УДК 004.852:004.4

Измайлов Егор Александрович, магистрант кафедры «Компьютерные системы и сети» (ИУ6), Московский государственный технический университет имени Н.Э. Баумана, г. Москва

СОХРАНЕНИЕ МОДЕЛЕЙ МАШИННОГО ОБУЧЕНИЯ ПРИ ИНТЕРГРАЦИИ PYTHON И C# ПРИЛОЖЕНИЙ

Аннотация

В статье рассматриваются методы сохранения обученных моделей машинного обучения при разработке прикладных систем, объединяющих Python и C# приложения. Выполнен анализ технологий сериализации моделей, используемых в экосистеме Python и .NET. Рассмотрены методы pickle, joblib, ONNX, skops.io, cloudpickle и ML.NET. Проведено сравнение подходов по критериям полноты сохранения модели, производительности, совместимости и удобства сопровождения. Особое внимание уделено интеграции моделей scikit-learn в C# приложение. В результате исследования определён наиболее подходящий способ сохранения моделей для разрабатываемой системы.

Annotation

The article discusses methods for saving trained machine learning models in the development of applied systems combining Python and C# applications. An analysis of model serialization technologies used in the Python and .NET ecosystems is carried out. The methods pickle, joblib, ONNX, skops.io, cloudpickle, and ML.NET are considered. The approaches are compared according to the criteria of model persistence completeness, performance, compatibility, and maintainability. Particular attention is paid to the integration of scikit-learn models into a C# application. As a result of the study, the most suitable method for model persistence in the developed system was determined.

Ключевые слова: машинное обучение, сериализация моделей, Python, C#, joblib, ONNX, scikit-learn.

Keywords: machine learning, model serialization, Python, C#, joblib, ONNX, scikit-learn.

Введение

Современные программные системы всё чаще используют методы машинного обучения для анализа данных и построения прогнозов. После завершения этапа обучения модель необходимо сохранить и обеспечить возможность её дальнейшего использования без повторного выполнения вычислений. Данный процесс называют сериализацией модели.

Выбор способа сохранения модели оказывает влияние на производительность системы, удобство сопровождения, совместимость программных компонентов и возможность интеграции между различными языками программирования. Особенно актуальной эта задача становится в проектах, где обучение выполняется средствами Python, а пользовательское приложение реализовано на платформе .NET.

Целью работы является анализ методов сохранения моделей машинного обучения и выбор наиболее подходящего решения для системы, объединяющей Python-компоненты и C# приложение.

Анализ предметной области

Разработка системы машинного обучения включает подготовку данных, обучение модели, оценку качества и последующее внедрение в программную среду. На этапе эксплуатации возникает необходимость сохранения результатов обучения таким образом, чтобы обеспечить быстрое восстановление модели и её дальнейшее использование.

Обученная модель представляет собой набор параметров и структур данных, сформированных в процессе обучения. Например, для линейной регрессии сохраняются коэффициенты модели и свободный член, для дерева решений – структура узлов и правила разбиения, а для ансамблевых методов – совокупность базовых алгоритмов и их параметры. В большинстве современных проектов дополнительно сохраняется pipeline предобработки данных, включающий масштабирование, кодирование признаков и другие этапы подготовки входной информации [3]. Для корректной работы системы необходимо сохранять не только саму модель, но и pipeline предобработки данных. Это особенно важно для библиотек scikit-learn, где обработка признаков и модель обычно используются совместно.

При выборе метода сохранения необходимо учитывать несколько критериев:

– полноту сохранения модели и pipeline;

– производительность при работе с большими массивами данных;

– совместимость с различными платформами;

– удобство сопровождения и обновления проекта.

Дополнительную сложность представляет интеграция Python-моделей в C# приложения. В рассматриваемой системе обучение выполняется средствами Python и библиотеки scikit-learn, а взаимодействие с пользователем реализовано через WinForms приложение на языке C#.

Методы сохранения моделей машинного обучения

В современных системах машинного обучения сохранение обученной модели является обязательным этапом разработки. После завершения процесса обучения модель должна быть сохранена таким образом, чтобы её можно было загрузить в рабочую среду без повторного выполнения вычислений. В большинстве случаев вместе с моделью сохраняются параметры предобработки данных, конфигурация pipeline и вспомогательные объекты [1].

Существует несколько подходов к сериализации моделей машинного обучения. Одни методы ориентированы исключительно на экосистему Python, другие обеспечивают межъязыковую совместимость и возможность интеграции с приложениями на C# и .NET [2]. Выбор подходящего метода зависит от архитектуры системы, требований к производительности и необходимости дальнейшего сопровождения проекта.

Библиотека Pickle

Pickle является стандартным модулем Python для сериализации объектов [6]. Он позволяет сохранять модель в файл и восстанавливать её без повторного обучения. Основными преимуществами метода являются простота использования и отсутствие необходимости установки дополнительных библиотек.

