ML.NET

Machine Learning framework for .Net

#CodeGen

#dotnetconf

@cloudgen_verona

Main sponsor

Community sponsors

Syncfusion®

Packt

MANNING

elmah.io

Udemy™

JET BRAINS

Local Event

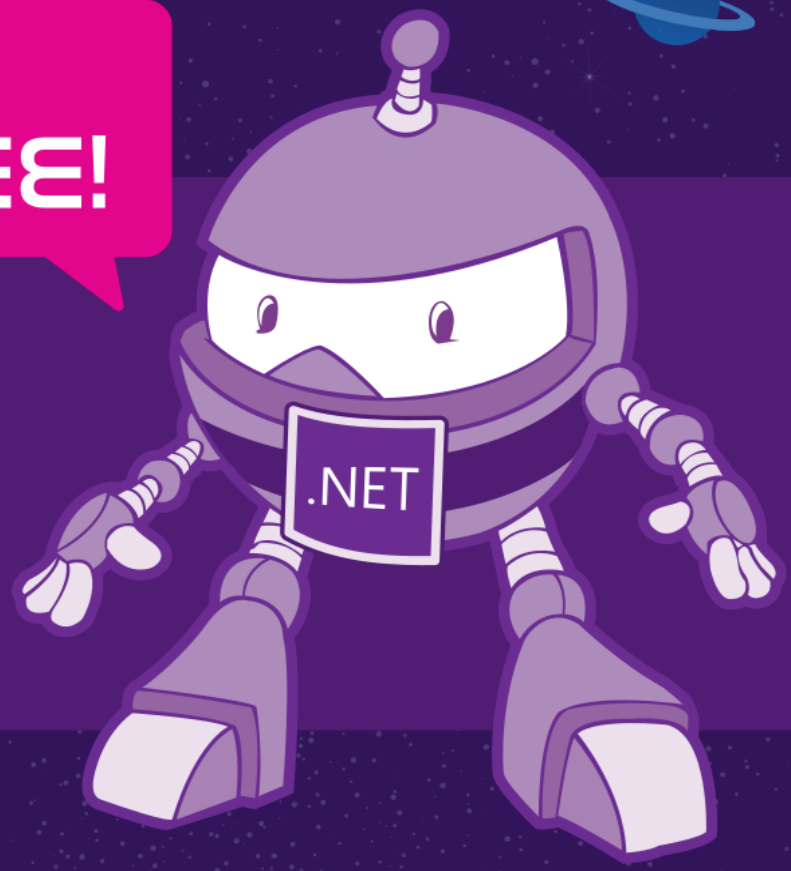
CloudGen
omnidi verso l'evoluzione

It's FREE!

.NET Conf

Discover the world of .NET

September 12-14, 2018



Tune in: www.dotnetconf.net

Marco Zamana

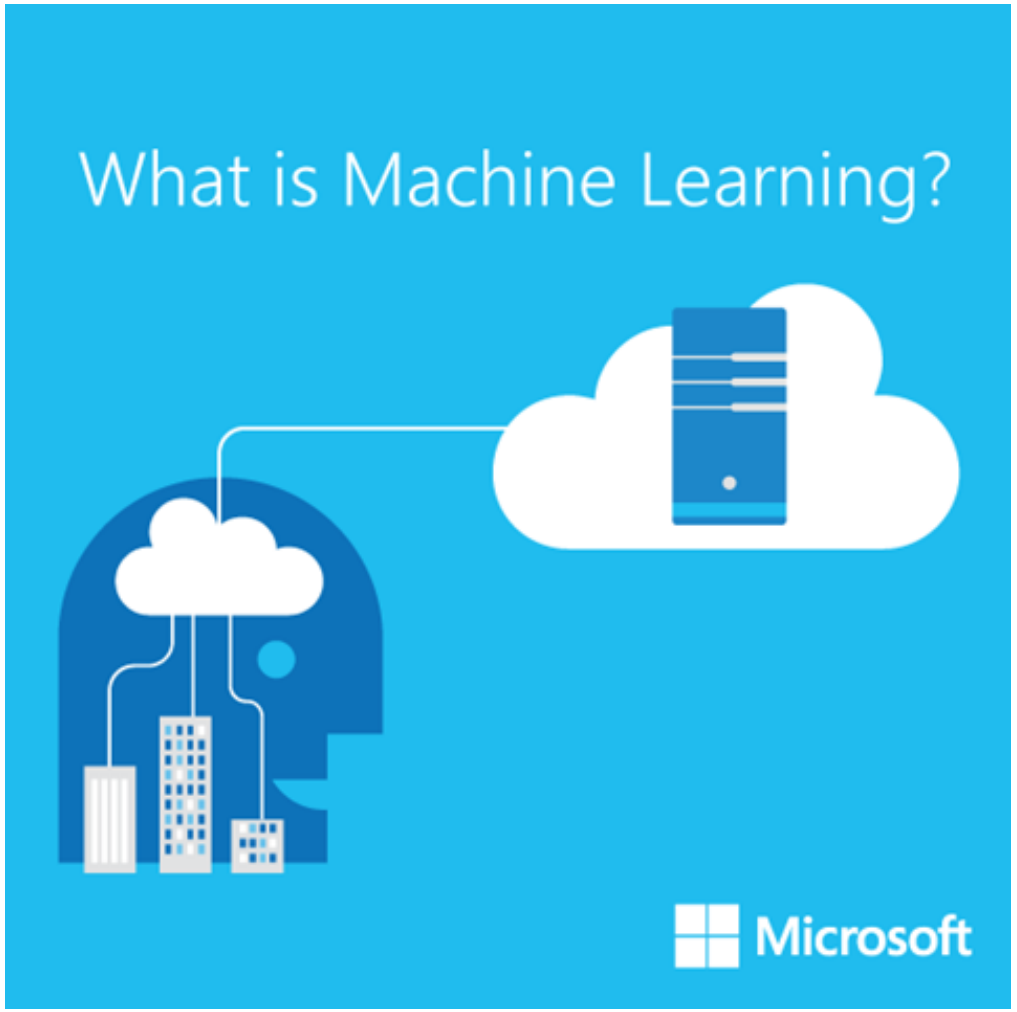
@marco_zamana

/MarcoZama

/zama-marco-zamana

ML.NET

Cosa è Machine Learning?



Machine learning è un tecnica di data science che permette di utilizzare dei dati per per prevedere comportamenti, risultati e tendenze futuri.

ML.NET

Cosa è Machine Learning?

Il Machine Learning crea

$f(x)$

Modello

Usando questi dati

Faccia

Faccia

Non è una faccia

Non è una faccia

Quindi il Machine Learning è un insieme di azioni

Prepara i tuoi dati

Costruisci e istruisci

Esegui

Ma per preparare i dati occorre sapere come analizzarli:

Quindi ci serve una rapida introduzione all'analisi scientifica dei dati.

Analisi dei dati

L’analisi scientifica dei dati

Esistono solo cinque domande a cui l'analisi scientifica dei dati risponde:

È A o B?

È strano?

In che quantità o in che numero?

In che modo sono organizzati i dati?

Qual è il prossimo passo da fare?

È A o B?

Questa è una domanda che ci permette di definire una classificazione a due classi o binaria.

È utile per qualsiasi domanda che può avere solo due risposte possibili.

Ad esempio:

Sarà possibile percorrere le prossime 1.000 miglia con questi pneumatici: Sì o No?

Cosa porta più clienti: un coupon da $ 5 o uno sconto del 25%?

Questa domanda può anche essere riformulata per includere più di due opzioni: È A o B o C o D e così via? Si tratta in questo caso della classificazione multiclasse ed è utile quando sono possibili più risposte o diverse migliaia di risposte. La classificazione multiclasse sceglie la più probabile.

È strano?

Questa è una domanda che ci permette di rilevare delle anomalie.

Il rilevamento delle anomalie segnala eventi o comportamenti imprevisti o insoliti. Fornisce degli indizi su dove cercare per individuare i problemi.

In che quantità o in che numero?

Questa è una domanda che ci permette di definire delle regressioni.

Possiamo effettuare delle previsioni numeriche e quindi rispondere a qualsiasi domanda che richiede informazioni numeriche

In che modo sono organizzati i dati?

Questa è una domanda che ci permette di definire il Clustering.

Cioè possiamo separare i dati in "gruppi" naturali per un'interpretazione più semplice.

Esempi comuni di domande di clustering sono:

A quali spettatori piacciono gli stessi tipi di film?

Quali modelli di stampanti condividono lo stesso malfunzionamento?

Riuscendo a capire come sono organizzati i dati, è possibile comprendere meglio, e prevedere, comportamenti ed eventi.

Qual è il prossimo passo da fare?

Questa è una domanda che ci permette di definire l'apprendimento per rinforzo.

L'apprendimento per rinforzo è stato ispirato dal modo in cui il cervello di topi ed esseri umani risponde alle punizioni e alle ricompense.

Questi algoritmi apprendono dai risultati e stabiliscono l'azione successiva.

In genere, l'apprendimento per rinforzo è una buona scelta per i sistemi automatizzati che devono prendere tante piccole decisioni senza la guida umana.

Gli algoritmi di apprendimento per rinforzo raccolgono i dati durante i processi, imparando dai tentativi e dagli errori.

Criteri dei dati

Criteri per i dati

Per ottenere le risposte desiderate dall'analisi scientifica dei dati, è necessario fornire materiale non elaborato di alta qualità.

In data science esistono alcuni componenti di cui deve essere eseguito il pull contemporaneamente. Tra questi:

- Rilevanti
- Connesso
- Accurati
- In quantità sufficiente

Rilevanza dei dati: la differenza

Irrelevant Data

Price of milk ($/gal)	Red Sox batting avg.	Blood alcohol content (%)
3.79	.304	.03
3.45	.320	.09
4.06	.259	.01
3.89	.298	.05
4.12	.332	.13
3.92	.270	.06
3.23	.294	.10

Relevant Data

Body mass (kg)	Margaritas	Blood alcohol content (%)
103	3	.03
67	5	.09
87	1	.01
52	2	.05
73	5	.13
79	3	.06
110	7	.10

ML.NET

Sono pronti i tuoi dati per l'analisi?

I dati a disposizione sono connessi?

