Diamond

Write code

(Regression-FastTree)

In this session

1. Task = Write C# program to predict diamond price
2. Create project
3. Add NuGet packages
4. Add using name space
5. Create data set input/output scheme
6. Set data set path
7. Create context
8. Load train data set
9. Create pipeline

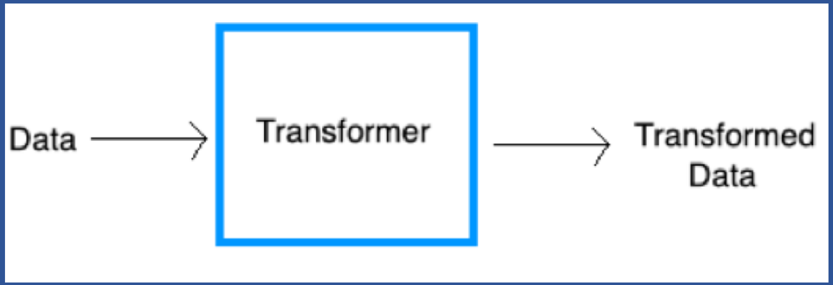
10. Transform data to numbers
11. Drop non-feature
12. Choose a learning algorithm
13. Train the model
14. Loads the test dataset
15. Creates the regression evaluator
16. Evaluates the model and creates metrics
17. Make Prediction

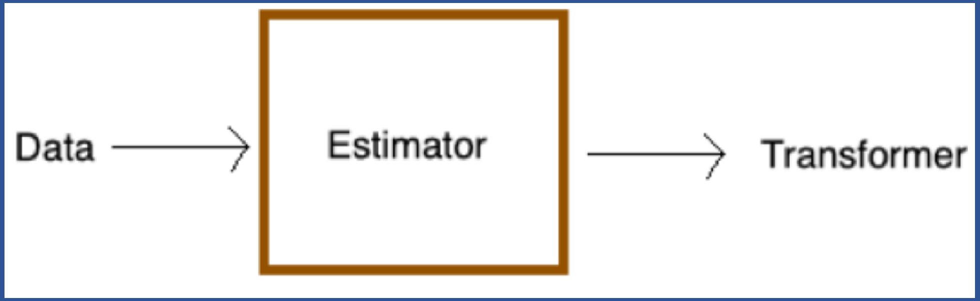
Data, Transformers, and Estimators

Data = tabular dataset

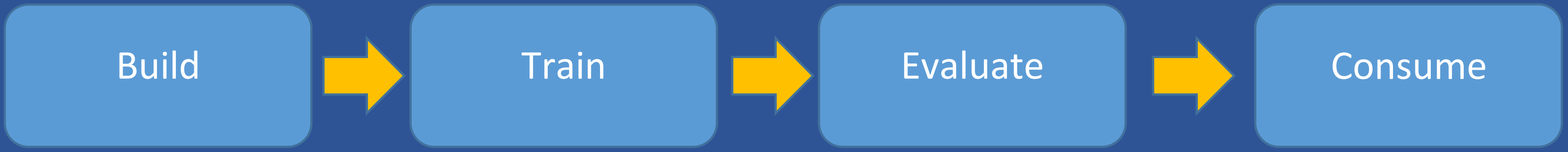
Transformer = transform dataset to compatible format

Estimator = user for create Transformers





Work flow

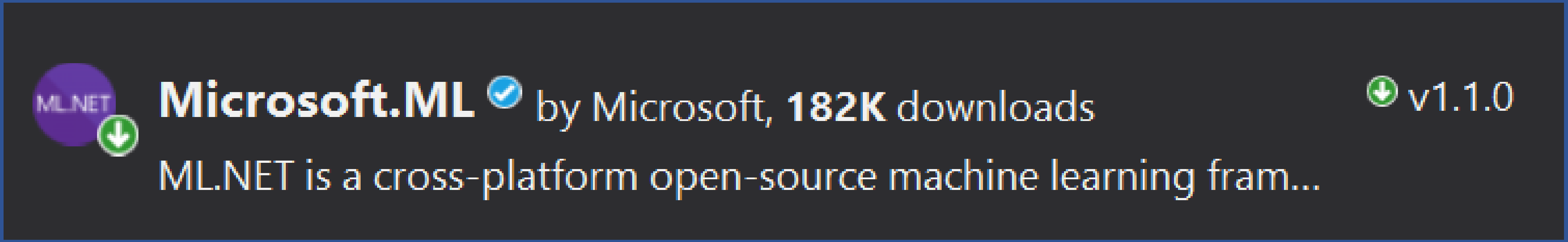


Write C# program to predict diamond price

Create new .NET CORE console app project name = “DiamondLarge”

Add NuGet Package

- Microsoft.ML
- Microsoft.ML.Fast


Microsoft.ML by Microsoft, **182K** downloads v1.1.0
ML.NET is a cross-platform open-source machine learning fram...

