# Дипломная работа на тему: **"Сравнение различных библиотек для машинного обучения: Scikit-learn, TensorFlow и PyTorch: Реализовать задачи классификации и регрессии с использованием Scikit-learn, TensorFlow и PyTorch, сравнить их производительность и удобство использования."**

**1. Введение**

**Обоснование выбора темы:**

1. На сегодняшний день машинное обучение (МО) стало одним из ключевых инструментов для решения широкого круга задач в различных областях науки и промышленности. Библиотеки и фреймворки, описывающие алгоритмы и инструменты для реализации моделей МО, стали основой этих инноваций. Среди них особо выделяются библиотеки **Scikit-learn**, **TensorFlow** и **PyTorch**, которые завоевали признание как ученых, так и практиков.

Каждая из этих библиотек обладает своими сильными сторонами и областью применения. **Scikit-learn** широко используется для решения базовых задач классификации и регрессии, обладая простотой и интуитивным интерфейсом. **TensorFlow** и **PyTorch** чаще используются для реализации сложных нейронных сетей, особенно в области глубокого обучения. Используя, модели классификации и регрессии, пользователи могут эффективно справляться с проектами по машинному обучению и достигать наилучших результатов.

2. Потребности рынка: существует растущая потребность в специалистах, обладающих навыками применения и сравнения различных моделей прогнозирования и анализа данных, таких как Регрессия и классификация, а также навыками работы с библиотеками глубокого обучения такими как Tensorflow, PyTorch. Компании стремятся использовать наиболее эффективные модели для улучшения качества своих продуктов и услуг.

3. Практическая значимость: Работа в области машинного обучения позволит применить полученные знания на практике и даст опыт использования библиотек TensorFlow, sсikit-learn и PyTorch и моделей классификации и регрессии. Это позволит создавать интересные и полезные для пользователей проекты.

4. Личный интерес и потенциальные перспективы: Навыки работы с библиотеками Tensorflow, skikit-learn и PyTorch востребованы на рынке труда, что открывает перспективы для карьерного роста и развития.

Таким образом, выбор темы “**Сравнение различных библиотек для машинного обучения: Scikit-learn, TensorFlow и PyTorch**: Реализовать задачи классификации и регрессии с использованием Scikit-learn, TensorFlow и PyTorch, сравнить их производительность и удобство использования” обусловлен ее актуальностью, потребностями рынка, практической значимостью и личным интересом, что делает эту тему подходящей для проведения дипломной работы.

**Определение цели и задач исследования:**

Цель данной дипломной работы — сравнить производительность, удобство использования и функциональные особенности библиотек **Scikit-learn**, **TensorFlow** и **PyTorch** при решении задач классификации и регрессии. В рамках работы будут проведены эксперименты для оценки скорости обучения, точности моделей, а также удобства использования каждой библиотеки.

**Задачи исследования:**

1. Обзор библиотек, представляющих модели классификации и регрессии, для выполнения задач машинного обучения .

2. Разработка сценариев сравнения: Разработать алгоритм сравнения эффективности библиотек, на основании использования моделей классификации и регрессии.

3. Проведение тестирования: Создать проект в Google Collab, в котором можно будет оценить эффективность различных моделей классификации и регрессии с использованием заявленных библиотек для МО, и провести их тестирование.

4. Написание дипломной работы: Составить дипломную работу, включающую в себя введение, обзор литературы, методологию и результаты исследования, анализ результатов, выводы и рекомендации.

Цели и задачи исследования направлены на получение практических результатов, которые позволят лучше понять разницу моделей классификации и регрессии, а также удобство использования библиотек TensorFlow, PyTorch и Scikit-learn.

**2. Основные понятия и определения**

Обзор основных понятий:

1**. Фреймворк (Framework)**: Программная платформа, которая предоставляет готовые компоненты и инструменты для разработки, в том числе машинного обучения. В данной работе будут использоваться такие фреймворки как PyTorch, Tensorflow и Scikit-learn.

2. **Обучение с учителем** (Supervised Learning) - обучение с учителем включает в себя обучение модели на размеченных данных. Размеченные данные означают, что для каждого примера в наборе данных известен правильный ответ или метка.