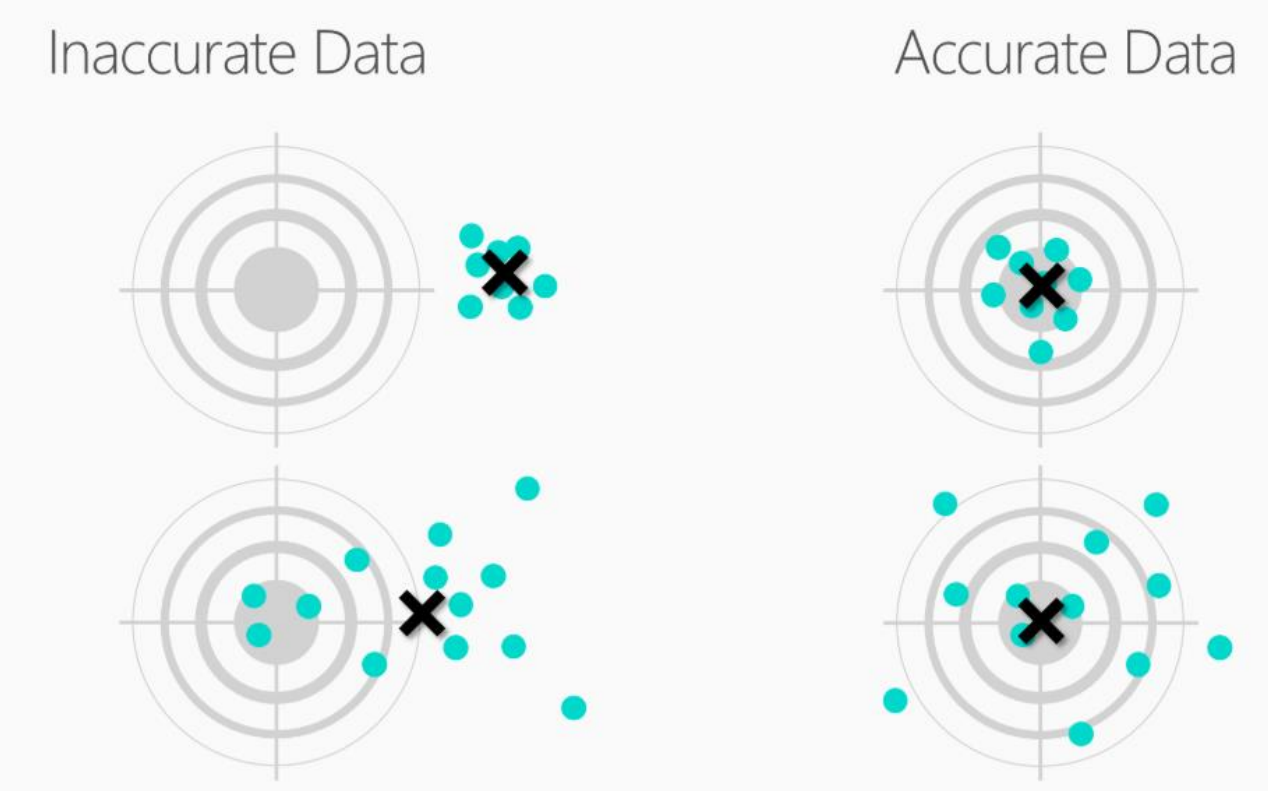
Disconnected Data

Grill temp. (Fahrenheit)	Weight of beef patty (lb)	Burger rating (out of 10)
	.33	8.2
	.24	5.6
550		7.8
725	.45	9.4
600		8.2
625		6.8
	.49	4.2

Connected Data

Grill temp. (Fahrenheit)	Weight of beef patty (lb)	Burger rating (out of 10)
575	.33	8.2
550	.24	5.6
550	.69	7.8
725	.45	9.4
600	.57	8.2
625	.36	6.8
550	.49	4.2

Il successivo check per il controllo dei dati è l'accuratezza degli stessi



Per quanto possa sembrare strano i dati in basso a dx sono accurati nonostante non siano precisi.
Sono distribuiti uniformemente sul bersaglio, quindi i nostri dati sono accurati.
Al contrario nei casi a dx, questi sono comunque fuori o non abbastanza uniformi al nostro bersaglio.

ML.NET

Sono pronti i tuoi dati per l'analisi?

I dati sono sufficienti?
Dati rilevanti, connessi, accurati e in quantità sufficiente rappresentano tutti gli ingredienti necessari per analisi di data science di alta qualità.

Porre una domanda ben strutturata

A una domanda vaga non è necessario rispondere con un nome o un numero.

A una domanda strutturate è obbligatorio rispondere con un nome o un numero.

ML.NET

Se fai una domanda, devi poter rispondere con i dati corretti

Si immagini di incontrare per strada un amico che non si vede per tanto tempo.

«Come va?»

Nonostante siete amici da 20 anni vi risponde in maniera vaga e confusa.

Noi siamo maledettamente curiosi e vogliamo costringerlo a essere più preciso.

Cosa facciamo?

Gli poniamo una domanda così precisa da impedirgli di rispondere genericamente.

Es.

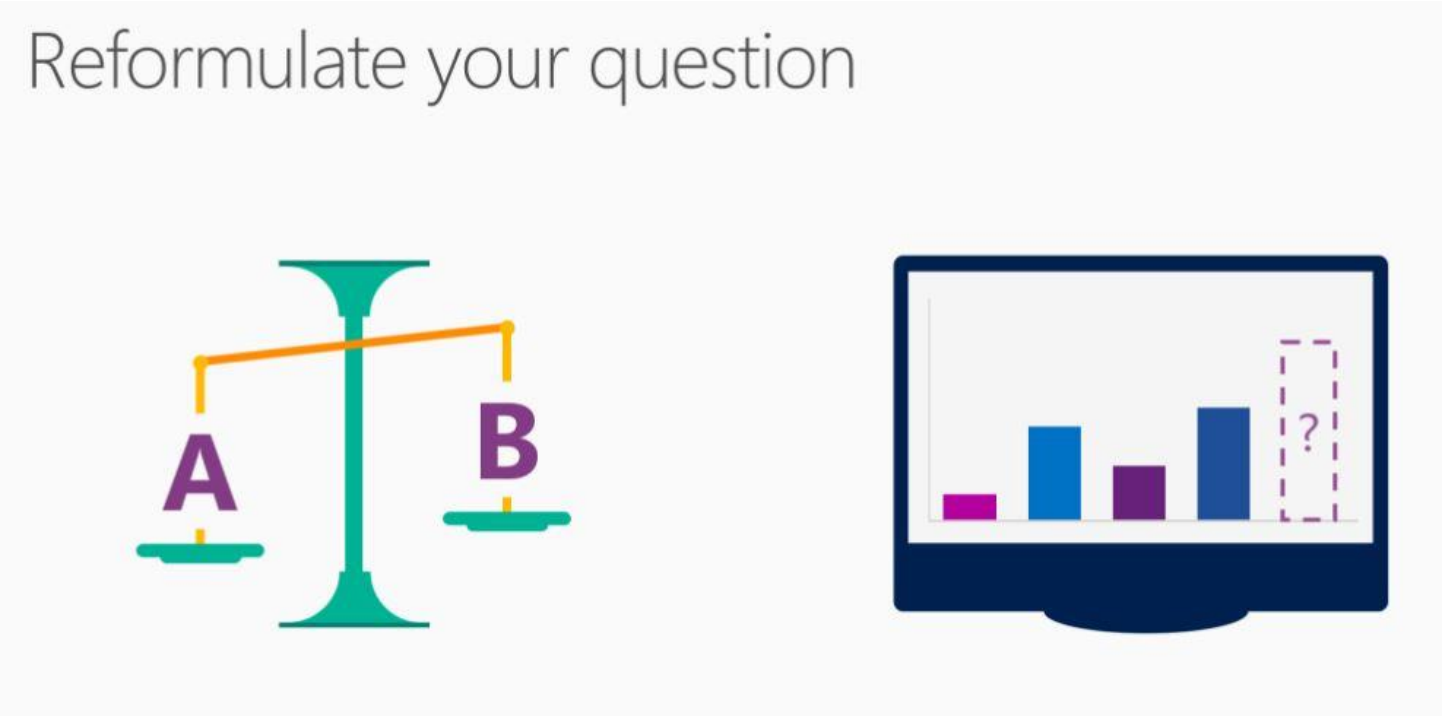
«Come va con il tuo lavoro da direttore?»

«Tuo figlio gioca ancora a calcio?»

Una destinazione è fondamentale per valutare l'esempio di risposta

Riformulare la domanda:

Come si pone una domanda è l'indicazione per individuare l'algoritmo che può fornire una risposta.



RICORDA:

Per creare un modello è essenziale la raccolta di dati rilevanti, accurati, connessi, in quantità sufficiente.

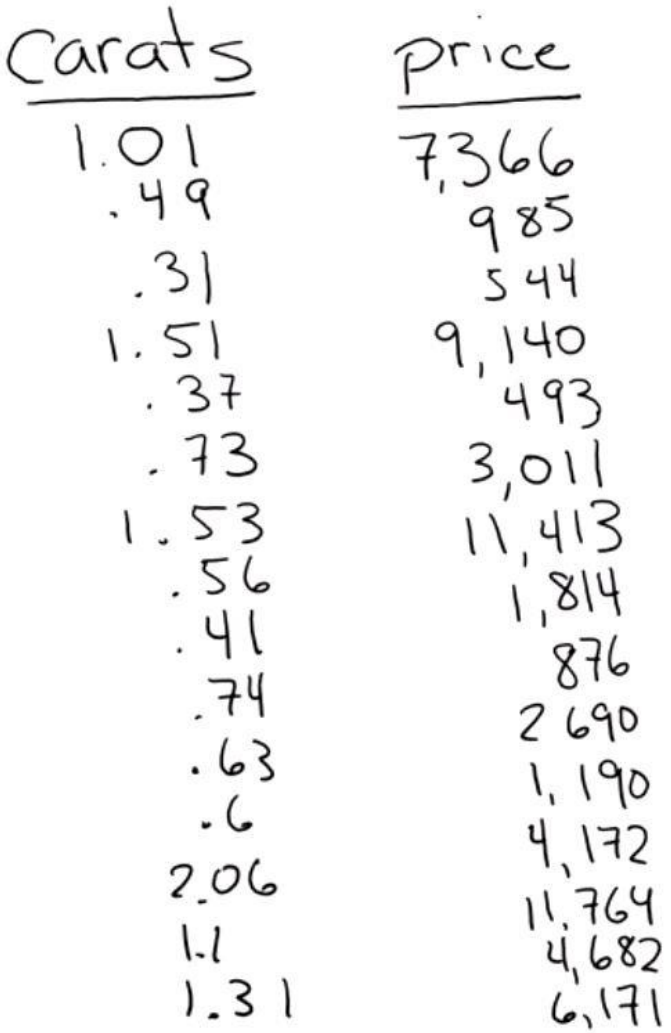
Un *modello* non è altro che una storia semplificata dei dati.

“ But Maybe

Demo strana

ML.NET

Prevedere la risposta



Carats	price
1.01	7,366
.49	985
.31	544
1.51	9,140
.37	493
.73	3,011
1.53	11,413
.56	1,814
.41	876
.74	2 690
.63	1, 190
.6	4, 172
2.06	11.764
1.1	4,682
1.31	6,171

Possiedo un anello che apparteneva a mia nonna con un'incastonatura da 1,35 carati e vorrei farmi un'idea sul suo prezzo.
Prendo un blocco note e una penna nella gioielleria e scrivo il prezzo di tutti i diamanti e il relativo peso in carati.
Iniziando dal primo diamante, pesa 1,01 carati e costa $ 7.366.

Adesso faccio la stessa cosa con tutti gli altri diamanti presenti in negozio.