Prepare data

Put Data Set in to D:\ml folder

Train Data Set

https://raw.githubusercontent.com/laploy/ML.NET/master/Diamond%20Large/diamonds-Large-Train.csv

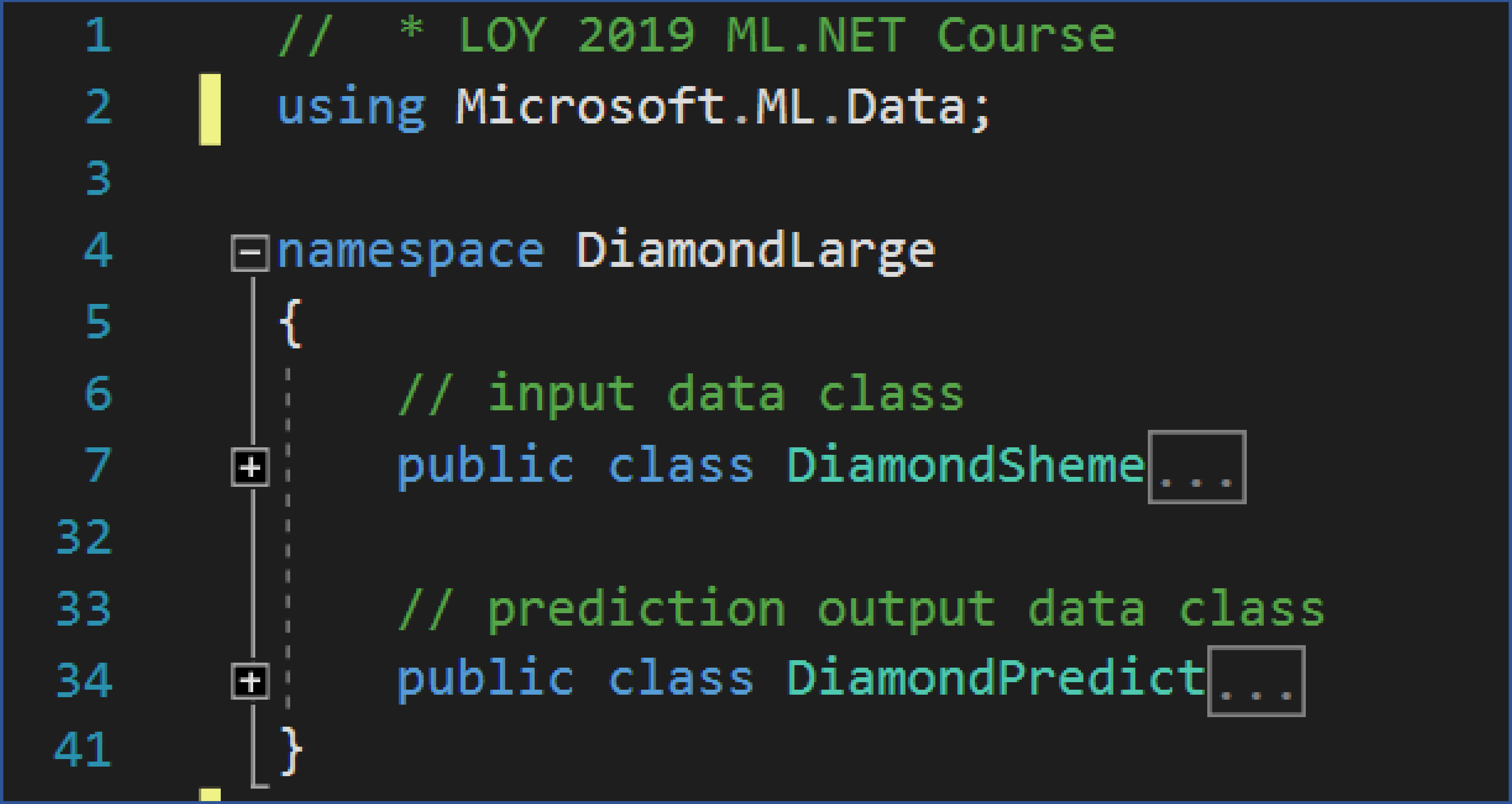
Score Data Set

https://github.com/laploy/ML.NET/blob/master/Diamond%20Large/diamonds-Large-Score.csv

