Skill ML.NET Forecasting

Forecast future values of a single sensor tag using ML.NET SSA (Singular Spectrum Analysis). The AI generates the full C# Script Class with time-series prediction engine, connects it to a live tag, creates output tags for forecast values with confidence bounds, and configures model persistence.

AI Integration Platform Skills Library Skill ML.NET Skill ML.NET Forecasting

- When to Use This Skill
- Prerequisites
- MCP Tools and Tables
- Step 1: Create Output Tags
- Step 2: Create the Script Class
  - ML.NET Namespace Declaration
  - Tag References Inside Script Classes
  - SSA Forecasting Pipeline
  - Full Class Example — Time-Series Forecasting
- Step 3: Write the Class via MCP
- Step 4: Create the Trigger
  - Expression (OnChange) or Task (Periodic)
  - ServerStartup — always wire LoadModel
- Step 5: Verify
- Common Pitfalls
- See also

When to Use This Skill

- User chose **Time-Series Forecasting — SSA** in the ML.NET router skill (Step 0)
- Predicting future values from a **single sensor** based on historical patterns
- User goals: tank level prediction, production rate forecasting, demand forecasting, predict future sensor readings

---

Prerequisites

- Solution open with the input tag already created and receiving data (live or simulated)
- Solution target platform set to **Multiplatform** (ML.NET requires .NET 8+)
- The user has confirmed: (1) which tag to forecast, (2) forecast horizon (steps ahead), (3) output folder path

MCP Tools and Tables

Category	Items
Tools	`get_table_schema`, `write_objects`, `get_objects`
Tables	`UnsTags`, `ScriptsClasses`, `ScriptsExpressions`, `ScriptsTasks`

Step 1: Create Output Tags

```
get_table_schema('UnsTags')
```

```
{
  "table_type": "UnsTags",
  "data": [
    { "Name": "<AssetPath>/ML/Forecast", "DataType": "Double", "Description": "Forecasted value" },
    { "Name": "<AssetPath>/ML/ForecastLower", "DataType": "Double", "Description": "Lower confidence bound"
},
    { "Name": "<AssetPath>/ML/ForecastUpper", "DataType": "Double", "Description": "Upper confidence bound"
},
    { "Name": "<AssetPath>/ML/LastPrediction", "DataType": "DateTime", "Description": "Timestamp of last
prediction" }
  ]
}
```

Replace `<AssetPath>` with the actual asset folder path (e.g., `Plant/Tank01`).

Step 2: Create the Script Class

ML.NET Namespace Declaration

> **Critical:** The field `AddMLNetNamespaces` does **not exist** and is silently ignored. Always use `NamespaceDeclarations` with a semicolon-separated string.

```
"NamespaceDeclarations": "Microsoft.ML;Microsoft.ML.Data;Microsoft.ML.Transforms;Microsoft.ML.Transforms.
TimeSeries;Microsoft.ML.Transforms.Text;Microsoft.ML.Trainers;Microsoft.ML.TimeSeries"
```

Tag References Inside Script Classes

Always use `@Tag.` prefix. ML.NET expects `float` but FrameworX tags use `double` — always cast with `(float)` when feeding ML.NET and `(double)` when writing to tags.

SSA Forecasting Pipeline

```
var pipeline = mlContext.Forecasting.ForecastBySsa(
    outputColumnName: "ForecastedValues",
    inputColumnName: nameof(SensorData.Value),
    windowSize: 10,
    seriesLength: 100,
    trainSize: trainingBuffer.Count,
    horizon: 5,
    confidenceLevel: 0.95f,
    confidenceLowerBoundColumn: "LowerBound",
    confidenceUpperBoundColumn: "UpperBound");
```