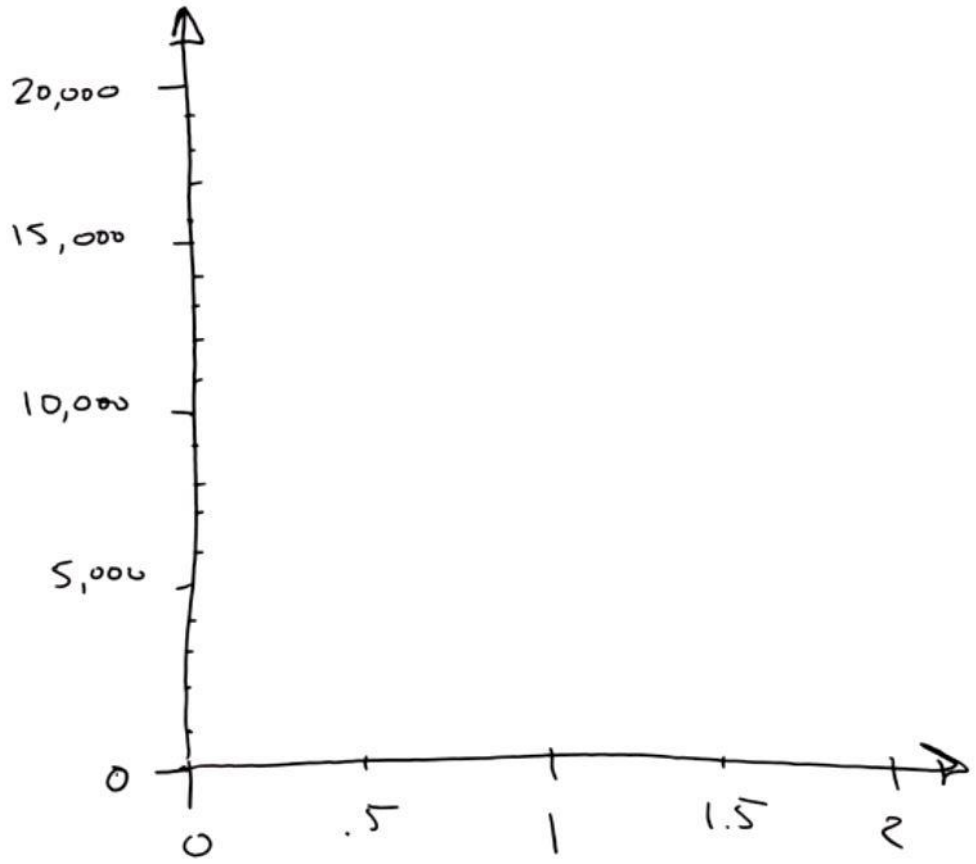
In base alla tabella di prima:

Porre una domanda ben strutturata

La domanda verrà adesso posta in modo ben strutturato: "Quale sarà il costo di acquisto di un diamante da 1,35 carati?"

Nell'elenco non c'è un diamante da 1,35 carati, quindi sarà necessario usare gli altri dati per ottenere una risposta a questa domanda.

Grafico dei dati esistenti

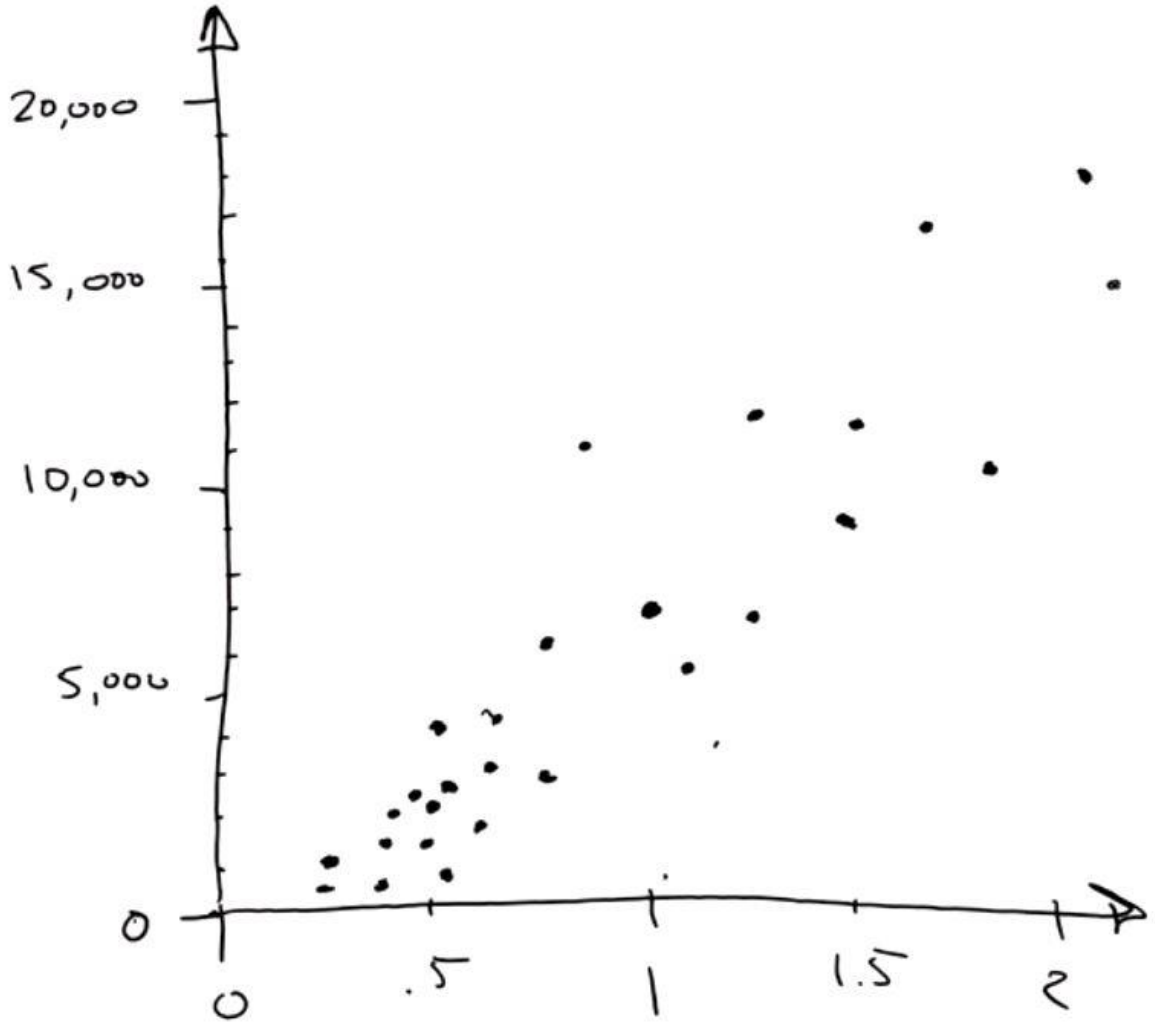


Per prima cosa verrà disegnata una linea numerica orizzontale, detta asse, su cui saranno indicati i pesi. L'intervallo dei pesi va da 0 a 2, quindi verranno disegnati una linea che copre questo intervallo e dei trattini per ciascun mezzo carato.

Adesso, per registrare il prezzo verrà tracciato un asse verticale che sarà connesso all'asse orizzontale del peso. L'asse verticale sarà suddiviso in unità di dollari. A questo punto viene fuori un set di assi coordinati.

ML.NET

Prevedere la risposta

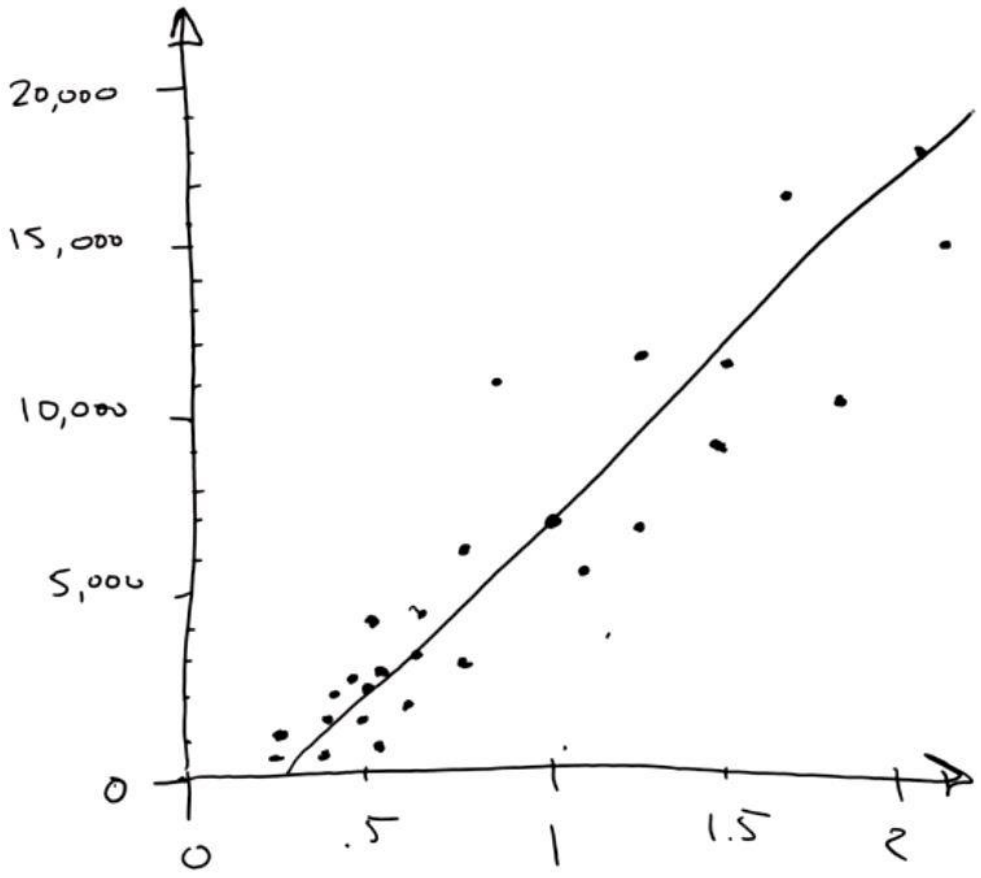


Questo è un *grafico di dispersione*, un'ottima soluzione per visualizzare i set di dati numerici.

Bisognerà fare la stessa cosa per ogni diamante presente nell'elenco.

Una volta terminato, questo è quello che si ottiene: un mucchio di punti, uno per ogni diamante.

Il modello ci siamo!



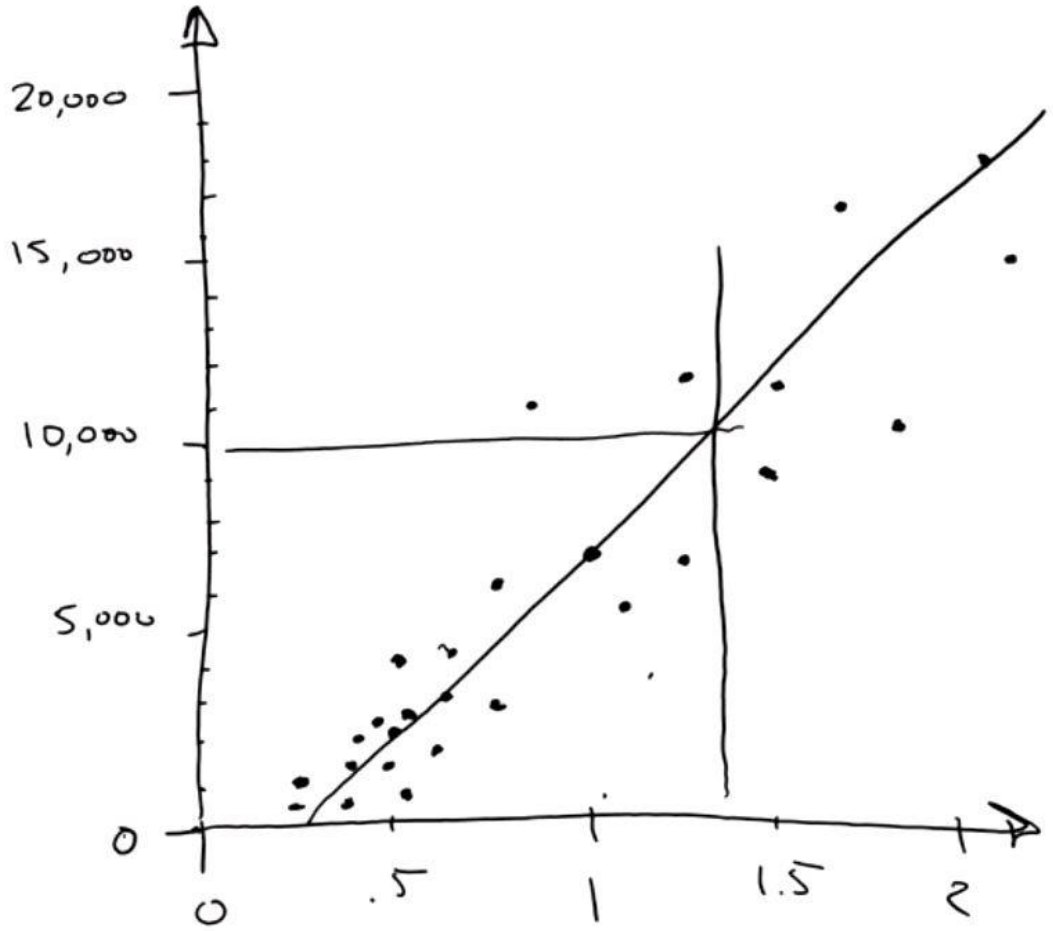
Il fatto che non tutti i punti si trovino esattamente sulla linea va bene.

