My recommender system

Preporuke u mom recommender systemu kreiraju se na osnovu podataka o narudzbama i stavkama narudzbe. Na primjer ukoliko neki korisnik cesto kupuje proizvod 1 i proizvod 2 zajedno, sistem ce preporucivati proizvod 2 ukoliko neko kupi proizvod 1. Metode koje su koristene u mom recommender sistemu su vezane za generisanje preporuke za dati proizvod, treniranje modela i cuvanje preporuka u bazu, te brisanje svih preporuka iz baze.

Preporuke proizvoda

Generisati cemo preporuke za dati proizvod. Kreira se model u recommender sistemu ukoliko vec nije inicijalizovan. Analizirati ce kupovine koje korisnik izvrsi te nauciti koji proizvodi idu zajedno. Nakon toga koristi se trenirani model koji ce preporuciti tri njabolja proizvoda. Ako narudzba sadrzi vise vise proizvoda kreirati ce parove koji su kupljeni zajedno nakon cega ce te parove spremiti u ratingentry listu.

```
lock (isLocked)
{
    if (mlContext == null)
    {
        mlContext = new MLContext();

        var tmpData = _context.Narudzbas.Include("StavkaNarudzbes").ToList();

        var data = new List<RatingEntry>();

        foreach (var x in tmpData)
        {
            if (x.StavkaNarudzbes.Count > 1)
            {
                var distinctItemId = x.StavkaNarudzbes.Select(y => y.ProizvodId).ToList();

                distinctItemId.ForEach(y =>
                {
                    var relatedItems = x.StavkaNarudzbes.Where(z => z.ProizvodId != y).ToList();

                    foreach (var z in relatedItems)
                    {
                        data.Add(new RatingEntry()
                        {
                            RatingId = (uint)y,
                            CoRatingId = (uint)z.ProizvodId,
                        });
```

Treniranje modela

```
    var traindata = mlContext.Data.LoadFromEnumerable(data);
    MatrixFactorizationTrainer.Options options = new MatrixFactorizationTrainer.Options();
    options.MatrixColumnIndexColumnName = nameof(RatingEntry.RatingId);
    options.MatrixRowIndexColumnName = nameof(RatingEntry.CoRatingId);
    options.LabelColumnName = "Label";
    options.LossFunction = MatrixFactorizationTrainer.LossFunctionTy
    options.Alpha = 0.01;
    options.Lambda = 0.025;
    options.NumberOfIterations = 100;
    options.C = 0.00001;

    var est = mlContext.Recommendation().Trainers.MatrixFactorization(options);

    modeltr = est.Fit(traindata);
}
```

uint RatingEntry.CoRatingId { get; set; }

Potrebno je pretvoriti data listu u ML.NET dataset nakon cega ce se definisati matrix factorization model. Definisati ce se stepen ucenja, regularizacija te broj iteracija nakon cega ce se pokrenuti trening modela.

Predikcija i generisanje preporuka

```
var allItems = _context.Proizvods.Where(x => x.ProizvodId != id);
var predictionResult = new List<Tuple<Database.Proizvod, float>>();

foreach (var item in allItems)
{
    var predictionEngine = mlContext.Model.CreatePredictionEngine<RatingEntry, Copurchase_prediction>(modeltr);
    var prediction = predictionEngine.Predict(new RatingEntry()
    {
        RatingId = (uint)id,
        CoRatingId = (uint)item.ProizvodId
    });

    predictionResult.Add(new Tuple<Database.Proizvod, float>(item, prediction.Score));
}
```

Uzimaju se svi porizvodi osim trenutnog. Model ce predvidjeti koliko je vjerovatno da su proizvodi kuljeni zajedno i poslije toga ce ih dodati u predicstionResult listu.

