ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Review Article

Open Access

Developing AI and ML Solutions with ML.Net

Naga Lalitha Sree Thatavarthi

USA

ABSTRACT

Emerging as game-changing technologies, artificial intelligence (AI) and machine learning (ML) have the power to upend a number of industries and inspire creativity. This study offers an extensive analysis of how AI and ML affect innovation. It looks at how various facets of innovation, such as business models, process optimisation, and product development, have been impacted by these technologies. The framework known as ML.NET was created by Microsoft ten years ago with the goal of easing the integration of machine learning models into sizable software programmes. This article presents the framework. We outline its design as well as the application requirements that influenced it. With regard to ML.NET, we specifically describe DataView, the central data abstraction that enables it to reliably and quickly capture whole prediction pipelines throughout the training and inference lifespan. After comparing ML.NET's performance against that of more recent arrivals, we conclude the research with a surprisingly positive analysis and a discussion of key lessons learned.

***Corresponding author**

Naga Lalitha Sree Thatavarthi, USA

Received: March 16, 2022; **Accepted:**March 23, 2022; **Published:** March 27, 2022

Keywords: Artificial intelligence, Machine learning, ML.NET

Introduction

Since machine learning (ML) is an artificial intelligence (AI) application, it is crucial to concentrate on developing systems and models that can access Big Data sets that are currently accessible. The system will then automatically adjust, learn, and improve predictability and performance through its settings to maximise and improve user experience and guarantee the efficacy of predictive analyses.

Furthermore, ML is a subfield of AI since all ML is an element of AI, but not all AI is an ML. The field of computer science known as machine learning (ML) aims to create and run preexisting algorithms to create generalised models that provide precise patterns and forecasts. ML algorithms, which are based on statistical and mathematical models and use historical data, enable machines to emulate human behaviour.

ML has grown significantly in recent years because to new computer technologies; yet, as many ML algorithms have been around for a while, many of their uses have remained unknown, AI needs the inventiveness of eminent data scientists and engineers to advance. Since humans can already do complicated jobs without the need for intelligence, AI and ML are not new fields; rather, data science's tools have helped them advance to a new level of success. As a result, human emotive decisions carry dangers and responsibility.

Looking back, a number of very large companies began to show interest in the development of artificial intelligence and to put significant effort into it. Seizing the chance, the Japanese government has announced intentions to supply 5th generation computers to support machine learning. Following that, IBM's Deep Blue computer, which was first to surpass all other computers in terms of information holding and interpretation in 1997, has made it possible for organisations to keep a large amount of data. And in the last 15 years, major international players like Google, Amazon, Baidu, and others have developed their profit-making capabilities by utilising AI and ML.

This paper presents ML.NET, a machine learning framework that enables developers to create and implement intricate AI/ML pipelines in their applications using cutting-edge machine learning models and data feature generators. An application can easily add a model by importing the ML.NET runtime and connecting the input/output data sources. Pipelines that have been developed and trained using ML.NET can be surfaced for prediction without any modification. In fact, training and prediction share the same code paths. Since ML.NET has been used by thousands of Microsoft data scientists and developers over the past ten years to incorporate machine learning models into hundreds of millions of goods and services, the platform's capacity to capture complete, end-to-end pipelines has been proven.

Large-scale artificial intelligence and machine learning are supported by ML.NET because of an internal design that incorporates concepts from relational database management systems and is represented by DataView, the platform's primary abstraction. In datasets larger than main memory, DataView can handle high dimensional data with grace and efficiency while providing compositional processing of schematized data. A DataView is an immutable, lazy-evaluated result of computations over one or more base tables or views, similar to views in relational databases (until it is compelled to materialise, for example, when several runs over the data are requested). Working sets can be larger than main memory because to DataView's internal streaming access to data.

Capabilities

- **Support for a Wide Range of Algorithms:** ML.NET is capable of handling a number of machine learning applications, including as clustering, regression, classification, anomaly detection, and more.
- **Model Interpretability:** A key component for several businesses, ML.NET emphasises understanding how models function and make decisions.
- **Integration with Different Frameworks:** TensorFlow, LightGBM, and ONNX deep learning models may be utilised by ML.NET, enabling developers to go beyond ML.NET's built-in features and make use of pre-trained models.
- **Cross-Platform:** ML.NET is cross-platform because it is a component of the.NET Core ecosystem. This implies that you can use Linux, macOS, or Windows for development.
- **New Models using C# or F#:** Using well-known.NET languages, developers can construct new machine learning models straight away.

