# ML.NET

## Einführung in das Machine-Learning-Framework für .NET-Entwickler

# Inhaltsverzeichnis


Einleitung

# Was ist ML.NET?

- Open-Source und Cross-Platform Framework für ML (machine learning) von Microsoft/ .NET Foundation [1][2]
- Für .NET-Entwickler ausgelegt, so dass man ML-Funktionalitäten in .NET-Anwendungen integrieren kann
- Aktuelle Version: 1.1.0 (05.06.2019)
- Lizenz: MIT

# Was ist ML.NET?

- Als erweiterbare Plattform konzipiert, so dass andere ML-Bibliotheken „konsumiert“ werden können
- Wird bereits im Microsoft Defender eingesetzt[3]
- Voraussetzung[4]:
  - .NET Framework 4.7.2 oder neuer
  - .NET Core 2.1 oder neuer
  - Bereitstellung über „Microsoft.ML“-NuGet-Pakete
  - [Optional] ML.NET AutoML

# Einsatzbereiche

# Einsatzbereiche[5][6]

## Klassifikation

- Binäre Klassifikation
- Multiklassen Klassifikation
- Clustering

## Regression

## Extensionable

- Bildklassifikation
- Objekterkennung in Bildern
- …

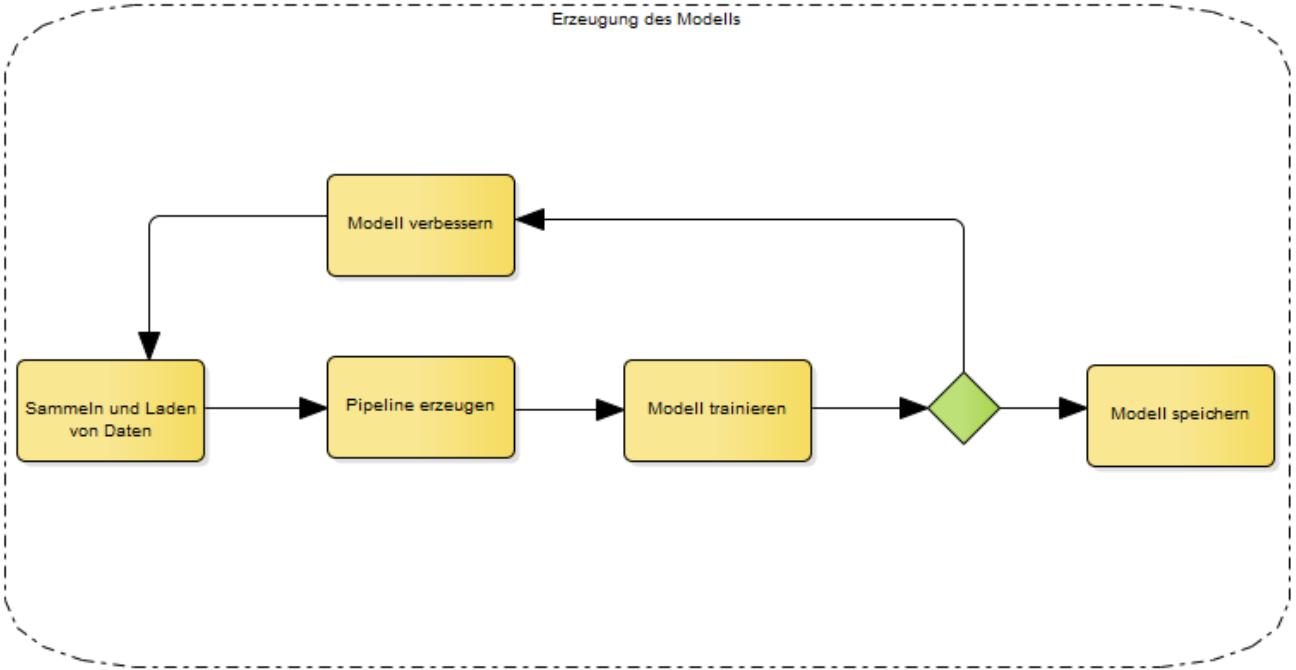
## Anomalie Erkennung

## Empfehlung

# Arbeitsablauf

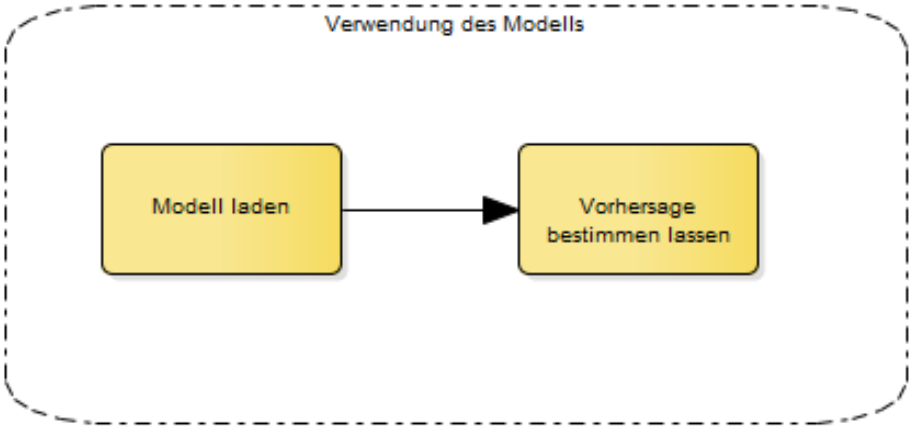
# Arbeitsablauf[6]

## Abschnitt 1 – Erzeugung des Modells:



# Arbeitsablauf[6]

## Abschnitt 2: Die Verwendung des Modells:



# Begrifflichkeiten

# Begrifflichkeiten

- Daten:
  - Daten können unterschiedlich aufgebaut sein, je nachdem was ich erreichen möchte[7]
  - Eine Datenaufbereitung ist meistens notwendig
  - Man unterscheidet zwischen „Label“ und „Feature“[8]
    - Label: Der Wert, den ich vorhergesagt bekommen möchte
    - Feature: verschiedene Datenwerte (Dimensionen) eines konkreten Datensatzes

# Begrifflichkeiten

- Trainer:
  - Als Trainer werden die jeweiligen Lernalgorithmen eines Einsatzbereiches bezeichnet[9]
  - Besitzt verschiedene Charakteristika[10]:
    - Zuordnung zum Einsatzbereich
    - Wird eine Normalisierung benötigt?
    - Wird ein Caching benötigt?
    - Muss ein zusätzliches NuGet-Paket geladen werden

# Konventioneller Weg in ML.NET

# Konventioneller Weg in ML.NET

- Ausgangssituation:
  - Ziel:
    - Bestimmung des Fahrpreises mittels Regression[11][12]
- Vorgehen:
  - (1) Abbildung des Labels und der Features aus dem Datensatz in konkrete C#-Klassen. Das „Label" muss dabei als „Score" annotiert werden.
  - (2) Erzeugung der Transformations-Pipeline samt Daten-Encoding, da die Features in numerischer Form benötigt werden und als eine Spalte abgebildet werden müssen

# Konventioneller Weg in ML.NET

- Vorgehen:
  - (3) Wähle einen beliebigen Trainer für Regressionen aus. Ausgewählt wurde dabei „FastTree[13]“
  - (4) Trainiere und speichere das Modell ab
  - (5) Evaluiere das Model anhand von der automatisch berechneten Bewertungsmetrik „Rsquared“
    - Bestimmtheitsmaß, wie gut passen die Daten in das Modell?
  - (6) Führe Vorhersagen durch

# Konventioneller Weg in ML.NET

- Probleme:
  - Auswahl des am geeignetsten Trainers und dessen Parameter-Konfiguration => iteratives „Try & Error"-Vorgehen
  - Der Aufwand beim „Try & Errors"-Vorgehen

# ML.NET AutoML

# ML.NET AutoML[14][15]

- AutoML steht für „Automatisiertes Machine Learning“
- Kein manuelles „Ausprobieren“ von verschiedenen Lernalgorithmen und Parametern mehr notwendig
- Ziel:
  - Optimales ML-Modell mit hoher Performance automatisiert zu generieren
  - Optimale Einstellung und Menge an verwendeten Parametern

# ML.NET AutoML

- Konkrete Möglichkeiten:
  - ML.NET CLI[16]
    - > mlnet auto-train --task binary-classification --dataset "customer-feedback.tsv" --label-column-name Sentiment
  - Verwendung der automatisierten ML-API per Code
  - ML.NET Model Builder
- Unterstützt folgende Einsatzbereiche:
  - Binäre Klassifikation
  - Multiklassen Klassifikation
  - Regression

# ML.NET Model Builder

# ML.NET Model Builder[17][18][19]

- Wird per Visual Studio Extension eingebunden
- Befindet sich noch im Preview-Status
- Voraussetzung[20]:
  - Visual Studio 2017 15.9.12
  - .NET Core 2.1 SDK
- Limitierung:
  - Es werden nur .tsv, .csv und SQL als Datenquellen unterstützt

# ML.NET Model Builder

## Trainingsdauer[21]:

| Datengröße | Dauer |
|---|---|
| 0 – 10 MB | 10 sek |
| 10 – 100 MB | 10 min |
| 100 – 500 MB | 30 min |
| 500 – 1 GB | 60 min |
| 1 GB+ | >3 Stunden |

