



A. MỤC TIÊU:

- Sử dụng được thư viện ML.net của Microsoft
- Phân tích và ứng dụng thuật toán thông minh vào sản phẩm phần mềm

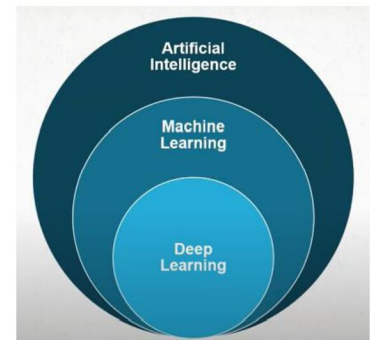
B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

C. NỘI DUNG THỰC HÀNH

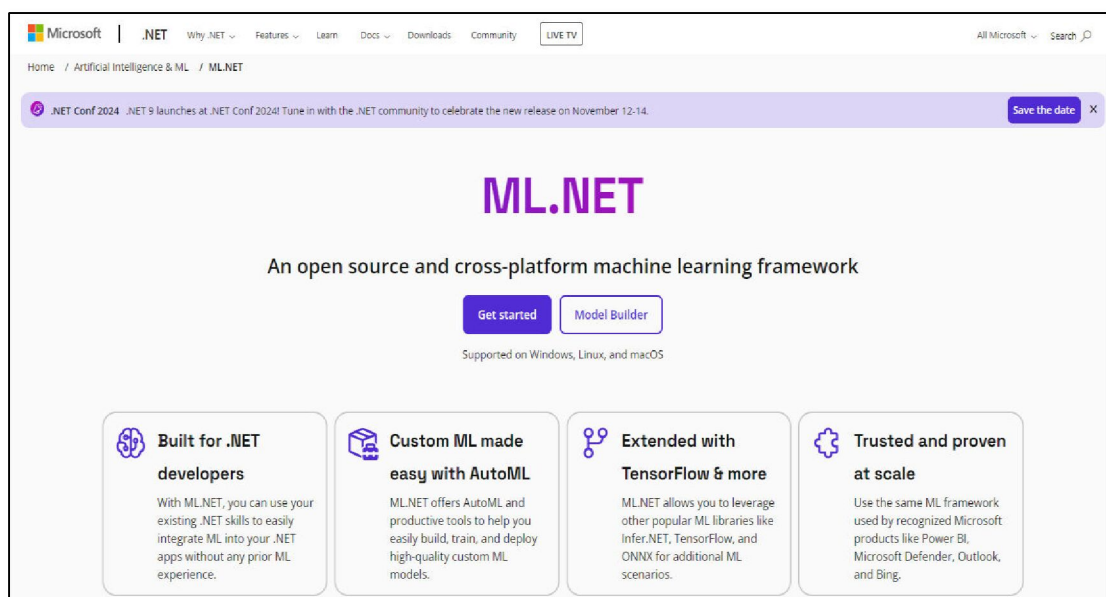
1. Tóm tắt lý thuyết

– Thư viện Machine Learning .Net (ML.net) là thư viện hỗ trợ các thuật toán liên quan tới AI và máy học của Microsoft, là thư viện mã nguồn mở (open-sources and cross-platform) và được ứng dụng trong nhiều lĩnh vực. Ví dụ: Google and Google App, Amazon, Computer Vision, Voice Assistants, Sentiment, Spam Detection, Fraud Detection, Advertisements, Self Driving Car, Forecasting, Tranlation, ...












– Có nhiều loại thuật toán nhưng tập trung ở 3 loại chính: Unsupervised Learning (Học không giám sát), Supervised Learning (Học có giám sát), Reinforcement Learning (Học tăng cường), ...

