Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Орловский государственный университет имени И.С. Тургенева»
Физико-математический факультет
Кафедра информатики

Гулый Михаил Владимирович

Применение алгоритмов машинного обучения к задачам в медицине

Курсовая работа по дисциплине
«Математическое моделирование»

Направление подготовки:
01.03.02 Прикладная математика и информатика
Направленность (профиль): Системное программирование
и компьютерные вычисления
Квалификация: бакалавр

Руководитель:

к.ф.-м.н., доц. Дорофеева В.И._____

Оценка _____

Орёл – 2022

Содержание

Введение

Обоснование выбора темы и ее актуальность

По оценкам Международного агентства по изучению рака IARC, каждый пятый человек в течение своей жизни заболевает онкологией. В 2019 году только в России было выявлено свыше 640 тыс. человек с онкозаболеваниями. Эксперты связывают рост смертности от онкологических заболеваний с общим старением населения. Шансы выжить у онкобольных зависят от того, на какой стадии обнаружен рак: чем раньше, тем лучше [1].

Искусственный интеллект (ИИ) и машинное обучение успешно применяются в медицине и решают широкий круг задач, постепенно превращаясь из вспомогательного инструмента в хороших помощников медицинского персонала. В основе работы ИИ лежит анализ медицинских данных и их обработка по заданным алгоритмам. В настоящее время анализируются не только данные объективного осмотра и анамнеза пациента, но и результаты анализов и обследований на медицинском оборудовании. Применение подобных инструментов повышает эффективность врача, избавляя его от выполнения ряда рутинных операций, таких как ведение части медицинской документации и описание нормы при проведении обследований. Одна из значимых проблем применения ИИ в медицине – подготовка корректных медицинских данных для обучения алгоритмов, так как для этого требуется большое количество времени специалистов узкого профиля. Возможным решением видится создание объединенной платформы хранения медицинских данных, где врачи смогут готовить данные для применения ИИ в своей специальности. Это позволит в будущем повысить эффективность применения машинного обучения в медицине благодаря анализу разноплановых данных из различных источников [2].

В мире наиболее частой формой рака у женщин является рак молочной железы. По оценкам экспертов ВОЗ, в мире ежегодно регистрируют до 1 млн новых случаев заболевания раком молочной железы

Для раннего выявления патологии молочных желез рекомендуется проходить УЗИ молочных желез 1 раз в год с 18 до 39 лет, а с 40 лет проходить маммографию 1 раз в год и УЗИ молочных желез по показаниям.

При обнаружении на маммографии или УЗИ молочных желез какого-либо образования (в том числе кист т.к. атипичные кисты могут иметь злокачественный характер) для лечащего врача необходимо узнать о том является ли это образование злокачественным или нет.

Тонкоигольная биопсия молочной железы является одним из информативных и малотравматичных методов диагностики рака. При ее выполнении исследуются клетки образования [4].

Степень разработанности проблемы

Искусственные нейронные сети и автоматический анализ данных используются для обнаружения и диагностики рака с середины 1980-х годов. Сегодня наиболее инновационные методы основаны на машинном обучении.

Мнения о зрелости и успешности машинного обучения в медицине для анализа изображений КТ, МРТ, маммограмм и так далее для обнаружения рака разделились. Суммарное количество проектов и позитивных научных публикаций по ML-диагностике рака растет, но, например, IBM как один из первых разработчиков, который тестировал IBM Watson Oncology больше чем в 50 клиниках, в 2020 году сократил штат сотрудников подразделения, а в 2021-м объявил о намерении его продать.

Несмотря на волну скептицизма, количество разработок и продуктов для обнаружения злокачественных новообразований увеличивается, включая рекомендованные FDA (Управлением по санитарному надзору за качеством

пищевых продуктов и медикаментов США). Например, лидерами скрининга рака легкого считаются Philips Healthcare и SIEMENS Healthineers. Google AI Healthcare и IBM Watson Oncology тоже популярны, хотя и не рекомендованы FDA. На рынке ML-решений существует большая конкуренция со стороны стартапов и проектов с открытым исходным кодом. На текущий момент FDA разрешили использование для медицинских целей свыше 80 ML-решений.

Машинное обучение сейчас активно тестируется для выявления рака мозга, груди, легкого, кожи, крови и печени. В России в 2020 году Центром диагностики и телемедицины был развернут проспективный эксперимент для тестирования ИИ-решений. Он стал самым большим научным исследованием в мире на эту тему. Алгоритмы для выявления рака молочной железы и рака легкого включили в московскую программу тестирования летом 2020 года после сервисов для диагностики COVID-19 [1].

Американские исследователи применили машинное обучение для эффективного прогнозирования развития злокачественной опухоли молочной железы. Алгоритм обучили на данных биопсии пациенток, а точность его предсказания составила 97,4 процента. Статья опубликована в Radiology.

Наиболее эффективным способом лечения злокачественной опухоли молочной железы является мастэктомия — хирургическое удаление молочной железы. Однако если опухоль доброкачественная, то даже квалифицированный специалист не всегда может с высокой точностью предсказать, насколько велик риск развития рака в будущем. Авторы новой работы представили алгоритм машинного обучения, который по данным биопсии опухоли молочной железы может с высокой точностью определить доброкачественную опухоль с повышенным риском развития рака груди и,

соответственно, выявить необходимость последующего хирургического вмешательства.

