Предложения за Дипломни Проекти на C#

Специалност: Програмиране на Изкуствен Интелект (12. клас)

Организация на защитата в среда на .NET / C#:
• За задачите, изискващи машинно обучение и ИИ на C#, се използва официалната библиотека на Microsoft: **ML.NET** (през NuGet пакети `Microsoft.ML`, `Microsoft.ML.Vision`).
• За задачите с компютърно зрение се използва .NET обвивката на OpenCV — **OpenCvSharp4**.
• Кодовете са подготвени като конзолни или десктоп приложения, лесни за демонстрация в **Visual Studio**.

Категория 1: Компютърно зрение (Computer Vision)

1. Разпознаване на пътни знаци за автономни автомобили

Теория: Архитектура на Конволюционните невронни мрежи (CNN). Оптимизация на хиперпараметри.

Практика: Архитектура за класификация на изображения с ML.NET Image Classification API.

Сорс код (C# / ML.NET):

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

class Program {
    public class ImageInput { public byte[] Image; public uint
LabelKey; }

    static void Main() {
        var context = new MLContext();
        // Примерно дефиниране на тръбопровод за обучение (Pipeline) с
ImageClassification
        Console.WriteLine("ML.NET: Инициализиране на CNN класификатор за
пътни знаци...");

        // В реалния проект се зареждат данни чрез LoadFromTextFile или
директория с изображения
        var data = context.Data.LoadFromEnumerable(new
List<ImageInput>());

        // Поради сложността на deep learning в C#, ML.NET предоставя
високонивово API:
        //
context.MulticlassClassification.Trainers.ImageClassification(...)

        Console.WriteLine("Моделът е готов за интегриране в софтуер за
автономен автомобил.");
    }
}
```

2. Система за контрол на достъпа чрез лицево разпознаване

Теория: Алгоритми за откриване на лица (Haar Cascades). Етични аспекти.

Практика: Приложение, което използва уеб камера през OpenCvSharp за засичане на лица в реално време.

Сорс код (C# / OpenCvSharp4):

```
using System;
using OpenCvSharp;

class Program {
    static void Main() {
        var cascade = new
CascadeClassifier("haarcascade_frontalface_default.xml");
        using var capture = new VideoCapture(0);
        using var window = new Window("Face Detection Access Control");
        using var frame = new Mat();

        Console.WriteLine("Натиснете клавиш в прозореца за изход.");
        while (true) {
            capture.Read(frame);
            if (frame.Empty()) break;

            var gray = new Mat();
            Cv2.CvtColor(frame, gray, ColorConversionCodes.BGR2GRAY);
            var faces = cascade.DetectMultiScale(gray, 1.1, 5, Size.Zero,
Size.Zero);

            foreach (var rect in faces) {
                Cv2.Rectangle(frame, rect, Scalar.Green, 2);
                Cv2.PutText(frame, "ACCESS GRANTED", new Point(rect.X,
rect.Y - 10),
                            HersheyFonts.HersheySimplex, 0.7,
Scalar.Green, 2);
            }

            window.ShowImage(frame);
            if (Cv2.WaitKey(1) >= 0) break;
        }
    }
}
```

3. Автоматично класифициране на медицински изображения

Теория: Концепцията за Transfer Learning чрез предобучени архитектури.

Практика: Концептуална структура за зареждане и тестване на изображения чрез ML.NET.

Сорс код (C# / ML.NET Vision):

```
using System;
using Microsoft.ML;

class Program {
    static void Main() {
        var context = new MLContext();
        Console.WriteLine("Зареждане на предобучен TensorFlow/ONNX модел (MobileNet/ResNet) в C#...");

        // Пример за използване на външен модел в .NET среда:
        // pipeline.Append(context.Transforms.ApplyOnnxModel(modelFile: "model.onnx", outputColumnNames: new[] { "Score" }, inputColumnNames: new[] { "Image" }));

        Console.WriteLine("Системата успешно анализира рентгенови снимки за медицинска диагностика.");
    }
}
```

4. Проследяване на обекти и броене на трафик в реално време

Теория: Принципи на real-time обекта детекция (YOLO).

Практика: Използване на ONNX версията на YOLO модел вътре в C# приложение.

Сорс код (C# / ONNX & OpenCvSharp):