Vracanje 3 najbolje preporuke

```
var finalResult = predictionResult.OrderByDescending(x => x.Item2).Select(x => x.Item1).Take(3).ToList();

if (finalResult != null)
    return _mapper.Map<List<Model.Proizvod>>(finalResult);
return null;
```

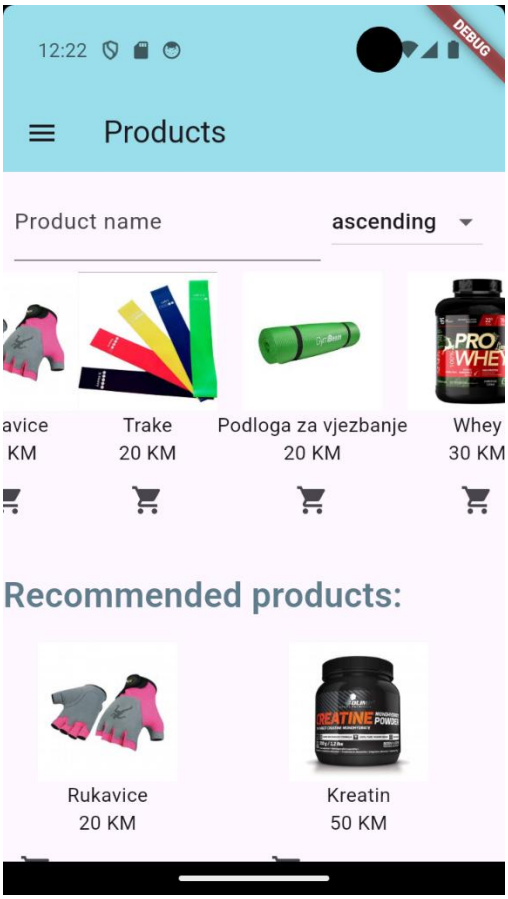
Sortirati ce preporuke po tacnosti nakon cega ce vratiti tri najbolje preporuke.

Trenira cijeli model i sprema rezultate u bazu

```
var stavkeNarudzbe = _context.StavkaNarudzbes.ToList();
var proizvodi = _context.Proizvods.ToList();

if (proizvodi.Count > 4 && stavkeNarudzbe.Count() > 2)
{
    List<Database.RecommendResult> recommendList = new List<Database.RecommendResult>();

    foreach (var proizvod in proizvodi)
    {

        var recommendedProducts = Recommend(proizvod.ProizvodId);


        var resultRecommend = new Database.RecommendResult()
        {
            ProizvodId = proizvod.ProizvodId,
            PrviProizvodId = recommendedProducts[0].ProizvodId,
            DrugiProizvodId = recommendedProducts[1].ProizvodId,
            TreciProizvodId = recommendedProducts[2].ProizvodId,
        };
        recommendList.Add(resultRecommend);
    }

    var list = _context.RecommendResults.ToList();
    var recordCount = list.Count();
    var proizvodiCount = _context.Proizvods.Count();
    if (recordCount != 0)
    {
        if (recordCount > proizvodiCount)
        {
            for (int i = 0; i < proizvodiCount; i++)
            {
                list[i].ProizvodId = recommendList[i].ProizvodId;
                list[i].PrviProizvodId = recommendList[i].PrviProizvodId;
                list[i].DrugiProizvodId = recommendList[i].DrugiProizvodId;
                list[i].TreciProizvodId = recommendList[i].TreciProizvodId;
            }
```

```
            for (int i = proizvodiCount; i < recordCount; i++)
            {
                _context.RecommendResults.Remove(list[i]);
            }
        }
        else
        {
            for (int i = 0; i < recordCount; i++)
            {
                list[i].ProizvodId = recommendList[i].ProizvodId;
                list[i].PrviProizvodId = recommendList[i].PrviProizvodId;
                list[i].DrugiProizvodId = recommendList[i].DrugiProizvodId;
                list[i].TreciProizvodId = recommendList[i].TreciProizvodId;
            }
            var num = recommendList.Count() - recordCount;

            if (num > 0)
            {
                for (int i = recommendList.Count() - num; i < recommendList.Count(); i++)
                {
                    _context.RecommendResults.Add(recommendList[i]);
                }
            }
        }
    }
    else
    {
        _context.RecommendResults.AddRange(recommendList);
    }
    await _context.SaveChangesAsync();
    return _mapper.Map<List<Model.RecommendResult>>(recommendList);
}
else
{
    throw new Exception("Not enough data to do recommmedation");
}
```

Za svaki proizvod pravi preporuke i sprema ih u bazu. Provjerava da li postoji dovoljno podataka prije nego krene treniranje.


12:22
DEBUG
Products
Product name
ascending
avice
KM
Trake
20 KM
Podloga za vjezbanje
20 KM
Whey
30 KM
Recommended products:
Rukavice
20 KM
Kreatin
50 KM