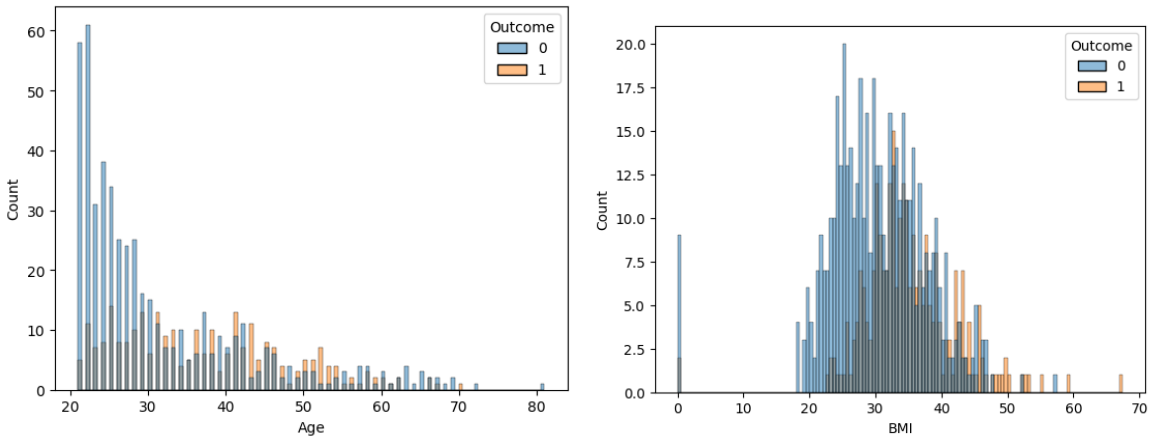

Рис. 6. Распределение количества больных и здоровы относительно возраста и ИМТ

Видим, что в молодом возрасте преобладают здоровые люди, но также замечаем, что диабет встречается у людей разного возраста. Что касается индекса массы тела, то диабет реже наблюдается у людей с низким BMI.

Выбрать наиболее подходящие для реализации методы решения поставленной задачи.

Для выбора метода решения поставленной задачи еще раз осознаем, что у нас есть размеченные данные (известен диагноз). Есть 8 входных числовых переменных, и одна выходная переменная (бинарная). Существует огромное множество методов машинного обучения, для рассмотрения и реализации в данной работе выберем: логистическую регрессию, деревья решений, нейронные сети.

Любая из этих моделей должна быть точной и иметь хорошую обобщающую способность. Поэтому все наши данные мы разделим на обучающую и тестовую выборки, на одних данных, мы будем проводить обучение модели (постройку параметров модели, на другом наборе проверять, не разучилась ли модель распознавать данные, которые она не видела).

```
# разделяем данные на входные и выходные
X = ds.drop(columns=['Outcome'])
Y = ds['Outcome']
# Разделение датасета на обучающую и тестовую выборки
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2,
random_state=11)
```

Логистическая регрессия

Логистическая регрессия — это разновидность множественной регрессии, общее назначение которой состоит в анализе связи между несколькими независимыми переменными и одной зависимой бинарной переменной. Во множественной линейной регрессии предполагается, что зависимая переменная является линейной функцией независимых переменных

$$y = a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_n x_n,$$

вместо предсказания бинарной переменной мы предсказываем непрерывную переменную со значениями на отрезке [0,1] при любых значениях независимых переменных. Это достигается применением следующего логит-преобразования

$$P = \frac{1}{1+e^{-y}},$$

где P — вероятность того, что произойдет интересующее событие. Тогда логистическую регрессию можно записать в виде

$$\ln\left(\frac{P}{1-P}\right) = a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_n x_n.$$

Следующие 4 строки кода позволяют создать модель model_LR, обучить ее на тестовом множестве и рассчитать точность на тестовом множестве.

```
from sklearn.linear_model import LogisticRegression
model_LR = LogisticRegression(max_iter=1000)
model_LR.fit(X_train, Y_train)
model_LR.score(X_test, Y_test)
```

Точность на тестовом множестве составила 0,74 (74%). Очень важно понимать сколько у нас ложноположительных и ложноотрицательных срабатываний

```
model_LR.predict(X_test)
y_pred = model_LR.predict(X_test)
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,y_pred)
```

Видим, что на тестовом множестве 89 правильно определенных здоровых и 25 больных.

```
array([[89, 11],
       [29, 25]])
```

Количество ошибок 11 ложноположительных (признали больными здоровых) и 29 ложноотрицательных срабатываний.

