Implementacija sistema preporuke

Korisnici imaju preporuku za nabavku sljedceg proizvoda na osnovu proizvoda koji su narucili. Navedeni sistem preopuke radi na principu conten-based filtering metode.   
  
Backend:

static MLContext mlContext = null;

static object isLocked = new object();

static ITransformer model = null;

public List<Product> Recommended(int id)

{

lock (isLocked)

{

if (mlContext == null)

{

mlContext = new MLContext();

var tmpData = \_context.Orders.Include("OrderItems").ToList();

var data = new List<ProductEntry>();

int num = 0;

foreach (var x in tmpData)

{

if (x.OrderItems.Count > 1)

{

var distinctItemId = x.OrderItems.Select(y => y.ProductId).ToList();

distinctItemId.ForEach(y =>

{

var relatedItems = x.OrderItems.Where(z => z.ProductId != y);

foreach (var z in relatedItems)

{

data.Add(new ProductEntry()

{

ProductID = (uint)y,

CoPurchaseProductID = (uint)z.ProductId,

});

}

});

}

}

if (data.Count == 0)

{

return null;

}

var trainData = mlContext.Data.LoadFromEnumerable(data);

MatrixFactorizationTrainer.Options options = new MatrixFactorizationTrainer.Options();

options.MatrixColumnIndexColumnName = nameof(ProductEntry.ProductID);

options.MatrixRowIndexColumnName = nameof(ProductEntry.CoPurchaseProductID);

options.LabelColumnName = nameof(ProductEntry.Label);

options.LossFunction = MatrixFactorizationTrainer.LossFunctionType.SquareLossOneClass;

options.Alpha = 0.01;

options.Lambda = 0.025;

options.NumberOfIterations = 100;

options.C = 0.00001;

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

model = est.Fit(trainData);

}

if (model == null)

{

return null;

}

var products = \_context.Products.Where(x => x.ProductId != id);

var predictionResult = new List<Tuple<Database.Product, 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.ProductId,

});

predictionResult.Add(new Tuple<Database.Product, float>(product, prediction.Score));

}

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

return Mapper.Map<List<Product>>(finalResult);

}

}

public class Copurchase\_prediction

{

public float Score { get; set; }

}

public class ProductEntry

{

[KeyType(count: 100)]

public uint ProductID { get; set; }

[KeyType(count: 100)]

public uint CoPurchaseProductID { get; set; }

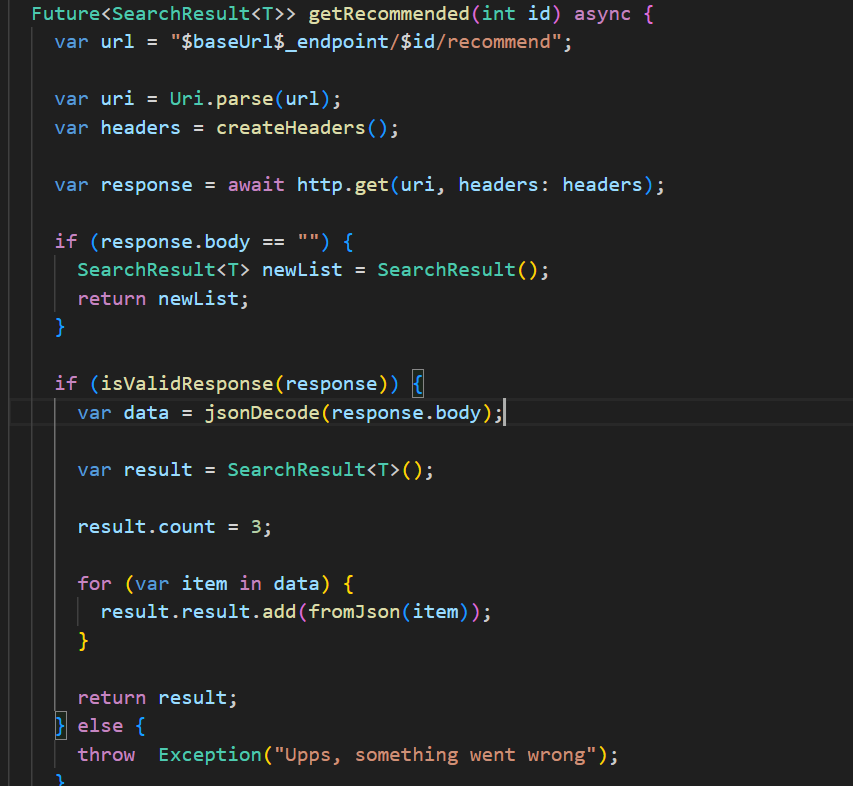
public float Label { get; set; }

}

}

Frontend

Request unutar ProductProvider-a:

  
  
Screen na kojem se prikazuju preporuceni proizvodi:   