Для этого исследователи собрали 1095 изображений биопсии грудной клетки, полученных от 1071 пациентки. Всего среди изображений исследователи обнаружили 1006 опухолей с высоким риском развития рака, из которых 115 привели к развитию рака молочной железы. Для обучения алгоритма ученые отобрали 671 изображение, а остальные использовали в качестве тестовой выборки.

Алгоритм был разработан при помощи метода машинного обучения random forest — классификатора, который основывается на работе множества деревьев решений. Такая система хранит особенности данных из обучающей выборки и затем применяет их для правильной классификации данных тренировочной выборки.

Система, разработанная исследователями, смогла правильно определить злокачественные опухоли с риском последующего развития рака молочной железы в 97,4 проценте случаев. По оценке авторов, ранняя диагностика патологии при помощи методов машинного обучения могла бы снизить количество необязательной мастэктомии на 30,6 процентов.

В декабре прошлого года ученые впервые успешно применили машинное обучение и анализ распределения пигмента для диагностики меланомы — об этом вы можете прочитать в нашей заметке. Также, здесь вы можете прочитать об алгоритме, который распознает одиночные (в том числе и раковые) клетки [3].

Предмет исследования

Исследование набора данных и создание модели машинного обучения. (написать шире)

Объект исследования

Набор данных о новообразованиях и их характеристиках, а также информация о том, является ли новообразование доброкачественным или злокачественным.

Цель работы

Создать модель машинного обучения, способную классифицировать новообразования с высокой точностью.

Основные задачи исследования

1. Исследование задачи на актуальность, анализ источников, соответствующих тематике работы.
2. Обоснование выбранных инструментов для проведения анализа данных и создания модели машинного обучения, описание признаков набора данных.
3. Анализ данных.
4. Создание различных моделей машинного обучения для данной задачи классификации новообразований.
5. Сравнение полученных моделей и выбор модели с самой высокой точностью предсказаний.

Структура работы

Работа состоит из введения, двух глав, заключения, списка источников и приложений.

Во введении рассматривается актуальность работы, ставится цель и обозначаются задачи, необходимые для достижения поставленной цели.

Первая глава представляет собой постановку задачи.

Во второй главе содержится обоснование выбора инструментов для исследовательского анализа данных и реализации моделей машинного

обучения, предварительная обработка данных, исследовательский анализ данных, а также создание моделей машинного обучения и сравнение их результатов.

В заключении делаются выводы о проделанной работе.

Приводится список использованных источников

Глава 1. Постановка задачи.

1.1 Описание предметной области

Диагностика онкологии имеет, несомненно, огромное значение в здравоохранении. От правильного диагноза зачастую зависит жизнь человека. Поэтому чрезвычайно важно классифицировать новообразование с максимально возможной точностью. Для достижения этой цели можно использовать модели машинного обучения.

1.2 Постановка задачи

Для создания нужной модели машинного обучения следует иметь набор данных, в котором хранятся результаты измерения параметров новообразований. К ведению статистики нужно относиться максимально серьезно, ведь недостаточно большой размера набора данных или пропущенные значения могут привести к некорректным результатам модели.

В связи с этим, следует выбрать набор данных, который бы не содержал пропущенных или некорректных значений и имел бы при этом достаточный размер для обучения модели.

Что касается моделей машинного обучения, необходимо подобрать оптимальные гиперпараметры, при которых модели будут выдавать наиболее высокую точность. Рекомендуется составить несколько моделей и сравнить их результаты, чтобы финальная модель обладала настолько высокой точностью, насколько это возможно.

1.3. Описание используемых инструментов для анализа данных и создания модели

В большинстве случаев для анализа данных, а также создания моделей машинного обучения используется язык программирования Python. Python – это высокоуровневый язык, который пользуется огромной

популярностью и подходит для решения задач самого разного рода, в том числе задач, связанных с машинным обучением.

В качестве библиотек для работы с данными будут использоваться:

- **Pandas** – это основной и самый главный модуль который будет использован в нашем анализе данных. Этот модуль предоставляет удобные структуры данных для манипулирования и обработки данных. Основной структурой является Dataframe который и будет использован далее. Dataframe представляет собой объект манипулирования и индексирования массивов двумерных данных.
- **NumPy** это open-source модуль для python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций. Они объединяются в высокоуровневые пакеты. NumPy (Numeric Python) предоставляет базовые методы для манипуляции с большими массивами и матрицами.

Для визуализации данных:

- Модуль **Matplotlib** используется для визуализации данных. С помощью этого модуля можно удобно строить графики и диаграммы, благодаря которым мы сможем проанализировать наш набор данных.
- Модуль **Seaborn** основанный на вышеупомянутом модуле Matplotlib расширит наши возможности визуализации данных. Этот модуль представляет высокоуровневый интерфейс для создания информативных графиков и различных диаграмм.

