.NET 人工智能的故事

自我介绍

Kinfey Lo – (卢建晖)
Microsoft Cloud Advocate

前微软MVP、Xamarin MVP和微软RD，拥有超过10年的云原生、人工智能和移动应用经验，为教育、金融和医疗提供应用解决方案。 Microsoft Iginte，Teched 会议讲师，Microsoft AI 黑客马拉松教练，目前在微软，为技术人员和不同行业宣讲技术和相关应用场景。

爱编程(Python， C# , TypeScript , Swift , Rust , Go ）

专注于人工智能，云原生，跨平台移动开发

Github : https://github.com/lokinfey

Email : kinfeylo@microsoft.com **Blog :** https://blog.csdn.net/kinfey

Developer Introduce

机器学习？深度学习？人工智能？

人工智能

是一种让计算机能够模仿人类智能的技术。
它包括机器学习

机器学习

是人工智能的一部分，它包括让计算机能够依靠经验更好地处理任务的多项技术

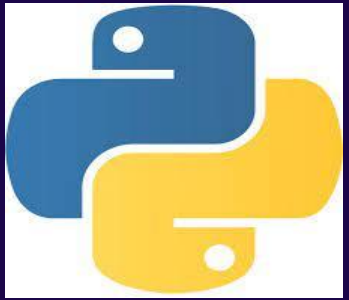
深度学习

又是机器学习的一部分，它以人工神经网络为基础，让计算机能够自我训练。

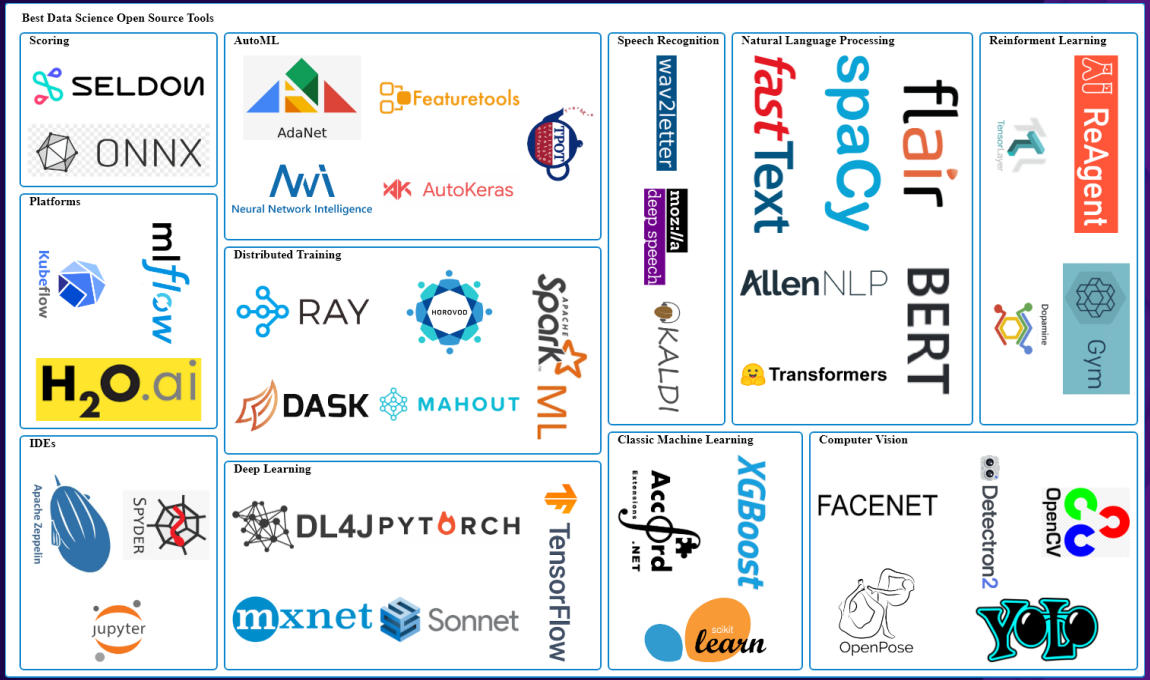
1950's 1960's 1970's 1980's 1990's 2000's 2010's

云原生开发挑战赛

人工智能主要技术栈



Python



云原生开发挑战赛

.NET 程序员如何做 AI ？能投入到实际生产吗?