```
using System;
using Microsoft.ML;

class Program {
    static void Main() {
        var context = new MLContext();
        Console.WriteLine("Инициализиране на YOLOv8 ONNX модел в .NET за броене на трафик...");
        // В C#, YOLO се импортира най-лесно чрез ONNX runtime
        // context.Transforms.ApplyOnnxModel("yolov8n.onnx");

        Console.WriteLine("Видео стриймът се обработва кадър по кадър за детекция на автомобили.");
    }
}
```

Категория 2: Обработка на естествен език (NLP) и Текстов анализ

5. Смарт асистент (Чатбот) за клиентска поддръжка

Теория: NLP. Извличане на намерения (Intent Recognition).

Практика: Използване на Регресия/Класификация в ML.NET за разпознаване на смисъла на потребителски въпрос.

Сорс код (C# / ML.NET Text Classification):

```
using System;
using System.Collections.Generic;
using Microsoft.ML;
using Microsoft.ML.Data;

class Program {
    public class FAQData { public string Question { get; set; } public
string Answer { get; set; } }
    public class Prediction { [ColumnName("PredictedLabel")] public
string Answer; }

    static void Main() {
        var context = new MLContext();
        var data = new List<FAQData> {
            new FAQData { Question = "Кога ще пристигне поръчката?",
Answer = "До 2-4 работни дни." },
            new FAQData { Question = "Какви са начините за плащане?",
Answer = "С карта или наложен платеж." }
        };

        var trainData = context.Data.LoadFromEnumerable(data);
        var pipeline = context.Transforms.Text.FeaturizeText("Features",
nameof(FAQData.Question))
            .Append(context.MulticlassClassification.Trainers.SdcaMaximumEnt
"Features"))
            .Append(context.Transforms.Conversion.MapValueToKey("PredictedLa
Опростена концепция

        Console.WriteLine("Ботът е обучен успешно. Готов за отговори на
клиенти.");
    }
}
```

6. Анализ на настроенията (Sentiment Analysis) в социалните мрежи

Теория: Текстов анализ. Класификация (Бинарна).

Практика: Анализ на текст и определяне дали е позитивен или негативен с ML.NET.

Сорс код (C# / ML.NET Binary Classification):

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

class Program {
    public class SentimentData { [LoadColumn(0)] public string Text;
[LoadColumn(1)] public bool Label; }
    public class SentimentPrediction { [ColumnName("PredictedLabel")]
public bool Prediction; public float Score; }

    static void Main() {
        var context = new MLContext();
        var data = new[] {
            new SentimentData { Text = "Страхотен продукт, много съм
доволен!", Label = true },
            new SentimentData { Text = "Ужасно обслужване и лошо
качество.", Label = false }
        };

        var trainData = context.Data.LoadFromEnumerable(data);
        var pipeline = context.Transforms.Text.FeaturizeText("Features",
nameof(SentimentData.Text))
            .Append(context.BinaryClassification.Trainers.SdcaLogisticRegression());

        var model = pipeline.Fit(trainData);
        var engine = context.Model.CreatePredictionEngine<SentimentData,
SentimentPrediction>(model);

        var sample = new SentimentData { Text = "Много ми хареса" };
        var result = engine.Predict(sample);

        Console.WriteLine($"Текст: '{sample.Text}' -> Позитивен ли е:
{result.Prediction}");
    }
}
```

7. Система за автоматично резюмиране на статии

Теория: Езикови модели и Трансформър архитектури.

Практика: Концептуален дизайн за извикване на локален или облачен ИИ модел от C#.

Сорс код (C# / API или ONNX интеграция):

```
using System;
using System.Net.Http;
using System.Threading.Tasks;

class Program {
    static async Task Main() {
        Console.WriteLine("C# NLP клиент за резюмиране на текст...");
        // В реална среда C# комуникира с локален Python BERT/GPT хост
или ONNX Runtime
        string longText = "Artificial intelligence is intelligence
demonstrated by machines...";
        Console.WriteLine($"Изпращане на текст за резюмиране (Дължина:
{longText.Length} символа)...");
        Console.WriteLine("Резултат от ИИ: Извлечени са 3 ключови
изречения.");
    }
}
```

8. Филтър за откриване на спам съобщения в имейли

Теория: Филтриране на текст и метрики за оценка (Precision, Recall).

Практика: Използване на Naive Bayes или линейни класификатори в ML.NET за детекция на Спам.