Output: `float[] ForecastedValues`, `float[] LowerBound`, `float[] UpperBound`

Full Class Example — Time-Series Forecasting

```
public class SensorData
{
    public float Value { get; set; }
}

public class ForecastOutput
{
    public float[] ForecastedValues { get; set; }
    public float[] LowerBound { get; set; }
    public float[] UpperBound { get; set; }
}

private static MLContext mlContext = new MLContext(seed: 0);
private static TimeSeriesPredictionEngine<SensorData, ForecastOutput> forecastEngine;
private static ITransformer model;
private static IDataView lastTrainingDataView;
private static bool modelTrained = false;
private static List<SensorData> trainingBuffer = new List<SensorData>();
private const int MinTrainingSize = 100;
private static readonly string ModelPath = Path.Combine(@Info.GetExecutionPath(), "<ClassName>.mlnet");

public int Predict(double inputValue)
{
    trainingBuffer.Add(new SensorData { Value = (float)inputValue });

    if (!modelTrained && trainingBuffer.Count >= MinTrainingSize)
        TrainModel();

    if (modelTrained)
        RunPrediction();

        return 1;
}

public void LoadModel()
{
    if (File.Exists(ModelPath))
    {
        model = mlContext.Model.Load(ModelPath, out _);
        forecastEngine = model.CreateTimeSeriesEngine<SensorData, ForecastOutput>(mlContext);
```

```
        modelTrained = true;
    }
}

private void TrainModel()
{
    lastTrainingDataView = mlContext.Data.LoadFromEnumerable(trainingBuffer);
    var pipeline = mlContext.Forecasting.ForecastBySsa(
        outputColumnName: "ForecastedValues",
        inputColumnName: nameof(SensorData.Value),
        windowSize: 10,
        seriesLength: 100,
        trainSize: trainingBuffer.Count,
        horizon: 5,
        confidenceLevel: 0.95f,
        confidenceLowerBoundColumn: "LowerBound",
        confidenceUpperBoundColumn: "UpperBound");

    model = pipeline.Fit(lastTrainingDataView);
    forecastEngine = model.CreateTimeSeriesEngine<SensorData, ForecastOutput>(mlContext);
    modelTrained = true;
    SaveModel();
}

private void SaveModel()
{
    mlContext.Model.Save(model, lastTrainingDataView.Schema, ModelPath);
}

private void RunPrediction()
{
    var forecast = forecastEngine.Predict();
    @Tag.<AssetPath>/ML/Forecast.Value = (double)forecast.ForecastedValues[0];
    @Tag.<AssetPath>/ML/ForecastLower.Value = (double)forecast.LowerBound[0];
    @Tag.<AssetPath>/ML/ForecastUpper.Value = (double)forecast.UpperBound[0];
    @Tag.<AssetPath>/ML/LastPrediction.Value = DateTime.Now;
}
```

Step 3: Write the Class via MCP

```
get_table_schema('ScriptsClasses')
```

```
{
  "table_type": "ScriptsClasses",
  "data": [
    {
      "Name": "<ClassName>",
      "Code": "CSharp",
      "Domain": "Server",
      "ClassContent": "Methods",
      "NamespaceDeclarations": "Microsoft.ML;Microsoft.ML.Data;Microsoft.ML.Transforms;Microsoft.ML.
Transforms.TimeSeries;Microsoft.ML.Transforms.Text;Microsoft.ML.Trainers;Microsoft.ML.TimeSeries",
      "Contents": "<AI-generated C# code from full class example above>"
    }
  ]
}
```

> Field names matter: the language field is `Code` (not `Language`), and the code body field is `Contents` (not `Code`). Using wrong field names results in silent data loss.

Step 4: Create the Trigger

Expression (OnChange) or Task (Periodic)

For single-input forecasting, an Expression OnChange trigger works well. For periodic forecasting (e.g., every 5 seconds), use a Task.

```
get_table_schema('ScriptsExpressions')
```

```
{
  "table_type": "ScriptsExpressions",
  "data": [
    {
      "Name": "ML_Forecast_<SensorName>",
      "ObjectName": "",
      "Expression": "@Script.Class.<ClassName>.Predict(@Tag.<AssetPath>.<Member>)",
      "Execution": "OnChange",
      "Trigger": ""
    }
  ]
}
```

> **`ObjectName` must be empty** when the ML class writes results internally. **Use `Trigger` (not `TriggerTag`)** — `TriggerTag` does not exist and is silently ignored.

ServerStartup — always wire LoadModel

Read the existing `ServerStartup` task first (document object — read-modify-write), then append:

```
Script.Class.<ClassName>.LoadModel();
```

Step 5: Verify

1. **Confirm Multiplatform** — ML.NET requires .NET 8+. Instruct the user: *"Solution Settings Target Platform = Multiplatform, then Product Modify."*
2. **Do NOT start the runtime automatically.**
3. **Wait for training** — needs `MinTrainingSize` data points (default 100)
4. **Check output tags** — verify `LastPrediction` timestamp updates and `Forecast` has reasonable values

---

Common Pitfalls

Mistake	Why It Happens	How to Avoid
Missing ML.NET namespaces	Used `AddMLNetNamespaces: true` (does not exist)	Always set `NamespaceDeclarations`
`CS0234` error in ScriptsTasks	Missing `@` prefix	Always use `@Script.Class.Name.Predict()`
Non-empty `ObjectName`	Void method + tag assignment	Leave `ObjectName` empty
Used `TriggerTag`	Field does not exist	Use `Trigger` field
`CS0246` Script Class not found	Build order issue	Set `BuildOrder: "1"` on the ML Script Class
`System.Math` load error	Solution targets .NET 4.8	Switch to Multiplatform (.NET 8+)
Wrong data types	ML.NET=float, tags=double	Cast with `(float)/(double)`
Model lost on restart	SaveModel/LoadModel missing	Always include both + wire LoadModel in ServerStartup
ForecastEngine null after LoadModel	Forgot to create engine after loading	Call `CreateTimeSeriesEngine` in LoadModel after `mlContext.Model.Load`

See also

- Skill ML.NET. Router skill that selects the appropriate ML.NET sub-skill based on user goals.
- Skill ML.NET Anomaly Detection. Sister sub-skill for SSA spike detection on a single sensor tag.
- Skill ML.NET Regression. Sister sub-skill for FastTree regression from multiple feature tags.
- Skill ML.NET Classification. Sister sub-skill for FastTree binary classification from multiple feature tags.