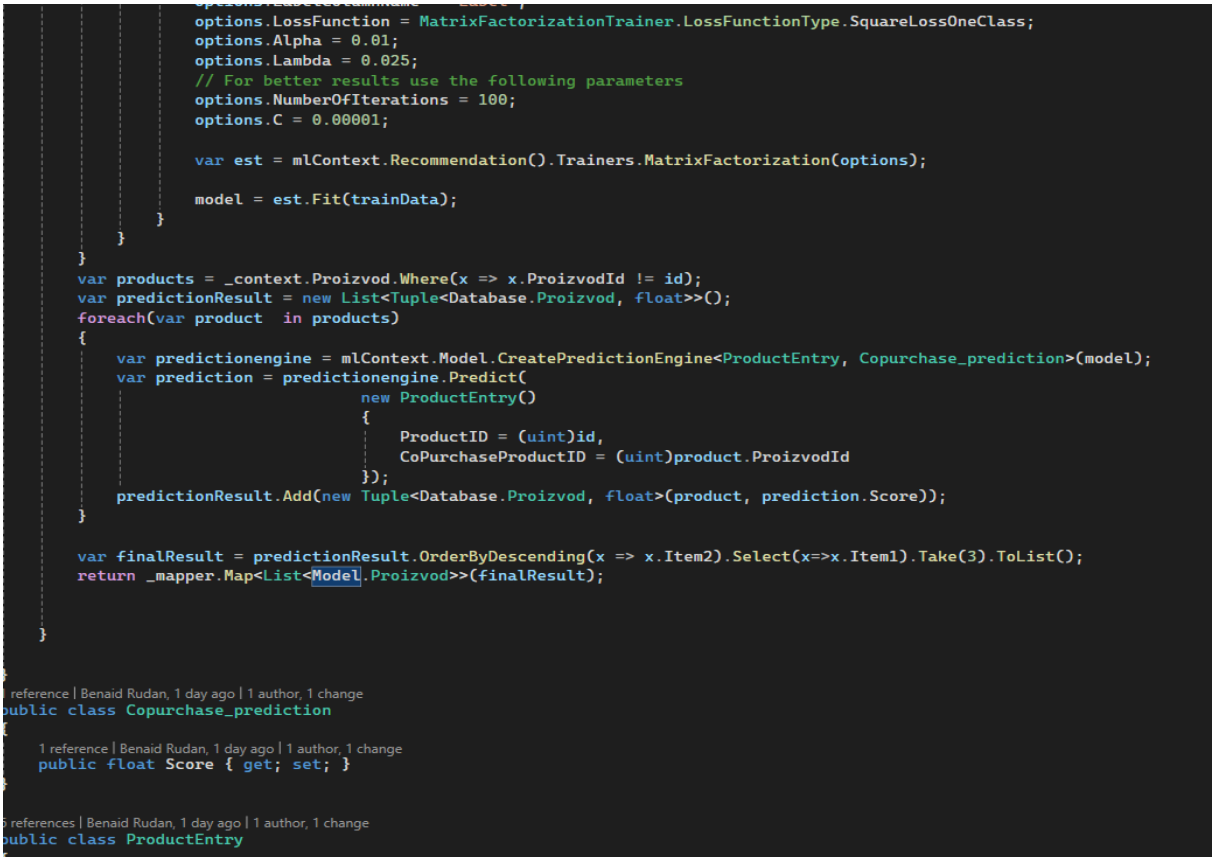
SISTEM PREPORUKE - Matrix Factorization algoritam