3. **Глубокое обучение** (Deep Learning) — это подраздел машинного обучения, который основан на использовании искусственных нейронных сетей с множеством слоев. Эти сети способны автоматически извлекать иерархические признаки из данных, что позволяет решать сложные задачи, такие как распознавание изображений, обработка естественного языка, генерация текста, речи и многое другое.

4. **Датасет (Dataset)** — это структурированная информация в табличном виде, где у каждого объекта прописаны определенные свойства: характеристики, связи или конкретные места. Этот механизм применяют для построения гипотез, анализа результатов или обучения нейросети на основе данных.

5. **Google Colab** – Это облачный сервис на основе Jupyter Notebook, не требующий установки, с бесплатным доступом к вычислительным ресурсам, включая графические процессоры и TPU. Colab применяется главным образом для машинного обучения, обработки данных и образовательных проектов.

6. **Эффективность модели (Model Efficiency)**: Оценка того, насколько точно и быстро модель выдает прогноз. Основные метрики для оценки эффективности включают достоверность (accuracy), полноту (recall), точность (precision), и f1-score, а также MSE, MAE, r^2.

7. **Достоверность (Accuracy)**: Процент правильно классифицированных объектов от общего числа объектов. Высокая точность указывает на то, что модель делает меньше ошибок.

8. **Полнота (Recall)**: Способность модели обнаруживать все истинные положительные примеры среди всех положительных примеров в данных. Высокая полнота означает, что модель пропускает меньше истинных положительных объектов.

9. **Точность (Precision)**: Способность модели правильно определять положительные примеры среди всех примеров, которые она классифицировала как положительные. Высокая точность означает, что модель делает меньше ложных срабатываний.

10. **F1-score**: Среднее гармоническое значение точности и полноты, которое используется как единая метрика для оценки эффективности модели, особенно когда имеется несбалансированный набор данных.

11. **MSE** (Mean Squared Error) — это средняя квадратичная ошибка, которая используется для оценки качества предсказаний модели. Она измеряет среднее значение квадратов разностей между предсказанными и фактическими значениями. Чем меньше MSE, тем точнее модель.

12. **MAE** (Mean Absolute Error) — это средняя абсолютная ошибка, метрика, используемая для оценки точности моделей машинного обучения, особенно в задачах регрессии. Она измеряет среднее абсолютное отклонение предсказанных значений от фактических.

13. **Коэффициент детерминации R2** (или R-квадрат) — это статистическая мера, которая показывает, насколько хорошо модель регрессии объясняет вариацию зависимой переменной (целевой переменной). Значение ближе к 1 указывает на лучшее соответствие модели данным.

14. **Классификация** - алгоритм предсказания дискретной (категориальной) функции (например, определить пол человека по его изображению)

15. **Регрессия** это алгоритм предсказания непрерывной функции (например, определить возраст человека по его изображению).

Данные термины помогают понять ключевые аспекты проверки эффективности моделей классификации и регрессии с использованием указанных фреймворков.

**3. Методы и подходы к разработке**

**Обоснование выбора датасетов и моделей**

Выбор датасета: Для задач регрессии выбраны датасеты, где есть: набор входных данных (например, изображения, числовые или текстовые признаки), и непрерывные выходные значения (например, цена, температура, время). Для задач классификации выбраны датасеты, которые имеют метки, представляющие категорию (например, футболка, свитер, ботинки и т. д.). Целевые значения в данном случае — дискретные классы (от 0 до 9), а не непрерывные данные, которые необходимы для регрессии.

Выбор модели: В библиотеках Scikit-learn, TensorFlow и PyTorch представлены различные модели классификации и регрессии. Я использовала такие модели как LogisticRegression, Sequential, LinearRegression.

**Архитектура создания моделей классификации и регрессии**

1. Импорт необходимых библиотек
2. Выбор и загрузка датасета
3. Изучение и нормализация датасета
4. Выбор и обучение модели
5. Оценка модели

**4. Обзор основных библиотек для разработки моделей классификации и регрессии**

Основные библиотеки, использованные в проекте – Numpy, Pandas, Matplotlib, Scikit-learn, TensorFlow, PyTorch, ниже более подробное описание.