Для создания моделей:

- **Scikit-learn** - один из наиболее широко используемых пакетов Python для Data Science и Machine Learning. Он позволяет выполнять множество операций и предоставляет множество

алгоритмов. Scikit-learn также предлагает отличную документацию о своих классах, методах и функциях, а также описание используемых алгоритмов.

- **Keras** — это библиотека глубокого обучения, представляющая из себя высокоуровневый API, написанный на Python и способный работать поверх TensorFlow, Theano или CNTK. Он был разработан с расчетом на быстрое обучение. Способность переходить от гипотез к результатам с наименьшими временными затратами является ключом к проведению успешных исследований. Также в отличии от Scikit-learn, Keras может проводить вычисления как на CPU, так и на GPU, что может существенно сократить время обучения моделей.

1.4 Обзор признаков набора данных и их значение

- diagnosis – целевая переменная. Может содержать ‘M’ (malignant – злокачественная опухоль) или ‘B’ (benign – доброкачественная опухоль)
- radius – среднее расстояние от центра до точек по периметру
- texture - стандартное отклонение значений шкалы серого
- perimeter - периметр
- area – площадь
- smoothness - локальное изменение длины радиуса
- compactness - компактность (периметр ^ 2 / площадь - 1.0)
- concavity - вогнутость (выраженность вогнутых участков контура)
- concave points - количество вогнутых участков контура
- symmetry - симметрия
- fractal dimension - фрактальная размерность

Среднее значение (mean), стандартная ошибка (SE) и «худшее» (worst) значение (среднее из трех наибольших значений) этих характеристик (кроме diagnosis) были вычислены для каждого изображения, в результате чего было получено 30 признаков. Например, третий признак — это среднее значение для радиуса, тринадцатый признак – стандартная ошибка (SE) для радиуса, двадцать третий признак - «худшее» (worst) значение для радиуса.

Глава 2. Создание модели машинного обучения

2.1 Предобработка данных

2.1.1 Импорт модулей и загрузка набора данных

Импортируем необходимые нам модули и загрузим наш набор данных

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
outputfile = 'data/Breast Cancer Wisconsin (Diagnostic) Data Set.csv'
df = pd.read_csv(outputfile)
```

Рис. 1 – Импорт модулей и загрузка набора данных

2.1.2 Проверка пропущенных значений

```
print('Всего пропущенных значений:', df.isnull().sum().sum())
```

✓ 0.3s

```
Всего пропущенных значений: 0
```

Рис. 2 – Проверка пропущенных значений

2.1.3 Разбиение набора данных

Добавим в переменную ‘y’ значения целевого признака, а в переменную ‘X’ всё остальное

```
X = df.drop(['diagnosis'], axis=1)
y = df['diagnosis']
```

Рис. 3 – Разбиение набора данных

2.1.4 Кодирование целевого признака

Для корректной работы большинства моделей, модели должны получать на вход закодированные значения (числа, а не строки). Закодируем переменную y при помощи LabelEncoder.

```
from sklearn.preprocessing import LabelEncoder
le  = LabelEncoder()
y = le.fit_transform(y)
```

Рис. 4 – Кодирование целевого признака при помощи LabelEncoder

Теперь 1 означает ‘M’ (malignant – злокачественная опухоль), а 0 означает ‘B’ (benign – доброкачественная опухоль).

При необходимости, закодированные значения можно декодировать, используя метод inverse_transform

```
y
```
✓ 0.4s

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
```

Рис. 5 – Закодированный целевой признак

```
le.inverse_transform(y)
```

```
Output exceeds the size limit. Open the full output data in a text editor
array(['M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M',
       'M', 'M', 'M', 'M', 'M', 'M', 'B', 'B', 'B', 'M', 'M', 'M', 'M',
       'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'B', 'M',
       'M', 'M', 'M', 'M', 'M', 'M', 'M', 'B', 'M', 'B', 'B', 'B', 'B',
       'B', 'M', 'M', 'B', 'M', 'M', 'B', 'B', 'B', 'B', 'M', 'B', 'M',
```

Рис. 6 – Целевой признак в декодированном виде

2.1.5 Преобразование данных

Для корректного обучения моделей данные должны быть преобразованы. Для этого будем использовать StandardScaler. Он преобразует наши данные таким образом, что распределение каждого признака будет иметь среднее значение 0 и стандартное отклонение 1.

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X = pd.DataFrame(ss.fit_transform(X), columns=X.columns)
```

Рис. 7 – Преобразование данных при помощи StandardScaler

2.2 Исследовательский анализ данных (EDA)

2.2.1 Сравнение количества доброкачественных и злокачественных новообразований

По диаграмме заметно, что в нашем наборе данных содержится 357 строк в которых новообразование является доброкачественным, и 212 строк в которых новообразование является злокачественным.

```
ax = sns.countplot(x=y)
for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()-50), size=20)
```

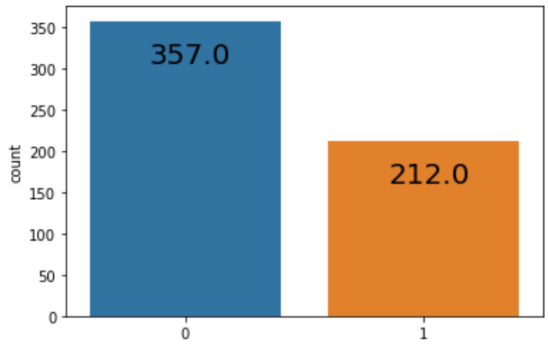

Рис. 8 – Сравнение количества доброкачественных и злокачественных новообразований

2.2.2 Получение характеристик признаков при помощи метода describe

Для дальнейшего анализа выберем те признаки, которые содержат в названии ‘mean’. В противном случае наши выводы будут слишком громоздкими.