开发环境的变化

.NET Interactive

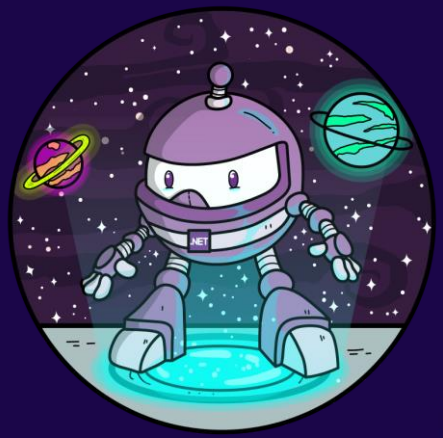
.NET Interactive 利用 .NET 的强大功能并将其嵌入到您的交互体验中。 以前所未有的方式在您的应用中共享代码、探索数据、编写和学习。

Notebook： Jupyter、nteract 和 Visual Studio Code

代码机器人

树莓派设备

可嵌入的脚本引擎REPL



云原生开发挑战赛

开发框架 – ML.NET

ML.NET

用于构建自定义 ML 模型的机器学习框架

技能重用
C# and F#

大规模实践
Azure, Office, Windows

定义ML模型更容易
Automated ML 和 Tools (Model Builder and CLI)

兼容性，扩展性
TensorFlow, ONNX and Infer.NET

免费，跨平台，开源

云原生开发挑战赛

ML.NET 相关组件

Developer friendly API for Machine Learning

ML Model Training & Consumption

Data
- IDataView (Dataset)
- File Loaders
- Database Loader
- Image Loader
- Data Transforms

Training ML Tasks

Classical ML
- Classification
- Regression
- Anomaly Detection
- Recommendation
- Time Series
- Ranking
- Clustering

Computer Vision
- Image Classification
- Object Detection (*)

(40 trainers/algorithms)

Model Consumption & Evaluation

Consumption
- Model
- Prediction Engine
- Prediction Engine Pool
- ONNX Export

Evaluation
- Model Evaluators
- Quality Metrics

Extensions & Tools
- ONNX Consumption
- TensorFlow Consumption
- AutoML
- CLI (Command Line Interface)
- Model Builder in VS
- Jupyter Notebooks

云原生开发挑战赛

ML.NET 相关组件

Extensions & Tools

- ONNX Consumption
- TensorFlow Consumption
- AutoML
- CLI (Command Line Interface)
- Model Builder in VS
- Jupyter Notebooks