**NumPy**

NumPy (Numerical Python) — это библиотека для работы с многомерными массивами и матрицами, а также для выполнения математических операций над ними. Она является основой для многих других библиотек, таких как Pandas и Scikit-learn.

Основные функции:

* Создание и манипуляции с массивами (np.array, np.zeros, np.ones, np.arange).
* Математические операции (линейная алгебра, статистика, тригонометрия).
* Индексация и срезы массивов.
* Работа с случайными числами (np.random).

Применение: NumPy используется для предобработки данных, выполнения математических операций и подготовки данных для обучения моделей.

**Pandas**

Pandas — это библиотека для работы с табличными данными (DataFrame и Series). Она предоставляет удобные инструменты для загрузки, обработки и анализа данных.

Основные функции:

* Загрузка данных из различных источников (CSV, Excel, SQL и др.).
* Очистка данных (удаление пропущенных значений, дубликатов).
* Фильтрация, сортировка и группировка данных.
* Объединение и разделение данных.
* Работа с временными рядами.

Применение: Pandas используется для предобработки данных, их анализа и подготовки к обучению моделей.

**Matplotlib**

Matplotlib — это популярная библиотека для создания визуализаций и графиков на языке программирования Python. Она предоставляет широкий набор инструментов для построения статических, анимированных и интерактивных графиков, которые могут быть использованы в научных исследованиях, анализе данных, машинном обучении и других областях.

**Scikit-learn**

Scikit-learn — это библиотека для машинного обучения, которая предоставляет инструменты для построения моделей классификации, регрессии, кластеризации и других задач.

Основные функции:

* Разделение данных на обучающую и тестовую выборки (train\_test\_split).
* Предобработка данных (масштабирование, кодирование категориальных переменных).
* Построение моделей (линейная регрессия, логистическая регрессия, деревья решений, SVM и др.).
* Оценка качества моделей (точность, F1-мера, ROC-AUC и др.).
* Кросс-валидация и подбор гиперпараметров.

Применение: Scikit-learn используется для обучения, оценки и настройки моделей машинного обучения.

**TensorFlow и PyTorch**

Эти библиотеки используются для глубокого обучения и построения нейронных сетей. Они предоставляют гибкие инструменты для создания и обучения сложных моделей.

Основные функции:

* Создание и обучение нейронных сетей.
* Работа с тензорами и автоматическое дифференцирование.
* Поддержка GPU для ускорения вычислений.

Применение: TensorFlow и PyTorch используются для задач, требующих глубокого обучения, таких как обработка изображений, текста и временных рядов.

**5. Проектирование проекта**

**Планирование проекта**

Выбор платформы и инструментов: Выбор подходящей платформы для машинного обучение (Google Collab) и определение наиболее подходящих библиотек (например, Numpy, Pandas, Scikit-learn, TensorFlow, PyTorch).

Определение структуры проекта: Разработка схемы проекта, включающей две части кода для выполнения задачи классификации и регрессии соответственно.

Разработка кода

**Технические требования**

Библиотеки: использовать Scikit-learn, TensorFlow, PyTorch для разработки моделей классификации и регрессии

Дополнительные инструменты: Numpy, Pandas, PIL, Matplotlib

Датасеты: Использование датасетов (например, Fashion MNIST, Diabetes, CIFAR10) для обучения моделей.

Интеграция с моделью машинного обучения: обучение модели на заданном датасете.

**6. Разработка в соответствии с созданной документацией**

**Планирование разработки**

Разработка была разделена на два основных блока: решение проблемы классификации с использованием вышеуказанных библиотек и решение проблемы регрессии с использованием тех же библиотек.

