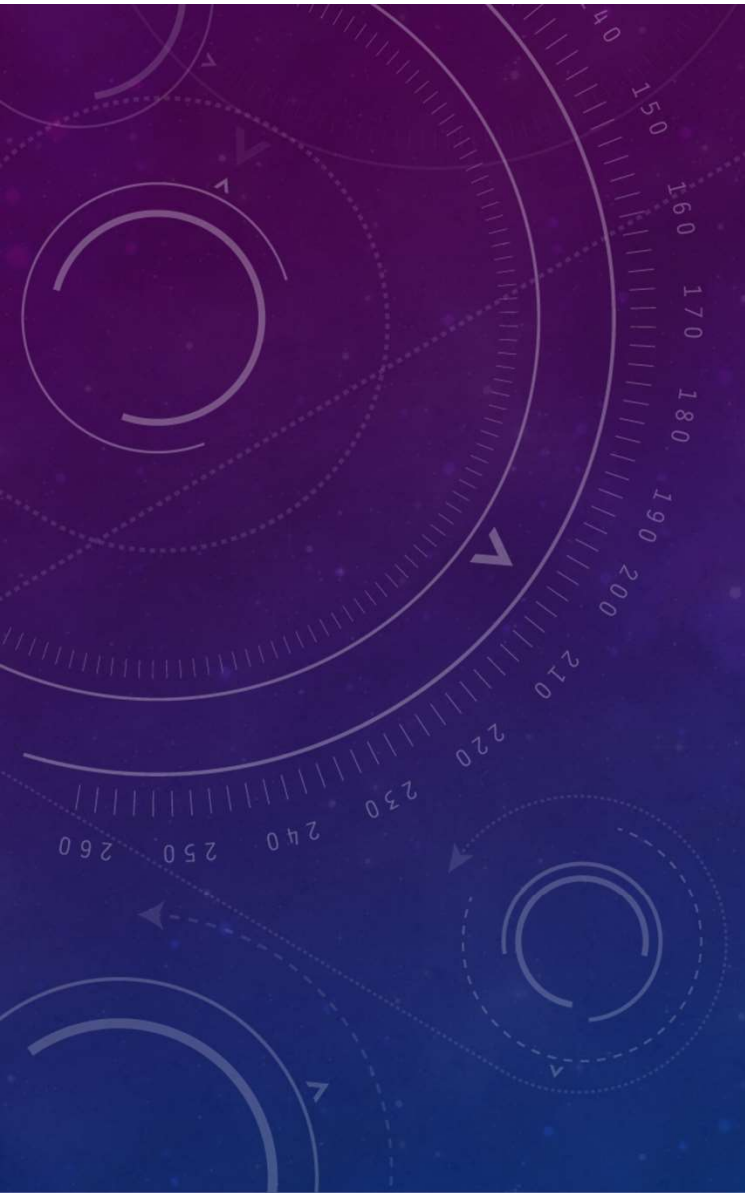


MACHINE LEARNING WITH .NET

Asmaliza Ahzan @ Emma

asmaliza@iverson.com.my

SETUP AND FIRST ML.NET APPLICATION

CHAPTER 2

SETUP ENVIRONMENT

- Visual Studio 2019
- Configurations/workloads
  - .NET Core (min: 3)
  - .NET Desktop Development
  - Universal Windows Platform Development
  - ASP.NET and Web Development
  - .NET Core Cross Platform Development