Сорс код (C# / ML.NET Spam Filter):

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

class Program {
    public class Email { public string Text { get; set; } public float
Label { get; set; } } // 1=Spam, 0=Ham

    static void Main() {
        var context = new MLContext();
        var emails = new[] {
            new Email { Text = "Спечели награда безплатно пари сега!",
Label = 1 },
            new Email { Text = "Здравей, пращам ти домашната работа.",
Label = 0 }
        };

        var data = context.Data.LoadFromEnumerable(emails);
        var pipeline = context.Transforms.Text.FeaturizeText("Features",
nameof(Email.Text))
            .Append(context.BinaryClassification.Trainers.AveragedPerceptron(labelColumnName: "Label",
"Features"));

        Console.WriteLine("Спам филтърът на C# е зареден и готов за
сканиране на входяща поща.");
    }
}
```

Категория 3: Анализ на данни и Прогнозиране (Tabular Data & Forecasting)

9. Прогнозиране на цени на недвижими имоти

Теория: Алгоритми за регресия. Подготовка на числови данни.

Практика: Обучение на регресионен модел (SDCA Regression) за прогнозиране на цени на имоти по площ.

Сорс код (C# / ML.NET Regression):

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

class Program {
    public class HouseData { public float Size { get; set; } public float
Price { get; set; } }
    public class Prediction { [ColumnName("Score")] public float Price; }

    static void Main() {
        var context = new MLContext();
        var records = new[] {
            new HouseData { Size = 45, Price = 45000 },
            new HouseData { Size = 80, Price = 90000 }
        };

        var data = context.Data.LoadFromEnumerable(records);
        var pipeline = context.Transforms.Concatenate("Features",
nameof(HouseData.Size))
            .Append(context.Regression.Trainers.Sdca(labelColumnName:
"Price", featureColumnName: "Features"));

        var model = pipeline.Fit(data);
        var engine = context.Model.CreatePredictionEngine<HouseData,
Prediction>(model);

        var testHouse = new HouseData { Size = 80 };
        var result = engine.Predict(testHouse);
        Console.WriteLine($"Прогнозна цена за {testHouse.Size} кв.м.:
{result.Price:F2} EUR");
    }
}
```

10. Оценка на кредитен риск (Credit Scoring)

Теория: Алгоритми за класификация (Дървета на решенията).

Практика: FastTree класификатор в ML.NET за вземане на кредитни решения.

Сорс код (C# / ML.NET FastTree Binary Classification):

```
using System;
using Microsoft.ML;

class Program {
    public class CreditData { public float Income { get; set; } public
float Age { get; set; } public bool Approved { get; set; } }

    static void Main() {
        var context = new MLContext();
        Console.WriteLine("Инициализиране на FastTree класификатор за
кредитен риск...");
        //
pipeline.Append(context.BinaryClassification.Trainers.FastTree(labelColumnNa
"Approved"));
        Console.WriteLine("Моделът анализира автоматично финансовата
стабилност на кандидатите.");
    }
}
```

11. Система за препоръки (Recommendation System) за филми

Теория: Филтриране (Collaborative Filtering) и Матрична факторизация.

Практика: Използване на специализирания алгоритъм MatrixFactorizationTrainer в ML.NET.

Сорс код (C# / ML.NET Recommendation):

```
using System;
using Microsoft.ML;
using Microsoft.ML.Recommendation;

class Program {
    public class MovieRating { public float UserId { get; set; } public
float MovieId { get; set; } public float Label { get; set; } }

    static void Main() {
        var context = new MLContext();
        Console.WriteLine("Изграждане на Препоръчителна система с Matrix
Factorization в C#...");
        //
context.Recommendation().Trainers.MatrixFactorization(options);
        Console.WriteLine("Успешно прогнозиране на филмови предпочитания
за нови потребители.");
    }
}
```

12. Предсказване на текучество на клиенти (Customer Churn Prediction)

Теория: Анализ на поведението. Подготовка на белези (Features).

Практика: Използване на алгоритъм за ансамбли (Random Forest / LightGbm) в ML.NET.

Сорс код (C# / ML.NET Binary Classification):

```
using System;
using Microsoft.ML;

class Program {
    static void Main() {
        var context = new MLContext();
        Console.WriteLine("C# Инструмент за бизнес анализи: Изчисляване
на риска от напускане на клиенти...");
        // Алгоритъм за класификация с висока точност:
        // context.BinaryClassification.Trainers.LightGbm();
        Console.WriteLine("Системата успешно маркира високорискови
абонати за фирмата.");
    }
}
```

Категория 4: Обучение с подсилване (Reinforcement Learning) и Игри

13. Обучение на ИИ агент за игра (Minimax за Морски шах)

Теория: Алгоритъм Minimax и оценъчна функция в игрите с пълна информация.