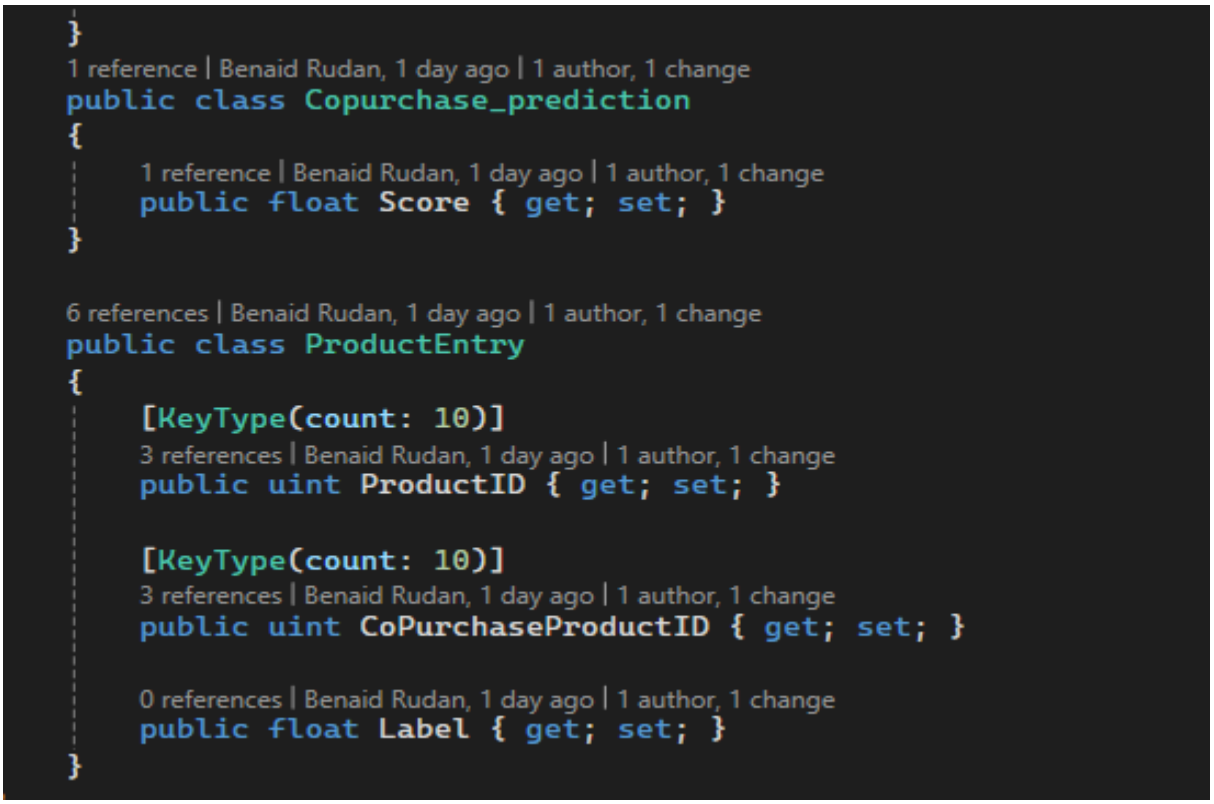
Sistem preporuke koristi Matrix Factorization algoritam iz ML.NET biblioteke za preporučivanje proizvoda na osnovu ko-kupovine (proizvodi koji se često kupuju zajedno). Algoritam analizira historijske narudžbe kako bi pronašao skrivene veze između proizvoda i predviđa koliko je vjerovatno da će korisnik kupiti određeni proizvod ako je već kupio drugi. Na kraju, sistem vraća top 3 preporučena proizvoda sortirana po ocjeni (score) dobijenoj iz modela.

