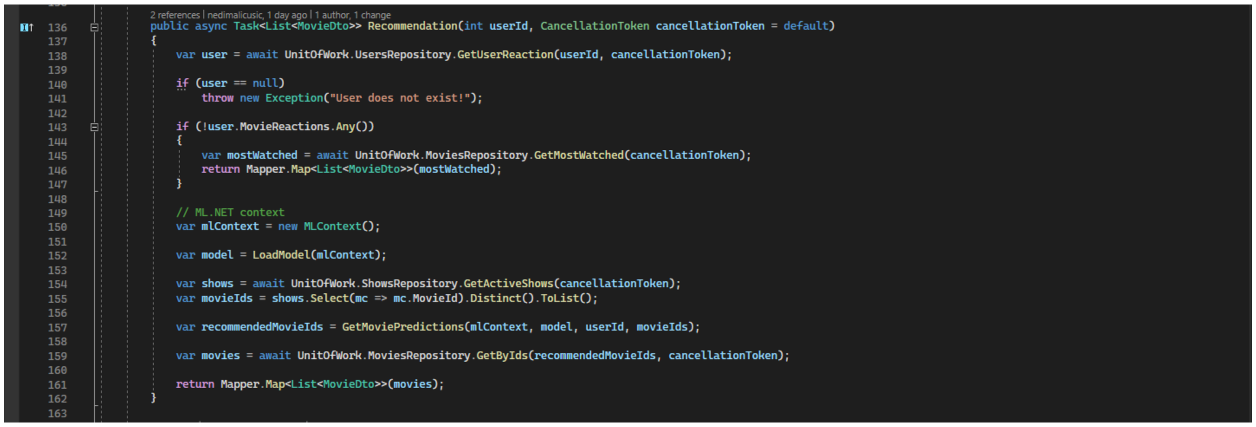
SISTEM PREPORUKE - Matrix Factorization algoritam

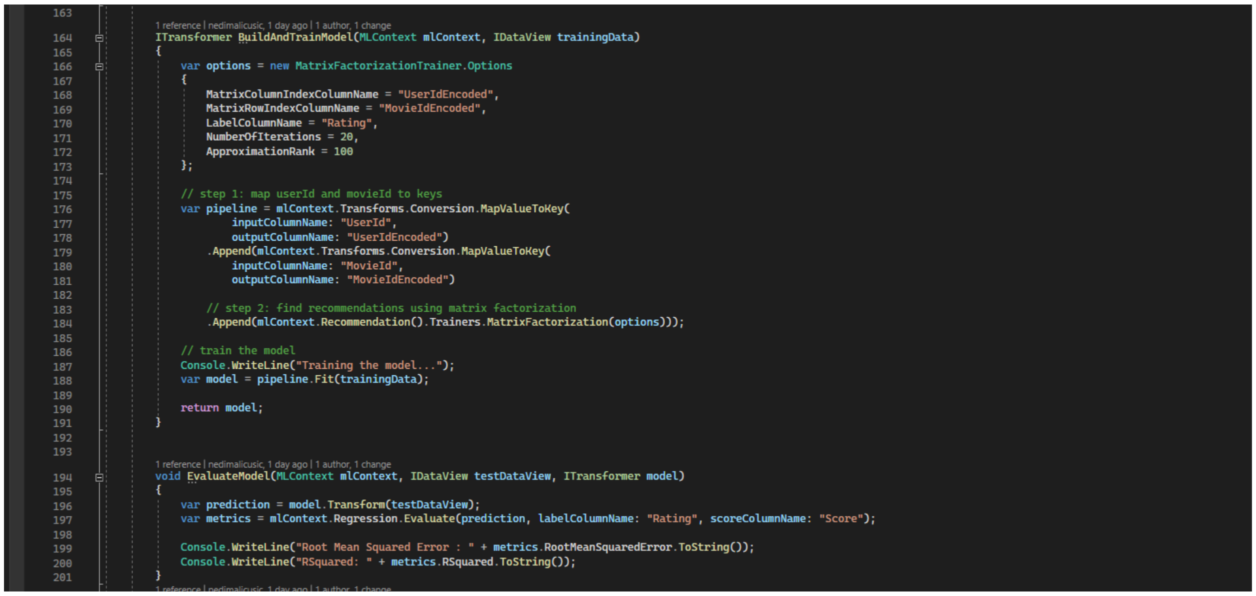
Sistem preporuke koristi Matrix Factorization algoritam iz ML.NET biblioteke kako bi predvidio koje filmove će korisnik najvjerovatnije željeti da pogleda. Model analizira historijske reakcije korisnika na filmove (ocjene) i uči skrivene obrasce između korisnika i filmova, poput žanrova, stilova ili tipova sadržaja koje određeni korisnici preferiraju. Kada korisnik zatraži preporuke, sistem izračunava očekivanu ocjenu (score) za svaki dostupni film i rangira ih prema vjerovatnoći da će se korisniku dopasti. Na kraju, vraća se lista top 3 preporučena filma sortirana prema najvišoj predviđenoj ocjeni.