Test Data Set

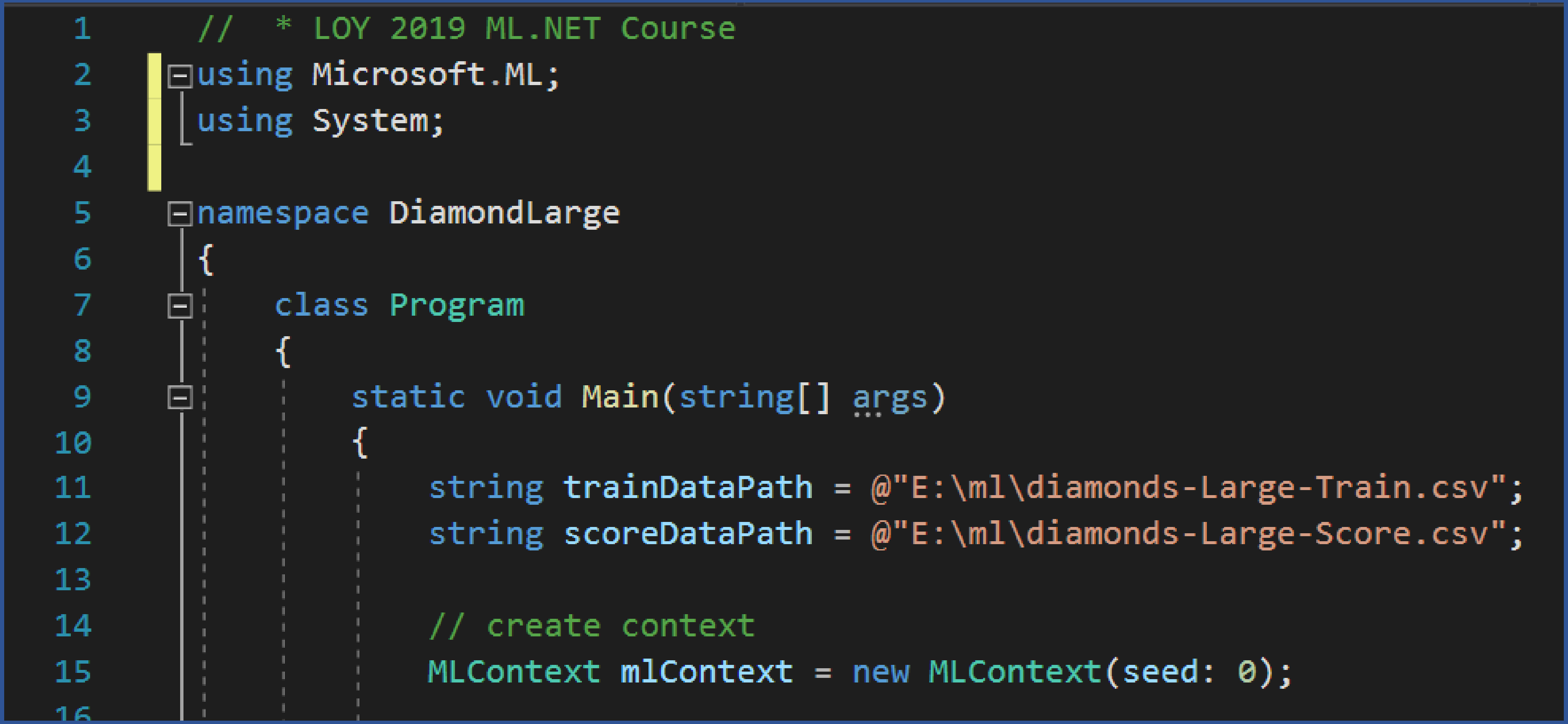
https://github.com/laploy/ML.NET/blob/master/Diamond%20Large/diamonds-Large-Test.csv