使用预训练的 **ONNX** 和 **TensorFlow** 模型

支持更多机器学习场景（例如图像分类和对象检测）

调查 **PyTorch**

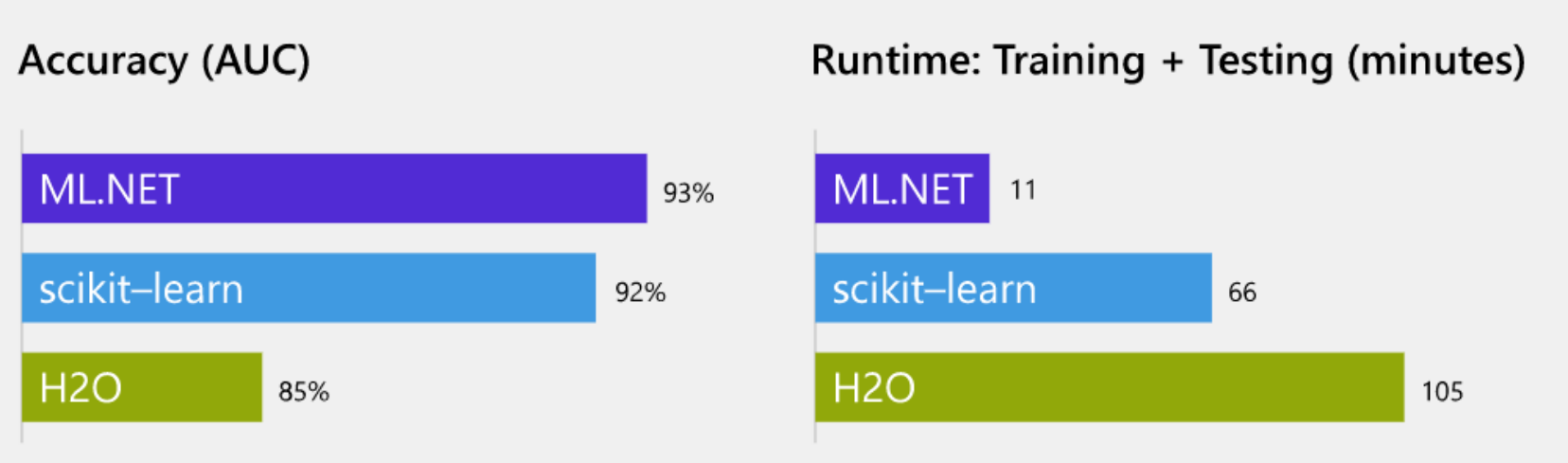
工装

使用 **AutoML** 选择最佳模型的模型构建器和 **CLI**

可以在 Jupyter Notebooks 中运行 .NET/ML.NET 代码

ML.NET 性能比较

ML.NET 在性能的实验评估* 中优于其他框架（更高的准确性和更低的运行时间）
在使用完整的 9GB 数据集进行训练时，scikit-learn 和 H2O 都会引发溢出/内存错误



使用约 900 MB 的 9GB 亚马逊评论数据集进行情绪分析的指标。

云原生开发挑战赛

ML.NET 支撑本地 到云的解决方案

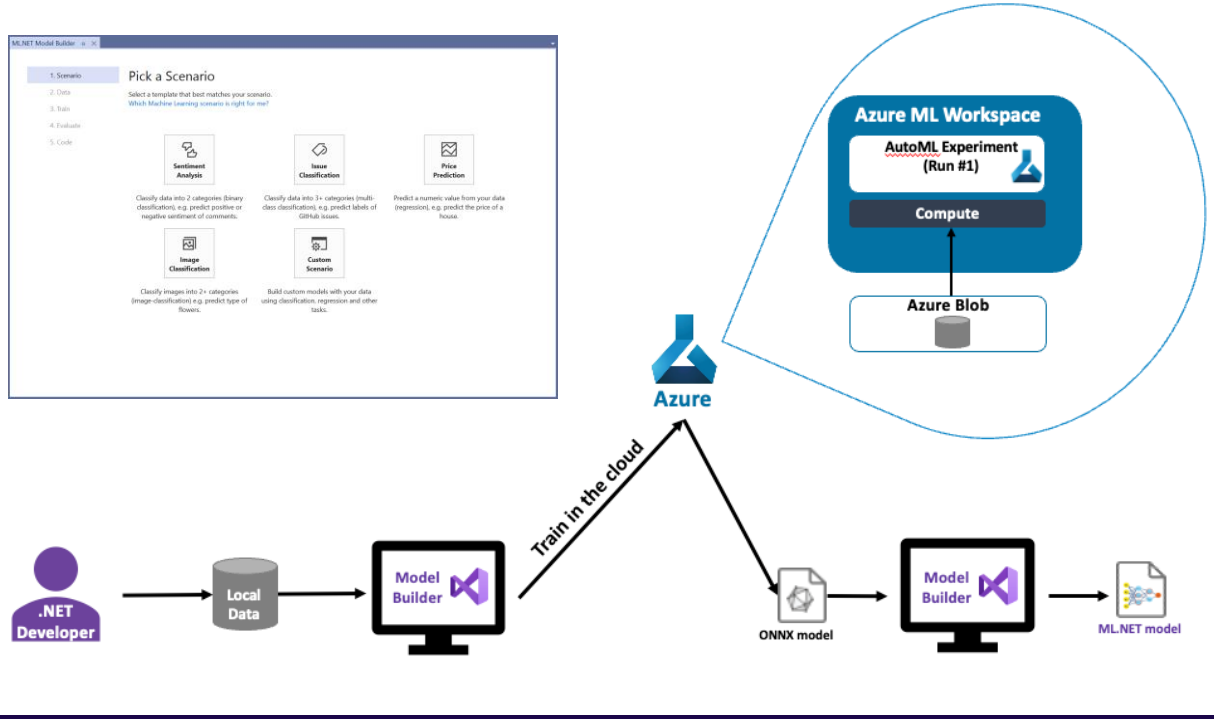
本地

用户使用本地 AutoML 在 Model Builder 中本地训练 ML 模型，以“尝试体验”、原型化并获得构建模型的信心

Azure

用户在模型生成器中通过简单、无缝的体验将培训扩展到 Azure

ML.NET Modoel Builder + Azure ML 的结合



云原生开发挑战赛

深度学习框架呢

TensorFlow.NET

https://github.com/SciSharp/TensorFlow.NET

Install TF.NET and TensorFlow binary through NuGet.

