DOKUMENTACIJA SISTEMA PREPORUKE

Aplikacija: BringMeHome
Student: IB200124, Amina Bardak

Svrha

Sistem preporuke aplikacije BringMeHome koristi mašinsko učenje (ML) kako bi se generisale personalizovane preporuke životinja za korisnike. Glavni cilj je da se poveća vjerovatnoća usvajanja, prikazujući korisnicima životinje koje najviše odgovaraju njihovim ranijim interakcijama i preference (životinje koje su dodali u favorite).

Tehnologije

- .NET ML.NET biblioteka
- Algoritam: Matrix Factorization (Matrična faktorizacija)

Ovaj algoritam je idealan za sisteme preporuka. On uči skrivene karakteristike iz korisničkih preference i na osnovu toga predviđa vjerovatnoću interakcije za nove.

Proces rada

Sistem preporuke funkcioniše kroz tri glavne faze: inicijalizaciju, trening i predikciju.

Kada se servis pokrene, metoda **EnsureModelIsLoadedOrTrained()** provjerava da li već postoji sačuvan, obučeni model u fajlu recommendation_model.zip.
Ako model postoji servis ga učitava direktno iz fajla. Ovo značajno ubrzava startovanje aplikacije i smanjuje opterećenje. Ako model ne postoji servis automatski pokreće proces treninga od nule.

Metoda **TrainAndSaveModel()** prikuplja sve podatke o favorizovanim životinjama (UserAnimalFavorites) iz baze podataka. Svaki par (korisnik, životinja) se tretira kao pozitivna interakcija. Ovi podaci se koriste za obuku Matrix Factorization modela.

Nakon uspješne obuke, model se serijalizuje i čuva kao .zip fajl. Ovo omogućava da se model ponovo koristi bez potrebe za ponovnim treningom, sve dok se ne prikupi veći broj novih podataka koji bi pokrenuli obnavljanje modela.

Metoda **GetRecommendationsForUser(userId, numberOfRecommendations)** služi za dobijanje preporuka za konkretnog korisnika. Prvo se filtriraju životinje koje su korisniku već omiljene. Za svaku preostalu, dostupnu životinju, model generiše Score koji predstavlja vjerovatnoću da će se ta životinja dopasti korisniku. Životinje se sortiraju po opadajućem Score-u. Servis vraća listu AnimalResponse objekata, uzimajući prvih **numberOfRecommendations** sa najvišim Score-om.

Putanja servisa sistema preporuke: ***bring_me_home\BringMeHome\BringMeHome.Services\Services***
Putanja sistema preporuke na frontend:
bring_me_home\bringmehomeUI\bringmehome_android\lib\screens

```
public void TrainAndSaveModel()
{
    lock (_lock)
    {
        _logger.LogInformation("Starting ML model training...");

        var userFavorites = _context.UserAnimalFavorites
                                    .Select(fav => new AnimalInteraction
                                    {
                                        UserId = (uint)fav.UserId,
                                        AnimalId = (uint)fav.AnimalId,
                                        Label = 1.0f
                                    }).ToList();

        if (!userFavorites.Any())
        {
            _logger.LogWarning("No user favorites found for ML model training. Model will not be trained.");
            _model = null;
            return;
        }

        var trainingDataView = _mlContext.Data.LoadFromEnumerable(userFavorites);

        var options = new MatrixFactorizationTrainer.Options
        {
            MatrixColumnIndexColumnName = nameof(AnimalInteraction.UserId),
            MatrixRowIndexColumnName = nameof(AnimalInteraction.AnimalId),
            LabelColumnName = nameof(AnimalInteraction.Label),
            LossFunction = MatrixFactorizationTrainer.LossFunctionType.SquareLossOneClass,
            Alpha = 0.01,
            Lambda = 0.025,
            NumberOfIterations = 100,
            Quiet = true
        };

        var pipeline = _mlContext.Recommendation().Trainers.MatrixFactorization(options);

        _model = pipeline.Fit(trainingDataView);
        _logger.LogInformation("ML model training finished.");

        _mlContext.Model.Save(_model, trainingDataView.Schema, ModelPath);
        _logger.LogInformation("ML model saved to {ModelPath}.", ModelPath);
    }
}
```