Esiste infatti il modello, ovvero la linea, anche se a ogni punto vengono associati alcuni *disturbi* o *varianze*.

Vi è quindi la perfetta relazione sottostante e vi è il mondo reale dinamico che aggiunge disturbo e incertezza.

Dal momento che si cerca di rispondere alla domanda *Quanto costa?*, è possibile parlare di *regressione*. E visto che viene usata una linea dritta, si tratta di una *regressione lineare*.

Usi il modello e trovi la risposta



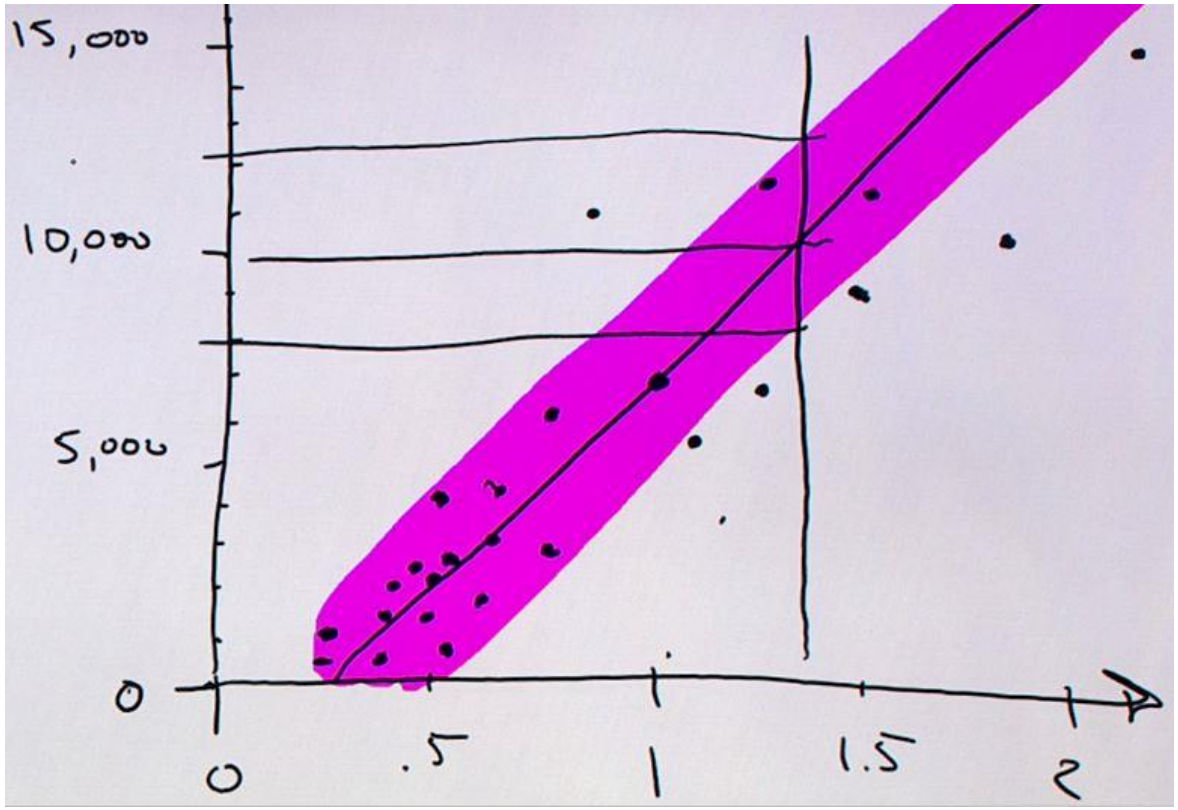
Il modello è pronto dobbiamo solo porgli la domanda:

Quando costerà un diamante da 1,35 carati?

Per rispondere a questa domanda, occorre partire da 1,35 carati e disegnare una linea verticale. Esattamente nel punto in cui incrocia la linea del modello, verrà tracciata una linea orizzontale verso l'asse dei dollari.
Va a colpire proprio i 10.000 dollari.
Boom Shakkalakka!

Quella è la risposta: un diamante da 1,35 carati costa circa $ 10.000.

Creare un intervallo di confidenza



E’ precisa questa previsione?

Per arrivare a capirlo bisogna disegnare un inviluppo attorno alla linea di regressione che include la maggior parte dei punti.

Questo inviluppo viene detto *intervallo di confidenza*: si è abbastanza sicuri che i prezzi rientrino in questo inviluppo, poiché è stato così per la maggior parte degli stessi prezzi in precedenza.

Cosa abbiamo fatto?

- Abbiamo posto una domanda a cui è stato possibile rispondere con i dati
- È stato creato un *modello* usando una *regressione lineare*.
- È stata realizzata una *previsione*, completa di *intervallo di confidenza*

ML.NET

E se avessimo voluto più informazioni?

Se fossero state necessarie ulteriori informazioni, come:

- il taglio del diamante
- le variazioni di colore (quanto si avvicina al bianco il diamante)
- il numero di inclusioni nel diamante

In quel caso, la matematica diventa utile.

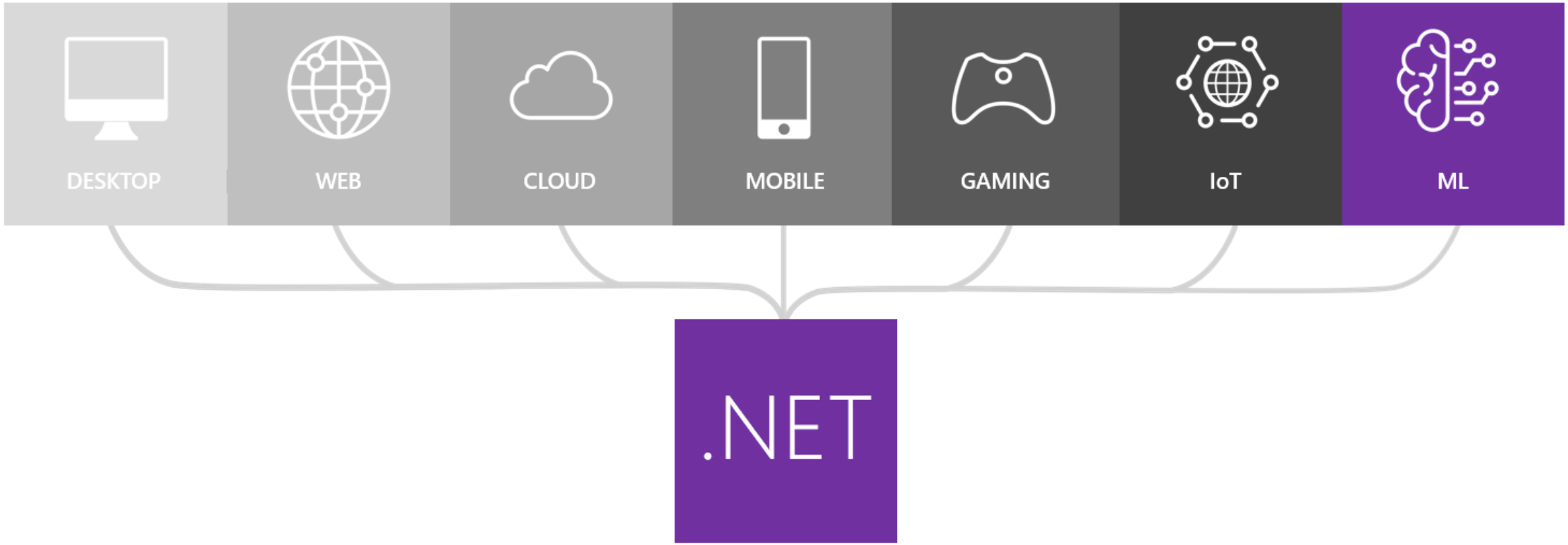
Se le colonne sono più di due, è difficile disegnare punti su carta.

La matematica consente di adattare una determinata linea o un determinato piano ai dati in modo molto preciso.

Il framework

ML.NET

Il framework



Cosa è?

ML.NET, un framework di apprendimento automatico multipiattaforma e open source.

ML.NET permette allo sviluppatore .NET di sviluppare i propri modelli dati e includere le ML personalizzate nelle proprie applicazioni.

Questo senza una precedente esperienza nello sviluppo o nell'ottimizzazione dei modelli di apprendimento automatico.

(But Maybe…(cit.))

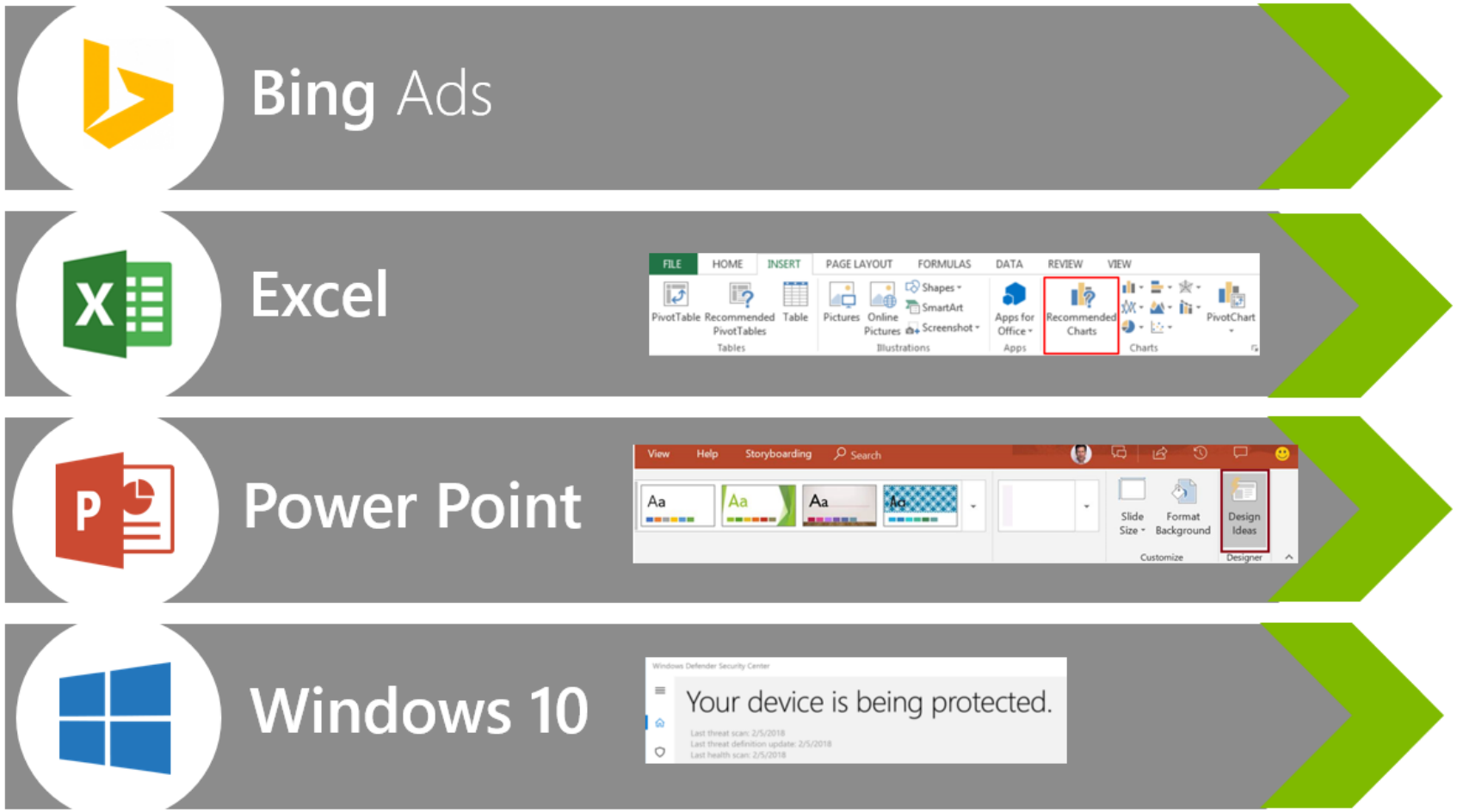
Nuovo progetto?