```
mean_columns = [col for col in df.columns if 'mean' in col]
mean_columns
```

✓ 0.3s

```
['radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'concave points_mean',
 'symmetry_mean',
 'fractal_dimension_mean']
```

Рис. 9 – Выбор признаков содержащих 'mean' в названии

Используя метод describe, посмотрим на характеристики признаков наших данных

```
df[mean_columns].round(2).describe().T
```

✓ 0.7s

	count	mean	std	min	25%	50%	75%	max
radius_mean	569.0	-0.000053	1.000906	-2.03	-0.69	-0.22	0.47	3.97
texture_mean	569.0	0.000141	1.000835	-2.23	-0.73	-0.10	0.58	4.65
perimeter_mean	569.0	-0.000018	1.000974	-1.98	-0.69	-0.24	0.50	3.98
area_mean	569.0	-0.000176	1.000852	-1.45	-0.67	-0.30	0.36	5.25
smoothness_mean	569.0	-0.000105	1.000758	-3.11	-0.71	-0.03	0.64	4.77
compactness_mean	569.0	0.000035	1.000784	-1.61	-0.75	-0.22	0.49	4.57
concavity_mean	569.0	0.000158	1.000797	-1.11	-0.74	-0.34	0.53	4.24
concave points_mean	569.0	-0.000018	1.000959	-1.26	-0.74	-0.40	0.65	3.93
symmetry_mean	569.0	-0.000123	1.001169	-2.74	-0.70	-0.07	0.53	4.48
fractal_dimension_mean	569.0	0.000193	1.001022	-1.82	-0.72	-0.18	0.47	4.91

Рис. 10 – Использование метода describe

Как мы можем наблюдать, среднее значение признаков действительно находится близко к нулю, а среднее среднеквадратическое отклонение находится близко к единице.

2.2.3 Создание тепловой карты

Построим тепловую карту (heatmap). На этой тепловой карте видно, как коррелируют признаки между собой.

```
fig, ax = plt.subplots(figsize=(14, 7))
sns.heatmap(df[mean_columns+['diagnosis']].corr(), ax=ax, annot=True, annot_kws={
        "size": 15, "weight": "bold"}, cmap="viridis");
```

✓ 1.1s

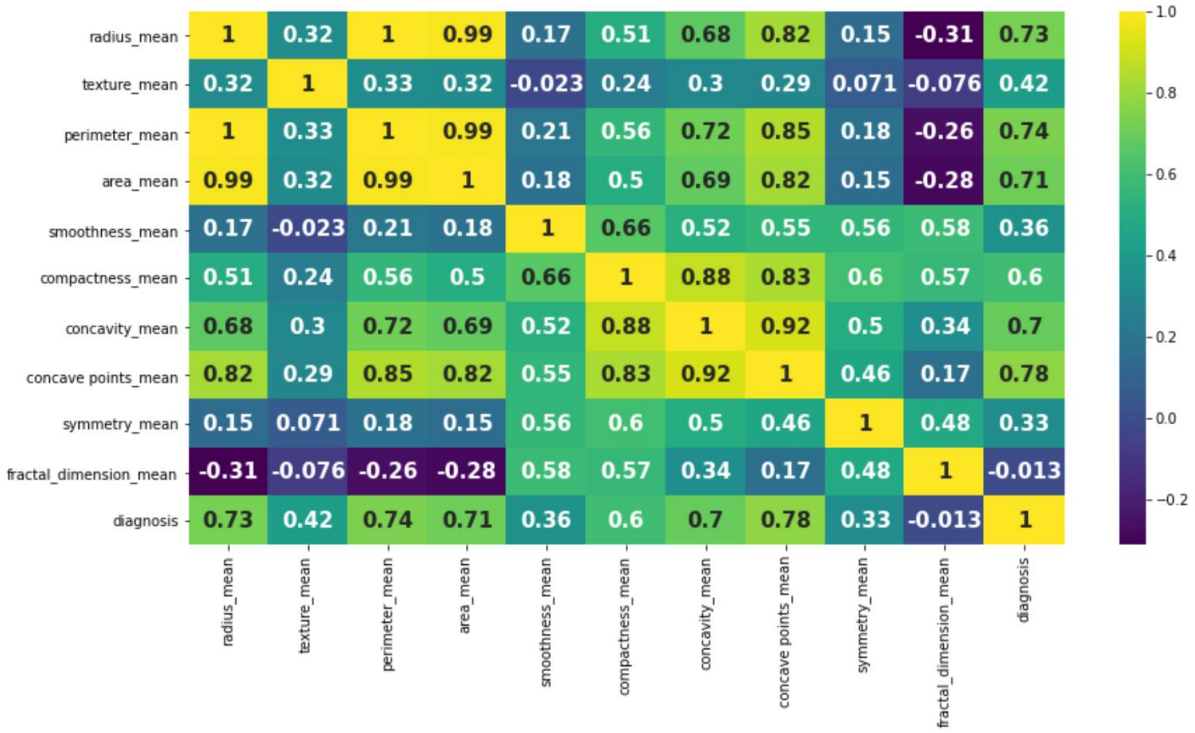

Рис. 11 – Создание тепловой карты

Рассматривая данную тепловую карту заметно, что наиболее сильно коррелируют с целевой переменной (diagnosis) такие признаки как radius_mean, perimeter_mean, area_mean, concavity_mean, concave points_mean. Признак fractal_dimension_mean наоборот, не оказывает почти никакого влияния на целевую переменную.

2.2.4 Визуализация взаимных отношений признаков при помощи pairplot

Построим pairplot на котором можно видеть, парные и индивидуальные распределения