- Tham khảo về ML.net tại: [ML.NET | Machine learning made for .NET](https://ml.net)



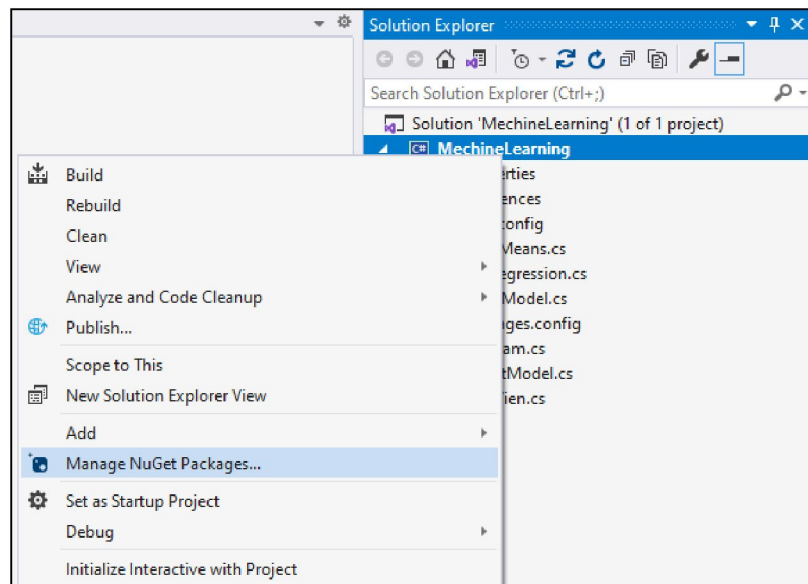
– ML.net hỗ trợ nhiều loại ứng dụng khác nhau và được sử dụng phổ biến trong thực tế (ML.net được hỗ trợ trên cả Net core framework và .Net framework):

 Sentiment analysis Analyze the sentiment of customer reviews using a binary classification algorithm.	 Product recommendation Recommend products based on purchase history using a matrix factorization algorithm.	 Price prediction Predict taxi fares based on parameters such as distance traveled using a regression algorithm.
 Customer segmentation Identify groups of customers with similar profiles using a clustering algorithm.	 Object detection Recognize objects in an image using an ONNX deep learning model.	 Fraud detection Detect fraudulent credit card transactions using a binary classification algorithm.
 Sales spike detection Detect spikes and changes in product sales using an anomaly detection model.	 Image classification Classify images (for example, broccoli vs. pizza) using a TensorFlow deep learning model.	 Sales forecasting Forecast future sales for products using a regression algorithm.

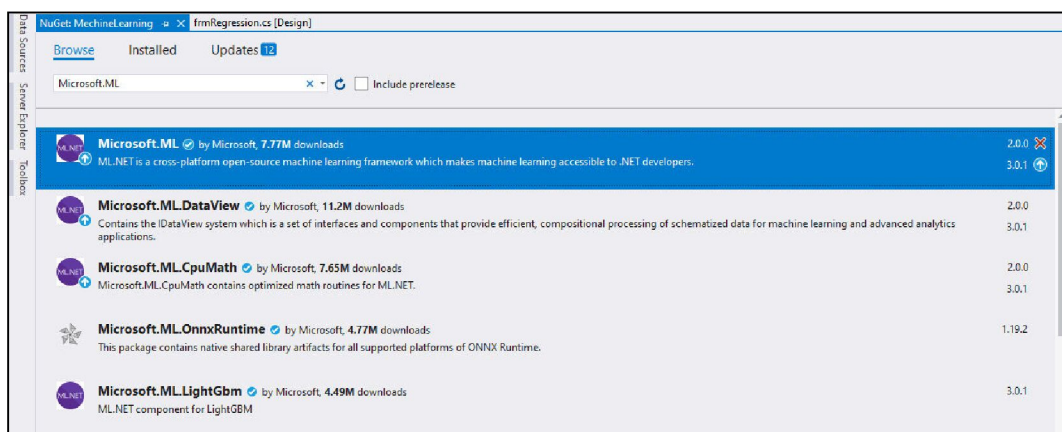
You can find [more ML.NET samples on GitHub](#), or take a look at the [ML.NET tutorials](#).