Дерево решений

Деревья решений — это способ представления правил в иерархической, последовательной структуре. Обычно каждый узел дерева включает проверку одного атрибута (независимой переменной). Иногда в узле две независимые переменные сравниваются друг с другом или вычисляется некоторая функция от одной или нескольких переменных.

При построении дерева решений формируются решающие правила и для каждого из них создается узел. Для каждого узла необходимо выбрать атрибут (атрибут ветвления), по которому будет производиться проверка правила. Метод, в соответствии с которым производится выбор атрибута ветвления на каждом шаге, называется алгоритмом построения дерева решений. При выборе атрибута ветвления должны обеспечиваться максимальная чистота. Большая чистота означает, что в узле будут представлены в основном объекты, относящиеся к одному классу. В идеальном случае в узле должны быть представлены объекты одного класса (не должно быть "примесей"). Кроме того, разбиение не должно создавать узлы, содержащие мало объектов (это значит, что правило применимо для малого числа объектов).

При построении деревьев решений применяют так называемые жадные алгоритмы. При построении дерева решений жадный алгоритм для каждого узла ищет оптимальный атрибут разбиения.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=1)
# Создание дерева решений с ограничение при максимальной глубине 2
model_DTC = DecisionTreeClassifier(max_depth=2)
# обучение классификатора
model_DTC = model_DTC.fit(x_train,y_train)
#проверка на тестовом множестве
y_pred = model_DTC.predict(x_test)
#Оценка точности
from sklearn import metrics
print("Точность на тестовом множестве:",metrics.accuracy_score(y_test,
y_pred)*100)
```

При построении дерева решений можно указывать такой параметр как максимальная глубина, при установке его в значение 2 мы получаем точность 0,7987 (79,87%). По таблице сопряженности можем видеть, что результат лучше, чем полученный с помощью логистической регрессии.

```
array([[89, 10],
       [21, 34]])
```

На тестовом множестве ошибки наблюдаются в 10+21=31 случае из 154.

Для иллюстрации самого дерева решений используются следующий набор команд