Limitations

- **Still Developing:** ML.NET is a more recent framework for machine learning than several well-known ones, such as TensorFlow or scikit-learn, while being more potent. Fewer features and less widespread community support may result from this.
- **Performance Overheads:** Although ML.NET provides a lot of useful features, there may be trade-offs in terms of performance when utilising frameworks that are more closely aligned with hardware.
- **Limited Deep Learning Support:** Compared to TensorFlow or PyTorch, ML.NET doesn't have as robust a native deep learning ecosystem. Even while both frameworks allow you to import models, ML.NET might not be the most effective tool for creating intricate neural networks from scratch.
- **Smaller Community:** The ML.NET community is smaller because of its relative novelty and specialisation. When compared to more well-known frameworks, this could mean less web resources, tutorials, or third-party tools.

Reasons to Implement

When Artificial Intelligence (AI) and Machine Learning (ML) are combined with ML.NET applications, software capabilities are increased to unprecedented heights.

- Enhanced precision and effectiveness AI and ML are used to increase the productivity and accuracy of ML.NET applications. Because complex algorithms are always learning and changing, they can eventually complete tasks with greater precision.
- An improved experience for users The combination of AI and ML enables a simple, customised user experience. Through the use of these technologies, ML.NET apps can tailor interactions to the preferences, behaviour, and habits of their users, resulting in an interface that is more intuitive and captivating.
- Maintenance that is Predictive Integrating AI and ML with ML.NET allows for the possibilities of predictive maintenance. The capacity of applications to anticipate issues and inefficiencies allows for proactive measures, reduces downtime, and enhances overall system performance.
- Fraud identification One area where AI and ML shine is in pattern and anomaly identification. Because they swiftly spot irregularities and enhance security procedures to stop fraudulent activity, these technologies are very useful in detecting fraud in. MLNET apps.
- Advanced Information AI and ML integration into ML.NET apps opens the door to advanced analytics. These tools analyse large-scale datasets, providing meaningful analysis and empowering businesses to make data-driven decisions that boost overall productivity and competitiveness.
- Artificial intelligence and chatbots It is simple to connect AI-powered chatbots and virtual assistants with ML.NET apps to provide users with engaging and responsive user interfaces. These intelligent agents facilitate a more engaging user experience through more effective communication, prompt inquiry responses, and streamlined user interactions. Employ ML.NET developers to maximise integration capabilities.

Related Work

Designed to be a Python machine learning tool, Scikit-learn primarily targets interactive use cases over datasets that fit into main memory. Scikit-learn's characteristic makes it less suitable for Big Data experimentation in a number of ways: runtime speed is frequently insufficient; huge datasets and feature sets are not supported; datasets can only be retrieved in batch from main memory rather than streaming. Python's global interpreter lock prevents multi-core processing from working natively, while some work has been done to address this problem for computations that are embarrassingly parallel, like cross-validation or tree ensemble models [1]. The DataView abstraction and a few additional database-inspired ML.NET approaches enable it to tackle the previously listed issues. Nevertheless, ML.NET offers NimbusML-based Sklearnlike Python bindings, making it simple for customers accustomed to the former to transition to ML.NET.

A few machine learning systems based on Scikit-learn constraints are MLLib [2]. Though they are not "data science languages," MLLib is enterprise-level environments for creating machine learning models for applications, in contrast to Scikit-learn and comparable to ML.NET. With the major focus being on in-memory distributed computation (based on Apache Spark), these JVM-based systems offer performance for huge datasets. In contrast, the primary objective of ML.NET is effective computing on a single machine.

Methodology

With the help of the.NET machine learning toolkit ML.NET, developers may create intricate pipelines for machine learning, assess them, and use the results straight for prediction. ML models that can be stacked or formed into ensembles are the next phase in pipeline construction, which typically consists of several transformation stages that feature and alter the raw input data. In actuality, they are Direct Acyclic Graphs (DAGs) of operators, therefore the term "pipeline" is a little misleading. Microsoft created the open-source, cross-platform machine learning framework ML.NET. It makes it possible for.NET developers to include machine learning into their apps without needing to learn a new programming language or possess specialised knowledge in the area.