2. Cách sử dụng ML.net

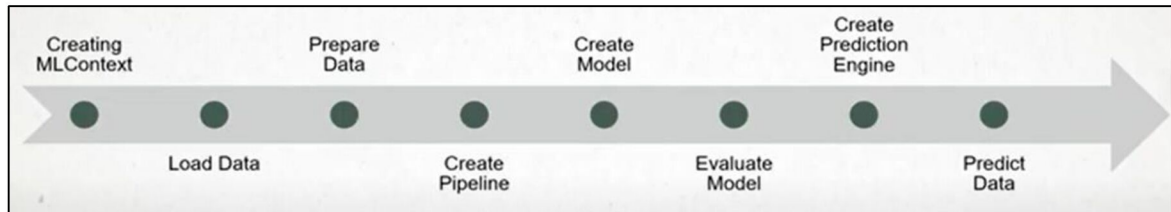
- Tạo project (lưu ý cần sử dụng framework 4.8 trở lên)
- Cài thư viện vào project, sử dụng Manage NuGet Packages ...



- Tìm thư viện ML.net (có thể sử dụng từ khóa **Microsoft.ML**) và chọn phiên bản phù hợp với .Net framework để cài đặt (framework 4.8 có thể cài bản ML.net 2.0)



- Các bước để sử dụng ML.net gồm:



Ví dụ:

```
//Bước 1. Tạo một ML Context
var ctx = new MLContext();

//Bước 2. Đọc dữ liệu từ file text để đưa vào mô hình huấn luyện
IDataView trainingData = ctx.Data
    .LoadFromTextFile<ModelInput>(dataPath, hasHeader: true);

//Bước 3. Xây dựng việc xử lý dữ liệu
var pipeline = ctx.Transforms.Text
    .FeaturizeText("Features", nameof(SentimentIssue.Text))
    .Append(ctx.BinaryClassification.Trainers
        .LbfgsLogisticRegression("Label", "Features"));

//Bước 4. Huấn luyện và tạo mô hình
ITransformer trainedModel = pipeline.Fit(trainingData);

//Bước 5. Sử dụng mô hình đã huấn luyện để làm dự đoán
var predictionEngine = ctx.Model
    .CreatePredictionEngine<ModelInput, ModelOutput>(trainedModel);
var sampleStatement = new ModelInput() { Text = "This is a horrible movie" };
var prediction = predictionEngine.Predict(sampleStatement);
```

3. Bài tập hướng dẫn

Bài 1. Sử dụng mô hình Regression để dự đoán lương của một nhân sự dựa vào số năm kinh nghiệm của họ với giao diện như sau :

	YearOfExperience	Salary
1	800	
2	900	
2.5	950	
3	1200	
4	1500	
4.2	1540	
4.3	1560	
5.8	1700	
5.9	1710	
6.5	1800	
8.8	2100	
10.5	2200	
12.1	2300	

Training

Prediction

YearsOfExperiment

Salary

Predict

Evaluation

MA Errors label9

MS Errors label10

RMS Errors label11

Loss Function label12

Hướng dẫn:

Xây dựng lớp dữ liệu đầu vào:

```
public class InputModel
{
    public float YearOfExperience { get; set; }
    public float Salary { get; set; }
}
```

Bước 1: Khai báo sử dụng thư viện ML

```
MLContext context = new MLContext();
```

Bước 2 : Load dữ liệu:

```
List<InputModel> lst = getData();
IDataView dataTran = context.Data.LoadFromEnumerable(lst);
```