```
import matplotlib.pyplot as plt
from sklearn import tree
features=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
plt.figure(figsize=(15,15))
tree.plot_tree(model_DTC,feature_names =
features,class_names=['0','1'],filled=True, rounded=True)
plt.show()
```

Здесь мы еще раз задаем названия входных переменных, выходные классы.

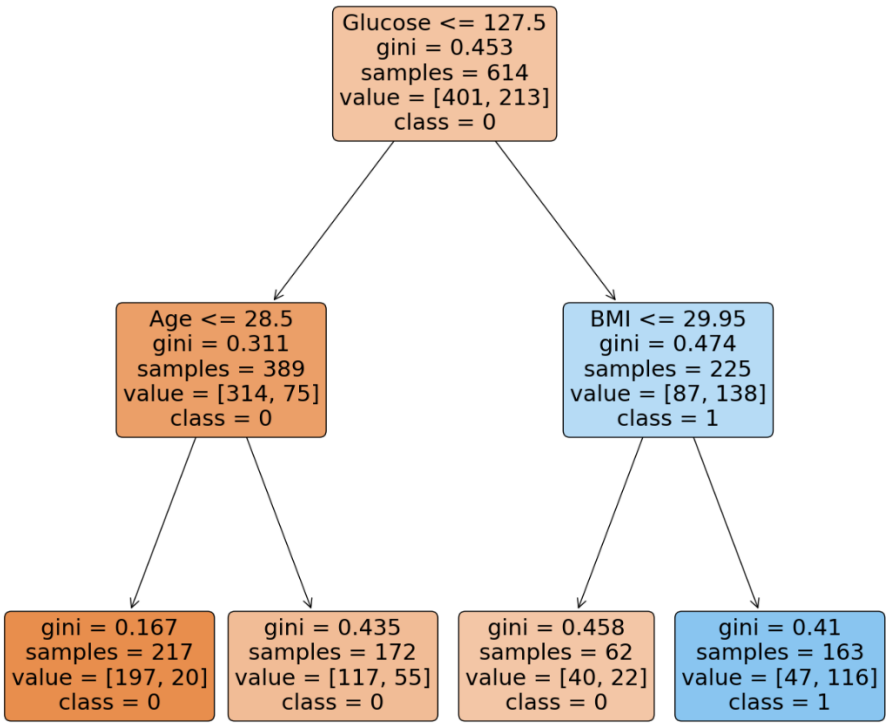

Рис. 7 Дерево решений при `max_depth=2`

Несомненным преимуществом дерева решений является, то, что мы можем воспроизвести правила классификации, они буквально формулируются на естественном языке.

Нейронная сеть

Искусственные нейронные сети— это модели машинного обучения, построенные из простых нелинейных элементов — искусственных нейронов, соединенных в сети.

На рисунке 8 представлены картинки однослойной сети, в ней входные сигналы взвешиваются (умножаются) на весовые коэффициенты w,

суммируются и подаются на функцию активации, на выходе получаем значение y. В случае сети с одним нейроном, можно сказать, что это логистическая регрессия.

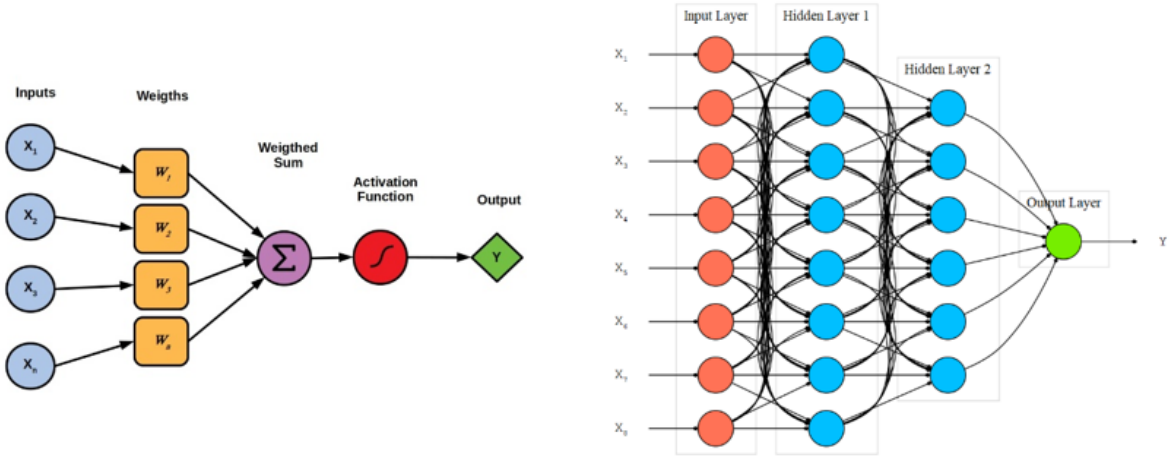

Рис. 8 Однослойная и двухслойная нейронные сети

Обучение нейронной сети сводится к настройке весовых коэффициентов, а также необходимо выбрать архитектуру самой сети, количество слоев, нейронов в каждом слое (их может быть достаточно много).