```
sns.pairplot(df[mean_columns+['diagnosis']], hue='diagnosis');
```

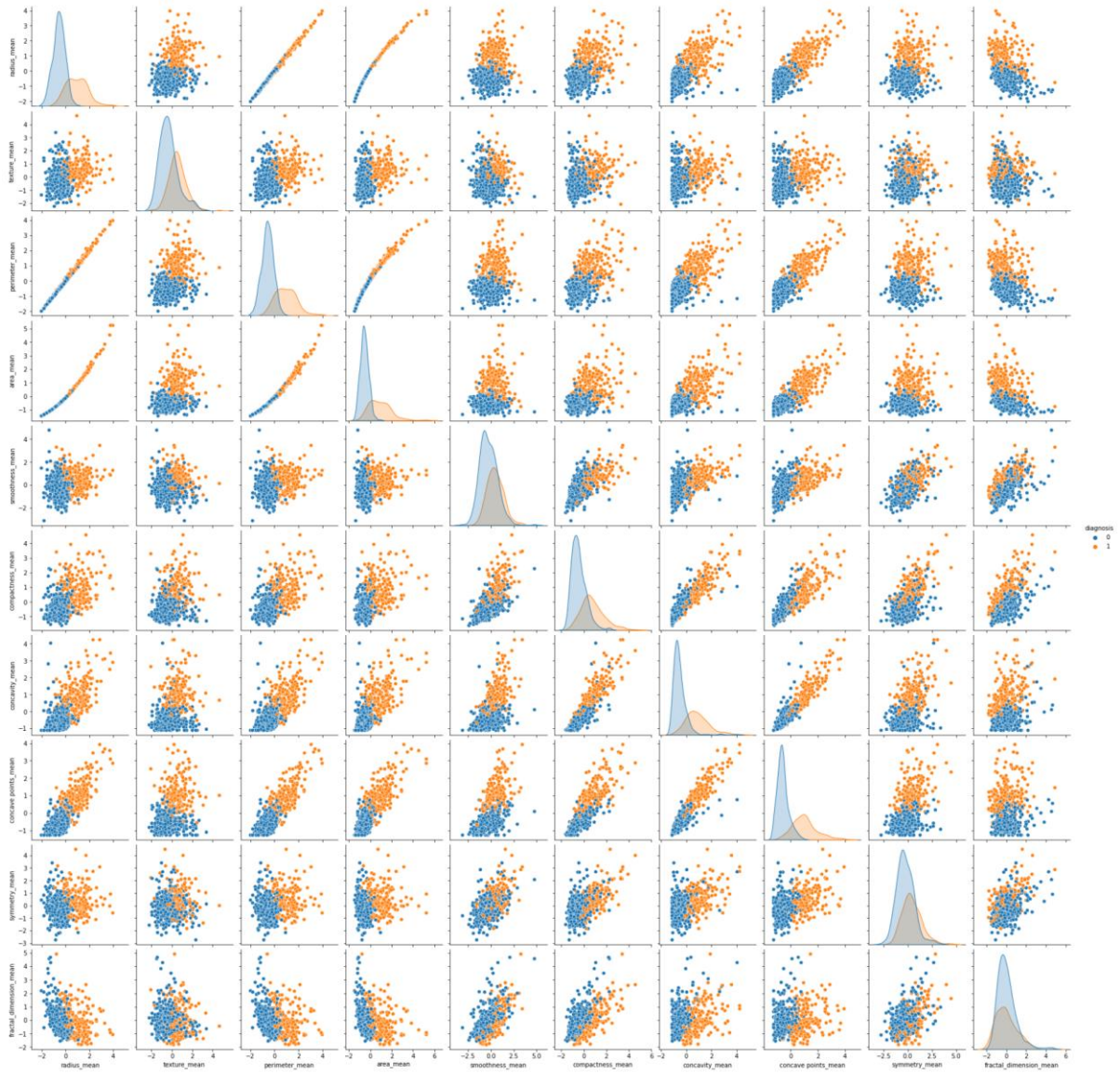

Рис. 12 – Создание графика парных и индивидуальных распределений

Как мы можем заметить исходя из данной диаграммы, во многих случаях можно мысленно разделить график на два кластера, в одном будут содержаться злокачественные (оранжевый) новообразования, а в другом доброкачественные (синий) образования. Например, для признака radius_mean, смотря на самую верхнюю строку, можно провести кривую, которая отделит злокачественные новообразования от доброкачественных.

2.3 Создание моделей машинного обучения

Для классификации целевого признака я воспользуюсь известными алгоритмами из библиотеки Sklearn. Для подбора оптимальных гиперпараметров я буду использовать GridSearchCV с 4-мя разбиениями. Этот механизм позволяет узнать, при каких гиперпараметрах достигается наибольшая accuracy (точность предсказаний).

2.3.1 Использование алгортима KNeighborsClassifier

Этот метод работает с помощью поиска кратчайшей дистанции между тестируемым объектом и ближайшими к нему классифицированным объектами из обучающего набора. Классифицируемый объект будет относится к тому классу, к которому принадлежит ближайший объект набора [5].

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedKFold
kfold = StratifiedKFold(n_splits=4, shuffle=True)
knn = KNeighborsClassifier()
grid = [{'n_neighbors': range(1, 40), 'weights': ['uniform', 'distance']}]
gridsearch = GridSearchCV(estimator=knn, param_grid=grid, cv=kfold)
gridsearch.fit(X, y)
print("Best parameters from gridsearch: {}".format(gridsearch.best_params_))
print("Accuracy: %0.3f" % gridsearch.best_score_)
```

```
Best parameters from gridsearch: {'n_neighbors': 8, 'weights': 'distance'}
Accuracy: 0.972
```