Với

```
private List<InputModel> getDataTrain()
{
    List<InputModel> lst = new List<InputModel>();
    lst.Add(new InputModel() { YearOfExperience = 1.0F, Salary = 800F});
    lst.Add(new InputModel() { YearOfExperience = 2.0F, Salary = 900F});
    lst.Add(new InputModel() { YearOfExperience = 2.5F, Salary = 950F});
    lst.Add(new InputModel() { YearOfExperience = 3.0F, Salary = 1200F});
    lst.Add(new InputModel() { YearOfExperience = 4.0F, Salary = 1500F});
    lst.Add(new InputModel() { YearOfExperience = 4.2F, Salary = 1540F});
    lst.Add(new InputModel() { YearOfExperience = 4.3F, Salary = 1560F});
    lst.Add(new InputModel() { YearOfExperience = 5.8F, Salary = 1700F});
    lst.Add(new InputModel() { YearOfExperience = 5.9F, Salary = 1710F});
    lst.Add(new InputModel() { YearOfExperience = 6.5F, Salary = 1800F});
    lst.Add(new InputModel() { YearOfExperience = 8.8F, Salary = 2100F});
    lst.Add(new InputModel() { YearOfExperience = 10.5F, Salary = 2200F});
    lst.Add(new InputModel() { YearOfExperience = 12.1F, Salary = 2300F});
    return lst;
}
```

Bước 3: Chuẩn bị dữ liệu

```
var estimator = context.Transforms.Concatenate("Features", new[]
                                                {"YearOfExperience"});
```

Bước 4: Tạo pipeline

```
var pipeline = estimator.Append(context.Regression.Trainers.Sdca(
    labelColumnName: "Salary", maximumNumberOfIterations: 100));
var model = pipeline.Fit(dataTran);
```


Bước 5: Huấn luyện và tạo mô hình

```
predictionEngine = context.Model.CreatePredictionEngine  
                                <InputModel, ResultModel>(model);
```

Trong đó Results là thông tin dự đoán

```
public class ResultModel  
{  
    [ColumnName("Score")]  
    public float Salary { get; set; }  
}
```

Bước 6: Đánh giá mô hình và hiển thị lên form

```
//Chuẩn bị dữ liệu đánh giá  
IDataView dataTest = context.Data.LoadFromEnumerable(getDataTest());  
dgvTestData.DataSource = getDataTest();  
var metrics = context.Regression.Evaluate(model.Transform(dataTest),  
                                           labelColumnName: "Salary");  
  
//Hiển thị kết quả lên form  
lblR2.Text = metrics.RSquared.ToString("0.00");  
lblMA.Text = metrics.MeanAbsoluteError.ToString("0.00");  
lblMS.Text = metrics.MeanSquaredError.ToString("0.00");  
lblRMS.Text = metrics.RootMeanSquaredError.ToString("0.00");  
lblLostFunction.Text = metrics.LossFunction.ToString("0.00");
```

Trong đó:

```
private List<InputModel> getDataTest()  
{  
    List<InputModel> lst = new List<InputModel>();  
    lst.Add(new InputModel() { YearOfExperience = 2.0F, Salary = 900F });  
    lst.Add(new InputModel() { YearOfExperience = 2.6F, Salary = 1000F });  
    lst.Add(new InputModel() { YearOfExperience = 2.8F, Salary = 1100F });  
    lst.Add(new InputModel() { YearOfExperience = 3.5F, Salary = 1400F });  
    lst.Add(new InputModel() { YearOfExperience = 4.0F, Salary = 1500F });  
    return lst;  
}
```

Sử dụng mô hình đã huấn luyện để dự đoán kết quả:

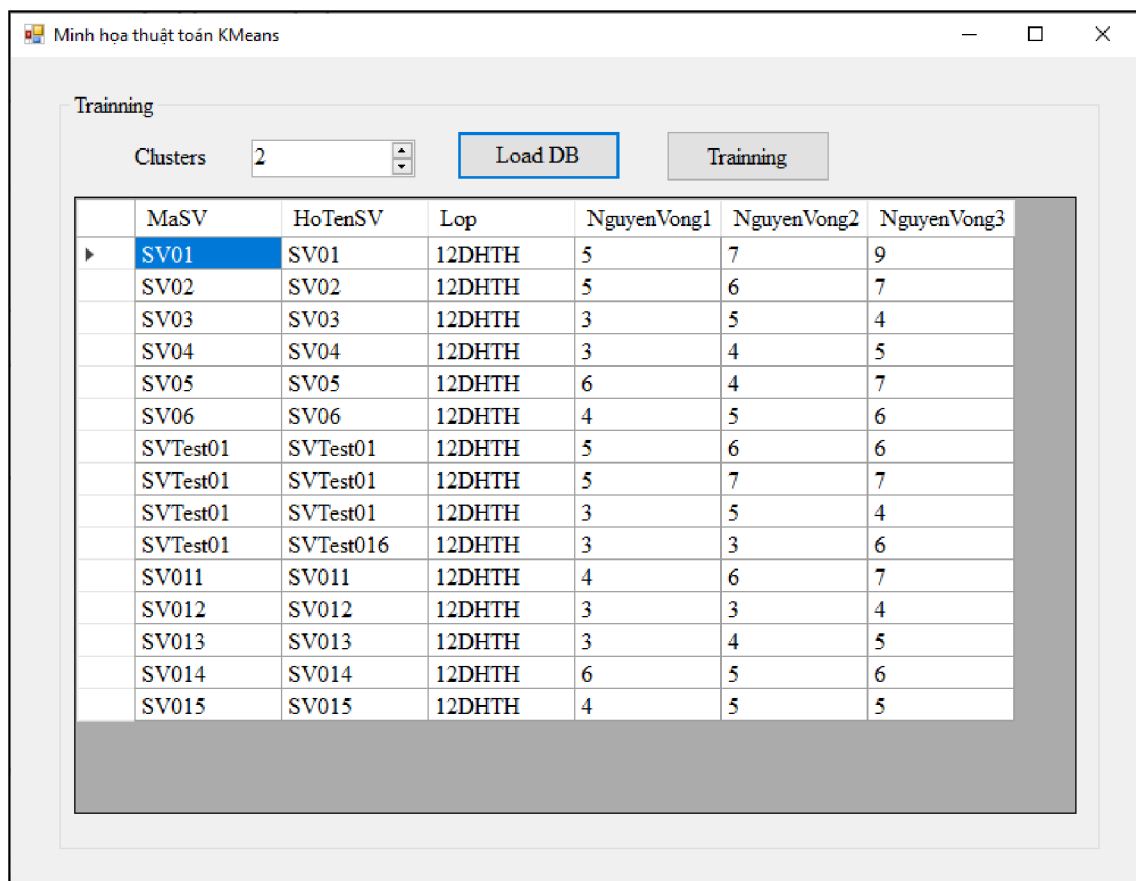
```
float year = float.Parse(txtNamKN.Text);  
InputModel user = new InputModel() {YearOfExperience = year};  
var result = predictionEngine.Predict(user);  
lblLuong.Text = result.Salary.ToString("0.00");
```

Bài 02. Xây dựng hệ thống hỗ trợ xếp phòng ký túc xá cho sinh viên, sao cho các sinh viên có nguyện vọng (*muốn ở chung với ai, phòng như thế nào, ...*) tương tự nhau thì được xếp cùng một phòng. Thông tin của mỗi sinh viên gồm: MSSV, Họ và tên, các số thể hiện mức độ về mỗi nguyện vọng của sinh viên (mỗi sinh viên có 03 nguyện vọng).

Phân tích: bài toán chia sinh viên thành nhiều nhóm (mỗi nhóm là 1 phòng) dựa trên nguyện vọng của các sinh viên. Đây là bài toán thuộc dạng bài toán phân cụm (Clusters) và sử dụng thuật toán K-Means.

Thuật toán K-Means là thuật toán rất quan trọng và được sử dụng phổ biến trong kỹ thuật phân cụm. Tư tưởng chính của thuật toán K-Means là tìm cách phân nhóm các đối tượng (objects) đã cho vào **K** cụm (**K** là số các cụm được xác định trước và **K** nguyên dương) sao cho tổng bình phương khoảng cách giữa các đối tượng đến tâm nhóm (centroid) là **nhỏ nhất** (Lưu ý: *mỗi đối tượng thuộc về đúng một cụm*).