YOUR FIRST ML.NET APPLICATION

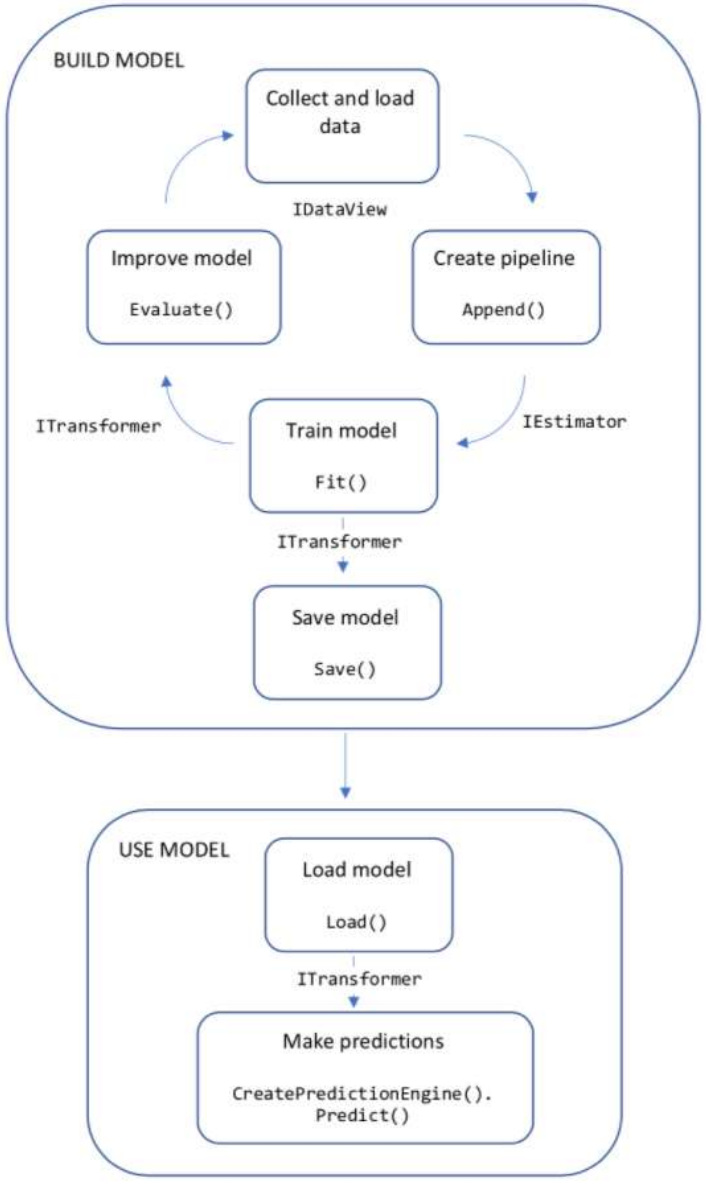
- Create a new .NET Core console application.
- Add Microsoft ML framework to the project.
  - Manage NuGet Packages > Microsoft.ML > Install
- Hello ML.NET World
  - This example constructs a linear regression model to predict house prices using house size and price data.

C# Copy

```
using System;
using Microsoft.ML;
using Microsoft.ML.Data;

class Program
{
    public class HouseData
    {
        public float Size { get; set; }
        public float Price { get; set; }
    }

    public class Prediction
    {
        [ColumnName("Score")]
        public float Price { get; set; }
    }

    static void Main(string[] args)
    {
        MLContext mlContext = new MLContext();

        // 1. Import or create training data
        HouseData[] houseData = {
            new HouseData() { Size = 1.1F, Price = 1.2F },
            new HouseData() { Size = 1.9F, Price = 2.3F },
            new HouseData() { Size = 2.8F, Price = 3.0F },
            new HouseData() { Size = 3.4F, Price = 3.7F } };
        IDataView trainingData = mlContext.Data.LoadFromEnumerable(houseData);

        // 2. Specify data preparation and model training pipeline
        var pipeline = mlContext.Transforms.Concatenate("Features", new[] { "Size" })
            .Append(mlContext.Regression.Trainers.Sdca(labelColumnName: "Price", maximumNumberOfIterations: 100));

        // 3. Train model
        var model = pipeline.Fit(trainingData);

        // 4. Make a prediction
        var size = new HouseData() { Size = 2.5F };
        var price = mlContext.Model.CreatePredictionEngine<HouseData, Prediction>(model).Predict(size);

        Console.WriteLine($"Predicted price for size: {size.Size*1000} sq ft= {price.Price*100:C}k");

        // Predicted price for size: 2500 sq ft= $261.98k
    }
}
```