ML.NET è stato originariamente sviluppato in Microsoft Research e si è evoluto significativamente nell'ultimo decennio.

Viene utilizzato in molti gruppi di prodotti in Microsoft come Windows, Bing, Azure e altri

ML.NET

Il framework



ML.NET

Il framework

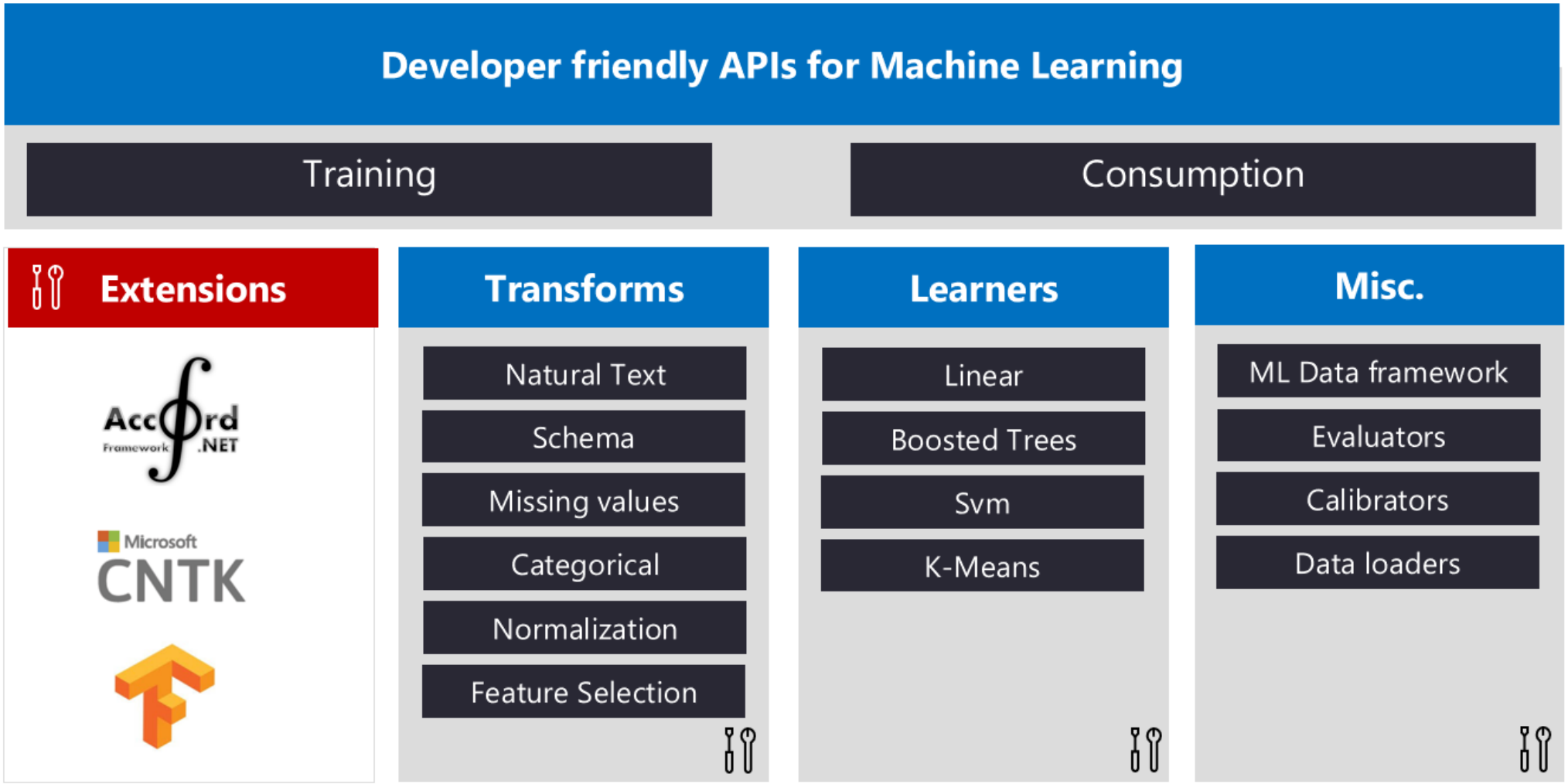
ML.NET è stato lanciato come parte di .NET Foundation e il repository contiene le API .NET C # per il training e l'uso dei modelli, oltre a una grande varietà di trasformazioni e di apprendimenti necessari per molte attività di ML come la regressione e classificazione.

ML.NET ha lo scopo di fornire il flusso di lavoro end to end per fondere assieme ML e applicazioni .NET per la pre-elaborazione, ingegneria delle funzioni, la modellazione, la valutazione e l'operatività.

ML.NET include tutti i tipi e il runtime. Assieme sono necessari per tutti gli aspetti dell'apprendimento automatico, compresi i tipi di dati principali, condotte estensibili, matematica ad alte prestazioni, strutture dati per dati eterogenei, supporto di strumenti e altro ancora.

ML.NET

Il framework



RoadMap

Nel tempo, ML.NET svilupperà altri scenari ML come i sistemi di raccomandazione, il rilevamento di anomalie e altri approcci, come l'apprendimento approfondito, sfruttando librerie di deep learning come TensorFlow, Caffe2 e CNTK e librerie generali di apprendimento automatico come Accord.Net

Demo 1
Analisi dei sentimenti

PROBLEMA

Il problema consiste nel predire se la recensione di un cliente è positiva o negativa.

Questo tipo di predizione è detta **Classificazione Binaria**

Questi commenti sono stati processati, cioè gestiti da delle persone, e ad ogni commento è stata assegnata una label:

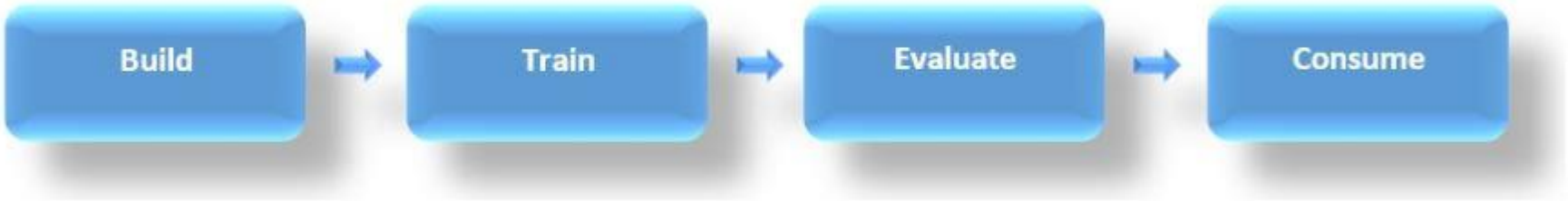
0 - negativo

1 – positivo

Costruiremo un modello che analizza una stringa e predice un valore di sentimento uguale a 1 o 0.

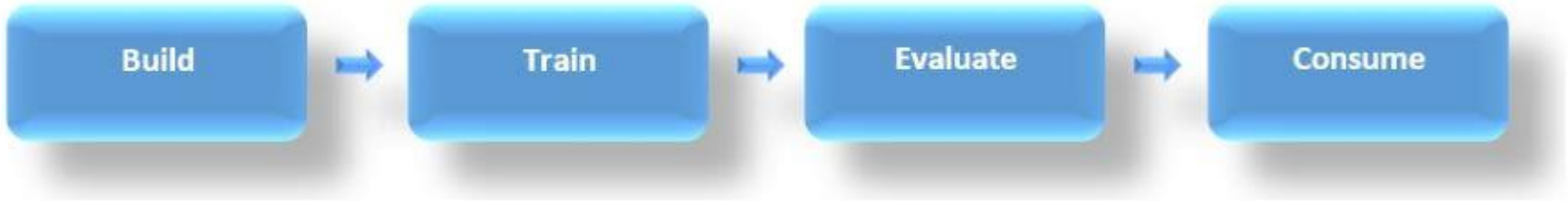
1- Costruire il modello

```
// LearningPipeline holds all steps of the learning process: data, transforms, learners.
var pipeline = new LearningPipeline();
// The TextLoader loads a dataset. The schema of the dataset is specified by passing a class containing
// all the column names and their types.
pipeline.Add(new TextLoader(TrainDataPath).CreateFrom<SentimentData>());
// TextFeaturizer is a transform that will be used to featurize an input column to format and clean the data.
pipeline.Add(new TextFeaturizer("Features", "SentimentText"));
// FastTreeBinaryClassifier is an algorithm that will be used to train the model.
// It has three hyperparameters for tuning decision tree performance.
pipeline.Add(new FastTreeBinaryClassifier() {NumLeaves = 5, NumTrees = 5, MinDocumentsInLeafs = 2});
```



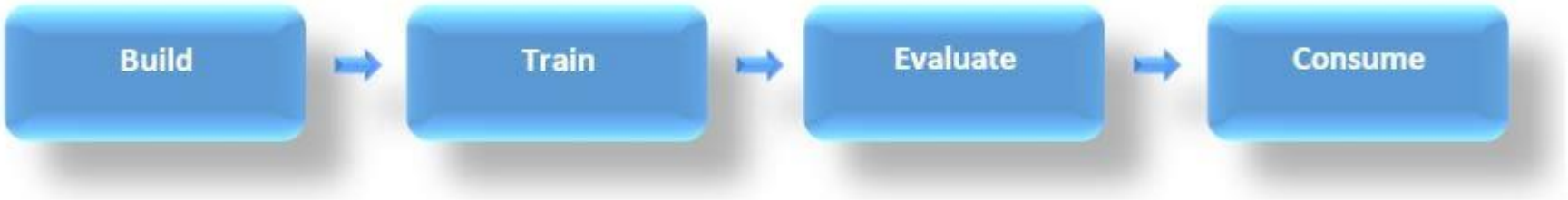
2 - Addestrare il modello

```
var model = pipeline.Train<SentimentData, SentimentPrediction>();
```