**Разработка**

Разработка моделей классификации с библиотеками Scikit-learn, Tensorflow, PyTorch:

Рис. 1 Модель классификации с использованием Scikit-learn



Рис. 2 Модель классификации с использованием TensorFlow

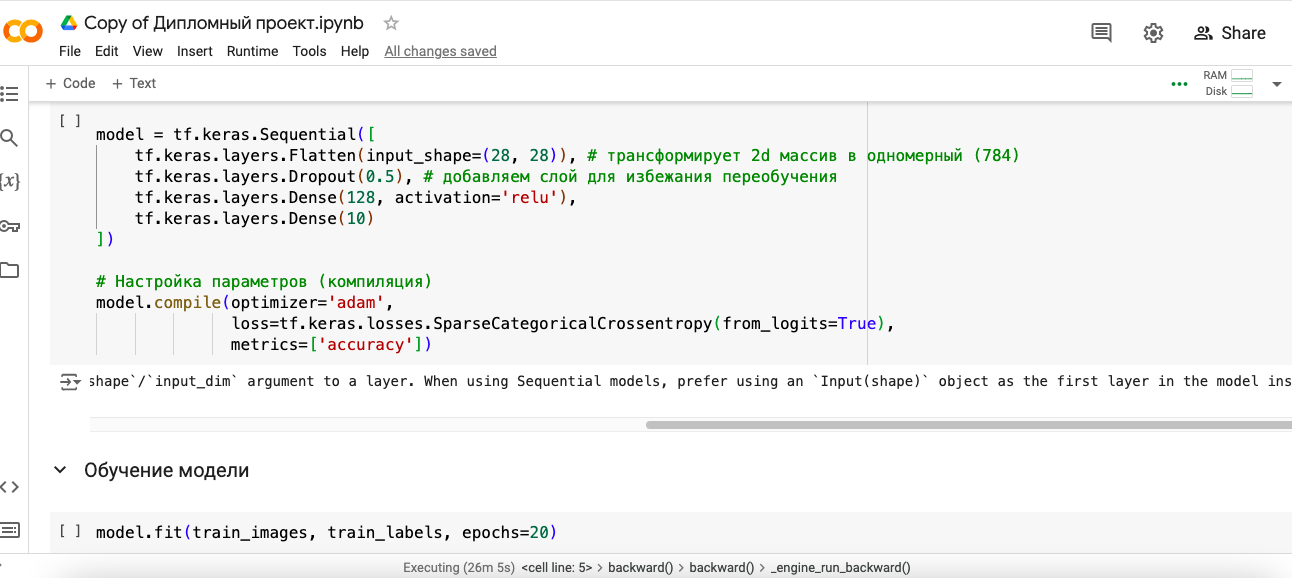
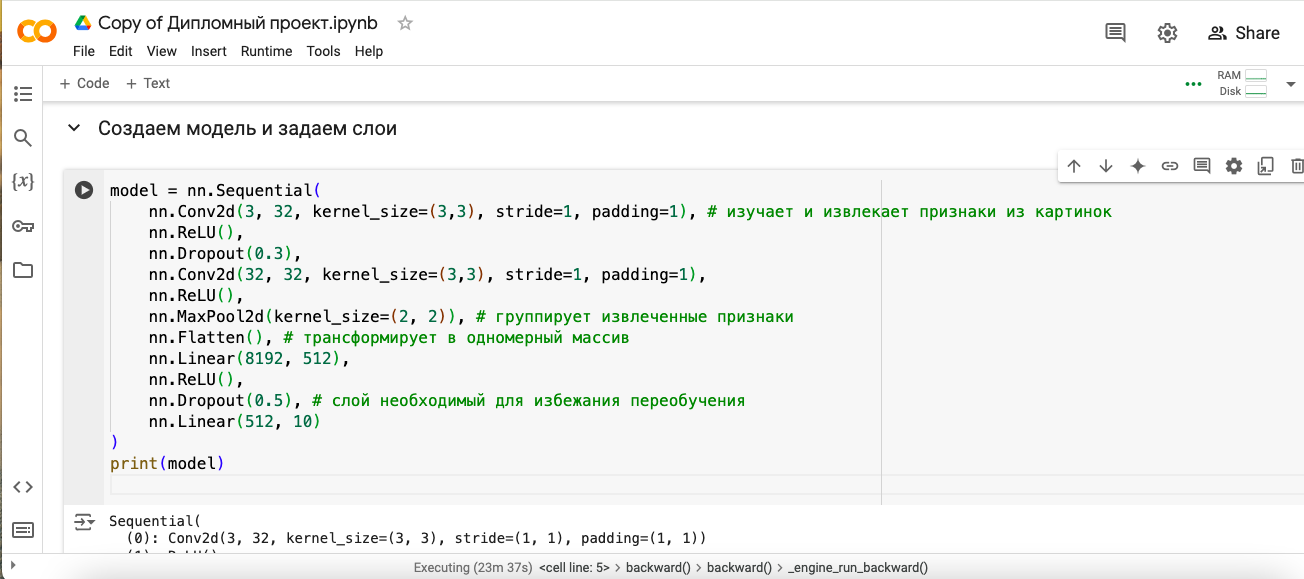