Modules

Data

- **Data Loading:** There are multiple ways to input data using ML.NET, from real-time sources to files (such CSV and TSV). Next, the information is shown in the IDataView format, which is a versatile and effective means of presenting huge datasets.
- **Data Transformation:** It's rare that raw data is available in a training-ready shape. To assist prepare and shape your data for the best model training, ML.NET offers a range of transformations, from text processing to normalisation.

Model

- **Method Selection:** The selection of a suitable method is contingent upon the nature of the problem (classification, regression, clustering, etc.). Deep neural networks to linear regressors are just a few of the options provided by ML.NET.
- **Model Definition:** entails building a pipeline that details the selected algorithm and the data transformations.

Training

Training comes next, after the model is defined and the data is ready. In order to minimise the discrepancy between the model's predictions and actual results, this procedure entails feeding the model data and modifying its internal parameters. Using the Fit() method on your established pipeline and providing your training data is how you would use ML.NET. As a result, a trained model is prepared for assessment and application.

Evaluation

It's crucial to assess the model's performance using unobserved data after training. By doing this, it is ensured that the model generalises successfully to new and novel data rather than just memorising the training set. ML.NET provides different metrics based on the task. You may examine accuracy, F1 score, or log loss, for instance, in classification issues. Metrics like mean absolute error or root mean squared error would be better suited for regression.

Advanced Use-Cases: ML.NET-Based Recommendation Systems

Recommendation systems have been used in many different contexts, such as product recommendations on e-commerce websites and movie suggestions on streaming services. Developers in the.NET ecosystem may construct these systems more easily thanks to the tools that ML.NET offers.

Overview of Recommendation Frameworks

The goal of recommendation systems is to make recommendations to consumers about products based on a variety of criteria, including user-to-user similarities, item similarities, and behaviour.

Mainly, there are two kinds:

- Filtering collaboratively based on user activity. It is likely that item 4 would be appreciated by user A and item 1 by user B if user A enjoys items 1, 2, and 3 and user B like items 2, 3, and 4.
- Using item attributes, content-based filtering is applied. It would be advised to watch more action films if the user enjoys them.

Conclusion

In the current application development process, machine learning is quickly moving from a specialised subject to a fundamental component. This brings up a lot of the early difficulties Microsoft had. A core set of these are addressed by ML.NET: it offers the scalability required to work on datasets of any size across a wide range of devices and environments; most importantly, it enables the efficient drafting and sharing of entire pipelines. It also brings machine learning onto the same technological stack as application development. It is no coincidence that ML.NET has these qualities; thousands of Microsoft data scientists have requested and provided insights on how to utilise it to build hundreds of services and products that are used on a daily basis by hundreds of millions of people globally [3-10].

Future Scope

Future research may compare several machine learning techniques in each of these financial domains, or it may concentrate on a particular application or domain. Grossus modo, our results are mapped out and cited in the part that follows, providing scholars with crucial data to assist them in their research into the growth of AI and ML applications in finance, emphasising the main area of their extensive array of advantages.

References

1. (2018) Joblib Documentation. Read the docs http://media.readthedocs.org/pdf/joblib/latest/joblib. pdf.
2. Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, et al. (2016) MLlib: Machine Learning in Apache Spark. JMLR 1-7.
3. Galen Andrew, Jianfeng Gao (2007) Scalable training of L1-regularized loglinear models. ICML '07: Proceedings of the 24th international conference on Machine learning 33-40.
4. (2019) Boost.Python. Wiki http://wiki.python.org/moi/boost.python.
5. (2018) Caffe2. http://caffe2.ai/.
6. (2014) Kaggle Challenge. Criteo http://labs.criteo.com/2014/02/ kaggle-display-advertising-challenge-dataset/.
7. Christopher Olston, Fangwei Li, Jeremiah Harmsen, Jordan Soyke, Kiril Gorovoy, et al. (2017) TensorFlow-Serving: Flexible, High-Performance ML Serving. In Workshop on ML Systems at NIPS https://research.google/pubs/tensorflow-serving-flexible-high-performance-ml-serving/.
8. Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J Franklin, Joseph E Gonzalez, et al. (2017) Clipper: A Low-Latency Online Prediction Serving System. Usenix https://www.usenix.org/system/files/conference/nsdi17/nsdi17-crankshaw.pdf.
9. Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, et al. (2016) TensorFlow: A system for large-scale machine learning. OSDI 265-283.
10. Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, et al. (2015) MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. Arvix https://www.arxiv.org/abs/1512.01274.