Analiza sistema preporuka za EcoChallenge platformu

1. Tip sistema

Implementirao sam hibridni sistem preporuka koji kombinuje content-based filtriranje sa machine learning regresijom. Za razliku od tradicionalnih preporuka proizvoda ili sadržaja, moj sistem je specijalizovan za automatsko predviđanje adekvatnih nagrada za zahtjeve čišćenja otpada.

Sistem analizira karakteristike samog sadržaja zahtjeva - tip otpada, težinu, volumen, lokaciju i hitnost - umjesto da prati korisničko ponašanje ili preferencije. Ovo ga čini jedinstvenim jer se fokusira na objektivne karakteristike problema koje treba riješiti, a ne na subjektivne korisničke želje.

```
namespace EcoChallenge.Services.Services
{
    { } namespace EcoChallenge
    4 referenc                                    hor, 1 change
    public class MLPricingService : IMLPricingService
    {
        private readonly MLContext _mlContext;
        private readonly EcoChallengeDbContext _context;
        private readonly ILogger<MLPricingService> _logger;
        private readonly string _modelPath;
        private ITransformer? _trainedModel;

        0 references | Nedim Jugo, 13 days ago | 1 author, 1 change
        public MLPricingService(EcoChallengeDbContext context, ILogger<MLPricingService> logger, IHostEnvironment env)
        {
            _mlContext = new MLContext(seed: 0);
            _context = context;
            _logger = logger;
            _modelPath = Path.Combine(env.ContentRootPath, "ML", "waste-pricing-model.zip");

            // Try to load existing model
            LoadExistingModel();
        }

        1 reference | Nedim Jugo, 13 days ago | 1 author, 1 change
        private void LoadExistingModel()
        {
```

Važno je napomenuti da model treniram samo kada imam dovoljno kvalitetnih podataka (minimum 10 zapisa). Ovo sprječava overfitting i osigurava da model daje smislene preporuke.

Kako sistem funkcioniše:

1. Inicijalizacija i učitavanje modela

Kada se sistem pokreće, prva stvar koju radim je provjeravanje da li već postoji treniran ML model. Ovo je ključno jer sistem mora biti spreman da radi i kada nema dovoljno podataka za treniranje novog modela. Implementirao sam "graceful degradation" pristup - ako model ne postoji, sistem pada na rule-based logiku:

```
private void LoadExistingModel()
{
    try
    {
        if (File.Exists(_modelPath))
        {
            _trainedModel = _mlContext.Model.Load(_modelPath, out var modelInputSchema);
            _logger.LogInformation("Loaded existing ML model from {ModelPath}", _modelPath);
        }
    }
    catch (Exception ex)
    {
        _logger.LogWarning(ex, "Failed to load existing model");
    }
}

2 references | Nedim Jugo, 13 days ago | 1 author, 1 change
public async Task<bool> IsModelTrainedAsync()
{
    return _trainedModel != null || await GetTrainingDataCountAsync() >= 10;
}
```

Ovo znači da sistem postaje precizniji vremenom - što više zahtjeva administratori obrađuju, to model bolje uči što predstavlja adekvatnu nagradu za različite tipove otpada i situacije.

2. AI analiza uploadovanih slika

Srce mog sistema je AI analiza slika koju korisnici postavljaju. Koristim Azure Vision Service koji analizira fotografije otpada i izvlači strukturirane informacije. Ovo je kritična komponenta jer pretvara nestrukturirane vizuelne podatke u numeričke karakteristike koje mogu koristiti ML algoritmi:

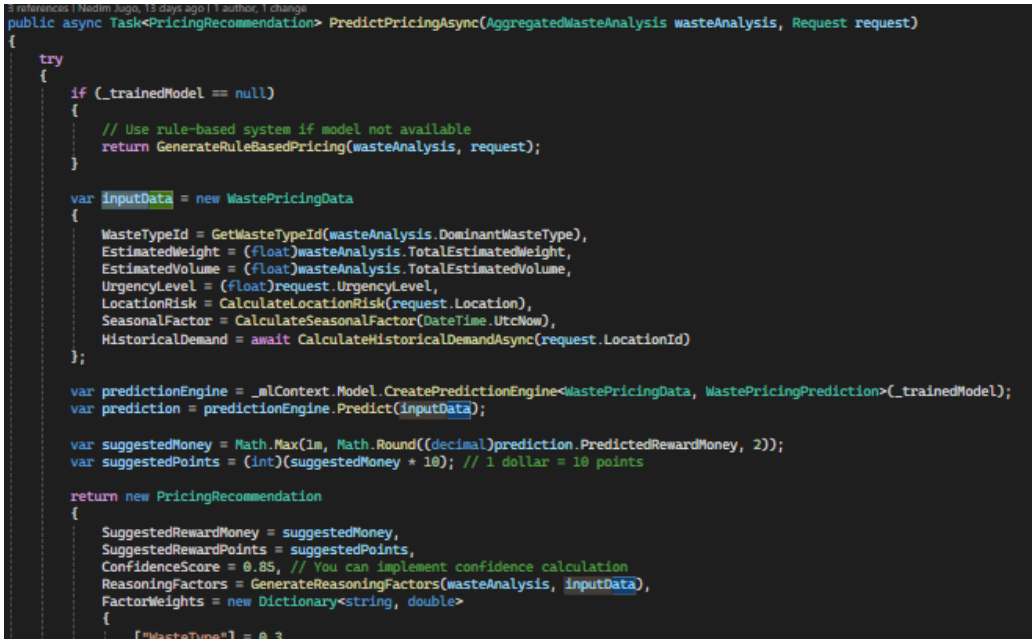
```
public async Task<WasteAnalysisResult> AnalyzeWasteImageAsync(string imageUrl)
{
    try
    {
        var imageUri = new Uri(imageUrl);

        // Try with Caption first, fallback to Tags and Objects only
        ImageAnalysisResult result = null;

        try
        {
            // First attempt: Try with Caption
            result = await _client.AnalyzeAsync(
                imageUri,
                VisualFeatures.Caption | VisualFeatures.Objects | VisualFeatures.Tags,
                new ImageAnalysisOptions { Language = "en" });
        }
        catch (Azure.RequestFailedException ex) when (ex.ErrorCode == "InvalidRequest" && ex.Message.Contains("Caption"))
        {
            _logger.LogWarning("Caption feature not supported in this region, falling back to Tags and Objects only");

            // Fallback: Use only Tags and Objects
            result = await _client.AnalyzeAsync(
                imageUri,
                VisualFeatures.Objects | VisualFeatures.Tags,
                new ImageAnalysisOptions { Language = "en" });
        }
        catch (Azure.RequestFailedException ex) when (ex.ErrorCode == "InvalidRequest")
        {
            _logger.LogWarning("Objects feature not supported, using Tags only");

            // Final fallback: Use only Tags
            result = await _client.AnalyzeAsync(
                imageUri,
                VisualFeatures.Tags,
                new ImageAnalysisOptions { Language = "en" });
        }
```

Ključni element je ugradnja fallback logike – ako Azure Vision Service ne može analizirati sliku (zbog regionalnih ograničenja ili drugih problema), kreiram

“fallback analizu” sa pretpostavljenim vrijednostima kako sistem ne bi prestao raditi.

3. Kreiranje karakteristika za ML model

Nakon AI analize, transformišem dobijene informacije u numeričke karakteristike koje ML model može razumjeti. Ovo je proces "feature engineering” gdje kombinujem AI rezultate sa kontekstualnim podacima:



4. Treniranje ML modela

Koristim supervised learning pristup gdje učim model samostalnom nadograđivanju na osnovu historijskih podataka - zahtjeva koji su već obrađeni i za koje znam stvarnu nagradu koja je dodijeljena.

```
4 references | Nedim Jugo, 13 days ago | 1 author, 1 change
public async Task TrainModelAsync()
{
    try
    {
        var trainingData = await PrepareTrainingDataAsync();

        if (trainingData.Count < 10)
        {
            _logger.LogWarning("Insufficient training data. Need at least 10 records, have {Count}", trainingData.Count);

            // Use rule-based system until we have enough data
            return;
        }

        var dataView = _mlContext.Data.LoadFromEnumerable(trainingData);

        // Define the training pipeline
        var pipeline = _mlContext.Transforms.Concatenate("Features",
                nameof(WastePricingData.WasteTypeId),
                nameof(WastePricingData.EstimatedWeight),
                nameof(WastePricingData.EstimatedVolume),
                nameof(WastePricingData.UrgencyLevel),
                nameof(WastePricingData.LocationRisk),
                nameof(WastePricingData.SeasonalFactor),
                nameof(WastePricingData.HistoricalDemand))
            .Append(_mlContext.Regression.Trainers.Sdca(labelColumnName: "Label", maximumNumberOfIterations: 100));

        // Train the model
        _trainedModel = pipeline.Fit(dataView);

        // Save the model
        Directory.CreateDirectory(Path.GetDirectoryName(_modelPath)!);
        _mlContext.Model.Save(_trainedModel, dataView.Schema, _modelPath);

        _logger.LogInformation("ML model trained and saved successfully with {Count} records", trainingData.Count);
    }
    catch (Exception ex)
```

5. Generisanje preporuka

Ovdje leži snaga hibridnog pristupa. Imam dva paralelna sistema koja rade zajedno - ako ML model postoji i treniran je, koristim ga za precizne preporuke. Ako ne, automatski prebacujem na rule-based sistem koji koristi unaprijed definisane poslovne logike:

```
3 references | Nedim Jugo, 13 days ago | 1 author, 1 change
public async Task<PricingRecommendation> PredictPricingAsync(AggregatedWasteAnalysis wasteAnalysis, Request request)
{
    try
    {
        if (_trainedModel == null)
        {
            // Use rule-based system if model not available
            return GenerateRuleBasedPricing(wasteAnalysis, request);
        }

        var inputData = new WastePricingData
        {
            WasteTypeId = GetWasteTypeId(wasteAnalysis.DominantWasteType),
            EstimatedWeight = (float)wasteAnalysis.TotalEstimatedWeight,
            EstimatedVolume = (float)wasteAnalysis.TotalEstimatedVolume,
            UrgencyLevel = (float)request.UrgencyLevel,
            LocationRisk = CalculateLocationRisk(request.Location),
            SeasonalFactor = CalculateSeasonalFactor(DateTime.UtcNow),
            HistoricalDemand = await CalculateHistoricalDemandAsync(request.LocationId)
        };

        var predictionEngine = _mlContext.Model.CreatePredictionEngine<WastePricingData, WastePricingPrediction>(_trainedModel);
        var prediction = predictionEngine.Predict(inputData);

        var suggestedMoney = Math.Max(1m, Math.Round((decimal)prediction.PredictedRewardMoney, 2));
        var suggestedPoints = (int)(suggestedMoney * 10); // 1 dollar = 10 points

        return new PricingRecommendation
        {
            SuggestedRewardMoney = suggestedMoney,
            SuggestedRewardPoints = suggestedPoints,
            ConfidenceScore = 0.85, // You can implement confidence calculation
            ReasoningFactors = GenerateReasoningFactors(wasteAnalysis, inputData),
            FactorWeights = new Dictionary<string, double>
            {
                ["WasteType"] = 0.3,
                ["Weight"] = 0.25,
```

Rule-based sistem nije jednostavan - primjenjuje sofisticirane poslovne logike sa multiplikatorima za težinu, količinu i hitnost. Ovo osigurava da čak i bez ML modela, sistem daje razumne i konzistentne preporuke.