# ML.NET Model Builder

- Bewertungsmetrik für eine binäre Klassifikation[22]:
  - Accuracy
    - Beschreibt die Genauigkeit
    - Es werden alle 4 Ergebnismöglichkeiten bei der Berechnung einbezogen
  - AUC (*Area under the curve)*
    - Verhältnis der richtigen positiven Ergebnisse zu den falschen positiven Ergebnissen unter Bestimmung der „True Positive Rate“ und der „False Positive Rate“
    - Das Ergebnis muss dabei größer 0,5 sein

# ML.NET Model Builder

- Bewertungsmetrik für eine binäre Klassifikation:
  - AUCPR (*Area under a Precision-Recall curve)*
    - Maß für den Erfolg einer Vorhersage, wenn die verwendeten Klassen sehr unausgewogen sind
    - Anhand der Verwendung der Präzision und der Wiedererkennung
  - F1-score
    - Gibt Auskunft über die Balance zwischen der Präzision und der Wiedererkennung

# Vergleich: ML.NET AutoML und konventioneller Weg

# Vergleich

- Als Ausgangsbasis dient das Fahrpreis-Beispiel
- Vergleich 1 [23]:

| AutoML | | Konventionell | |
|---|---|---|---|
| Algorithmus: | LightGbmRegression | | FastTreeRegression |
| Rsquared | 0,9515 | | 0,92 |
| Vorhersage | 15,4954 | | 15,7855 |

- Vergleich 2:

| AutoML | | Konventionell | |
|---|---|---|---|
| Algorithmus: | LightGbmRegression | | LightGbmRegression |
| Rsquared | 0,9515 | | 0,92 |
| Vorhersage | 15,4954 | | 15,6766 |

# Vergleich

## Vergleich 3:

| AutoML | | Konventionell | |
|---|---|---|---|
| Algorithmus: | LightGbmRegression | | LightGbmRegression mit 1000 Iterationsvorgängen |
| Rsquared | 0,9515 | | 0,92 |
| Vorhersage | 15,4954 | | 15,5983 |

# Verbesserung der Modellgenauigkeit

# Verbesserung der Modellgenauigkeit[24]

- Definieren des Problems
- Bereitstellen weiterer Datenbeispiele
- Hinzufügen von Kontext zu den Daten
- Verwendung von aussagekräftigen Daten und Features
- Kreuzvalidierung
- Hyperparameteroptimierung
- Auswahl eines anderen Algorithmus

# Quellen


- [1] https://docs.microsoft.com/de-de/dotnet/machine-learning/
- [2] https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet
- [3] https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet/customers/microsoft-defender
- [4] https://github.com/dotnet/machinelearning
- [5] https://docs.microsoft.com/de-de/dotnet/machine-learning/tutorials/
- [6] https://docs.microsoft.com/de-de/dotnet/machine-learning/how-does-mldotnet-work
- [7] https://docs.microsoft.com/de-de/dotnet/machine-learning/how-to-guides/prepare-data-ml-net
- [8] https://docs.microsoft.com/de-de/dotnet/machine-learning/how-to-guides/train-machine-learning-model-ml-net#working-with-default-column-names
- [9] https://docs.microsoft.com/de-de/dotnet/machine-learning/how-to-choose-an-ml-net-algorithm
- [10] https://docs.microsoft.com/de-de/dotnet/api/microsoft.ml.trainers.fasttree.fasttreeregressiontrainer?view=ml-dotnet
- [11] https://docs.microsoft.com/de-de/dotnet/machine-learning/tutorials/predict-prices
- [12] https://github.com/dotnet/machinelearning-samples/tree/master/samples/csharp/getting-started/Regression_TaxiFarePrediction
- [13] https://www.nuget.org/packages/Microsoft.ML.FastTree/1.3.1/
- [14] https://docs.microsoft.com/de-de/dotnet/machine-learning/automl-overview
- [15] https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet#automl
- [16] https://docs.microsoft.com/de-de/dotnet/machine-learning/automate-training-with-cl

# Quellen


- [17] https://github.com/dotnet/machinelearning-samples/tree/master/modelbuilder
- [18] https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet/model-builder
- [19] https://dotnet.microsoft.com/learn/ml-dotnet/get-started-tutorial/intro
- [20] https://github.com/dotnet/machinelearning-samples/tree/master/modelbuilder#installation
- [21] https://github.com/dotnet/machinelearning-samples/tree/master/modelbuilder#train
- [22] https://docs.microsoft.com/de-de/dotnet/machine-learning/resources/metrics
- [23] https://www.nuget.org/packages/Microsoft.ML.LightGbm/1.3.1/
- [24] https://docs.microsoft.com/de-de/dotnet/machine-learning/resources/improve-machine-learning-model-ml-net

# Vielen Dank!