Недостатком pickle является зависимость от Python-окружения и версий библиотек. Кроме того, загрузка файлов из непроверенных источников может представлять угрозу безопасности, поскольку при десериализации возможно выполнение произвольного кода.

Библиотека Joblib

Joblib представляет собой специализированную библиотеку для сохранения объектов Python, содержащих большие массивы NumPy [1]. Данный метод широко используется совместно со scikit-learn.

Преимуществами joblib являются высокая скорость загрузки и сохранения моделей, поддержка сжатия файлов и возможность сохранения полного pipeline обработки данных. Это позволяет сохранять не только модель, но и этапы подготовки признаков.

Для рассматриваемой системы joblib является удобным вариантом, поскольку C# приложение может взаимодействовать с Python-компонентом через API или локальный сервис, не работая напрямую с внутренним форматом модели.

Формат ONNX

ONNX представляет собой универсальный формат обмена моделями машинного обучения [2]. Его основным преимуществом является независимость от конкретного языка программирования и возможность запуска модели без Python.

В C# приложениях ONNX используется совместно с ONNX Runtime. Это позволяет выполнять прогнозирование непосредственно средствами .NET.

Недостатком метода является необходимость конвертации моделей и ограниченная поддержка некоторых pipeline и пользовательских компонентов.

Библиотека Skops.io

Skops.io является альтернативным способом сохранения моделей scikit-learn с повышенным вниманием к безопасности сериализации [4]. Метод позволяет анализировать содержимое файла модели и ограничивать использование неподдерживаемых типов объектов.

Несмотря на преимущества в области безопасности, skops.io пока используется значительно реже, чем joblib, и имеет меньшую поддержку в существующих проектах.

Библиотека Cloudpickle

Cloudpickle расширяет возможности стандартного pickle и позволяет сохранять более сложные объекты Python, включая пользовательские функции и динамически создаваемые классы [7].

Метод полезен в исследовательских проектах и распределённых вычислениях, однако для типовых моделей scikit-learn его использование чаще всего является избыточным.

Платформа ML.NET

ML.NET является платформой машинного обучения для экосистемы .NET [5]. Она использует собственный формат хранения моделей и обеспечивает удобную интеграцию с C# приложениями.

Однако модели ML.NET плохо совместимы с Python-библиотеками, поэтому данный подход эффективен преимущественно в проектах, полностью реализованных на платформе .NET.

Сравнительный анализ методов сохранения

Для интеграции моделей машинного обучения в C# приложение могут использоваться несколько архитектурных подходов.

Первый вариант предполагает использование Python как отдельного сервиса. В этом случае C# приложение передаёт входные данные Python-компоненту, который загружает модель через joblib и выполняет прогнозирование.

Второй вариант основан на использовании ONNX Runtime. После конвертации модели в формат ONNX прогнозирование может выполняться непосредственно в C# приложении без запуска Python.

Третий вариант предполагает использование ML.NET и полный перенос логики машинного обучения в экосистему .NET.

Для разработки и исследования моделей машинного обучения использовалась среда Jupiter Notebook, обеспечивающая удобную работу с данными, визуализацией и тестированием. В качестве языка программирования использовался Python, это обусловлено большим количеством специализированных библиотек для анализа данных и машинного обучения. Исходя из этого, в разрабатываемой системе наиболее подходящей является первая архитектура. Однако для выбора наиболее подходящего метода сохранения обученных моделей был произведён сравнительный анализ. Его результаты представлены в таблицах 1 и 2.

Таблица 1 – Анализ преимуществ и недостатков

Метод сохранения	Основные преимущества	Основные недостатки
pickle	Встроен в стандартную библиотеку Python, прост в использовании, не требует	Имеются риски безопасности при загрузке файлов из ненадёжных источников, менее

	установки дополнительных модулей	эффективен для крупных моделей
joblib	Оптимизирован для больших массивов данных, прост в применении, широко используется в машинном обучении	Требует Python-среду для загрузки модели, ограниченная межъязыковая совместимость
ONNX	Кроссплатформенность, возможность запуска без Python, удобство интеграции в C# приложения	Процесс конвертации сложнее, поддерживаются не все модели и операции
skops.io	Повышенная безопасность сериализации, современный подход к хранению моделей	Менее распространён, ограниченная поддержка сторонних инструментов
cloudpickle	Поддерживает нестандартные объекты Python и сложную логику	Сильная зависимость от исходного кода и структуры проекта
ML.NET	Нативная интеграция с .NET, отсутствие необходимости в Python	Ограниченная совместимость с Python моделями Scikit-learn