Create data set input/output scheme



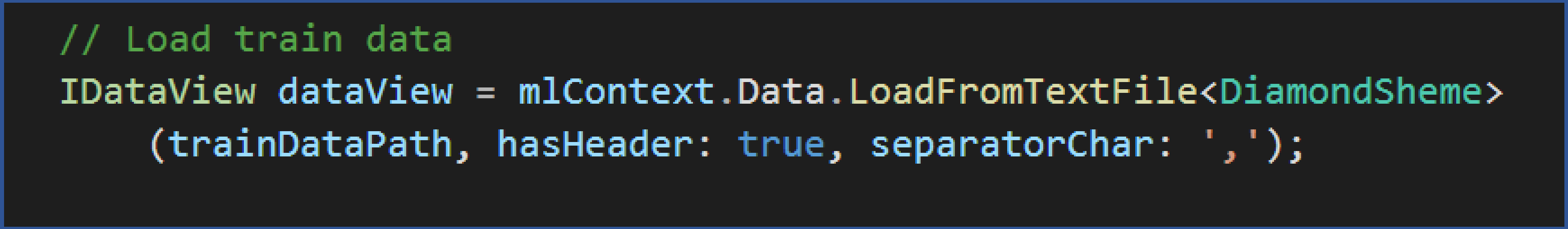
```
//  * LOY 2019 ML.NET Course
using Microsoft.ML.Data;

namespace DiamondLarge
{
    // input data class
    public class DiamondSheme...

    // prediction output data class
    public class DiamondPredict...
}
```

Create ML context



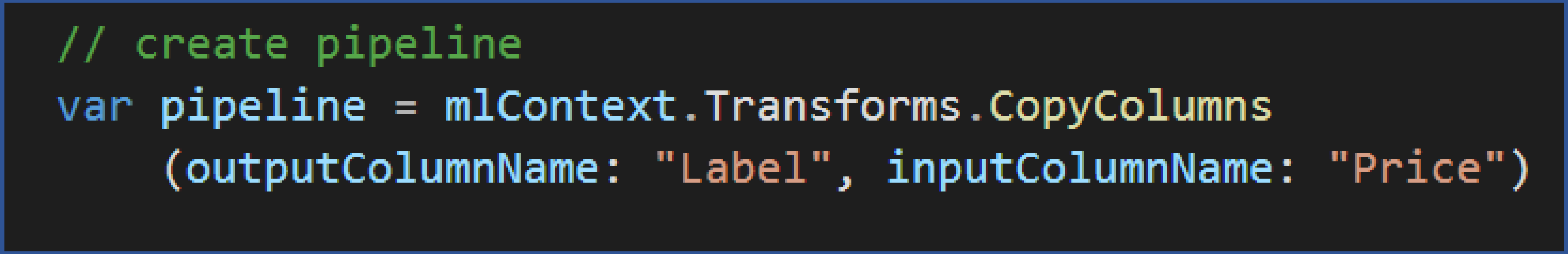
```
//  * LOY 2019 ML.NET Course
using Microsoft.ML;
using System;

namespace DiamondLarge
{
    class Program
    {
        static void Main(string[] args)
        {
            string trainDataPath = @"E:\ml\diamonds-Large-Train.csv";
            string scoreDataPath = @"E:\ml\diamonds-Large-Score.csv";

            // create context
            MLContext mlContext = new MLContext(seed: 0);
```

Load train data set



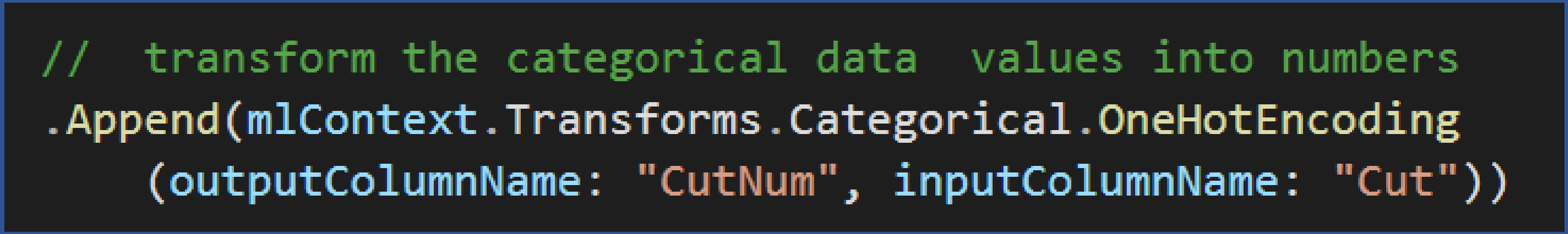
```
// Load train data
IDataView dataView = mlContext.Data.LoadFromTextFile<DiamondSheme>
    (trainDataPath, hasHeader: true, separatorChar: ',');
```