Рис. 3 Модель классификации с использованием PyTorch



Разработка моделей регрессии с библиотеками Scikit-learn, Tensorflow, PyTorch:

Рис. 4 Модель регрессии с использованием Scikit-learn

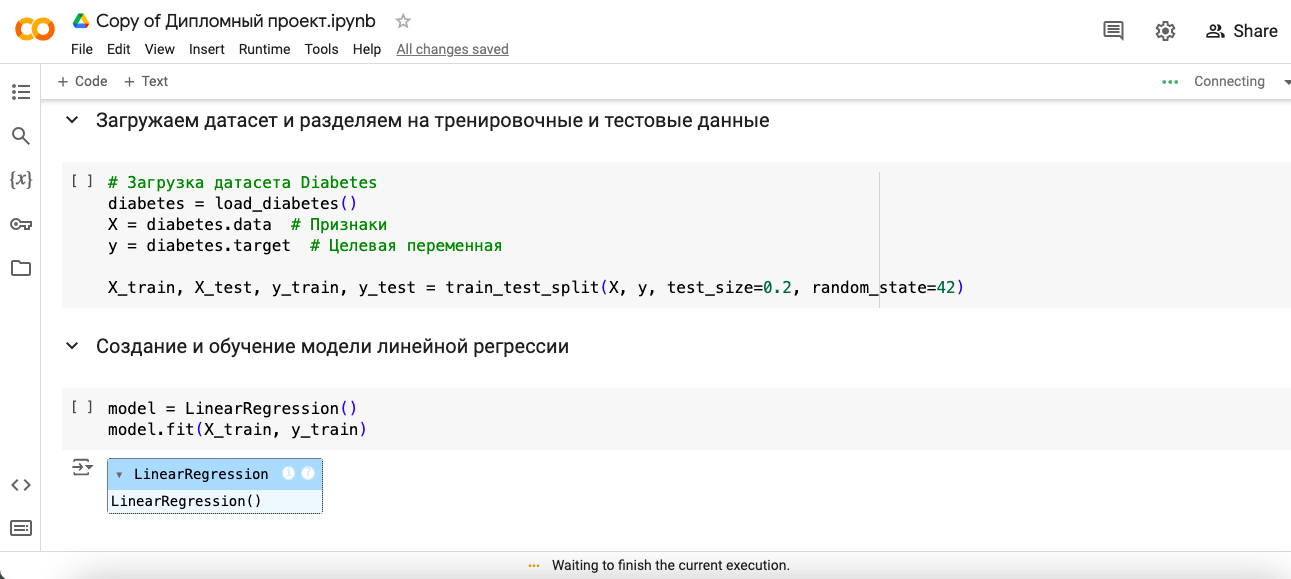


Рис. 5 Модель регрессии с использованием TensorFlow

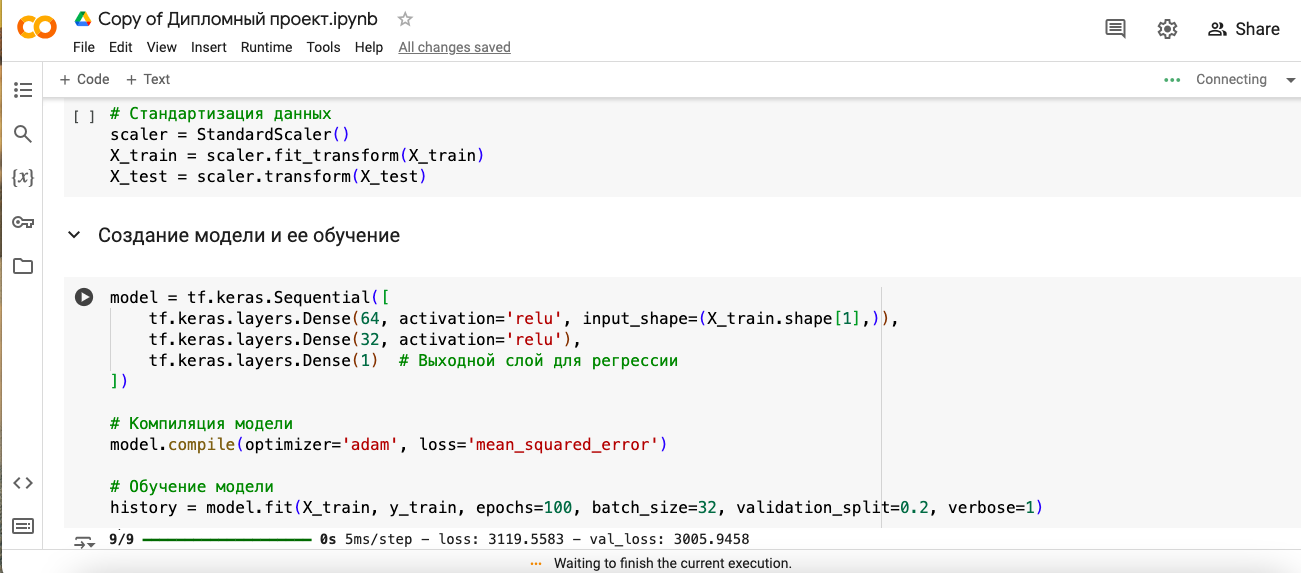