Рис. 13 – Использование алгортима KNeighborsClassifier

2.3.2 Использование алгортима LogisticRegression

Логистическая регрессия выводит прогнозы о точках в бинарном масштабе — нулевом или единичном. Если значение чего-либо равно либо больше 0.5, то объект классифицируется в большую сторону (к единице). Если значение меньше 0.5 — в меньшую (к нулю).

У каждого признака есть своя метка, равная только 0 или только 1. Логистическая регрессия является линейным классификатором и поэтому

используется, когда в данных прослеживается какая-то линейная зависимость [5].

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(max_iter=10000)
grid = [{'C': np.logspace(-1, 10, 10)}]
gridsearch = GridSearchCV(
    estimator=log_reg, param_grid=grid,  cv=kfold)
gridsearch.fit(X, y)
print("Best parameters from gridsearch: {}".format(gridsearch.best_params_))
print("Accuracy: %0.3f" % gridsearch.best_score_)
```

```
Best parameters from gridsearch: {'C': 1.6681005372000592}
Accuracy: 0.979
```

Рис. 14 – Использование алгортима LogisticRegression

2.3.3 Использование алгортима LinearSVC

Работа метода опорных векторов заключается в рисовании линии между разными кластерами точек, которые нужно сгруппировать в классы. С одной стороны линии будут точки, принадлежащие одному классу, с другой стороны — к другому классу.

Классификатор будет пытаться увеличить расстояние между рисуемыми линиями и точками на разных сторонах, чтобы увеличить свою «уверенность» определения класса. Когда все точки построены, сторона, на которую они падают — это класс, которому эти точки принадлежат [5].

```
from sklearn.svm import LinearSVC
LSVC = LinearSVC(max_iter=100000)
grid = [{'C': np.linspace(0.1, 20)}]
gridsearch = GridSearchCV(estimator=LSVC, param_grid=grid,  cv=kfold)
gridsearch.fit(X, y)
print("Best parameters from gridsearch: {}".format(gridsearch.best_params_))
print("Accuracy: %0.3f" % gridsearch.best_score_)
```

```
Best parameters from gridsearch: {'C': 0.1}
Accuracy: 0.975
```

Рис. 15 – Использование алгортима LinearSVC

2.3.4 Использование алгортима RandomForestClassifier

Random forest (с англ. — «случайный лес») — алгоритм машинного обучения, заключающийся в использовании комитета (ансамбля) решающих деревьев. Алгоритм сочетает в себе две основные идеи: метод бэггинга Бреймана, и метод случайных подпространств . Алгоритм применяется для задач классификации, регрессии и кластеризации. Основная идея заключается в использовании большого ансамбля решающих деревьев, каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим [6].

```
from sklearn.ensemble import RandomForestClassifier
grid = {'n_estimators': np.arange(150, 251, 10)}
rfc = RandomForestClassifier(n_jobs=-1)
gridsearch = GridSearchCV(estimator=rfc, param_grid=grid, cv=kfold)
gridsearch.fit(X, y)
print("Best parameters from gridsearch: {}".format(gridsearch.best_params_))
print("Accuracy: %0.3f" % gridsearch.best_score_)
```

```
Best parameters from gridsearch: {'n_estimators': 210}
Accuracy: 0.967
```

Рис. 16 – Использование алгортима RandomForestClassifier

2.3.5 Использование алгортима ExtraTreesClassifier

Extremely Randomized Trees (или Extra-Trees) - это метод ансамблевого обучения. Метод создает дополнительные деревья в подвыборках наборов данных и применяет большинство голосов для улучшения предсказуемости классификатора. Благодаря такому подходу метод уменьшает дисперсию. Метод применяет случайные пороговые значения для каждой характеристики подвыборок, чтобы получить наилучшее из пороговых значений в качестве правила разделения.