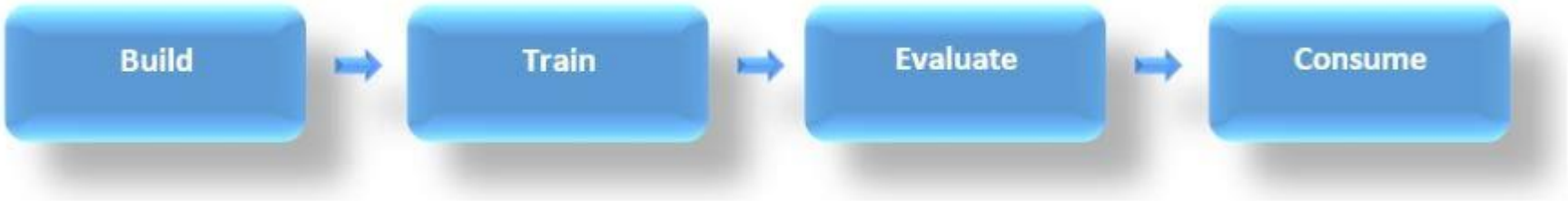
3 – Valuta il modello

```
var testData = new TextLoader(TestDataPath).CreateFrom<SentimentData>();

var evaluator = new BinaryClassificationEvaluator();
var metrics = evaluator.Evaluate(model, testData);
```



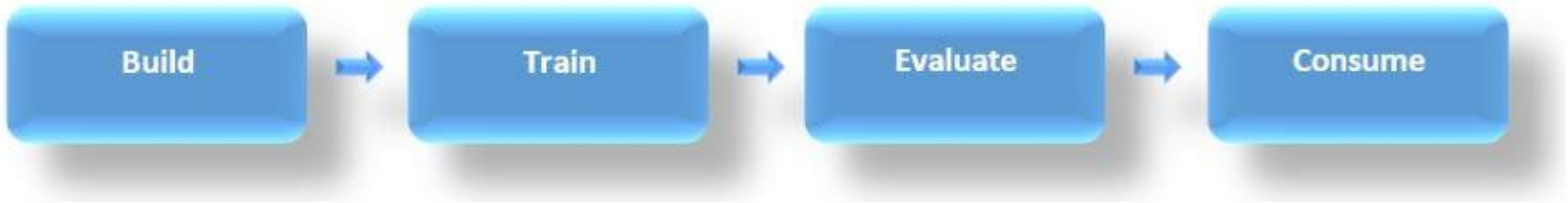
4 – Consuma il modello

```
var predictions = model.Predict(TestSentimentData.Sentiments);
```



4 – Consuma il modello

```
internal static readonly IEnumerable<SentimentData> Sentiments = new[]
{
    new SentimentData
    {
        SentimentText = "Contoso's 11 is a wonderful experience",
        Sentiment = 0
    },
    new SentimentData
    {
        SentimentText = "The acting in this movie is very bad",
        Sentiment = 0
    },
    new SentimentData
    {
        SentimentText = "Joe versus the Volcano Coffee Company is a great film.",
        Sentiment = 0
    }
};
```



Demo 2

Multiclassificazione di Iris

PROBLEMA

Questo problema è incentrato sulla previsione del tipo di un fiore dell'iris (setosa, versicolor o virginica) in base ai parametri del fiore come la lunghezza del petalo, la larghezza del petalo, ecc.

Per risolvere questo problema dovremmo costruire un modello che prende in input 4 parametri:

- petal length
- petal width
- sepal length
- sepal width

PROBLEMA

E predire a che tipo iris appartiene il fiore:

- setosa
- versicolor
- Virginica

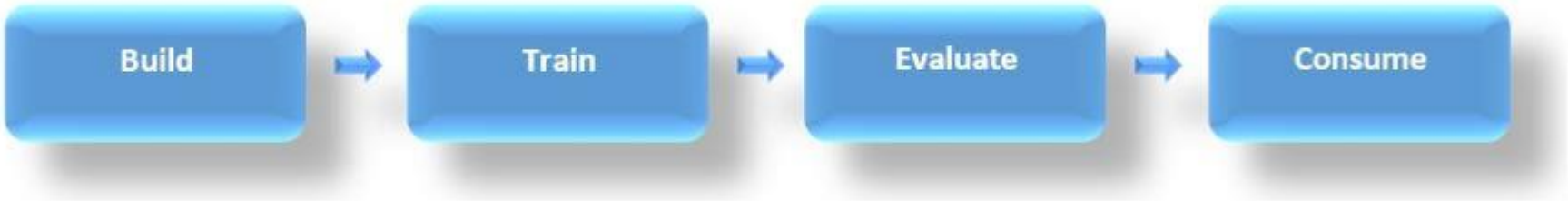
Per essere precisi, il modello restituirà le probabilità che il fiore appartenga a ciascun tipo.

1 – Costruisci il modello

```
// LearningPipeline holds all steps of the learning process: data, transforms, learners.
var pipeline = new LearningPipeline
{

    // The TextLoader loads a dataset. The schema of the dataset is specified by passing a class containing
    // all the column names and their types.
    new TextLoader(TrainDataPath).CreateFrom<IrisData>(),

    When ML model starts training, it looks for two columns: Label and Features.
    // Transforms
    //              like in this example, no extra actions required.
    // Label:   values that should be predicted. If you have a field named Label in your data type,
    //          If you don't have it, copy the column you want to predict with ColumnCopier transform:
    //              new ColumnCopier(("FareAmount", "Label"))
    // Features: all data used for prediction. At the end of all transforms you need to concatenate
    //              all columns except the one you want to predict into Features column with
    //              ColumnConcatenator transform:
    new ColumnConcatenator("Features",
        "SepalLength",
        "SepalWidth",
        "PetalLength",
        "PetalWidth"),
    // StochasticDualCoordinateAscentClassifier is an algorithm that will be used to train the model.
    new StochasticDualCoordinateAscentClassifier()
}
```

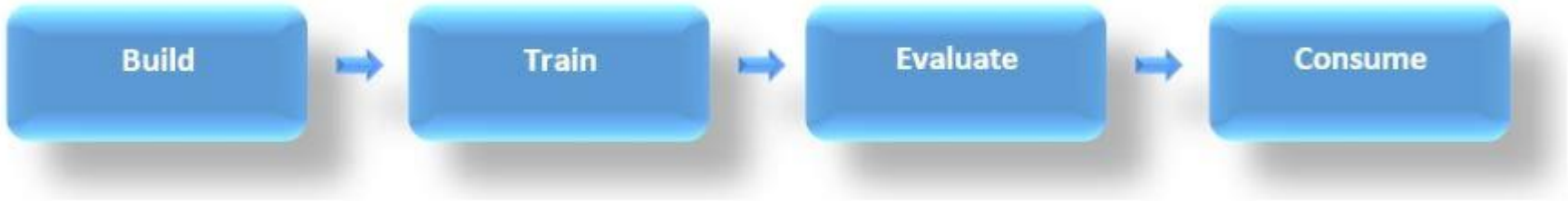
2 - Addestrare il modello

```
var model = pipeline.Train<IrisData, IrisPrediction>();
```



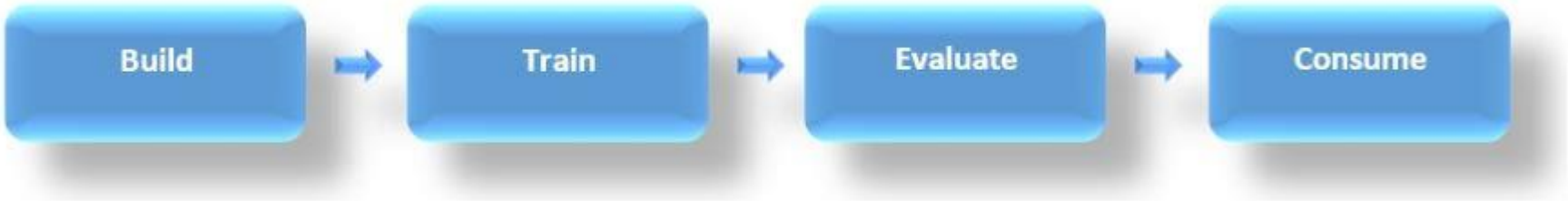
3 – Valuta il modello

```
var testData = new TextLoader(TestDataPath).CreateFrom<IrisData>();

var evaluator = new ClassificationEvaluator {OutputTopKAcc = 3};
var metrics = evaluator.Evaluate(model, testData);
```



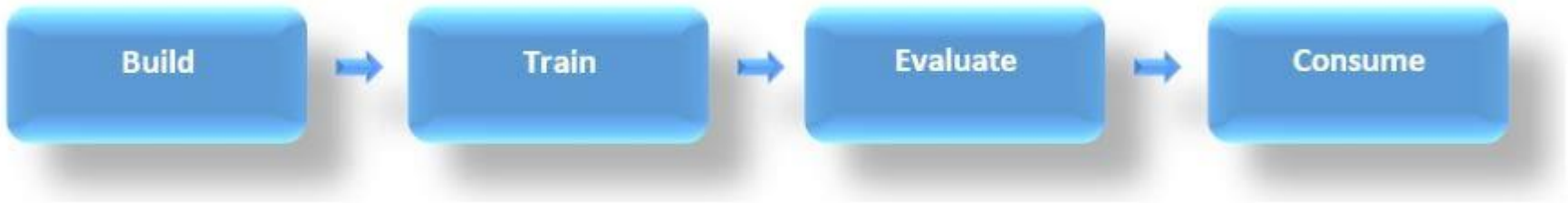
4 – Consuma il modello

```
var prediction = model.Predict(TestIrisData.Iris1);

Console.WriteLine($"Actual: setosa.     Predicted probability: setosa:      {prediction.Score[0]:0.####}");
Console.WriteLine($"                                           versicolor:  {prediction.Score[1]:0.####}");
Console.WriteLine($"                                           virginica:   {prediction.Score[2]:0.####}");
```



4 – Consuma il modello

```
internal class TestIrisData
{
    internal static readonly IrisData Iris1 = new IrisData()
    {
        SepalLength = 3.3f,
        SepalWidth = 1.6f,
        PetalLength = 0.2f,
        PetalWidth= 5.1f,
    }
    (...)
}
```



Demo 3
Clustering di Iris

PROBLEMA

Per dimostrare l'API di clustering in azione, utilizzeremo tre tipi di fiori di iris:

- setosa
- versicolor
- virginica

Tutti loro sono memorizzati nello stesso set di dati.

Anche se il tipo di questi fiori è noto, non lo useremo ed eseguiremo l'algoritmo di clustering solo sui parametri dei fiori come la lunghezza del petalo, la larghezza del petalo, ecc. Il compito è raggruppare tutti i fiori in tre diversi cluster. Ci aspetteremmo che i fiori di diversi tipi appartengano a cluster diversi.

PROBLEMA

Tutti i fiori sono memorizzati nello stesso set di dati.

Anche se il tipo di questi fiori è noto, non lo useremo ed eseguiremo l'algoritmo di clustering solo sui parametri dei fiori come la lunghezza del petalo, la larghezza del petalo, ecc.

Il compito è raggruppare tutti i fiori in tre diversi cluster.

Ci aspetteremmo che i fiori di diversi tipi appartengano a cluster diversi.