Các đặc trưng cần sử dụng để giải quyết bài toán đó là thông tin của sinh viên: cụ thể ở đây là nguyện vọng 1, nguyện vọng 2, nguyện vọng 3.



	MaSV	HoTenSV	Lop	NguyenVong1	NguyenVong2	NguyenVong3
▶	SV01	SV01	12DHTH	5	7	9
	SV02	SV02	12DHTH	5	6	7
	SV03	SV03	12DHTH	3	5	4
	SV04	SV04	12DHTH	3	4	5
	SV05	SV05	12DHTH	6	4	7
	SV06	SV06	12DHTH	4	5	6
	SVTest01	SVTest01	12DHTH	5	6	6
	SVTest01	SVTest01	12DHTH	5	7	7
	SVTest01	SVTest01	12DHTH	3	5	4
	SVTest01	SVTest016	12DHTH	3	3	6
	SV011	SV011	12DHTH	4	6	7
	SV012	SV012	12DHTH	3	3	4
	SV013	SV013	12DHTH	3	4	5
	SV014	SV014	12DHTH	6	5	6
	SV015	SV015	12DHTH	4	5	5

Hướng dẫn: Tương tự như bài tập 1, ta lần lượt thực hiện các bước như sau :

Bước 0: Khai báo mô hình

```
//Khởi tạo sử dụng thư viện ML
```

```
MLContext mlContext = new MLContext(seed: 1);
```

```
//seed: là một số bất kỳ, phân biệt với các xử lý khác
```

Bước 1: Load dữ liệu

```
IDataView fullData = mlContext.Data.LoadFromEnumerable(getSinhViens());
```

Với

```
public List<SinhVien> getSinhViens()
{
    List<SinhVien> lst = new List<SinhVien>();
    SinhVien sv1 = new SinhVien("SV01", "SV01", "12DHTH", 5, 7, 9); lst.Add(sv1);
    SinhVien sv2 = new SinhVien("SV02", "SV02", "12DHTH", 5, 6, 7); lst.Add(sv2);
    SinhVien sv3 = new SinhVien("SV03", "SV03", "12DHTH", 3, 5, 4); lst.Add(sv3);
    SinhVien sv4 = new SinhVien("SV04", "SV04", "12DHTH", 3, 4, 5); lst.Add(sv4);
    SinhVien sv5 = new SinhVien("SV05", "SV05", "12DHTH", 6, 4, 7); lst.Add(sv5);
    SinhVien sv6 = new SinhVien("SV06", "SV06", "12DHTH", 4, 5, 6); lst.Add(sv6);
    SinhVien sv10 = new SinhVien("SVTest01", "SVTest01", "12DHTH", 5, 6, 6); lst.Add(sv10);
    SinhVien sv9 = new SinhVien("SVTest01", "SVTest01", "12DHTH", 5, 7, 7); lst.Add(sv9);
    SinhVien sv8 = new SinhVien("SVTest01", "SVTest01", "12DHTH", 3, 5, 4); lst.Add(sv8);
    SinhVien sv7 = new SinhVien("SVTest01", "SVTest016", "12DHTH", 3, 3, 6); lst.Add(sv7);
    SinhVien sv11 = new SinhVien("SV011", "SV011", "12DHTH", 4, 6, 7); lst.Add(sv11);
    SinhVien sv12 = new SinhVien("SV012", "SV012", "12DHTH", 3, 3, 4); lst.Add(sv12);
    SinhVien sv13 = new SinhVien("SV013", "SV013", "12DHTH", 3, 4, 5); lst.Add(sv13);
    SinhVien sv14 = new SinhVien("SV014", "SV014", "12DHTH", 6, 5, 6); lst.Add(sv14);
    SinhVien sv15 = new SinhVien("SV015", "SV015", "12DHTH", 4, 5, 5); lst.Add(sv15);

    return lst;
}
```