Putanja: C:\Users\DT

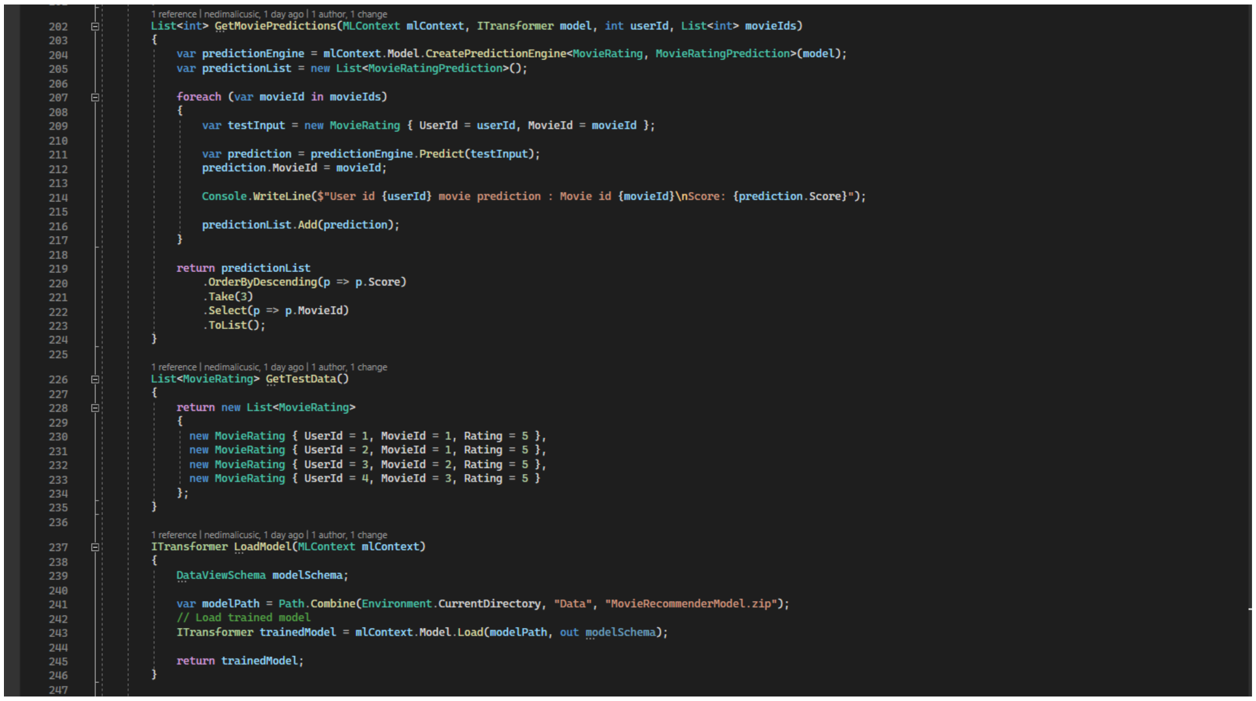
User\source\repos\eCinema\eCinema-Seminarski\eCinema\eCinema.Application\Services\MoviesService.cs