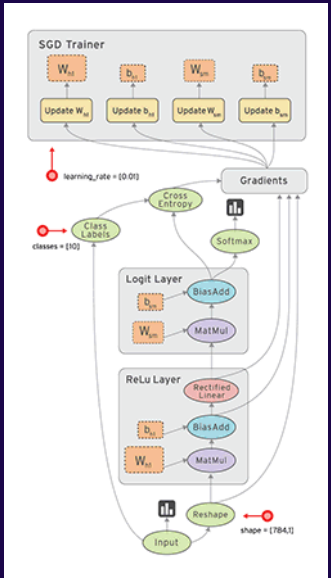
```
### install tensorflow C#/F# binding
PM> Install-Package TensorFlow.NET
### install keras for tensorflow
PM> Install-Package TensorFlow.Keras

### Install tensorflow binary
### For CPU version
PM> Install-Package SciSharp.TensorFlow.Redist

### For GPU version (CUDA and cuDNN are required)
PM> Install-Package SciSharp.TensorFlow.Redist-Windows-GPU
```

Import TF.NET and Keras API in your project.

```
using static Tensorflow.Binding;
using static Tensorflow.KerasApi;
using Tensorflow;
using NumSharp;
```

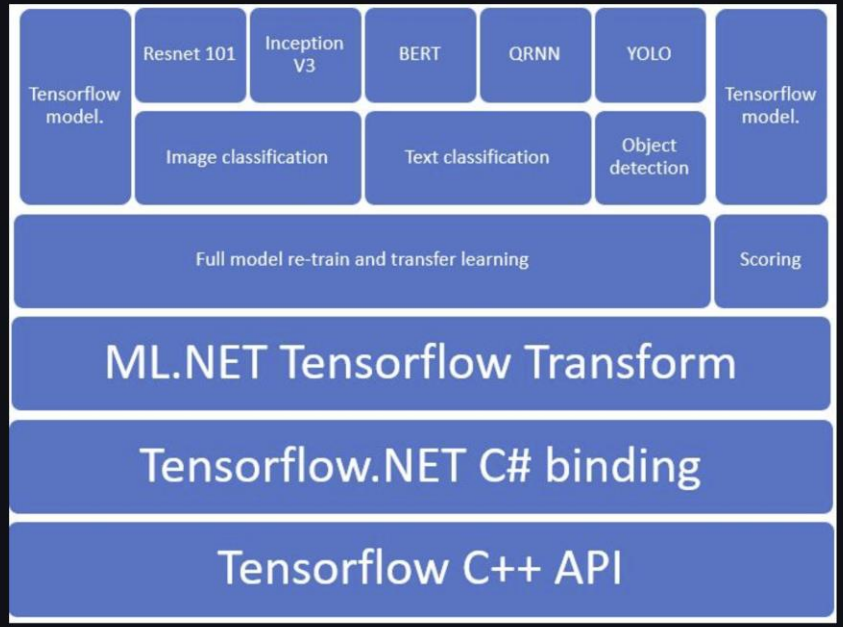


云原生开发挑战赛

TensorFlow.NET 与 ML.NET

Solution

ML .NET does not have a DNN training infrastructure so we plan to use Tensorflow in the backend through the C# bindings created by Tensorflow .NET.



云原生开发挑战赛

TorchSharp

https://github.com/dotnet/TorchSharp

TorchSharp 是一个 .NET 库，可提供对支持 PyTorch 的库的访问。 它是 .NET 基金会的一部分。重点是将 libtorch 提供的 API 与张量相关联。 设计意图是尽可能接近 Pytorch 体验，同时仍然在有意义的地方利用 .NET 静态类型系统的优势。 例如：当 Pytorch 为特定参数定义多个有效类型时，依赖于方法重载。

```
using TorchSharp;
using static torch.nn;

var lin1 = Linear(1000, 100);
var lin2 = Linear(100, 10);
var seq = Sequential(("lin1", lin1), ("relu1", ReLU()), ("drop1", Dropout(0.1)), ("lin2", lin2));

var x = torch.randn(64, 1000);
var y = torch.randn(64, 10);

var optimizer = torch.optim.Adam(seq.parameters());
var loss = functional.mse_loss(Reduction.Sum);

for (int i = 0; i < 10; i++) {
    var eval = seq.forward(x);
    var output = loss(eval, y);

    optimizer.zero_grad();

    output.backward();

    optimizer.step();
}
```