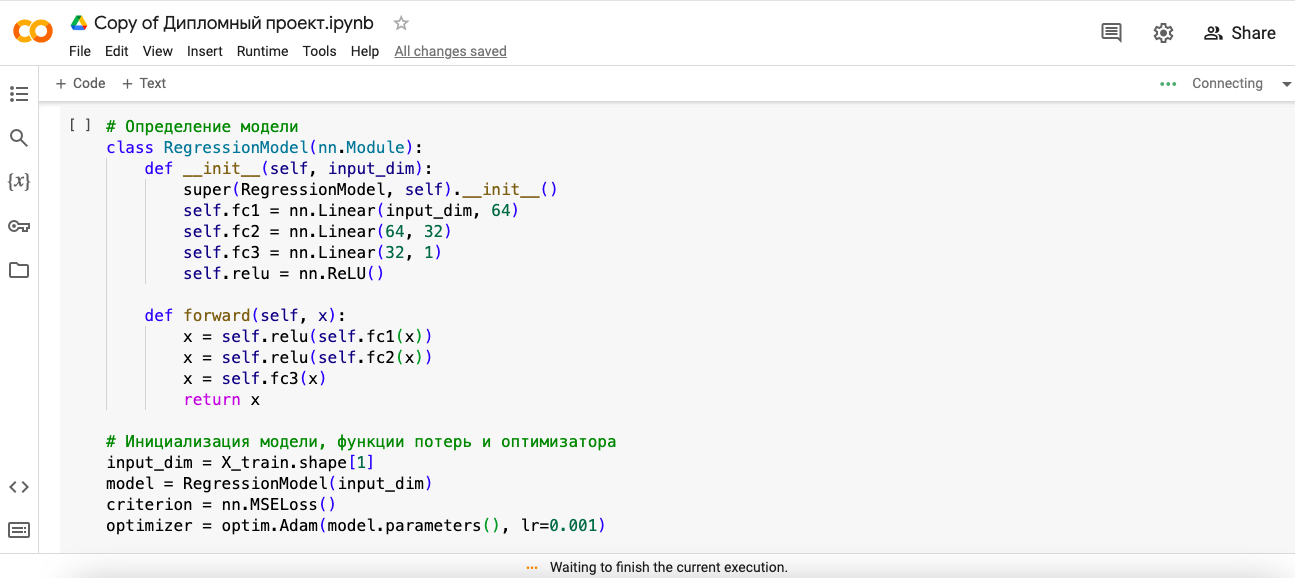


Рис. 6 Модель регрессии с использованием PyTorch



**7. Анализ и интерпретация результатов**

**Сравнение моделей**

Были протестированы различные модели, включая LogisticRegression, LinearRegression, Sequential на различных наборах данных с использованием библиотек PyTorch, TensorFlow и Scikit-learn.