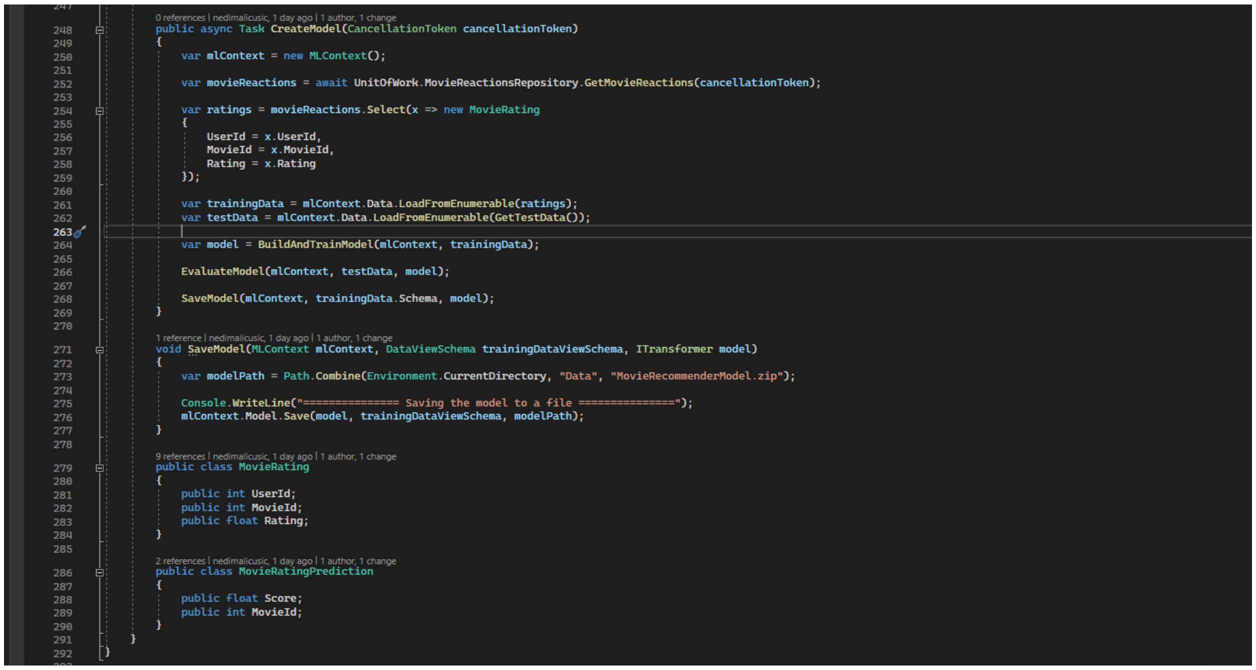
```
2 references | nedimalicusic, 1 day ago | 1 author, 1 change
public async Task<List<MovieDto>> Recommendation(int userId, CancellationToken cancellationToken = default)
{
    var user = await UnitOfWork.UsersRepository.GetUserReaction(userId, cancellationToken);

    if (user == null)
        throw new Exception("User does not exist!");

    if (!user.MovieReactions.Any())
    {
        var mostWatched = await UnitOfWork.MoviesRepository.GetMostWatched(cancellationToken);
        return Mapper.Map<List<MovieDto>>(mostWatched);
    }

    // ML.NET context
    var mlContext = new MLContext();

    var model = LoadModel(mlContext);

    var shows = await UnitOfWork.ShowsRepository.GetActiveShows(cancellationToken);
    var movieIds = shows.Select(mc => mc.MovieId).Distinct().ToList();

    var recommendedMovieIds = GetMoviePredictions(mlContext, model, userId, movieIds);

    var movies = await UnitOfWork.MoviesRepository.GetByIds(recommendedMovieIds, cancellationToken);

    return Mapper.Map<List<MovieDto>>(movies);
}
```



```
1 reference | nedimalicusic, 1 day ago | 1 author, 1 change
ITransformer BuildAndTrainModel(MLContext mlContext, IDataView trainingData)
{
    var options = new MatrixFactorizationTrainer.Options
    {
        MatrixColumnIndexColumnName = "UserIdEncoded",
        MatrixRowIndexColumnName = "MovieIdEncoded",
        LabelColumnName = "Rating",
        NumberOfIterations = 20,
        ApproximationRank = 100
    };

    // step 1: map userId and movieId to keys
    var pipeline = mlContext.Transforms.Conversion.MapValueToKey(
            inputColumnName: "UserId",
            outputColumnName: "UserIdEncoded")
        .Append(mlContext.Transforms.Conversion.MapValueToKey(
            inputColumnName: "MovieId",
            outputColumnName: "MovieIdEncoded")

        // step 2: find recommendations using matrix factorization
        .Append(mlContext.Recommendation().Trainers.MatrixFactorization(options)));

    // train the model
    Console.WriteLine("Training the model...");
    var model = pipeline.Fit(trainingData);

    return model;
}


1 reference | nedimalicusic, 1 day ago | 1 author, 1 change
void EvaluateModel(MLContext mlContext, IDataView testDataView, ITransformer model)
{
    var prediction = model.Transform(testDataView);
    var metrics = mlContext.Regression.Evaluate(prediction, labelColumnName: "Rating", scoreColumnName: "Score");

    Console.WriteLine("Root Mean Squared Error : " + metrics.RootMeanSquaredError.ToString());
    Console.WriteLine("RSquared: " + metrics.RSquared.ToString());
}
```