CODE FLOW

- Collect and load training data into an IDataView object
- Specify a pipeline of operations to extract features and apply a machine learning algorithm
- Train a model by calling Fit() on the pipeline
- Evaluate the model and iterate to improve
- Save the model into binary format, for use in an application
- Load the model back into an ITransformer object
- Make predictions by calling CreatePredictionEngine.Predict()


BUILD MODEL
Collect and load data
IDataView
Improve model
Evaluate()
Create pipeline
Append()
ITransformer
Train model
Fit()
IEstimator
ITransformer
Save model
Save()
USE MODEL
Load model
Load()
ITransformer
Make predictions
CreatePredictionEngine().
Predict()

MACHINE LEARNING MODEL

- An ML.NET model is an object that contains transformations to perform on your input data to arrive at the predicted output.
- The model specifies the steps needed to transform your input data into a prediction.
- With ML.NET, you can train a custom model by specifying an algorithm, or you can import pre-trained TensorFlow and ONNX models.

DATA PREPARATION

- In most cases, the data that you have available isn't suitable to be used directly to train a machine learning model.
- The raw data needs to be prepared, or pre-processed, before it can be used to find the parameters of your model.
- Your data may need to be converted from string values to a numerical representation. You might have redundant information in your input data.
- You may need to reduce or expand the dimensions of your input data.
- Your data might need to be normalized or scaled.

PIPELINE

- A machine learning pipeline is a way to codify and automate the workflow it takes to produce a machine learning model.
- Machine learning pipelines consist of multiple sequential steps that do everything from data extraction and preprocessing to model training and deployment.

TRAINING MODEL

- A machine learning training model is a process in which a machine learning (ML) algorithm is fed with sufficient training data to learn from.

MODEL EVALUATION

- Once you have trained your model, how do you know how well it will make future predictions?
- With ML.NET, you can evaluate your model against some new test data.
- Each type of machine learning task has metrics used to evaluate the accuracy and precision of the model against the test data set.

C# Copy

```
HouseData[] testHouseData =
{
    new HouseData() { Size = 1.1F, Price = 0.98F },
    new HouseData() { Size = 1.9F, Price = 2.1F },
    new HouseData() { Size = 2.8F, Price = 2.9F },
    new HouseData() { Size = 3.4F, Price = 3.6F }
};

var testHouseDataView = mlContext.Data.LoadFromEnumerable(testHouseData);
var testPriceDataView = model.Transform(testHouseDataView);

var metrics = mlContext.Regression.Evaluate(testPriceDataView, labelColumnName: "Price");

Console.WriteLine($"R^2: {metrics.RSquared:0.##}");
Console.WriteLine($"RMS error: {metrics.RootMeanSquaredError:0.##}");

// R^2: 0.96
// RMS error: 0.19
```

MODEL EVALUATION TECHNIQUES

- R-Squared
  - The most common interpretation of r-squared is how well the regression model fits the observed data. For example, an r-squared of 60% reveals that 60% of the data fit the regression model. Generally, a higher r-squared indicates a better fit for the model.
- Root Mean Square Error (RMSE)
  - Frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed.
  - RMSE is always non-negative, and a value of 0 (almost never achieved in practice) would indicate a perfect fit to the data.

PREDICTION TYPES IN BINARY CLASSIFICATION

- True negative: Properly classified as negative
- True positive: Properly classified as positive
- False negative: Improperly classified as negative
- False positive: Improperly classified as positive

EVALUATION METRICS

Metrics	Descriptions
Accuracy	This metric is calculated simply as the ratio of correctly classified predictions to total classifications.
Precision	Precision is defined as the proportion of true results over all the positive results in a model.
Recall	Recall is the fraction of all correct results returned by the model.
F-score	F-scores give another perspective on the performance of the model compared to simply looking at accuracy. The range of values is between 0 and 1, with an ideal value of 1.

EVALUATION METRICS

Metrics	Descriptions
Area Under the Curve (AUC)	The area under the curve plotted with true positives on the y-axis and false positives on the x-axis.
Average Log Loss and Training Log Loss	The average log loss is effectively expressing the penalty for wrong results in a single number by taking the difference between the true classification and the one the model predicts. Training log loss represents the uncertainty of the model using probability versus the known values.

THE END