Gli input del modello sono i seguenti parametri

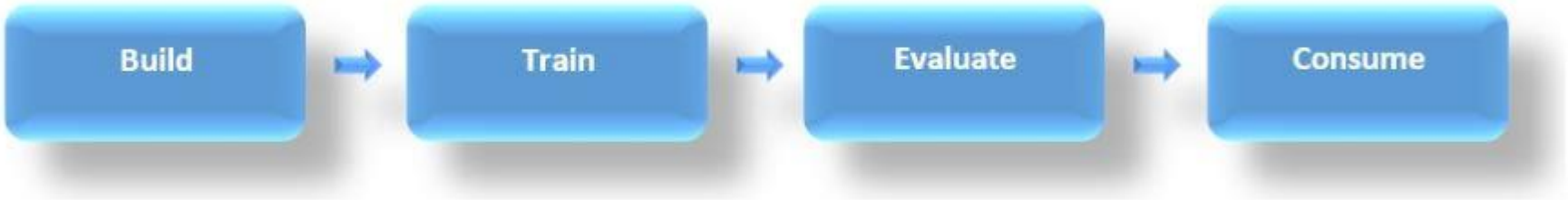
- petal length
- petal width
- sepal length
- sepal width

1 – Costruisci il modello

```
// LearningPipeline holds all steps of the learning process: data, transforms, learners.
var pipeline = new LearningPipeline
{
    // The TextLoader loads a dataset. The schema of the dataset is specified by passing a class containing
    // all the column names and their types.
    new TextLoader(DataPath).CreateFrom<IrisData>(useHeader: true),
    // ColumnConcatenator concatenates all columns into Features column
    new ColumnConcatenator("Features",
        "SepalLength",
        "SepalWidth",
        "PetalLength",
        "PetalWidth"),
    // KMeansPlusPlusClusterer is an algorithm that will be used to build clusters. We set the number of clusters to
    new KMeansPlusPlusClusterer() { K = 3 }
};
```

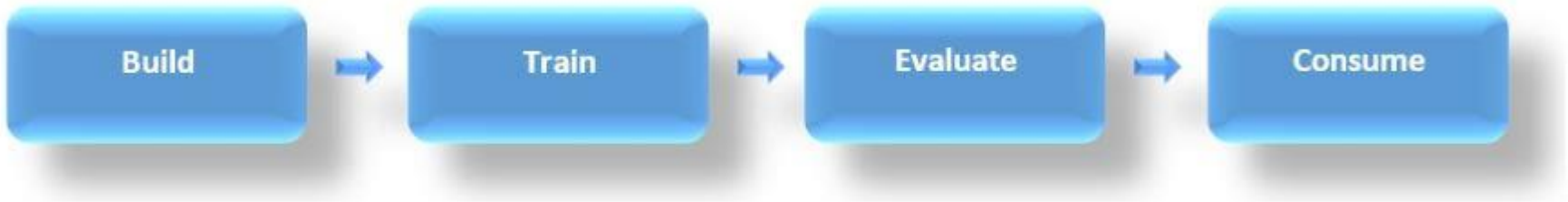
2 - Addestrare il modello

```
var model = pipeline.Train<IrisData, ClusterPrediction>();
```



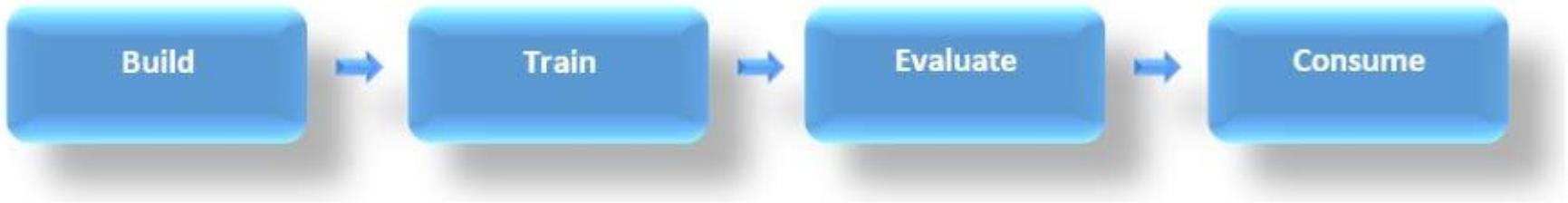
3 – Consuma il modello

```
var prediction1 = model.Predict(TestIrisData.Setosa1);
```



3 – Consuma il modello

```
internal class TestIrisData
{
    internal static readonly IrisData Iris1 = new IrisData()
    {
        SepalLength = 3.3f,
        SepalWidth = 1.6f,
        PetalLength = 0.2f,
        PetalWidth= 5.1f,
    }
    (...)
}
```



Demo 3

Pronostico tariffa di taxi in NY

PROBLEMA

Questo problema è incentrato sulla previsione della tariffa di un viaggio in taxi a New York City.

A prima vista, può sembrare che dipenda semplicemente dalla distanza percorsa.

Tuttavia, i venditori di taxi a New York addebiteranno importi variabili per altri fattori come passeggeri aggiuntivi, pagando con carta di credito anziché contanti e così via.

Questa previsione può essere utilizzata in applicazioni per i fornitori di taxi per fornire agli utenti e ai conducenti una stima delle tariffe di corsa.

PROBLEMA

Per risolvere questo problema costruiremo un modello che prende in input i seguenti valori:

- vendor ID
- rate code
- passenger count
- trip time
- trip distance
- payment type

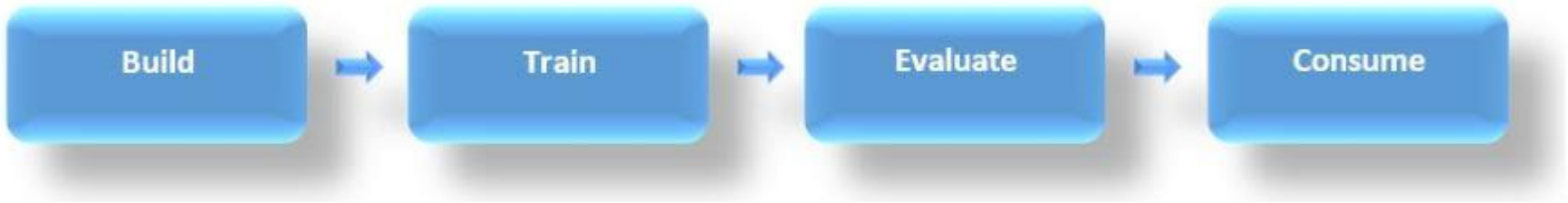
E cercheremo di predire il costo della corsa

1 – Costruisci il modello

```
// LearningPipeline holds all steps of the learning process: data, transforms, learners.
var pipeline = new LearningPipeline
{
    // The TextLoader loads a dataset. The schema of the dataset is specified by passing a class containing
    // all the column names and their types. This will be used to create the model, and train it.
    new TextLoader(TrainDataPath).CreateFrom<TaxiTrip>(separator:','),

    // Transforms
    // When ML model starts training, it looks for two columns: Label and Features.
    // Label:   values that should be predicted. If you have a field named Label in your data type,
    //              no extra actions required.
    //          If you don't have it, like in this example, copy the column you want to predict with
    //              ColumnCopier transform:
    new ColumnCopier(("FareAmount", "Label")),

    // CategoricalOneHotVectorizer transforms categorical (string) values into 0/1 vectors
    new CategoricalOneHotVectorizer("VendorId",
        "RateCode",
    "PaymentType"),
    // Features: all data used for prediction. At the end of all transforms you need to concatenate
    //              all columns except the one you want to predict into Features column with
    //              ColumnConcatenator transform:
    new ColumnConcatenator("Features",
        "VendorId",
        "RateCode",
        "PassengerCount",
        "TripDistance",
        "PaymentType"),
        //FastTreeRegressor is an algorithm that will be used to train the model.
    new FastTreeRegressor()
    };
```

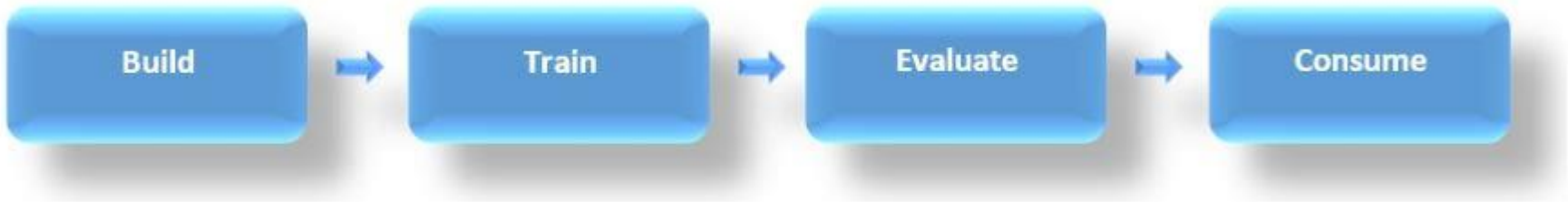
2 - Addestrare il modello

```
var model = pipeline.Train<TaxiTrip, TaxiTripFarePrediction>();
```



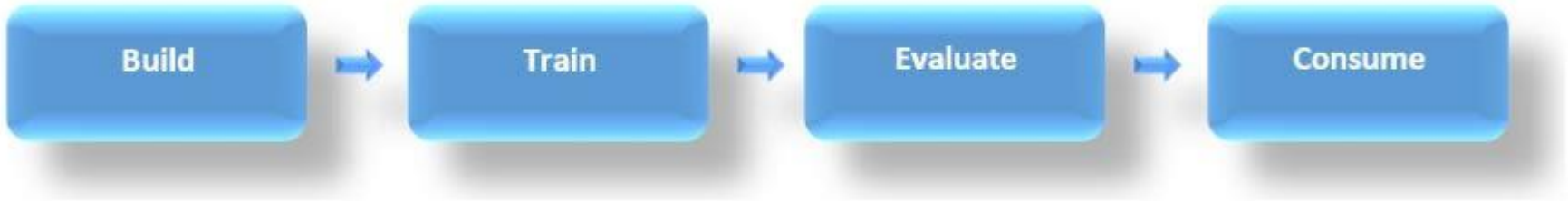
3 – Valuta il modello

```
var testData = new TextLoader(TestDataPath).CreateFrom<TaxiTrip>(separator: ',');

var evaluator = new RegressionEvaluator();
var metrics = evaluator.Evaluate(model, testData);
```



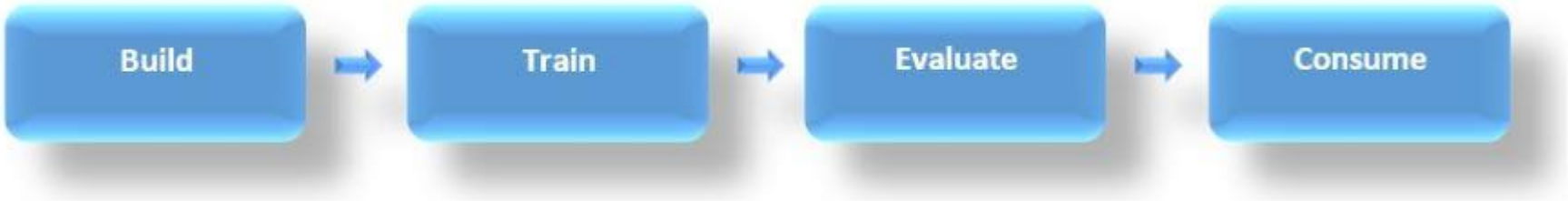
4 – Consuma il modello

```
var prediction = model.Predict(TestTaxiTrips.Trip1);
Console.WriteLine($"Predicted fare: {prediction.FareAmount:0.####}, actual fare: 29.5");
```



4 – Consuma il modello

```
internal class TestTaxiTrips
    {
        internal static readonly TaxiTrip Trip1 = new TaxiTrip
        {
            VendorId = "VTS",
            RateCode = "1",
            PassengerCount = 1,
            TripDistance = 10.33f,
            PaymentType = "CSH",
            FareAmount = 0 // predict it. actual = 29.5
        };
    }
```



“ But Maybe

ML.NET

Problemi di cuore

Quale Algoritmo è quello giusto?

Dipende da quello che si deve fare.
Per ognuno la documentazione è abbastanza esaustiva.