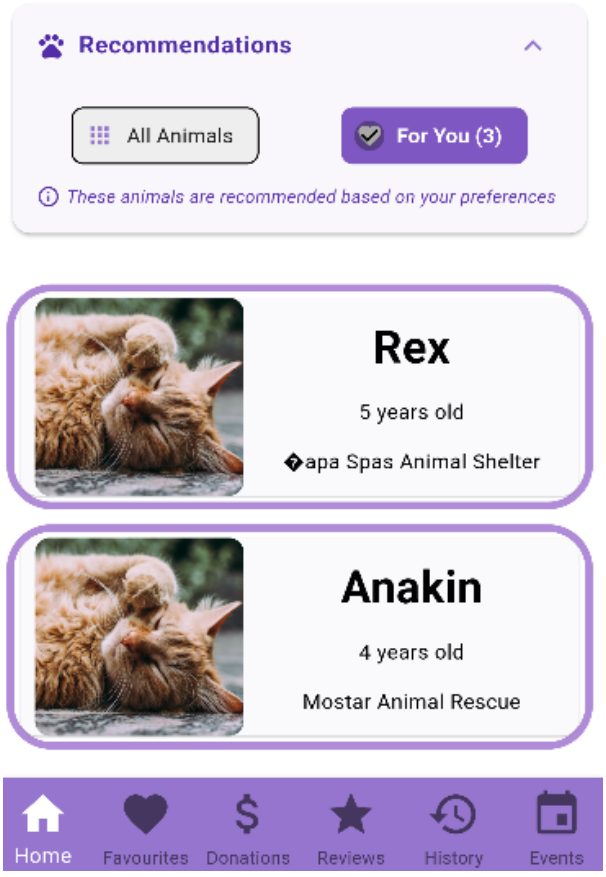
1. Source kod treniranja modela

```
public List<AnimalResponse> GetRecommendationsForUser(int userId, int numberOfRecommendations)
{
    if (_model == null)
    {
        _logger.LogWarning("ML model not available for predictions. Returning empty list.");
        return new List<AnimalResponse>();
    }

    var predictionEngine = _mlContext.Model.CreatePredictionEngine<AnimalInteraction, AnimalRecommendationPrediction>(_model);

    var userFavoritedAnimalIds = _context.UserAnimalFavorites
                                         .Where(f => f.UserId == userId)
                                         .Select(f => f.AnimalId)
                                         .ToHashSet();

    var availableAnimals = _context.Animals
        .Include(a => a.Breed)
            .ThenInclude(b => b.Species)
        .Include(a => a.Shelter)
        .Include(a => a.Status)
        .Include(a => a.Color)
        .Include(a => a.AnimalTemperament)
        .Where(a => !userFavoritedAnimalIds.Contains(a.AnimalID) && a.StatusID == GetAvailableStatusId())
        .ToList();

    if (!availableAnimals.Any())
    {
        _logger.LogInformation("No available animals for user {UserId} after filtering favorited ones.", userId);
        return new List<AnimalResponse>();
    }

    var predictions = new List<(Animal Animal, float Score)>();

    foreach (var animal in availableAnimals)
    {
        var prediction = predictionEngine.Predict(new AnimalInteraction
        {
            UserId = (uint)userId,
            AnimalId = (uint)animal.AnimalID
        });
        predictions.Add((animal, prediction.Score));
    }

    var recommendedAnimals = predictions
        .OrderByDescending(p => p.Score)
        .Select(p => MapToResponse(p.Animal))
        .Take(numberOfRecommendations)
        .ToList();

    _logger.LogInformation("Generated {Count} recommendations for user {UserId}.", recommendedAnimals.Count, userId);
    return recommendedAnimals;
}
```

2. Source kod dobivanje preporuke za specifičnog korisnika

```
protected override async Task ExecuteAsync(CancellationToken stoppingToken)
{
    _logger.LogInformation("ML Model Retraining Service running.");

    while (!stoppingToken.IsCancellationRequested)
    {
        _logger.LogInformation("ML Model Retraining Service: Waiting for next retraining cycle...");
        await Task.Delay(_retrainingInterval, stoppingToken);

        if (stoppingToken.IsCancellationRequested)
        {
            break;
        }

        _logger.LogInformation("ML Model Retraining Service: Initiating model retraining.");

        using (var scope = _scopeFactory.CreateScope())
        {
            try
            {
                var mlRecommendationService = scope.ServiceProvider.GetRequiredService<MlRecommendationService>();
                mlRecommendationService.TrainAndSaveModel();
                _logger.LogInformation("ML Model Retraining Service: Model retraining completed successfully.");
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "ML Model Retraining Service: Error during model retraining.");
            }
        }
    }

    _logger.LogInformation("ML Model Retraining Service stopped.");
}
```

3. Source kod za ponovno obučavanje modela



4. Izgled preporuka na mobilnoj aplikaciji za korisnika