**Логистическая регрессия** является одной из самых простых и понятных моделей для задач бинарной классификации. В случае мультикласса алгоритм обучения использует схему one-vs-rest (OvR). Модель остается простой и удобной в использовании.

**Модель Sequential** — это упрощённая архитектура для создания слоев один за другим (последовательно). Каждый слой получает на вход данные от предыдущего слоя и передает результат следующему..

Модель **LinearRegression** (линейная регрессия) — это один из базовых алгоритмов машинного обучения, используемый для решения задач регрессии. Ее основная цель — найти линейную зависимость между независимыми переменными (признаками) и зависимой переменной (целевой переменной).

Основные метрики, такие как достоверность, полнота, точность, f1-score, mse, mae, R^2, были использованы для оценки производительности каждой модели.

**Сравнение использованных библиотек :**

1. По назначению

* **Scikit-learn** предназначена для классического машинного обучения (обучение с учителем, без учителя, предобработка данных). Простой и удобный интерфейс для работы с алгоритмами (например, линейная регрессия, SVM, случайные леса, кластеризация). Не подходит для глубокого обучения (нейронные сети).
* **TensorFlow**: Фреймворк для глубокого обучения, но также поддерживает классическое машинное обучение. Имеет библиотеку Keras для быстрого создания моделей, а также низкоуровневый API для тонкой настройки. Подходит для создания сложных нейронных сетей, включая сверточные (CNN), рекуррентные (RNN) и трансформеры.
* **PyTorch**: Также фреймворк для глубокого обучения. Его синтаксис очень похож на синтаксис Python, что делает его популярным среди разработчиков. Подходит для создания сложных моделей, включая нейронные сети, и активно используется в академических исследованиях.

2. По простоте использования

* **Scikit-learn**: Очень проста в использовании, идеальна для начинающих.
* **TensorFlow**: Сложен в использовании, но с наличием Keras (встроенным в TensorFlow) значительно упрощает создание моделей.
* **PyTorch**: Более интуитивный и гибкий, чем TensorFlow.

3. По производительности

* **Scikit-learn**: Оптимизирована для работы на CPU. Не поддерживает GPU, что ограничивает ее применение для больших данных и глубокого обучения.
* **TensorFlow**: Поддерживает GPU и TPU (Tensor Processing Units), что делает его очень производительным для глубокого обучения. Имеет встроенные оптимизации для распределенных вычислений.
* **PyTorch**: Также поддерживает GPU и TPU. Производительность сравнима с TensorFlow.

4. Поддержка и экосистема

* **Scikit-learn**: Часть экосистемы SciPy, интегрируется с Pandas, NumPy и Matplotlib. Поддержка ограничена классическим ML.
* **TensorFlow**: Имеет мощную экосистему, включая TensorFlow Lite (для мобильных устройств), TensorFlow.js (для веба) и TensorFlow Serving (для развертывания моделей). Поддержка от Google.
* **PyTorch**: Экосистема включает TorchVision, TorchText, TorchAudio и другие инструменты. Поддержка от Facebook. PyTorch активно используется в исследованиях, включая NLP и компьютерное зрение.

5. Применение

* **Scikit-learn**: Классическое машинное обучение: классификация, регрессия, кластеризация, уменьшение размерности. Не подходит для глубокого обучения.
* **TensorFlow**: Глубокое обучение: нейронные сети, компьютерное зрение, NLP, рекомендательные системы. Промышленное развертывание и масштабирование.
* **PyTorch**: Глубокое обучение: исследования, прототипирование, NLP, компьютерное зрение. Популярен в академической среде.

**Рекомендации по выбору фреймворка в зависимости от цели**

Выбор зависит от ваших задач: если вы работаете с классическими алгоритмами, выбирайте Scikit-learn; если вам нужны нейронные сети и промышленное развертывание — TensorFlow; если вы исследователь и цените гибкость — PyTorch.

**8. Заключение**

**Обзор выполненной работы**

Проект по тестированию различных библиотек машинного обучения был выполнен успешно, все необходимые требования были соблюдены и получены соответствующие результаты.

