Kharlamov Mikhail, 24B-44

ML.NET

Что это такое и с чем его едят?

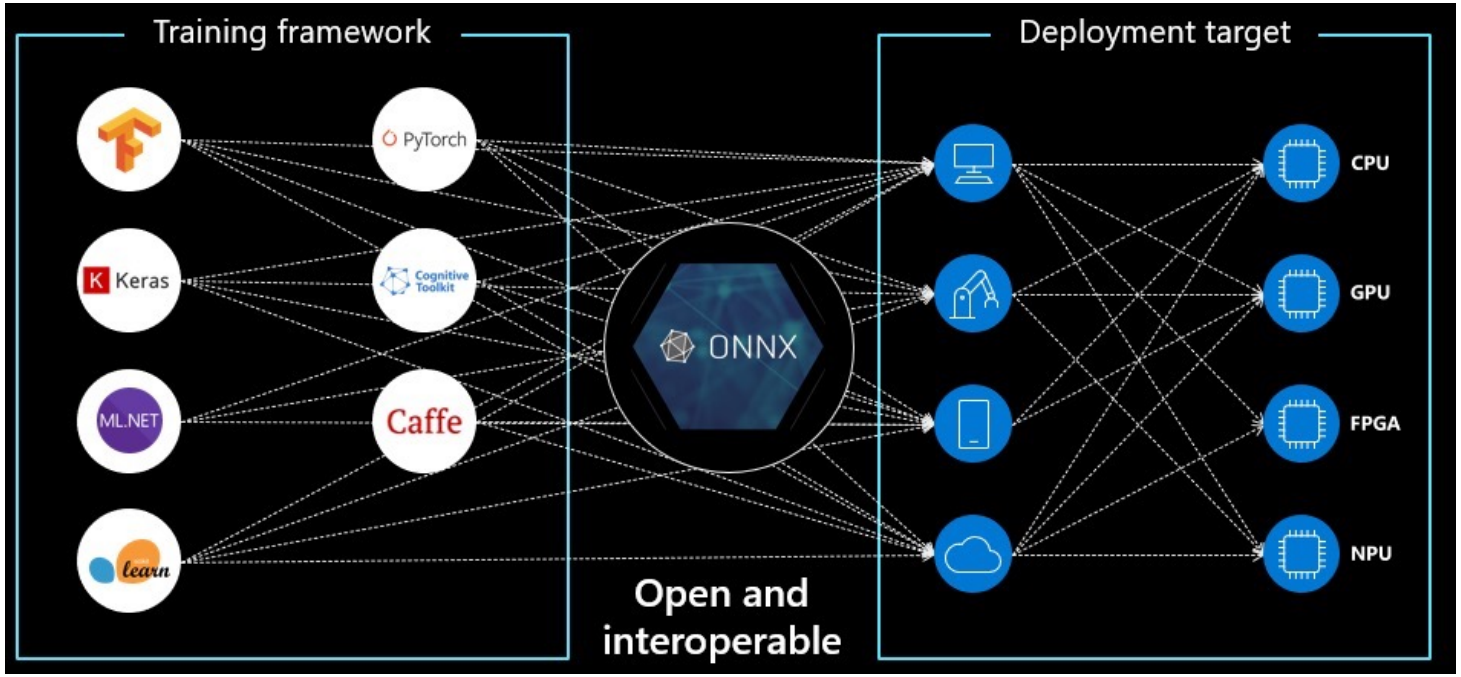
- ML.NET – это бесплатная открытая библиотека со средствами машинного обучения для языков программирования C# и F#.
- Изначально поддерживала только элементарные методы машинного обучения.
- На данный момент поддерживает классический ML, ожидается поддержка глубокого обучения.

