

# WEB-ТЕХНОЛОГИИ - ЛЕКЦИИ

02.03.03 -Математическое обеспечение и администрирование информационных систем,  
направленность (профиль) -разработка и администрирование информационных систем

<http://vikchas.ru>

## Лекция 4. Тема «Основы WEB- программирования для машинного обучения ML.NET»

Часовских Виктор Петрович  
доктор технических наук, профессор кафедры  
ШИиКМ, ФГБОУ ВО «Уральский  
государственный экономический университет

Екатеринбург 2025

## WEB-программирование и машинное обучение с [ML.NET](#)

[ML.NET](#) представляет собой фреймворк машинного обучения от Microsoft, который интегрируется с экосистемой .NET, делая его особенно привлекательным для разработчиков веб-приложений на платформе [ASP.NET](#).

### Основы [ML.NET](#)

[ML.NET](#) — это кроссплатформенный фреймворк машинного обучения с открытым исходным кодом, который позволяет интегрировать модели машинного обучения непосредственно в .NET-приложения. Ключевые особенности:

- Написан на C# и оптимизирован для .NET-экосистемы
- Поддерживает различные сценарии ML: классификация, регрессия, кластеризация, обнаружение аномалий и другие
- Позволяет использовать предобученные модели или обучать собственные
- Имеет интеграцию с другими фреймворками ML через ONNX

### Архитектура веб-приложений с [ML.NET](#)

#### 1. Встроенные модели ML в [ASP.NET](#)

Самый прямолинейный подход — интеграция [ML.NET](#) непосредственно в проект [ASP.NET](#):

CSHARP

// Пример контроллера ASP.NET Core с интегрированной ML.NET моделью

[ApiController]

[Route("api/[controller]")]

public class PredictionController : ControllerBase

{

    private readonly PredictionEngine<ModelInput, ModelOutput>  
    \_predictionEngine;

    public PredictionController(PredictionEngine<ModelInput, ModelOutput>  
    predictionEngine)

{

```
_predictionEngine = predictionEngine;  
}
```

```
[HttpPost]
```

```
public ActionResult<ModelOutput> Predict([FromBody] ModelInput input)  
{  
    var prediction = _predictionEngine.Predict(input);  
    return prediction;  
}  
}
```

При регистрации сервисов в Startup.cs:

CSHARP

```
public void ConfigureServices(IServiceCollection services)  
{  
    // Загрузка модели и создание PredictionEngine  
    var mlContext = new MLContext();  
    var model = mlContext.Model.Load("model.zip", out var schema);  
  
    // Регистрация PredictionEngine как синглтон  
    services.AddSingleton(sp =>  
        mlContext.Model.CreatePredictionEngine<ModelInput,  
        ModelOutput>(model));  
  
    services.AddControllers();  
}
```

## 2. Микросервисная архитектура с ML-сервисом

Для более масштабируемых решений можно вынести ML-компонент в отдельный микросервис:

- **Отдельный [ML.NET API](#):** Специализированный микросервис для ML-задач

- **gRPC-интерфейс:** Эффективное взаимодействие между веб-приложением и ML-сервисом
- **Асинхронная обработка:** Использование очередей сообщений для обработки ML-задач в фоновом режиме

## Интеграция [ML.NET](#) с веб-технологиями

### 1. [ASP.NET Core](#) и [ML.NET](#)

CSHARP

// Пример регистрации ML.NET Model Builder в ASP.NET Core

```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        // Регистрация ML.NET контекста
        services.AddSingleton<MLContext>(new MLContext());

        // Регистрация сервиса прогнозирования
        services.AddTransient<IModelService, ModelService>();

        // Регистрация обработчиков данных
        services.AddTransient<IDataProcessor, DataProcessor>();

        services.AddControllersWithViews();
    }
}
```

### 2. Blazor и [ML.NET](#)

Blazor позволяет создавать интерактивные веб-интерфейсы с возможностью запуска [ML.NET](#) моделей как на сервере, так и в WebAssembly:

CSHARP

@page "/predict"

@inject PredictionEngine<ModelInput, ModelOutput> PredictionEngine

```
<h1>Прогнозирование</h1>
```

```
<EditForm Model="@input" OnValidSubmit="HandleSubmit">
    <InputNumber @bind-Value="input.Feature1" />
    <InputNumber @bind-Value="input.Feature2" />
    <button type="submit">Получить прогноз</button>
</EditForm>
```

```
@if (prediction != null)
{
    <p>Прогноз: @prediction.Score</p>
}
```

```
@code {
    private ModelInput input = new();
    private ModelOutput? prediction;

    private void HandleSubmit()
    {
        prediction = PredictionEngine.Predict(input);
    }
}
```

### 3. SignalR для обновлений в реальном времени

Для отправки результатов ML-прогнозов в реальном времени:

CSHARP

```
// Hub для отправки обновлений ML-прогнозов
```

```
public class PredictionHub : Hub
{
```

```

private readonly IModelService _modelService;

public PredictionHub(IModelService modelService)
{
    _modelService = modelService;
}

public async Task RequestPrediction(ModelInput input)
{
    var result = _modelService.Predict(input);
    await Clients.Caller.SendAsync("ReceivePrediction", result);
}

public async Task SubscribeToStreamPredictions()
{
    // Логика подписки на поток прогнозов
}
}

```

## Практические сценарии использования [ML.NET](#) в веб-приложениях

### 1. Интеллектуальная классификация данных

#### CSHARP

// Сервис классификации текстовых данных

```

public class TextClassificationService
{
    private readonly PredictionEngine<TextData, TextPrediction>
    _predictionEngine;

    public TextClassificationService(MLContext mlContext)
    {

```

```

        // Загрузка модели классификации текста
        var model = mlContext.Model.Load("text_classification_model.zip", out _);
        _predictionEngine = mlContext.Model.CreatePredictionEngine<TextData,
        TextPrediction>(model);
    }

    public string ClassifyText(string text)
    {
        var prediction = _predictionEngine.Predict(new TextData { Text = text });
        return prediction.Category;
    }
}

```

## 2. Рекомендательные системы

### CSHARP

```

// Контроллер для рекомендаций продуктов
[ApiController]
[Route("api/[controller]")]
public class RecommendationsController : ControllerBase
{
    private readonly IProductRecommender _recommender;

    public RecommendationsController(IProductRecommender recommender)
    {
        _recommender = recommender;
    }

    [HttpGet("user/{userId}")]
    public IActionResult GetRecommendationsForUser(int userId)
    {

```

```
        var recommendations = _recommender.GetTopRecommendations(userId, 5);  
        return Ok(recommendations);  
    }  
}
```

### 3. Обнаружение аномалий в веб-трафике

CSHARP

// Middleware для обнаружения аномалий в запросах

```
public class AnomalyDetectionMiddleware
```

```
{
```

```
    private readonly RequestDelegate _next;
```

```
    private readonly IAnomalyDetector _anomalyDetector;
```

```
    public AnomalyDetectionMiddleware(RequestDelegate next, IAnomalyDetector  
anomalyDetector)
```

```
    {
```

```
        _next = next;
```

```
        _anomalyDetector = anomalyDetector;
```

```
    }
```

```
    public async Task InvokeAsync(HttpContext context)
```

```
    {
```

```
        // Сбор метрик запроса
```

```
        var requestFeatures = ExtractRequestFeatures(context);
```

```
        // Проверка на аномалии
```

```
        var isAnomaly = _anomalyDetector.CheckForAnomaly(requestFeatures);
```

```
        if (isAnomaly)
```

```
        {
```

```
// Логирование или блокировка аномального запроса
context.Response.StatusCode = 403;
await context.Response.WriteAsync("Suspicious request detected");
return;
}

await _next(context);
}
}
```

## Инструменты и оптимизации для [ML.NET](#) в веб-приложениях

### 1. [ML.NET](#) Model Builder

Visual Studio имеет интегрированный инструмент [ML.NET](#) Model Builder, который значительно упрощает создание и интеграцию моделей:

- Визуальный интерфейс для создания моделей
- Автоматический выбор алгоритмов (AutoML)
- Генерация готового кода для интеграции

### 2. Оптимизация производительности

- **Pooling PredictionEngine:** Создание пула экземпляров PredictionEngine для высоконагруженных веб-приложений
- **Batch-предсказания:** Обработка нескольких запросов одновременно для повышения производительности
- **Кэширование результатов:** Сохранение результатов предсказаний для частых запросов

### 3. Мониторинг и отладка ML-моделей в веб-среде

- Интеграция с Application Insights для отслеживания производительности ML-компонентов
- Логирование метрик и предсказаний
- A/B тестирование различных моделей

## Заключение

[ML.NET](#) предоставляет мощные инструменты для интеграции машинного обучения в веб-приложения на платформе .NET. От простых встроенных моделей до сложных микросервисных архитектур — [ML.NET](#) обеспечивает гибкость и производительность, необходимые для современных интеллектуальных веб-приложений. Благодаря тесной интеграции с экосистемой .NET, разработчики могут легко добавлять функции машинного обучения в существующие [ASP.NET](#) приложения, расширяя их возможности без необходимости осваивать новые технологические стеки.