```
from keras.models import Sequential
from keras.layers import Dense
import numpy

# создаем модели, добавляем слои один за другим
model_NN2 = Sequential()
model_NN2.add(Dense(8, input_dim=8, activation='relu')) #входной слой
model_NN2.add(Dense(16, activation='relu'))
model_NN2.add(Dense(15, activation='relu'))
model_NN2.add(Dense(1, activation='sigmoid')) # выходной слой

# компилируем модель, используем градиентный спуск adam
model_NN2.compile(loss="binary_crossentropy", optimizer="adam",
metrics=['accuracy'])

# обучаем нейронную сеть
model_NN2.fit(X_train, Y_train, epochs = 1000, batch_size=10)

# оцениваем результат
scores = model_NN2.evaluate(X, Y)
print("\n%s: %.2f%%" % (model_NN2.metrics_names[1], scores[1]*100))
```

Следует заметить, что до сих пор не существует аналитических методов выбора параметров нейронных сетей. Выбор структуры сети производится на

основе опыта исследователя и результатов проведенных экспериментов. Была проведена достаточно большая серия экспериментов с различными архитектурами, здесь отображены лучшие варианты удавалось получать точности от 84-92%. Различные варианты объясняются различными архитектурами сети, а также начальной значений весовых коэффициентов сети (задаются случайным образом).

Сравнение полученных моделей

Большую точность показали нейросетевые модели. Но нужно понимать, чем более простой алгоритм, тем он грубее, но при этом легче объяснить полученные результаты, схематичное изображение представлено на рисунке 9. Наиболее мощные алгоритмы способны находить сложные нелинейные зависимости, но их интерпретация является непростой задачей. На практике необходимо находить компромисс между точностью и простотой.

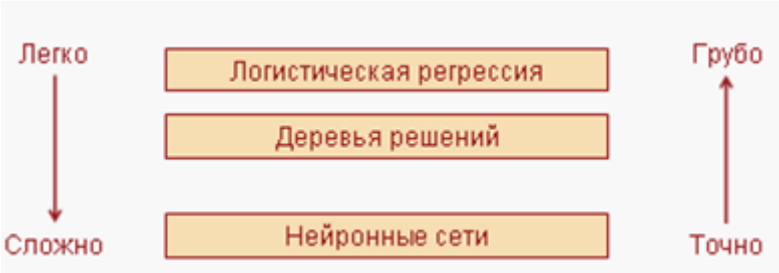

Рисунок 9. Методы машинного обучения

Алгоритмы машинного обучения становятся более точными по мере сбора и обработки данных с течением времени. В результате повышается точность и эффективность. Алгоритмы машинного обучения могут применятся в медицине от скрининга пациента до назначения правильных лекарств. Всегда есть опасения по поводу того, что ИИ заменит человек, но думаю, что в медицинской области разрабатываемы системы все равно остаются подсказчиком, и никоим образом не заменяют самого врача диагноста.

Заключение

1. В ходе теоретических исследований были изучены основные понятия и методы, связанные с машинным обучением.
2. Для работы с методами машинного обучения выбрали Python, tensorflow и sklearn. Для отображения данных и визуализации Pandas, seaborn и matplotlib.
3. Для решения поставленной задачи были выбраны методы логистическая регрессия, дерево решений, нейронные сети.
4. Для каждого выбранного метода была проведена серия экспериментов, получены следующие результаты точности классификации:

   Логистическая регрессия 74%

   Дерево решений 79,87%

   Нейронные сети 84-92%

В процессе работы были выполнены все поставленные задачи и достигнута цель. Полученные в работе результаты и опыт работы с могут быть ***в перспективе*** использованы при создании медицинских систем на основе алгоритмов машинного обучения.

Список литературы и источников

1. Паклин Н. Б., Орешков В. И. Бизнес-аналитика: от данных к знаниям.— СПб.: Питер, 2013. — 704 с.

2. Бурков А. Машинное обучение без лишних слов. — СПб.: Питер, 2020. — 192 с

3. https://www.kaggle.com/datasets/piyushborhade/diabetes-dataset/

4. Применение логистической регрессии в медицине и скоринге [Электронный ресурс] URL: https://basegroup.ru/community/articles/logis-medic-scoring