Таблица 2 – Сравнение по критериям

Метод сохранения	Полнота сохранения модели	Производительность	Совместимость с C# и другими платформами	Удобство сопровождения

pickle	Сохраняет объект модели целиком, включая параметры и часть pipeline	Средняя скорость работы, плохо оптимизирован для больших массивов NumPy	Подходит в основном только для Python	Требует совпадения версий библиотек и Python
joblib	Полностью сохраняет модель, pipeline, параметры и объекты Scikit-learn	Высокая скорость сохранения и загрузки, поддержка сжатия, эффективная работа с NumPy	Основная работа выполняется в Python, но возможна интеграция через API или отдельный сервис	Удобен при сопровождени и проектов Scikit-learn, легко используется в реальных приложениях
ONNX	Сохраняет структуру и параметры модели, но поддержка pipeline ограничена	Высокая скорость выполнения после конвертации, хорошая оптимизация для рабочей среды	Отлично подходит для C#, .NET, C++, мобильных и облачных платформ	Требует контроля совместимости при конвертации моделей

Продолжение таблицы 2

skops.io	Сохраняет модели Scikit-learn с упором на структуру и безопасность	Производительность ниже по сравнению с joblib	В основном ориентирован на Python	Более безопасен для обмена моделями
cloudpickle	Позволяет сохранять сложные объекты, пользовательские функции и классы	Скорость ниже joblib при работе с большими данными	Используется преимущественно внутри Python	Возможны проблемы совместимости при изменении кода проекта
ML.NET	Сохраняет модели экосистемы .NET и данные обработки	Хорошая производительность внутри .NET среды	Полностью совместим с C# и платформой .NET	Удобен для сопровождения C# приложений

Сравнительный анализ показал, что наиболее подходящим решением для рассматриваемой системы является библиотека joblib [1]. Данный метод обеспечивает:

– полное сохранение модели и pipeline;

– высокую производительность при работе с массивами NumPy;

– простую интеграцию со scikit-learn;

– удобство практического использования.

Формат ONNX обеспечивает лучшую совместимость с платформой .NET и позволяет выполнять модель без запуска Python-интерпретатора [2]. Однако использование данного подхода требует дополнительной конвертации моделей, проверки корректности работы после преобразования и подготовки входных данных в формате, поддерживаемом ONNX Runtime. Кроме того, не все pipeline и пользовательские компоненты scikit-learn корректно конвертируются в универсальный формат ONNX. Использование ML.NET

также не является оптимальным решением, поскольку обучение модели выполняется средствами Python.

Заключение

В ходе работы были рассмотрены основные методы сохранения моделей машинного обучения и выполнен их сравнительный анализ. Исследование

показало, что при использовании Python и библиотеки scikit-learn наиболее практичным решением является библиотека joblib.

Данный подход обеспечивает полное сохранение модели и pipeline предобработки, высокую производительность и удобную интеграцию с Python-компонентами системы. Использование joblib позволяет разделить этапы обучения и эксплуатации модели, а также упростить сопровождение программного обеспечения.

Список использованных источников

1. Документация Joblib [Электронный ресурс]. – URL: https://joblib.readthedocs.io/en/stable/ (дата обращения: 25.04.2026).

2. Документация ONNX Runtime [Электронный ресурс]. – URL: https://onnxruntime.ai/ (дата обращения: 27.04.2026).

3. Документация scikit-learn [Электронный ресурс]. – URL: https://scikit-learn.org/stable/model_persistence.html (дата обращения: 28.04.2026).

4. Документация skops.io [Электронный ресурс]. – URL: https://skops.readthedocs.io/en/stable/persistence.html (дата обращения: 28.04.2026).

5. Документация ML.NET [Электронный ресурс]. – URL: https://learn.microsoft.com/ru-ru/dotnet/machine-learning/ (дата обращения: 25.04.2026).

6. Документация pickle [Электронный ресурс]. – URL: https://docs.python.org/3/library/pickle.html (дата обращения: 28.04.2026).

7. Документация Cloudpickle [Электронный ресурс]. – URL: https://github.com/cloudpipe/cloudpickle (дата обращения: 28.04.2026).

References

1. Joblib. Official documentation [Electronic resource]. – URL: https://joblib.readthedocs.io/en/stable/ (date of access: 25.04.2026).

2. ONNX Runtime. Official documentation [Electronic resource]. – URL: https://onnxruntime.ai/ (date of access: 27.04.2026).

3. scikit-learn. Model persistence documentation [Electronic resource]. – URL: https://scikit-learn.org/stable/model_persistence.html (date of access: 28.04.2026).

4. skops.io. Model persistence documentation [Electronic resource]. – URL: https://skops.readthedocs.io/en/stable/persistence.html (date of access: 28.04.2026).

5. ML.NET. Official documentation [Electronic resource]. – URL: https://learn.microsoft.com/ru-ru/dotnet/machine-learning/ (date of access: 25.04.2026).

6. Python Software Foundation. pickle documentation [Electronic resource]. – URL: https://docs.python.org/3/library/pickle.html (date of access: 28.04.2026).

7. Cloudpickle. Official documentation [Electronic resource]. – URL: https://github.com/cloudpipe/cloudpickle (date of access: 28.04.2026).