```
from sklearn.ensemble import ExtraTreesClassifier
grid = {'n_estimators': np.arange(150, 251, 10), 'bootstrap': [True, False]}
etc = ExtraTreesClassifier(n_jobs=-1)
gridsearch = GridSearchCV(estimator=etc, param_grid=grid, cv=kfold)
gridsearch.fit(X, y)
print("Best parameters from gridsearch: {}".format(gridsearch.best_params_))
print("Accuracy: %0.3f" % gridsearch.best_score_)
```

```
Best parameters from gridsearch: {'bootstrap': False, 'n_estimators': 190}
Accuracy: 0.970
```

Рис. 17 – Использование алгортима ExtraTreesClassifier

2.3.6 Использование алгортима GradientBoostingClassifier

Градиентный бустинг — это техника машинного обучения для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений.

Бустинг — это техника построения ансамблей, в которой предсказатели построены не независимо, а последовательно [7].

```
from sklearn.ensemble import GradientBoostingClassifier
gbc = GradientBoostingClassifier()
grid = {'n_estimators': np.arange(150, 250, 10), 'learning_rate': [1, 0.1, 0.01],
        'subsample': [1.0, 0.5], 'max_features': np.arange(1, len(df.columns), 5)}
gridsearch = GridSearchCV(estimator=gbc, param_grid=grid, cv=kfold, n_jobs=-1)
gridsearch.fit(X, y)
print("Best parameters from gridsearch: {}".format(gridsearch.best_params_))
print("Accuracy: %0.3f" % gridsearch.best_score_)
```

```
Best parameters from gridsearch: {'learning_rate': 0.1, 'max_features': 6, 'n_estimators': 210, 'subsample': 0.5}
Accuracy: 0.974
```

Рис. 18 – Использование алгортима GradientBoostingClassifier

2.4 Создание моделей глубокого обучения (Deep Learning)

Глубокое обучение — это разновидность машинного обучения на основе искусственных нейронных сетей. Процесс обучения называется глубоким, так как структура искусственных нейронных сетей состоит из нескольких входных, выходных и скрытых слоев. Каждый слой содержит единицы, преобразующие входные данные в сведения, которые следующий слой может использовать для определенной задачи прогнозирования. Благодаря этой структуре компьютер может обучаться с помощью собственной обработки данных [8].

2.4.1 Модель № 1

Итак, в качестве первой модели будем использовать модель с одним слоем, который содержит 12 нейронов.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score


def create_model():
    model_1 = Sequential()
    model_1.add(Dense(12, input_shape=(X.shape[1],), activation='sigmoid'))
    model_1.add(Dense(1, activation='sigmoid'))
    model_1.compile(optimizer='sgd', loss="binary_crossentropy",
                    metrics=["accuracy"])
    return model_1


classifier = KerasClassifier(build_fn=create_model, epochs=200, verbose=False)
kfold = StratifiedKFold(n_splits=4, shuffle=True)
result_acc = cross_val_score(estimator=classifier, X=X, y=y, cv=kfold)

print('accuracy is {:.3f}'.format(result_acc.mean()))
```

```
accuracy is 0.967
```

Рис. 19 – Первая модель глубокого обучения

2.4.2 Модель № 2

Вторая модель будет содержать 2 слоя, в каждом слое содержится по 6 нейронов

```
def create_model():
    model_2 = Sequential()
    model_2.add(Dense(6, input_shape=(X.shape[1],), activation="relu"))
    model_2.add(Dense(6,  activation="relu"))
    model_2.add(Dense(1, activation="sigmoid"))
    model_2.compile(optimizer='rmsprop', loss="binary_crossentropy",
                    metrics=["accuracy"])
    return model_2


classifier = KerasClassifier(build_fn=create_model, epochs=200, verbose=False)
kfold = StratifiedKFold(n_splits=4, shuffle=True)
result_acc = cross_val_score(estimator=classifier, X=X, y=y, cv=kfold)

print('accuracy is {:.3f}'.format(result_acc.mean()))
```

```
accuracy is 0.977
```

Рис. 20 – Вторая модель глубокого обучения

2.4.3 Модель № 3

Третья модель будет содержать 3 слоя, в каждом слое содержится 10 нейронов. Также, после добавления второго и третьего слоёв, мы добавляем Dropout. С помощью этого приёма мы с вероятностью 25% отключаем нейроны с предшествующем слое. Такой приём увеличивает качество обучения на тренировочных данных, а также повышает качество предсказаний модели на новых тестовых данных

```
def create_model():
    model_3 = Sequential()
    model_3.add(Dense(10, input_shape=(X.shape[1],), activation="relu"))
    model_3.add(Dense(10,  activation="relu"))
    model_3.add(Dropout(0.25))
    model_3.add(Dense(10,  activation="relu"))
    model_3.add(Dropout(0.25))
    model_3.add(Dense(1, activation="sigmoid"))
    model_3.compile(optimizer='adam', loss="binary_crossentropy",
                    metrics=["accuracy"])
    return model_3


classifier = KerasClassifier(build_fn=create_model, epochs=200, verbose=False)
kfold = StratifiedKFold(n_splits=4, shuffle=True)
result_acc = cross_val_score(estimator=classifier, X=X, y=y, cv=kfold)

print('accuracy is {:.3f}'.format(result_acc.mean()))
```

```
accuracy is 0.981
```