云原生开发挑战赛

Download

TorchSharp is distributed via the NuGet gallery: https://www.nuget.org/packages/TorchSharp/

To use TorchSharp, you also need one of the LibTorch backend packages: https://www.nuget.org/packages?q=libtorch, specifically one of

- `libtorch-cpu-linux-x64` (CPU, Linux)
- `libtorch-cpu-win-x64` (CPU, Windows)
- `libtorch-cpu-osx-x64` (CPU, OSX)
- `libtorch-cpu` (CPU, references all three, larger download but simpler)
- `libtorch-cuda-11.3-linux-x64` (CPU/CUDA 11.3, Linux)

  > NOTE: Due to the presence of very large native binaries, using the `libtorch-cuda-11.3-linux-x64` package requires .NET 6, e.g. .NET SDK version `6.0.100-preview.5.21302.13` or greater.

- `libtorch-cuda-11.3-win-x64` (CPU/CUDA 11.3, Windows)

Alternatively you can access the libtorch native binaries via direct reference to existing local native binaries of LibTorch installed through other means (for example, by installing PyTorch using a Python package manager). You will have to add an explicit load of the relevant native library, for example:

```
using System.Runtime.InteropServices;
NativeLibrary.Load("/home/gunes/anaconda3/lib/python3.8/site-packages/torch/lib/libtorch.so")
```

云原生开发挑战赛

TF.NET vs TorchSharp

应用模型

如何在桌面端或手机端应用中使用模型

模型文件中不仅包含了所有的权重矩阵，还记录了该神经网络的数据流图。这样不同平台、不同语言都可以根据模型文件里的信息构建网络、加载权重矩阵、执行前向计算。

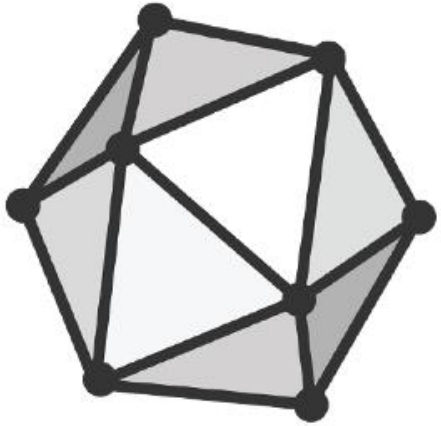
数据科学家

机器学习工程师

ONNX

OPEN NEURAL NETWORK EXCHANGE

ONNX是一个开放式的规范，定义了可扩展的计算图模型、标准数据类型以及内置的运算符。该文件在存储结构上可以理解为是一种层级的结构



ONNX Runtime – Multi-language API (C#)

C#

```
SessionOptions options = new SessionOptions();
var session = new InferenceSession("model.onnx", options))
var inputMeta = session.InputMetadata;
var container = new List<NamedOnnxValue>();
// Prepare input data – container.
// Run the inference
var results = session.Run(container);
```

Example reference:
https://github.com/microsoft/onnxruntime/blob/master/csharp/sample/Microsoft.ML.OnnxRuntime.InferenceSample/Program.cs