Среди моделей классификации, с заданными параметрами и настройками, наиболее эффективной оказалась модель классификации с использованием библиотеки TensorFlow (Sequential). Обучение длилось 1 минуту и дало более точные прогнозы. Точность определения категорий на тестовых данных составила 86%. После увеличения количества эпох точность улучшилась до 87%.

Среди моделей регрессии, с заданными параметрами и настройками, наиболее эффективной оказалась модель регрессии с использованием PyTorch. Точность предсказания составила 87%.

**Дальнейшие планы**

Будет полезным создание сводной статистики по всем моделям, со всеми метриками эффективности, чтобы можно было выбирать наиболее подходящую модель на основе полных данных.

Планируется добавление новых методов предобработки данных, а также добавление дополнительных слоев и более точная настройка гиперпараметров, а также более глубокое изучение функционала PyTorch и Tensorflow для дальнейших целей машинного обучения.

## **Вывод**

Для реализации задач классификации и регрессии с использованием Scikit-learn, можно воспользоваться встроенными моделями, такими как LogisticRegression для классификации и LinearRegression для регрессии. Scikit-learn предоставляет простой и интуитивно понятный интерфейс, который позволяет быстро обучить модель с минимальными усилиями. Например, для классификации данных Fashion MNIST можно использовать всего несколько строк кода, включая загрузку данных, разделение на обучающую и тестовую выборки, обучение модели и оценку ее точности. Аналогично, для регрессии можно использовать набор данных California Housing, где процесс включает те же этапы, но с использованием модели линейной регрессии. Scikit-learn также предлагает широкий выбор метрик для оценки производительности моделей, таких как accuracy, precision, recall для классификации и MSE, MAE для регрессии.

TensorFlow, с другой стороны, предоставляет более гибкий и мощный инструментарий для создания и обучения моделей машинного обучения, особенно в контексте глубокого обучения. Для задач классификации и регрессии можно использовать API Keras, который интегрирован в TensorFlow и значительно упрощает процесс создания нейронных сетей. Например, для классификации данных MNIST можно создать простую полносвязную нейронную сеть с несколькими слоями, используя Sequential. Для регрессии можно использовать аналогичный подход, но с изменением функции потерь на MSE и выходного слоя на один нейрон. TensorFlow также поддерживает распределённые вычисления и работу с GPU, что делает его предпочтительным выбором для задач, требующих высокой производительности и масштабируемости.

PyTorch, в свою очередь, отличается динамическим графом вычислений, что делает его более гибким и удобным для исследовательских задач. Для реализации классификации и регрессии в PyTorch необходимо вручную определить архитектуру модели, функцию потерь и оптимизатор. Например, для классификации данных CIFAR-10 можно создать сверточную нейронную сеть, используя модули nn.Module и nn.Sequential. Для регрессии можно использовать аналогичный подход, но с изменением функции потерь на MSE и выходного слоя на один нейрон. PyTorch также предоставляет удобные инструменты для работы с данными, такие как DataLoader и Dataset, которые упрощают процесс загрузки и предобработки данных. Благодаря своей гибкости и простоте отладки, PyTorch часто используется в исследовательских проектах и прототипировании.

Сравнивая производительность и удобство использования этих библиотек, можно отметить, что Scikit-learn является наиболее простым и быстрым решением для задач классификации и регрессии, особенно когда речь идет о небольших наборах данных и простых моделях. TensorFlow и PyTorch, напротив, предоставляют более мощные инструменты для работы с глубокими нейронными сетями и большими объемами данных, но требуют большего уровня знаний и усилий для настройки и обучения моделей. Выбор между этими библиотеками зависит от конкретных задач, требований к производительности и предпочтений разработчика.

В заключение, можно сказать, что Scikit-learn, TensorFlow и PyTorch представляют собой три столпа современного машинного обучения, каждый из которых имеет свои уникальные преимущества и области применения. Их постоянное развитие и адаптация к новым требованиям делают их незаменимыми инструментами для разработчиков и исследователей. Независимо от того, какая задача стоит перед вами, выбор подходящей библиотеки позволит вам эффективно решить её и достичь поставленных целей.