Рис. 21 – Третья модель глубокого обучения

2.5 Сравнение результатов

Результаты вышеупомянутых моделей были занесены в датафрейм для большей наглядности. По итогу получился следующий датафрейм

	name	accuracy
0	knn 9 uniform	0.971880
1	log reg C=1.6681005372000592	0.978935
2	LinearSVC C=0.5061224489795918	0.975414
3	RandomForestClassifier 220	0.966611
4	ExtraTreesClassifier 220 bootstrap=False 180	0.970095
5	GBC lr=0.1 mf=11 n_est=180 subsample=0.5	0.973653
6	model_1	0.966611
7	model_2	0.977187
8	model_3	0.980683

Рис. 22 – Сравнение результатов используемых моделей

Как мы можем видеть, наибольшей точностью предсказаний (accuracy) обладает третья нейронная сеть.

Другие исследования по данной задаче вы можете посмотреть по ссылке: https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/code .

Большинство людей сумели создать модель, классифицирующую новообразования с точностью от 0.95 до 0.98, но далеко не многие проводили кросс-валидацию (особенно для моделей глубокого обучения), что ставиты результаты по сомнение.

Заключение

В данной работе были представлены исследовательский анализ данных, создание моделей машинного обучения и сравнение эффективности этих моделей. Приведено обоснование выбора набора данных и инструментов для анализа набора данных и создания моделей. Описаны все признаки набора данных и произведена предобработка, в ходе которой целевой признак был закодирован, а нецелевые признаки были преобразованы для работы с моделями машинного обучения. Произведён исследовательский анализ данных. Реализованы 6 популярных алгоритмов машинного обучения с подбором оптимальных гиперпараметров для этих алгоритмов. Также созданы и протестированы 3 нейронные сети. Результаты точности предсказаний были вычислены при помощи кросс-валидации, что подчёркивает корректность результатов. Проведено сравнение результатов точности предсказаний, а также выбрана модель с самой высокой точностью.

Цель данной работы была выполнена – удалось создать модель машинного обучения, способную классифицировать новообразования с высокой точностью.

Были выполнены задачи:

1. Исследованы задачи на актуальность, анализ источников, соответствующих тематике работы.
2. Обоснован выбор инструментов для проведения анализа данных и создания моделей.
3. Проведены предобработка данных и описание признаков набора данных.
4. Проведен исследовательский анализ данных.
5. Созданы модели машинного обучения.

6. Произведено сравнение различных моделей, а также выбор модели с самой высокой точностью.

В результате проделанной работы, наилучшая модель имеет точность предсказаний 0.980683. Это достаточно высокий результат. Данная модель может иметь практическое применение в классификации новообразований полученных с помощью изображений биопсии грудной клетки.

Список источников

1. Как машинное обучение помогает в борьбе с онкологией [Электронный ресурс]. – Режим доступа: https://trends.rbc.ru/trends/industry/610032979a7947c814cd616c

2. Применение искусственного интеллекта для анализа медицинских данных [Электронный ресурс]. – Режим доступа: https://www.almclinmed.ru/jour/article/view/1190?locale=ru_RU

3. Машинное обучение предскажет необходимость мастэктомии [Электронный ресурс]. – Режим доступа: https://nplus1.ru/news/2017/10/18/breast-cancer-machine-learning

4. Тонкоигольная аспирационная биопсия молочной железы [Электронный ресурс]. – Режим доступа: https://www.zb-tula.ru/diagnostika/minimalno-invazivnye-vmeshatelstva-pod-ultrazvukovym-kontrolem/tonkoigolnaya-aspiratsionnaya-biopsiya-molochnoy-zhelezy/

5. Обзор методов классификации в машинном обучении с помощью Scikit-Learn [Электронный ресурс]. – Режим доступа: https://tproger.ru/translations/scikit-learn-in-python/#:~:text=%D0%BA%D0%B0%D0%B6%D0%B4%D0%BE%D0%B3%D0%BE%20%D0%B8%D0%B7%20%D0%BD%D0%B8%D1%85.-,%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%20k%2D%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B8%D1%85%20%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9%20(K%2DNearest%20Neighbors),%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D0%BE%D0%BC%D1%83%20%D0%BF%D1%80%D0%B8%D0%BD%D0%B0%D0%B4%D0%BB%D0%B5%D0%B6%D0%B8%D1%82%20%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B8%D0%B

9%20%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%20%D0%BD%D0%B0%D0%B1%D0%BE%D1%80%D0%B0.

6. Random forest [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Random_forest

7. Градиентый бустинг — просто о сложном [Электронный ресурс]. – Режим доступа: https://neurohive.io/ru/osnovy-data-science/gradientyj-busting/

8. Глубокое обучение, машинное обучение и искусственный интеллект [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/azure/machine-learning/concept-deep-learning-vs-machine-learning