Per i task invece vi rimando qui:
https://github.com/dotnet/docs/blob/a848fb04b815b5337458a17668b3481e9f27370e/docs/machine-learning/resources/tasks.md

Learner Name	Binary classification	Multi-class classification	Regression	Ranking	Clustering	Anomaly Detection
Averaged Perceptron	✔					
FastForest	✔		✔			
FastTree	✔		✔	✔		
Generalized Additive Models	✔		✔			
K-Means++					✔	
Linear SVM	✔					
Logistic Regressor	✔	✔				
Naïve Bayes		✔				
Online Gradient Descent			✔			
Ordinary Least Squares			✔			
PCA Anomaly Detector						✔
Poisson Regression			✔			
Stochastic Dual Coordinate Ascent	✔	✔	✔			
Stochastic Gradient Descent	✔					
Tweedie FastTree			✔			

Applicazione
Problemi di cuore?

Verona 5/10/2012

Dimissioni Ospedale per morbo di Still:

Morbo di Still - è una **malattia infiammatoria sistemica**, dal carattere cronico, che si contraddistingue principalmente per febbre alta, rush cutaneo, mal di gola e dolore articolare.

Per la sua somiglianza con l'**artrite reumatoide**, il morbo di Still viene considerato anche una forma particolare di **artrite**.

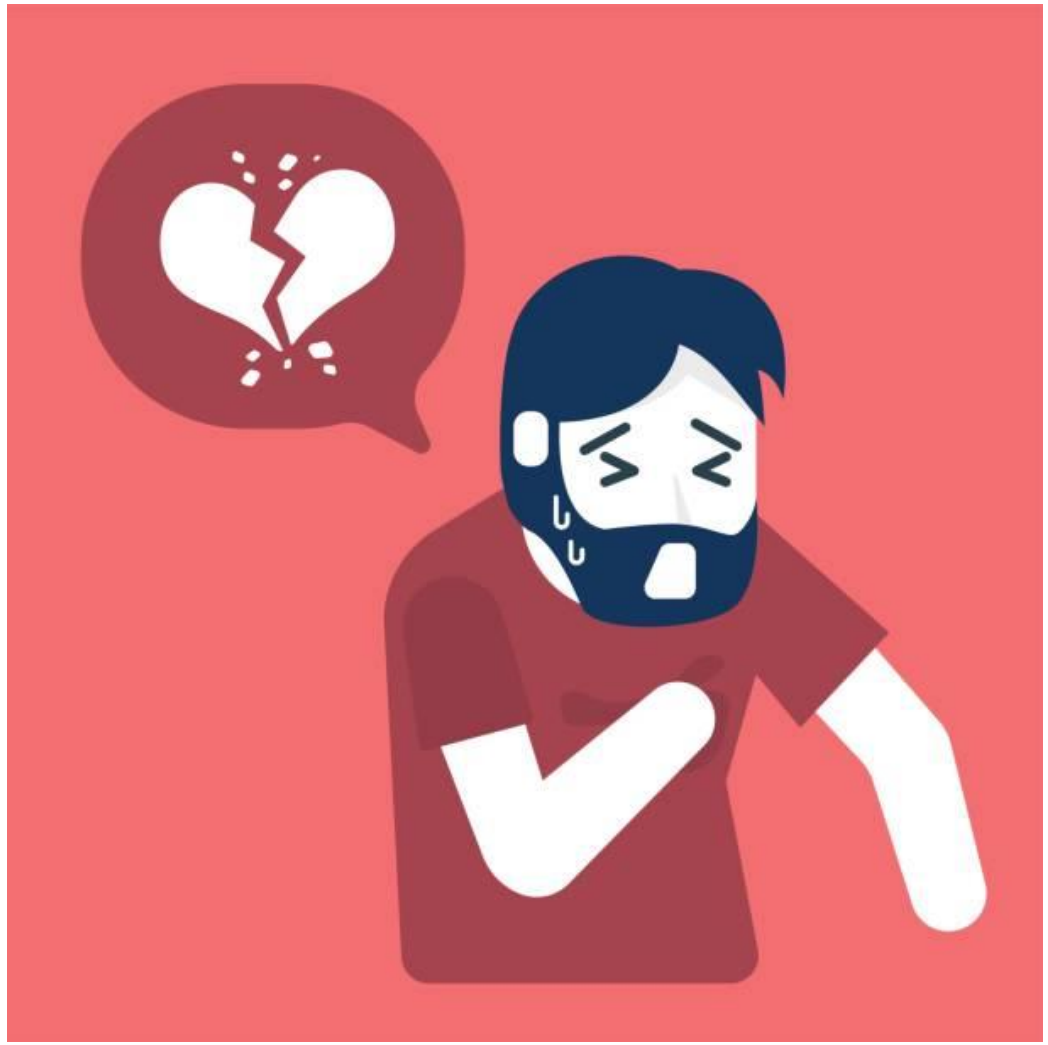
Tratto da https://www.my-personaltrainer.it/salute-benessere/morbo-di-still.html

Verona 02/03/2017

L'**ipertensione arteriosa** è uno stato, costante e non occasionale, in cui la pressione arteriosa a riposo risulta più alta rispetto agli standard fisiologici considerati normali.

L'ipertensione è una tra le malattie più diffuse nei Paesi industrializzati; colpisce, infatti, circa il 20% della popolazione adulta e rappresenta uno dei maggiori problemi clinici dei tempi moderni.
L'ipertensione arteriosa è conosciuta anche come "**killer silenzioso**", perché non comporta alcun sintomo e agisce nell'ombra, degenerando in complicanze severe, talvolta dall'esito mortale.

Tratto da https://www.my-personaltrainer.it/ipertensione/ipertensione.html



CodeGen Heart Prediction

L’idea è quella di creare un’applicazione complessa per fare capire dove inserire il framework di ML.NET e come utilizzarlo.

Ci serve un dataset e la sua analisi

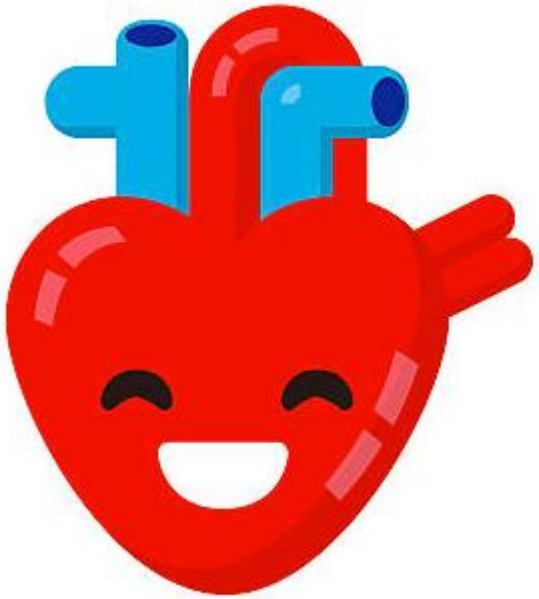
ML.NET

Problemi di cuore


```
1. #3 (age)
2. #4 (sex)
3. #9 (cp)
    chest pain type
    -- Value 1: typical angina
    -- Value 2: atypical angina
    -- Value 3: non-anginal pain
    -- Value 4: asymptomatic

4. #10 (trestbps)
    resting blood pressure (in mm Hg on admission to the hospital)

5. #12 (chol)
    serum cholestoral in mg/dl

6. #16 (fbs)
    (fasting blood sugar > 120 mg/dl)

7. #19 (restecg)
    resting electrocardiographic results
    -- Value 0: normal
    -- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
    -- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

8. #32 (thalach)
    maximum heart rate achieved

9. #38 (exang)
     exercise induced angina (1 = yes; 0 = no)

10. #40 (oldpeak)
    ST depression induced by exercise relative to rest

11. #41 (slope)
    the slope of the peak exercise ST segment
    -- Value 1: upsloping
    -- Value 2: flat
    -- Value 3: downsloping

12. #44 (ca)
    number of major vessels (0-3) colored by flourosopy

13. #51 (thal)
    3 = normal; 6 = fixed defect; 7 = reversable defect
```

ML.NET

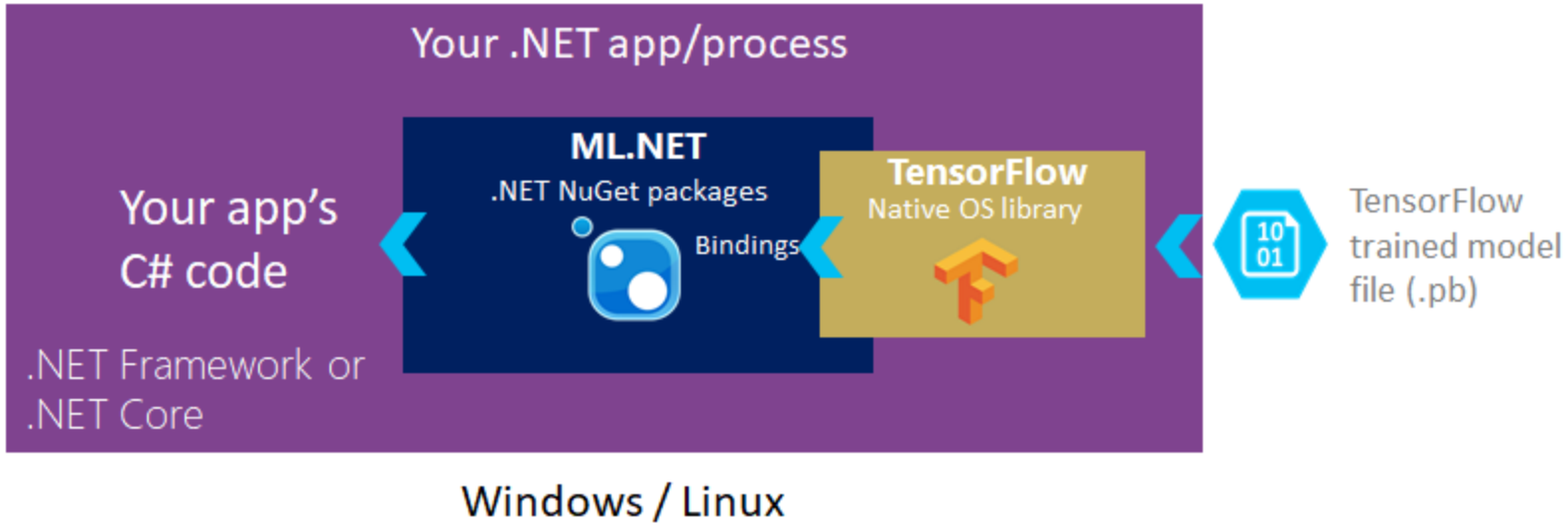
Problemi di cuore



Ddiagnosis of heart disease
(angiographic disease status)

> 0 heart disease

ML.NET 0.5



- Added a TensorFlow model scoring transform (TensorFlowTransform) to ML.NET v0.5
- New ML.NET API proposal exploration for you to provide feedback for upcoming versions

ML.NET 0.5

Riporto dal Blog dotNet

Why ML.NET is switching from the LearningPipeline API to a new API?

Specifically, the new ML.NET API offers attractive features which aren't possible with the current LearningPipeline API:

Strongly-typed API: This new Strongly-typed API takes advantage of C# capabilities so errors can be discovered in compilation time along with improved Intellisense in the editors.

Better flexibility: This API provides a decomposable train and predict process, eliminating rigid and linear pipeline execution. With the new API, execute a certain code path and then fork the execution so multiple paths can re-use the initial common execution. For example, share a given transforms' execution and transformed data with multiple learners and trainers, or decompose pipelines and add multiple learners.

Grazie

Domande?


/MarcoZama


@marco_zamana


/zama-marco-zamana