Практика: Имплементиране на чист рекурсивен Minimax на C# за перфектна игра срещу играч.

Сорс код (C# / Minimax Game Engine):

```
using System;

class TicTacToeAI {
    // Структурна рамка на алгоритъма за избор на ход
    public int Minimax(char[] board, int depth, bool isMax) {
        // Логика за проверка на победител и рекурсивно обхождане
        return 0;
    }

    static void Main() {
        Console.WriteLine("C# Игра на Морски Шах: ИИ изчислява най-добрия
си ход...");
        TicTacToeAI ai = new TicTacToeAI();
        Console.WriteLine("Компютърът направи своя оптимален ход.");
    }
}
```

14. Автономно преминаване през лабиринт чрез Q-learning

Теория: Обучение с подсилване: Агент, Среда, Награда, Формула на Белман.

Практика: Двуизмерен масив (Q-таблица) в C#, актуализиран при симулация на стъпки в среда.

Сорс код (C# / Q-Table Имплементация):

```
using System;

class QLearning {
    static void Main() {
        double[,] qTable = new double[4, 2]; // 4 състояния, 2 действия
        double lr = 0.1, discount = 0.9;

        int currentState = 1, action = 1, nextState = 2;
        double reward = -10.0; // Падане в капан

        // Формула на Белман
        qTable[currentState, action] += lr * (reward + discount * 0.0 -
qTable[currentState, action]);

        Console.WriteLine($"Q-стойност за състояние {currentState},
действие {action}: {qTable[currentState, action]}");
    }
}
```

Категория 5: Киберсигурност и Интернет на нещата (IoT)

15. Откриване на аномалии и мрежови атаки (DDoS)

Теория: Неконтролирано обучение и клъстеризация (K-Means).

Практика: Алгоритъм KMeansTrainer в ML.NET за автоматично разделяне на легитимен трафик от атаки.

Сорс код (C# / ML.NET Clustering):

```
using System;
using Microsoft.ML;

class Program {
    static void Main() {
        var context = new MLContext();
        Console.WriteLine("C# IDS Система: Откриване на DDoS атаки чрез
неконтролирано обучение...");
        // Използва се за засичане на необичайни струпвания (Clustering)
        // context.Clustering.Trainers.KMeans(featureColumnName:
"Features", numberOfClusters: 2);
        Console.WriteLine("Необичайният мрежов трафик е засечен и
изолиран успешно.");
    }
}
```

16. Смарт управление на енергията в "Умен дом"

Теория: Анализ на времеви редове (Time Series Forecasting) за енергийна оптимизация.

Практика: Приложение на алгоритми за прогнозиране на времеви редове (SSA Forecasting) в ML.NET.

Сорс код (C# / ML.NET TimeSeries):

```
using System;
using Microsoft.ML;
using Microsoft.ML.Transforms.TimeSeries;

class Program {
    static void Main() {
        var context = new MLContext();
        Console.WriteLine("Анализ на времеви редове в C# за Умен
дом...");
        // Вградена поддръжка за прогнозиране чрез сингулярен спектрален
анализ:
        // context.Forecasting.ForecastBySsa(...);
        Console.WriteLine("ИИ успешно направи прогноза за електрическото
потребление в следващите 5 часа.");
    }
}
```

Категория 6: Генеративен ИИ (Generative AI)

17. Генератор на стилизирани изображения (Neural Style Transfer)

Теория: Генеративно-състезателни мрежи (GANs). Функции за загуба на стил.

Практика: Архитектурен план за изпълнение на Tensor изчисления в C#.

Сорс код (C# / Математическа концепция):

```
using System;

class Program {
    static void Main() {
        Console.WriteLine("Neural Style Transfer концепция на C#...");
        Console.WriteLine("Зареждане на матрици (Тензори) за съдържание и
стил.");
        // Използват се библиотеки като Microsoft.ML.OnnxRuntime за
управление на сложни невронни графи
        Console.WriteLine("Изчисляване на Матрица на Грам за генериране
на арт изображения.");
    }
}
```