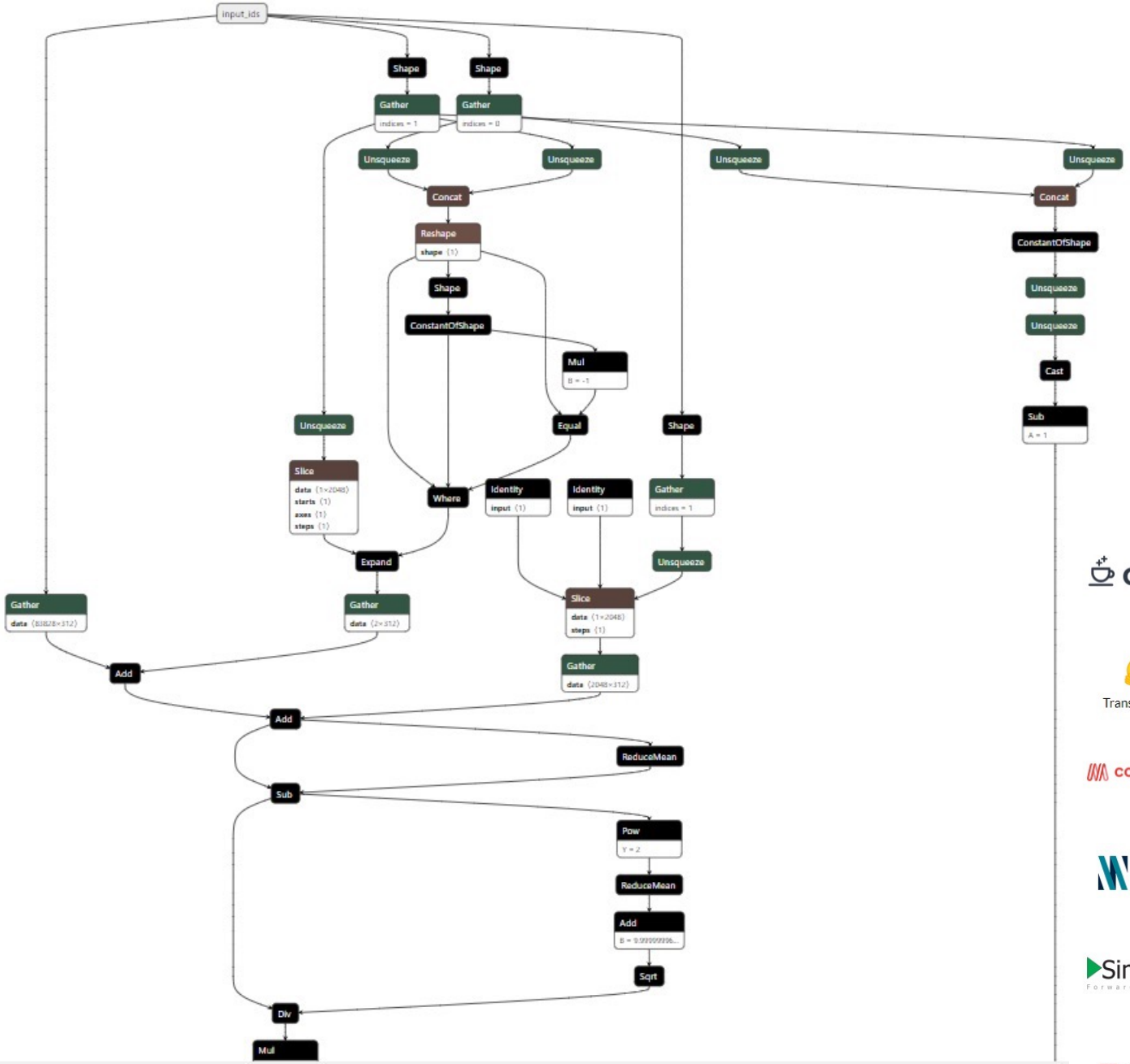
Зачем, а главное, зачем?..

- Работает в эко-системе .NET
- Поддерживает интеграцию с Python при использовании NimbusML
- И самое главное: инференс и поддержка формата ONNX

Инференс, ONNX и другие латинские фразы

- Инференсом ML-модели называют процесс её работы на конечном устройстве.
- ONNX (Open Neural Network Exchange) – открытый формат представления нейросети, который связывает представление моделей в разных фреймворках.