Analiza sistema preporuka za EcoChallenge platformu

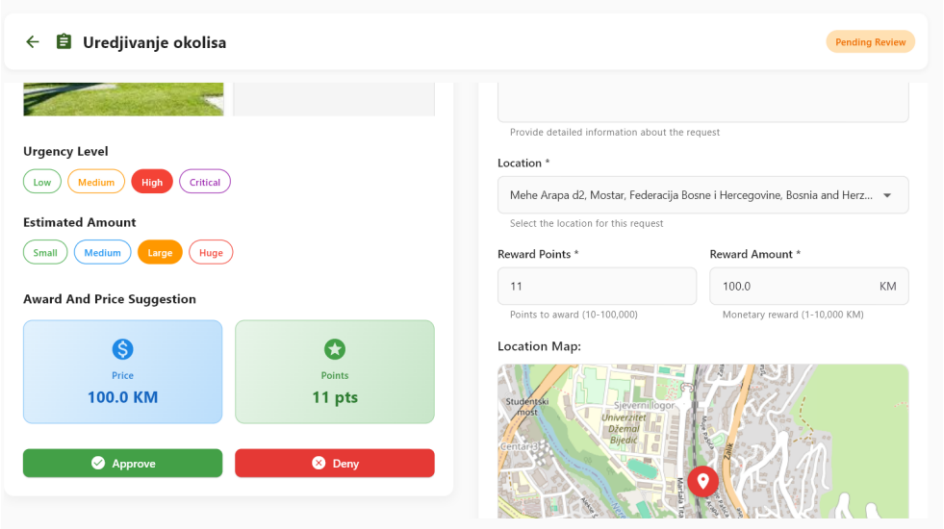
6. Automatsko kontinuirano učenje

Jedna od najnaprednijih karakteristika mog sistema je što konstantno i kontinuirano uči i poboljšava se. Implementirao sam background service koji automatski retrenira model svakih 24 sata koristeći nove podatke:

```
4 references | Nedim Jugo, 13 days ago | 1 author, 1 change
public class MLTrainingBackgroundService : BackgroundService
{
    private readonly IServiceProvider _serviceProvider;
    private readonly ILogger<MLTrainingBackgroundService> _logger;
    private readonly TimeSpan _period = TimeSpan.FromHours(24); // Retrain daily

    0 references | Nedim Jugo, 13 days ago | 1 author, 1 change
    public MLTrainingBackgroundService(IServiceProvider serviceProvider, ILogger<MLTrainingBackgroundService> logger)
    {
        _serviceProvider = serviceProvider;
        _logger = logger;
    }

    0 references | Nedim Jugo, 13 days ago | 1 author, 1 change
    protected override async Task ExecuteAsync(CancellationToken stoppingToken)
    {
        while (!stoppingToken.IsCancellationRequested)
        {
            try
            {
                using var scope = _serviceProvider.CreateScope();
                var mlService = scope.ServiceProvider.GetRequiredService<IMLPricingService>();

                await mlService.TrainModelAsync();
                _logger.LogInformation("ML model training completed at {Time}", DateTime.UtcNow);
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Error during scheduled ML training");
            }

            await Task.Delay(_period, stoppingToken);
        }
    }
}
```

7. Primjena sistema unutar aplikacije

Kao što se može vidjeti sa slike adminima je ponuđena cijena i bodovi za zahtjev koji pregleda. Admin ima mogućnost izmjene cijene i bodova ukoliko pregleda sliku i shvati da je sistem napravio manju grešku.



Analiza sistema preporuka za EcoChallenge platformu

Zaključak

Kreirao sam sofisticiran hibridni sistem preporuka koji predstavlja inovativnu primjenu AI i machine learning tehnologija u domenu upravljanja otpadom. Sistem uspješno kombinuje tri komplementarne komponente:

- AI-powered content analizu kroz Azure Vision Service.
- Naprednu machine learning regresiju sa kontinuiranim učenjem.
- Inteligentnu rule-based logiku kao sigurnosnu mrežu.

Najveća snaga ovog pristupa leži u njegovoj praktičnosti i robusnosti. Sistem nije teoretski eksperiment već produkcijski spreman alat koji može odmah početi da radi i postepeno se poboljšava. Hibridna priroda osigurava da nikad ne prestane da funkcioniše, dok kontinuirano učenje garantuje da vremenom postaje precizniji.

Za buduća poboljšanja, preporučio bih dodavanje collaborative filtering komponenti za personalizaciju preporuka i implementaciju feedback mehanizama koji bi omogućili korisnicima da ocijene kvalitet dobijenih nagrada, što bi dodatno poboljšalo preciznost sistema.