Putanja: C:\Users\DT User\source\repos\RSII_Seminarski\eBarbershop.Services\ProizvodService.cs

```
static MLContext mlContext = null;
static object isLocked = new object();
static ITransformer model = null;

2 references | Benaid Rudan, 1 day ago | 1 author, 1 change
public List<Model.Proizvod> Recommend(int id)
{
    lock(isLocked)
    {
        if (mlContext == null)
        {
            mlContext = new MLContext();
            var tmpData = _context.Narudzba.Include("NarudzbaProizvodis").ToList();
            var data = new List<ProductEntry>();
            foreach (var x in tmpData)
            {
                if (x.NarudzbaProizvodis.Count > 1)
                {
                    var distinctItemId = x.NarudzbaProizvodis.Select(y => y.ProizvodId)
                        .ToList();
                    distinctItemId.ForEach(y =>
                    {
                        var relatedItems = x.NarudzbaProizvodis.Where(z => z.ProizvodId != y);
                        foreach(var z in relatedItems)
                        {
                            data.Add(new ProductEntry()
                            {
                                ProductID = (uint)y,
                                CoPurchaseProductID = (uint)z.ProizvodId
                            });
                        }
                    });
                }
                var trainData = mlContext.Data.LoadFromEnumerable(data);

                MatrixFactorizationTrainer.Options options = new MatrixFactorizationTrainer.Options();
                options.MatrixColumnIndexColumnName = nameof(ProductEntry.ProductID);
                options.MatrixRowIndexColumnName = nameof(ProductEntry.CoPurchaseProductID);
                options.LabelColumnName = "Label";
                options.LossFunction = MatrixFactorizationTrainer.LossFunctionType.SquareLossOneClass;
                options.Alpha = 0.01;
                options.Lambda = 0.025;
```



```
                options.LossFunction = MatrixFactorizationTrainer.LossFunctionType.SquareLossOneClass;
                options.Alpha = 0.01;
                options.Lambda = 0.025;
                // For better results use the following parameters
                options.NumberOfIterations = 100;
                options.C = 0.00001;

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

                model = est.Fit(trainData);
            }
        }
    }
    var products = _context.Proizvod.Where(x => x.ProizvodId != id);
    var predictionResult = new List<Tuple<Database.Proizvod, float>>();
    foreach(var product  in products)
    {
        var predictionengine = mlContext.Model.CreatePredictionEngine<ProductEntry, Copurchase_prediction>(model);
        var prediction = predictionengine.Predict(
                                new ProductEntry()
                                {
                                    ProductID = (uint)id,
                                    CoPurchaseProductID = (uint)product.ProizvodId
                                });
        predictionResult.Add(new Tuple<Database.Proizvod, float>(product, prediction.Score));
    }

    var finalResult = predictionResult.OrderByDescending(x => x.Item2).Select(x=>x.Item1).Take(3).ToList();
    return _mapper.Map<List<Model.Proizvod>>(finalResult);


}

1 reference | Benaid Rudan, 1 day ago | 1 author, 1 change
public class Copurchase_prediction
{
    1 reference | Benaid Rudan, 1 day ago | 1 author, 1 change
    public float Score { get; set; }
}

6 references | Benaid Rudan, 1 day ago | 1 author, 1 change
public class ProductEntry
```



```
}
1 reference | Benaid Rudan, 1 day ago | 1 author, 1 change
public class Copurchase_prediction
{
    1 reference | Benaid Rudan, 1 day ago | 1 author, 1 change
    public float Score { get; set; }
}

6 references | Benaid Rudan, 1 day ago | 1 author, 1 change
public class ProductEntry
{
    [KeyType(count: 10)]
    3 references | Benaid Rudan, 1 day ago | 1 author, 1 change
    public uint ProductID { get; set; }

    [KeyType(count: 10)]
    3 references | Benaid Rudan, 1 day ago | 1 author, 1 change
    public uint CoPurchaseProductID { get; set; }

    0 references | Benaid Rudan, 1 day ago | 1 author, 1 change
    public float Label { get; set; }
}
```

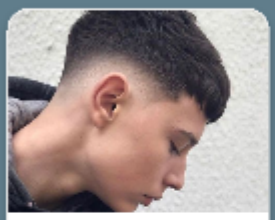
Putanja gdje se prikazuju preporuke: C:\Users\DT User\source\repos\RSII_Seminarski\UI\ebarbershop_mobile\lib\screens\home_screen.dart

eBarbershop admin

Dobrodošao admin

Početna strana Galerija

Najnovije vijesti



Dobrodošli u eBa...

Radujemo se vašim posjetama u našem no...

Nova linija proizv...

Uveli smo nove profesionalne proizvod...

Preporučeni proizvodi

Brijačka krema

10,50 KM

Šampon za suvu

15,00 KM

Ulje za

Početna Trgovina Usluge Recenzije Profil Termini