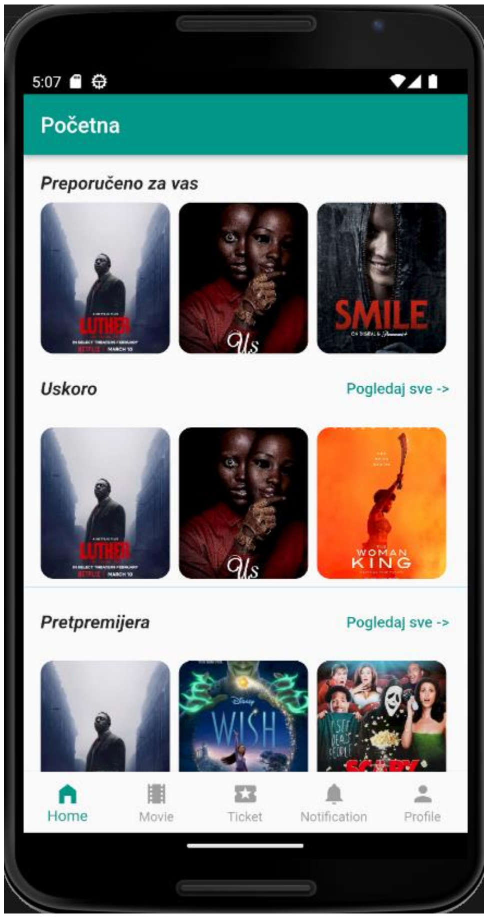


```
1 reference | nedimalicusic, 1 day ago | 1 author, 1 change
List<int> GetMoviePredictions(MLContext mlContext, ITransformer model, int userId, List<int> movieIds)
{
    var predictionEngine = mlContext.Model.CreatePredictionEngine<MovieRating, MovieRatingPrediction>(model);
    var predictionList = new List<MovieRatingPrediction>();

    foreach (var movieId in movieIds)
    {
        var testInput = new MovieRating { UserId = userId, MovieId = movieId };

        var prediction = predictionEngine.Predict(testInput);
        prediction.MovieId = movieId;

        Console.WriteLine($"User id {userId} movie prediction : Movie id {movieId}\nScore: {prediction.Score}");

        predictionList.Add(prediction);
    }

    return predictionList
        .OrderByDescending(p => p.Score)
        .Take(3)
        .Select(p => p.MovieId)
        .ToList();
}

1 reference | nedimalicusic, 1 day ago | 1 author, 1 change
List<MovieRating> GetTestData()
{
    return new List<MovieRating>
    {
      new MovieRating { UserId = 1, MovieId = 1, Rating = 5 },
      new MovieRating { UserId = 2, MovieId = 1, Rating = 5 },
      new MovieRating { UserId = 3, MovieId = 2, Rating = 5 },
      new MovieRating { UserId = 4, MovieId = 3, Rating = 5 }
    };
}

1 reference | nedimalicusic, 1 day ago | 1 author, 1 change
ITransformer LoadModel(MLContext mlContext)
{
    DataViewSchema modelSchema;

    var modelPath = Path.Combine(Environment.CurrentDirectory, "Data", "MovieRecommenderModel.zip");
    // Load trained model
    ITransformer trainedModel = mlContext.Model.Load(modelPath, out modelSchema);

    return trainedModel;
}
```



```
        0 references | nedimalicusic, 1 day ago | 1 author, 1 change
        public async Task CreateModel(CancellationToken cancellationToken)
        {
            var mlContext = new MLContext();

            var movieReactions = await UnitOfWork.MovieReactionsRepository.GetMovieReactions(cancellationToken);

            var ratings = movieReactions.Select(x => new MovieRating
            {
                UserId = x.UserId,
                MovieId = x.MovieId,
                Rating = x.Rating
            });

            var trainingData = mlContext.Data.LoadFromEnumerable(ratings);
            var testData = mlContext.Data.LoadFromEnumerable(GetTestData());

            var model = BuildAndTrainModel(mlContext, trainingData);

            EvaluateModel(mlContext, testData, model);

            SaveModel(mlContext, trainingData.Schema, model);
        }

        1 reference | nedimalicusic, 1 day ago | 1 author, 1 change
        void SaveModel(MLContext mlContext, DataViewSchema trainingDataViewSchema, ITransformer model)
        {
            var modelPath = Path.Combine(Environment.CurrentDirectory, "Data", "MovieRecommenderModel.zip");

            Console.WriteLine("=============== Saving the model to a file ===============");
            mlContext.Model.Save(model, trainingDataViewSchema, modelPath);
        }

        9 references | nedimalicusic, 1 day ago | 1 author, 1 change
        public class MovieRating
        {
            public int UserId;
            public int MovieId;
            public float Rating;
        }

        2 references | nedimalicusic, 1 day ago | 1 author, 1 change
        public class MovieRatingPrediction
        {
            public float Score;
            public int MovieId;
        }
    }
}
```

Putanja gdje se prikazuju preporuke: C:\Users\DT User\source\repos\eCinema\eCinema-Seminarski\UI\eCinema_mobile\lib\screens\HomeScreen.dart


5:07
Početna
Preporučeno za vas
Uskoro
Pogledaj sve ->
Pretpremijera
Pogledaj sve ->
Home
Movie
Ticket
Notification
Profile