//và chia bộ dữ liệu thành 2 phần: 80% để train và 20% để test

```
DataOperationsCatalog.TrainTestData trainTestData =
    mlContext.Data.TrainTestSplit(fullData, testFraction: 0.2);
trainingDataView = trainTestData.TrainSet;
testingDataView = trainTestData.TestSet;
```

Bước 2 : Cấu hình thông tin cần sử dụng

```
var dataProcessPipeline = mlContext.Transforms.Concatenate(
    "Features",
    nameof(SinhVien.NguyenVong1),
    nameof(SinhVien.NguyenVong2),
    nameof(SinhVien.NguyenVong3)
);
```

Bước 3: Huấn luyện mô hình

```
var trainer = mlContext.Clustering.Trainers.KMeans(
    featureColumnName: "Features",
    numberOfClusters: 2
);
var trainingPipeline = dataProcessPipeline.Append(trainer);
var trainedModel = trainingPipeline.Fit(trainingDataView);
```

Bước 4: Đánh giá mô hình, sử dụng dữ liệu Test đã chia ở trên để đánh giá mô hình

```
IDataView predictions = trainedModel.Transform(testingDataView);  
var metrics = mlContext.Clustering.Evaluate(  
    predictions,  
    scoreColumnName: "Score",  
    featureColumnName: "Features"  
);
```

Bước 5: Lưu mô hình về file.zip để sử dụng sau này

```
string ModelRelativePath = @"..\MLModels\KiTucXaModel.zip";  
string ModelPath = GetAbsolutePath(ModelRelativePath);  
mlContext.Model.Save(trainedModel,  
    trainingDataView.Schema,  
    ModelPath);  
MessageBox.Show("Hoàn thành quá trình huấn luyện", "Thông báo");
```

Bước 6: Thử nghiệm

```
var sampleStudent = new SinhVien()  
{  
    HoTenSV = "Sinh viên Test",  
    Lop = "12DHTH01",  
    MaSV = "2001210001",  
    NguyenVong1 = 6,  
    NguyenVong2 = 7,  
    NguyenVong3 = 5  
};  
//Load model đã lưu để sử dụng  
ITransformer model = mlContext.Model.Load(ModelPath, out var  
    modelInputSchema);  
//Dự đoán kết quả  
var predEngine = mlContext.Model.CreatePredictionEngine<SinhVien,  
    SVPrediction>(model);  
var resultprediction = predEngine.Predict(sampleStudent);  
MessageBox.Show(resultprediction.Distance[0].ToString());  
MessageBox.Show(resultprediction.Distance[1].ToString());  
MessageBox.Show("Bạn sinh viên này được xếp ở cụm " +  
    resultprediction.SelectedClusterId.ToString());
```


Một số lớp liên quan cần được cài đặt:

Lớp SinhVien: là thông tin đầu vào của mỗi sinh viên

```
public class SinhVien
{
    3 references
    public string MaSV { get; set; }
    3 references
    public string HoTenSV { get; set; }
    3 references
    public string Lop { get; set; }
    3 references
    public float NguyenVong1 { get; set; }
    3 references
    public float NguyenVong2 { get; set; }
    3 references
    public float NguyenVong3 { get; set; }
    1 reference
    public SinhVien()...
    19 references
    public SinhVien(string ma, string ten, string lop,
        float nv1, float nv2, float nv3)...
}
```

Lớp SVPrediction: Lưu kết quả sau khi phân cụm kế thừa lớp SinhVien

```
public class SVPrediction:SinhVien
{
    1 reference
    [ColumnName("PredictedLabel")]
    public uint SelectedClusterId { get; set; }

    2 references
    [ColumnName("Score")]
    public float[] Distance { get; set; }
}
```

D. BÀI TẬP TỰ LÀM

1. Xây ứng dụng thuật toán Regression để dự đoán điểm tốt nghiệp của một sinh viên khi biết một số điểm học phần.
2. Xây dựng ứng dụng dự đoán giá nhà dựa vào diện tích.
3. Xây dựng ứng dụng tư vấn hỗ trợ chọn chuyên ngành cho sinh viên ngành CNTT.