Caffe2 · Yandex CatBoost · Chainer · Cognitive Toolkit · CoreML

Transformers · Keras · LibSVM · MATLAB · MindSpore

composer · mxnet · MyCaffe · NCNN · NeoML

Neural Network Libraries · PaddlePaddle · PyTorch · SAS · SIEMENS

Simio · SINGA · SciKit Learn · Tengine · TensorFlow

Tribuo · XGBoost · WOLFRAM · ZAMA

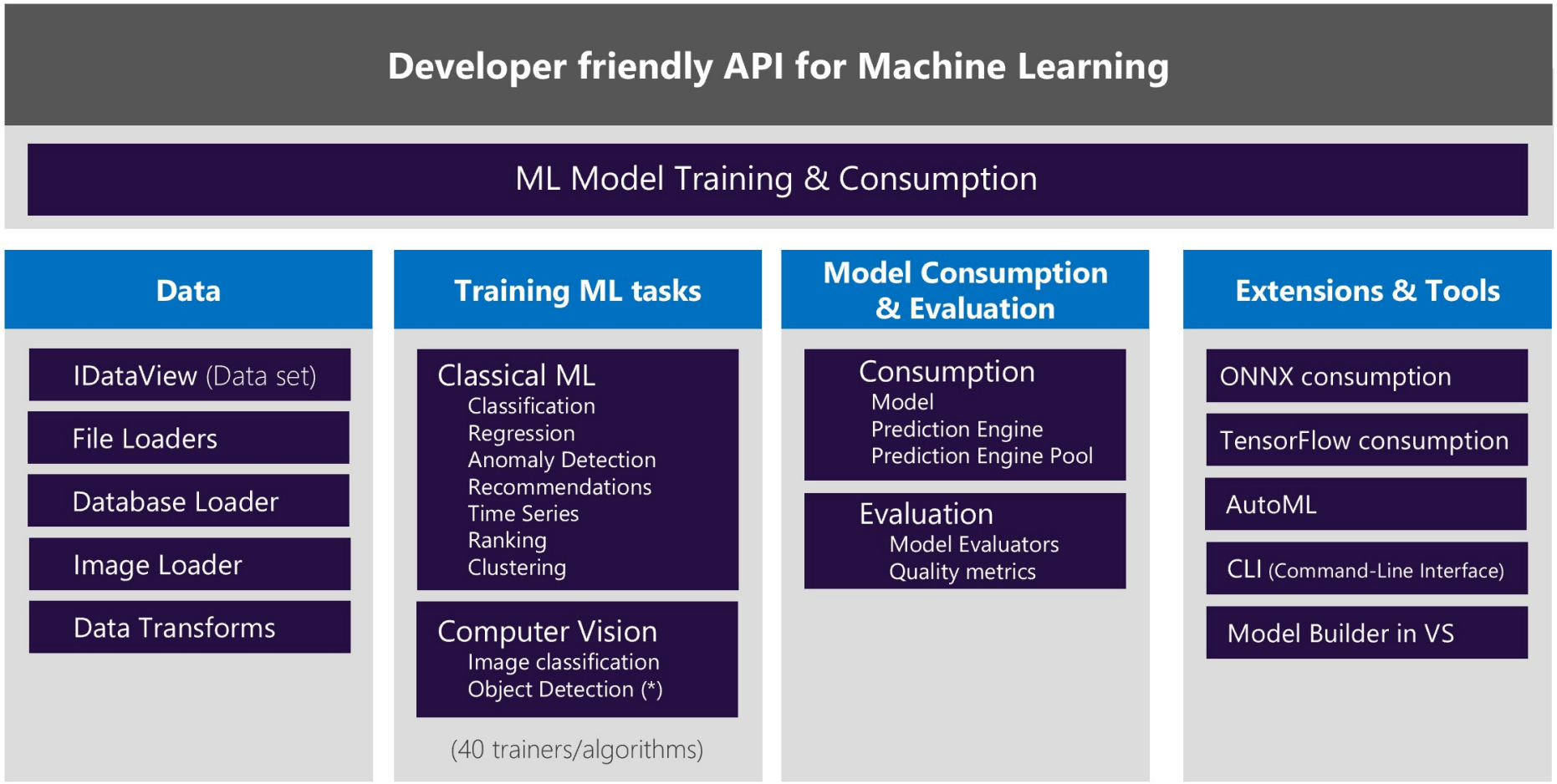
Работа с данными. DataView

- Аналог библиотеки pandas в Python и, конкретно, его объекта DataFrame
- Необходимо для очень удобного и крайне высокоуровневого препроцессинга данных перед обучением моделей

```
// Remove rows with missing values in the "Age" and "YearsExperience" columns
var view = context.Data.FilterRowsByMissingValues(data, "Age", "YearsExperience");

// Remove rows where "Age" is less than 20 or greater than 80
var view = context.Data.FilterRowsByColumn(data, "Age", lowerBound: 20, upperBound: 8

// Remove the "Age" column
var estimator = context.Transforms.DropColumns("Age");
var view = estimator.Fit(data).Transform(data);

// Replace missing values in the "Age" column
var estimator = context.Transforms.ReplaceMissingValues("Age",
      replacementMode: MissingValueReplacingEstimator.ReplacementMode.Mean);
var view = estimator.Fit(data).Transform(data);
```

Структура

ML.NET components



(*) Object detection coming soon after v1.4-Preview

Классический ML

- Классификация (SdcaLogisticRegression, LightGBM, FastTree)
- Регрессия (OnlineGradientDescent, FastForest)
- Кластеризация (KMeans)
- Ранжирование (LightGBM)
- Обнаружение аномалий (RandomizedPca, OneClassSVM)

TorchSharp

- Это интеграция PyTorch-подобного API в .NET для:
  - Загрузки предобученных нейросетей (например, BERT, ResNet).
  - Обучения своих моделей (но это сложнее, чем в Python).
  - Использования GPU (через CUDA, если установлены драйверы).
- Это очень похоже на torch, но не до конца