关注51Aspx

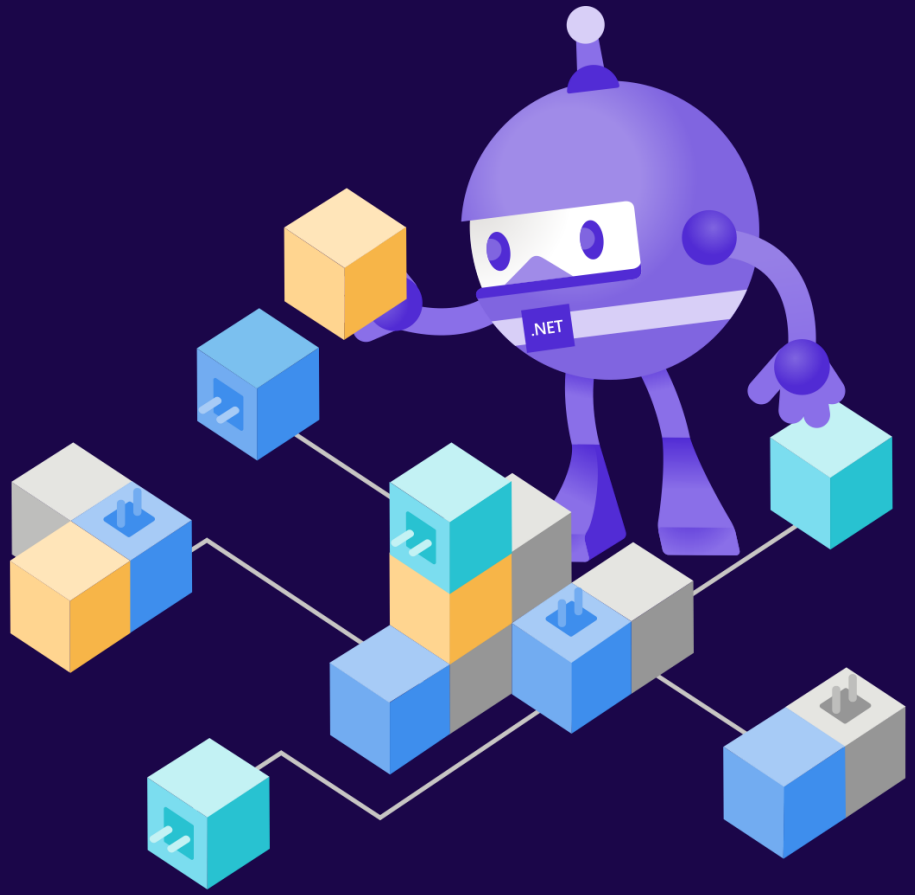
51Aspx
源码服务专家

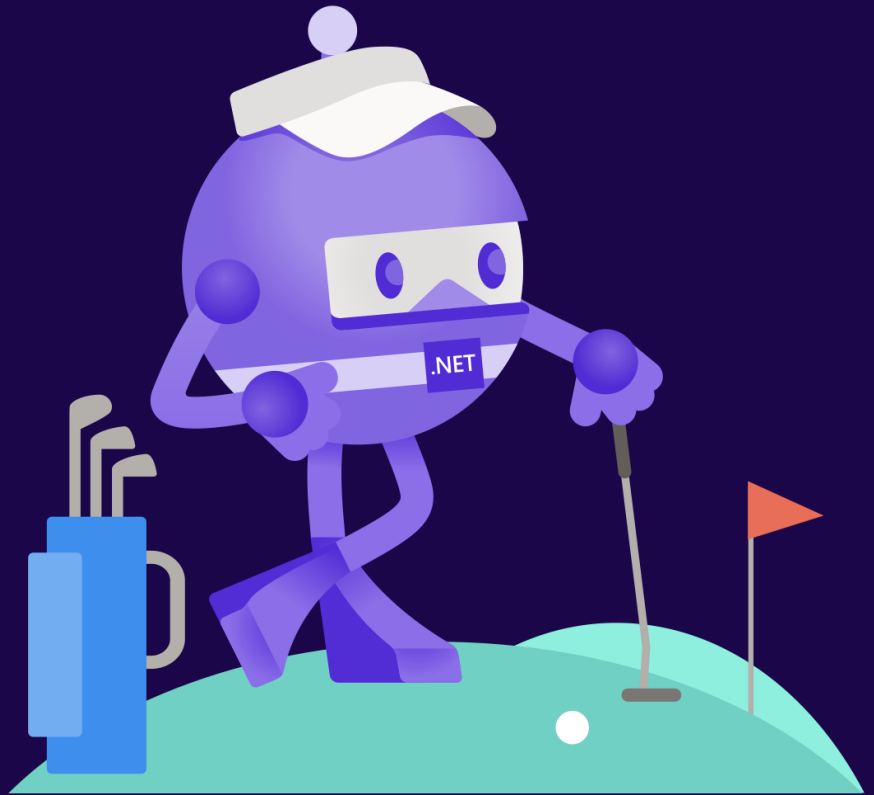
感谢收看

大赛详情链接

https://club.51aspx.com/netscec/

扫码报名



总结

谢谢