Create pipeline



```
// create pipeline
var pipeline = mlContext.Transforms.CopyColumns
    (outputColumnName: "Label", inputColumnName: "Price")
```

Transform data to numbers

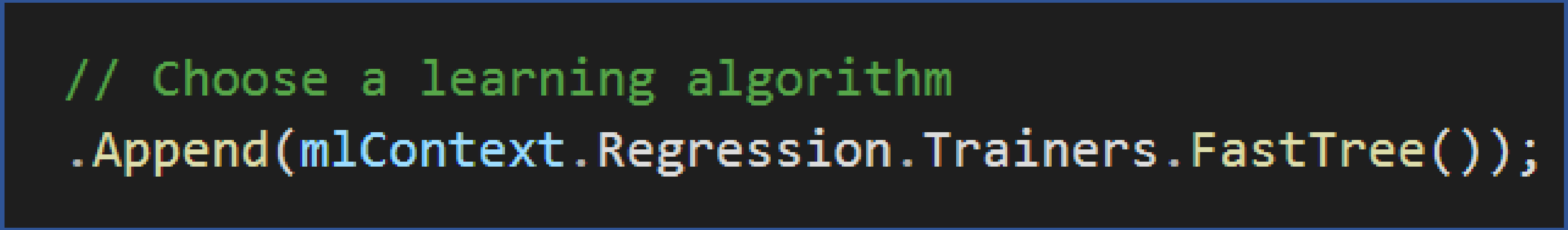


```
//  transform the categorical data  values into numbers
.Append(mlContext.Transforms.Categorical.OneHotEncoding
    (outputColumnName: "CutNum", inputColumnName: "Cut"))
```

Drop non-feature

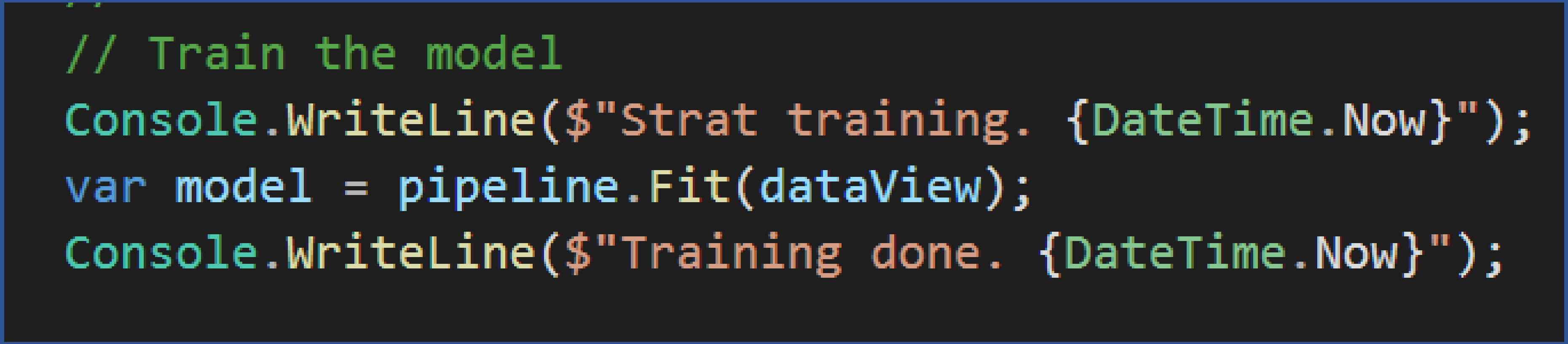
```
// combines all of the feature columns into the Features column
.Append(mlContext.Transforms.Concatenate
    ("Features", "Carat", "CutNum", "ColorNum", "ClearityNum", "Depth",
    "Table", "LengthX", "WidthY", "DepthZ"))
```

