Recommender system

1. Implementacija recommender sistema

Implementacija recommender sistema se nalazi na putanji - \eVetCare\eVetCare.Services\RecommendationService.cs

```
public void GenerateRecommendationsForPet(int petId)
{
    var ownerId = _context.Pets
        .Where(p => p.PetId == petId)
        .Select(p => p.OwnerId)
        .FirstOrDefault();

    var trainingData = _context.InvoiceItems
        .Join(_context.Invoices, ii => ii.InvoiceId, i => i.InvoiceId, (ii, i) => new { i.AppointmentId, ii.ServiceId })
        .Join(_context.Appointments, x => x.AppointmentId, a => a.AppointmentId, (x, a) => new { a.Pet.OwnerId, x.ServiceId })
        .Select(x => new ServiceRating
        {
            OwnerId = x.OwnerId,
            ServiceId = x.ServiceId,
            Label = 1f
        })
        .ToList();

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

    var pipeline = _mlContext.Transforms.Conversion
        .MapValueToKey("OwnerIdEncoded", nameof(ServiceRating.OwnerId))
        .Append(_mlContext.Transforms.Conversion.MapValueToKey("ServiceIdEncoded", nameof(ServiceRating.ServiceId)))
        .Append(_mlContext.Recommendation().Trainers.MatrixFactorization(new MatrixFactorizationTrainer.Options
        {
            MatrixRowIndexColumnName = "OwnerIdEncoded",
            MatrixColumnIndexColumnName = "ServiceIdEncoded",
            LabelColumnName = "Label",
            NumberOfIterations = 20,
            ApproximationRank = 100
        }));

    var model = pipeline.Fit(dataView);
```

```
    var predictionEngine = _mlContext.Model.CreatePredictionEngine<ServiceRating, ServiceScore>(model);

    var allServiceIds = _context.Services.Select(s => s.ServiceId).ToList();

    var predictions = allServiceIds
        .Select(sid => new
        {
            ServiceId = sid,
            Score = predictionEngine.Predict(new ServiceRating { OwnerId = ownerId, ServiceId = sid }).Score
        })
        .OrderByDescending(x => x.Score)
        .Take(3)
        .ToList();

    var recommendedServices = _context.Services
        .Where(s => predictions.Select(p => p.ServiceId).Contains(s.ServiceId))
        .Select(s => s.Name)
        .ToList();

    var content = "Recommended services: " + string.Join(", ", recommendedServices);

    _context.Recommendations.Add(new Recommendation
    {
        PetId = petId,
        Content = content,
        CreatedAt = DateTime.Now
    });

    _context.SaveChanges();
}
```

2. Ideja

Napravljen je recommender system za veterinarsku aplikaciju. Ideja je da se na osnovu onoga sto su vlasnici ljubimaca ranije koristili kao usluge, predlozi koje bi usluge mogle biti potrebne/zanimljive vlasniku odredjenog ljubimca.

3. Kako to radi

Iz baze se uzimaju parovi (OwnerId, ServiceId) iz historije (fakture → stavke fakture → termini → ljubimci) i trenira se model kolaborativnog filtriranja (matrix factorization) u ML.NET-u. Zatim za konkretnog vlasnika se izracuna skor svidjanja za sve usluge, poslozi se prema skoru i uzmu top preporuke.

4. Koraci

1. Nadjem vlasnika za datog ljubimca (petId)

Prvo iz tabele ljubimaca dobijem OwnerId za dati PetId, jer preporuke pravim za vlasnika (vlasnik placa usluge).

```
public void GenerateRecommendationsForPet(int petId)
{
    var ownerId = _context.Pets
        .Where(p => p.PetId == petId)
        .Select(p => p.OwnerId)
        .FirstOrDefault();
```

2. Sastavim trening podatke iz baze

Spojim InvoiceItems → Invoices → Appointments → Pets i izvucem sve kombinacije vlasnik -> usluga koje su se stvarno desile. Svaku takvu kombinaciju tretiram kao pozitivan signal.

```
var trainingData = _context.InvoiceItems
    .Join(_context.Invoices, ii => ii.InvoiceId, i => i.InvoiceId, (ii, i) => new { i.AppointmentId, ii.ServiceId })
    .Join(_context.Appointments, x => x.AppointmentId, a => a.AppointmentId, (x, a) => new { a.Pet.OwnerId, x.ServiceId })
    .Select(x => new ServiceRating
    {
        OwnerId = x.OwnerId,
        ServiceId = x.ServiceId,
        Label = 1f
    })
    .ToList();
```

3. Treniram model preporuka (matrix factorization)

```
.Append(_mlContext.Recommendation().Trainers.MatrixFactorization(new MatrixFactorizationTrainer.Options
{
    MatrixRowIndexColumnName = "OwnerIdEncoded",
    MatrixColumnIndexColumnName = "ServiceIdEncoded",
    LabelColumnName = "Label",
    NumberOfIterations = 20,
    ApproximationRank = 100
}));

model = pipeline.Fit(dataView);
```

4. Izracunam skor za sve usluge I uzmem top N

Za konkretnog vlasnika prodjem kroz sve usluge, izracunam skor, sortiram od najveceg prema manjem, I uzmem npr. top tri.

```
    var predictionEngine = _mlContext.Model.CreatePredictionEngine<ServiceRating, ServiceScore>(model);

    var allServiceIds = _context.Services.Select(s => s.ServiceId).ToList();

    var predictions = allServiceIds
        .Select(sid => new
        {
            ServiceId = sid,
            Score = predictionEngine.Predict(new ServiceRating { OwnerId = ownerId, ServiceId = sid }).Score
        })
        .OrderByDescending(x => x.Score)
        .Take(3)
        .ToList();
```

5. Snimim rezultat u bazu

Od ServiceId iz top liste dohvatim nazive usluga i zapišem u tabelu Recommendations kao tekst (npr. “Preporučene usluge: …”), zajedno s vremenom kreiranja. Tako kasnije lako prikažem preporuku u UI-ju.

```
        var recommendedServices = _context.Services
            .Where(s => predictions.Select(p => p.ServiceId).Contains(s.ServiceId))
            .Select(s => s.Name)
            .ToList();

        var content = "Recommended services: " + string.Join(", ", recommendedServices);

        _context.Recommendations.Add(new Recommendation
        {
            PetId = petId,
            Content = content,
            CreatedAt = DateTime.Now
        });

        _context.SaveChanges();
    }
```