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

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

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

var optimizer = torch.optim.Adam(seq.parameters());

for (int i = 0; i < 10; i++) {
    using var eval = seq.forward(x);
    using var output = functional.mse_loss(eval, y, Reduction.Sum);

    optimizer.zero_grad();

    output.backward();

    optimizer.step();
}
```

А что с GPU в .NET?

- Ответ прост: ML.NET не умеет работать с GPU. Так что никакого настоящего Deep Learning на .NET нет и быть не может
- В дальнейшем интеграция работы с GPU во фреймворк планируется
- TorchSharp умеет работать с GPU (но только с CUDA)

Итоги

- ML.NET – очень крутой фреймворк машинного обучения на C# и F#
- Вся его крутость состоит в поддержке ONNX – универсального формата моделей машинного обучения
- Это позволяет интегрировать ML-модели, обученные на другом языке, в экосистемы приложений .NET

C# Program.cs

```
using Microsoft.ML;
using Microsoft.ML.Data;

var context = new MLContext();

var data = context.Data.LoadFromTextFile<BostonHousing>( path: "HousingData.csv", hasHeader: true,
    separatorChar: ',');

var view = context.Data.FilterRowsByMissingValues(data, params columns: "CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE",
    "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT", "MEDV");

var trainTestData = context.Data.TrainTestSplit(data, testFraction: 0.2, seed: 0);
var trainData = trainTestData.TrainSet;
var testData = trainTestData.TestSet;

var pipeline :EstimatorChain<RegressionPredictionTransformer<...>>? = context.Transforms.Concatenate( outputColumnName: "Features",
        params inputColumnNames: "CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE",
        "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT") // ColumnConcatenatingEstimator
    .Append( estimator: context.Transforms.NormalizeMinMax( outputColumnName: "Features")) // EstimatorChain<NormalizingTransformer>
    .Append( estimator: context.Regression.Trainers.LbfgsPoissonRegression(
        labelColumnName: "MEDV",
        featureColumnName: "Features",
        l1Regularization: 0f,
        l2Regularization: 0f
    ));

var model :TransformerChain<RegressionPredictionTransformer<...>>? = pipeline.Fit(trainData);

var predictions :IDataView? = model.Transform(testData);
var metrics = context.Regression.Evaluate(predictions, labelColumnName: "MEDV");

Console.WriteLine($"R-Squared: {metrics.RSquared}");
Console.WriteLine($"Mean Absolute Error: {metrics.MeanAbsoluteError}");
Console.WriteLine($"Mean Squared Error: {metrics.MeanSquaredError}");
```

14 1

```
public class BostonHousing
{
    [LoadColumn( fieldIndex: 0)]
    public float CRIM { get; set; }

    [LoadColumn( fieldIndex: 1)]
    public float ZN { get; set; }

    [LoadColumn( fieldIndex: 2)]
    public float INDUS { get; set; }

    [LoadColumn( fieldIndex: 3)]
    public float CHAS { get; set; }

    [LoadColumn( fieldIndex: 4)]
    public float NOX { get; set; }

    [LoadColumn( fieldIndex: 5)]
    public float RM { get; set; }

    [LoadColumn( fieldIndex: 6)]
    public float AGE { get; set; }

    [LoadColumn( fieldIndex: 7)]
    public float DIS { get; set; }

    [LoadColumn( fieldIndex: 8)]
    public float RAD { get; set; }

    [LoadColumn( fieldIndex: 9)]
    public float TAX { get; set; }

    [LoadColumn( fieldIndex: 10)]
    public float PTRATIO { get; set; }
```

```
/Users/mikhailkharlamov/Documents/SPbU/Cou
R-Squared: 0,7925908621046165
Mean Absolute Error: 3,126210268806009
Mean Squared Error: 22,730372898598603

Process finished with exit code 0.
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import PoissonRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


column_names = ["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE",
    "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT", "MEDV"]

df = pd.read_csv("HousingData.csv", names=column_names, na_values="?", comment="\t", sep=",", skiprows=22)
df = df.dropna()

features = ["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE",
            "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT"]
target = ["MEDV"]

X_train, X_test, y_train, y_test = train_test_split(df[features], df[target], test_size=0.2, random_state=42)

model = PoissonRegressor(alpha=0.1, max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print(f"R2: {r2_score(y_test, y_pred):.3f}")
print(f"MAE: {mean_absolute_error(y_test, y_pred):.3f}")
print(f"MSE: {mean_squared_error(y_test, y_pred):.3f}")
```

```
R2: 0.801
MAE: 2.479
MSE: 15.483
```

Список литературы

- https://dotnet.microsoft.com/ru-ru/apps/ai/ml-dotnet
- https://habr.com/ru/companies/jugru/articles/495208/
- https://habr.com/ru/companies/microsoft/articles/436728/
- https://habr.com/ru/companies/rostelecom/articles/704844/
- https://ru.wikipedia.org/wiki/ML.NET

- https://github.com/mikhail-kharlamov/CSharp-Homeworks/blob/machineLearning/MachineLearning/MachineLearning/Program.cs