Choose a learning algorithm



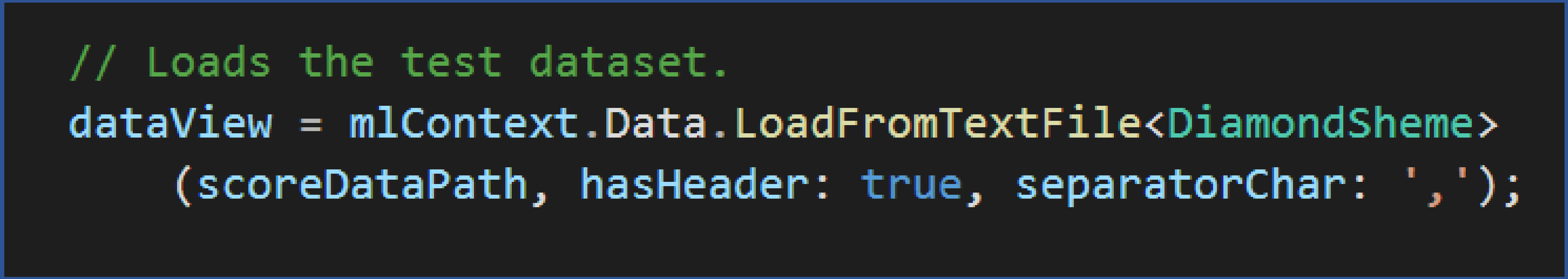
```
// Choose a learning algorithm
.Append(mlContext.Regression.Trainers.FastTree());
```

Train the model



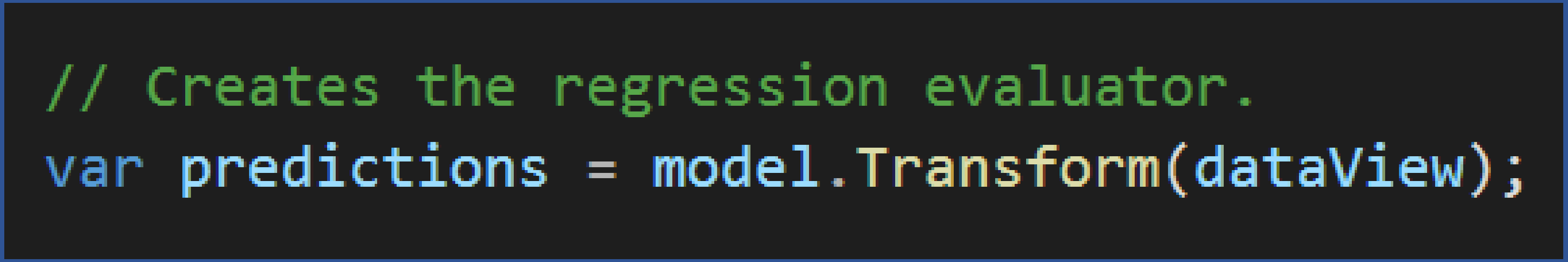
```
// Train the model
Console.WriteLine($"Strat training. {DateTime.Now}");
var model = pipeline.Fit(dataView);
Console.WriteLine($"Training done. {DateTime.Now}");
```

Loads the test dataset



```
// Loads the test dataset.
dataView = mlContext.Data.LoadFromTextFile<DiamondSheme>
    (scoreDataPath, hasHeader: true, separatorChar: ',');
```

Creates the regression evaluator



```
// Creates the regression evaluator.
var predictions = model.Transform(dataView);
```

Evaluates the model and creates metrics

```
// Evaluates the model and creates metrics.
var metrics = mlContext.Regression.Evaluate(predictions, "Label", "Score");

Console.WriteLine();
Console.WriteLine($"*************************************************");
Console.WriteLine($"* Model quality metrics evaluation ");
Console.WriteLine($"*------------------------------------------------");

// RSquared is another evaluation metric of the regression models.
// RSquared takes values between 0 and 1.The closer its value is to 1, the better the model is
Console.WriteLine($"* RSquared Score: {metrics.RSquared:0.##}");

// RMS is one of the evaluation metrics of the regression model.
// The lower it is, the better the model is
Console.WriteLine($"* Root Mean Squared Error: {metrics.RootMeanSquaredError:#.##}");
```

Make Prediction

```
// create engine
var predictionFunction = mlContext.Model.CreatePredictionEngine
    <DiamondSheme, DiamondPredict>(model);

// make one test diamond data we want to predict
var myPredict = new DiamondSheme()
{
    Id = "34973",
    Carat = 0.83f,
    Cut = "Very Good",
    Color = "G",
    Clearity = "SI2",
    Depth = 63.1f,
    Table = 61f,
    Price = 2259f,
    LengthX = 5.96f,
    WidthY = 5.92f,
    DepthZ = 3.75f
};

// make prediction
var prediction = predictionFunction.Predict(myPredict);
```

What's next ?

Create and save ML model