## **Машинное обучение в среде Windows Server 2022: [ML.NET](#) и конкуренты**

### **[ML.NET](#) в контексте Windows Server 2022**

Windows Server 2022 предоставляет стабильную и высокопроизводительную платформу для запуска решений машинного обучения в корпоративной среде. [ML.NET](#), как родной фреймворк от Microsoft, имеет ряд преимуществ при использовании на этой платформе:

#### **Преимущества [ML.NET](#) на Windows Server 2022:**

##### **1. Нативная интеграция:**

- Бесшовная работа в экосистеме .NET и Windows
- Оптимизация под Windows-инфраструктуру
- Поддержка Windows-аутентификации и безопасности

##### **2. Производительность:**

- Оптимизация для серверных CPU Intel/AMD
- Возможность использования аппаратного ускорения с поддержкой CUDA
- Низкое потребление ресурсов по сравнению с некоторыми другими фреймворками

##### **3. Масштабируемость:**

- Интеграция с Windows Server Failover Clustering
- Поддержка балансировки нагрузки Windows

- Работа с Hyper-V для виртуализации ML-среды

## **Альтернативные решения для Windows Server 2022**

### **1.TensorFlow и PyTorch на Windows**

#### **Преимущества:**

- Более широкое сообщество и экосистема
- Богатая библиотека предобученных моделей
- Более глубокие возможности для сложных задач глубокого обучения

#### **Недостатки:**

- Производительность на Windows может быть ниже, чем на Linux
- Необходимость настройки Python-окружения
- Сложности с интеграцией в .NET-приложения

### **2.Microsoft Cognitive Toolkit (CNTK)**

#### **Преимущества:**

- Нативная оптимизация для Windows
- Высокая производительность для нейронных сетей
- Хорошая интеграция с другими службами Microsoft

#### **Недостатки:**

- Менее активная разработка по сравнению с [ML.NET](#)
- Ограниченное сообщество по сравнению с TensorFlow/PyTorch
- Меньший выбор готовых моделей

### **3.[H2O.ai](#) на Windows**

#### **Преимущества:**

- Ориентированность на бизнес-задачи
- Автоматизация ML (AutoML)
- Хорошая визуализация и интерпретируемость

#### **Недостатки:**

- Не так тесно интегрирован с Windows Server как [ML.NET](#)
- Требуется Java-среда
- Платная модель для расширенных функций

## Сравнительный анализ для Windows Server 2022

| Аспект                        | <u>ML.NET</u> | TensorFlow/PyTorch | CNTK       | <u>H2O.ai</u> |
|-------------------------------|---------------|--------------------|------------|---------------|
| Интеграция с Windows          | ★★★★★<br>★    | ★★★★☆☆             | ★★★★★<br>☆ | ★★★★☆<br>☆    |
| Интеграция с .NET             | ★★★★★<br>★    | ★★★☆☆☆             | ★★★★☆<br>☆ | ★★★☆☆<br>☆    |
| Производительность на Windows | ★★★★★<br>☆    | ★★★★☆☆             | ★★★★★<br>☆ | ★★★★☆<br>☆    |
| Глубокое обучение             | ★★★★☆<br>☆    | ★★★★★★             | ★★★★★<br>☆ | ★★★★☆<br>☆    |
| Традиционное ML               | ★★★★★<br>☆    | ★★★★☆☆             | ★★★★☆<br>☆ | ★★★★★<br>★    |
| Масштабируемость              | ★★★★★<br>☆    | ★★★★★★             | ★★★★☆<br>☆ | ★★★★★<br>☆    |
| Корпоративные функции         | ★★★★★<br>☆    | ★★★★☆☆             | ★★★★☆<br>☆ | ★★★★★<br>☆    |
| Экосистема/сообщество         | ★★★★☆<br>☆    | ★★★★★★             | ★★★☆☆<br>☆ | ★★★★☆<br>☆    |

### Рекомендации по выбору

Рекомендуется [ML.NET](#), если:

- Ваша инфраструктура и приложения основаны на технологиях Microsoft
- Требуется тесная интеграция с .NET-приложениями
- Важны корпоративные функции безопасности Windows Server
- Требуются стандартные задачи ML: классификация, регрессия, кластеризация

Рекомендуются альтернативные решения, если:

- Требуются сложные модели глубокого обучения (TensorFlow/PyTorch)
- Нужно максимальное количество предобученных моделей (TensorFlow/PyTorch)
- Требуются продвинутые возможности AutoML ([H2O.ai](https://h2o.ai))
- Проект не ограничен экосистемой Microsoft

### **Практические соображения по развертыванию**

Для [ML.NET](#) на Windows Server 2022:

- Используйте Windows Server Core для максимальной производительности
- Рассмотрите возможность интеграции с контейнерами Windows
- Используйте SQL Server как хранилище данных для ML-моделей
- Интегрируйте с Windows Authentication для безопасного доступа к моделям

В корпоративной среде Windows Server 2022 [ML.NET](#) предоставляет наиболее органичное решение, особенно для компаний, уже инвестировавших в экосистему Microsoft. Однако для определенных узкоспециализированных задач глубокого обучения может потребоваться дополнение другими фреймворками.