

JMFLORESZAZO

software craftsman & digital creative



ML para desarrollares .NET

Bienvenidos

Acerca de...



Jose María Flores Zazo, autor

*¡Hola! Gracias por entrar en “ML para desarrolladores .NET”.
Espero poder aportarte los conocimientos mínimos y necesarios
para que puedas ponerlo en práctica.*



Introducción

Resumen del curso

ML – Machine Learning

El desarrollo de soluciones de aprendizaje automático (ML) es una habilidad muy solicitada hoy en día. Esta habilidad no está limitada a los científicos de datos. A los desarrolladores de aplicaciones .NET tradicionales cada vez más se les está pidiendo que integren algoritmos de ML en sus aplicaciones de cliente.

ML.NET hace que sea más fácil para los desarrolladores de .NET aprovechar el poder de ML en aplicaciones .NET sin pasar por la curva de aprendizaje de otros lenguajes como Python o R.

Veremos la API de ML.NET para darte una idea de como usar ML en .NET, nos familiarizaremos como los conceptos de ML y como ML.NET puede ayudarnos a construir modelos y servicios de ML, y a consumirlos en un proyecto de escritorio y Web.

Y finalmente conocerás unos recursos muy valiosos en caso de querer crear soluciones más avanzadas de ML en la plataforma .NET.



Requisitos previos y herramientas

Resumen del curso

01

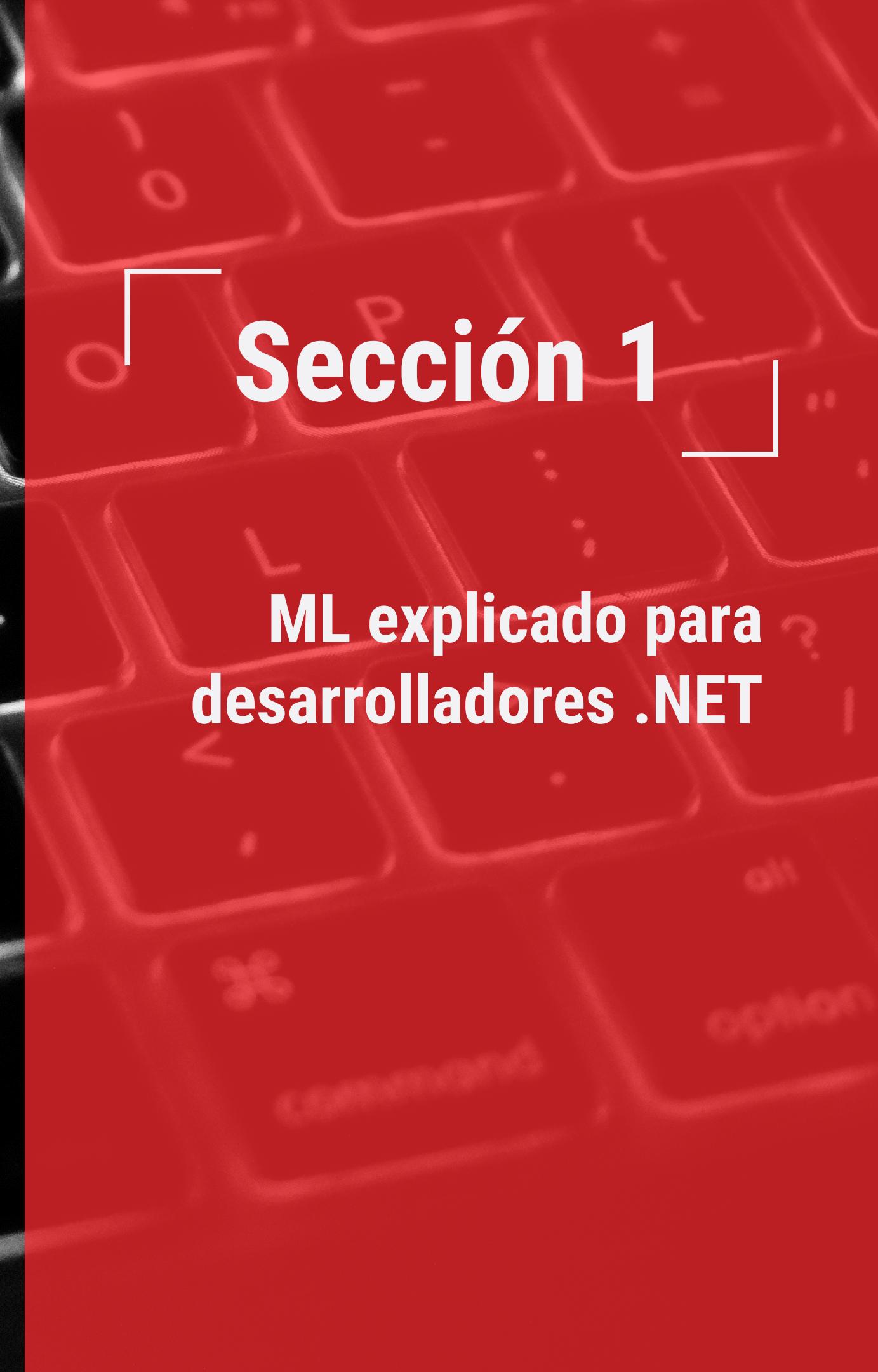
Conocimientos sobre
Codificación en C#

02

Entorno de desarrollo
Visual Studio 2019



¡Comencemos!



Introducción (1/2)

Sección 1

Jerga – Lenguaje especializado

Si eres nuevo en el mundo del ML, no te saltes esta sección. Habitualmente los desarrolladores estamos habituados a estudiar con un caso práctico. Pero en este mundo del ML es necesario que domines algunos conceptos antes de escribir una línea de código.

En el mundo del ML alguno de los términos que se utilizan son los mismos que el mundo del desarrollo, pero tienen un significado completamente diferente y nos ayudará en una conversación con científicos de datos:

- **Dataset:** conjunto de datos, datos de origen que proporcionas a un algoritmo de ML para que pueda aprender los patrones de los cuales poder hacer predicciones al respecto. La mayoría de los datos están estructurados (como los pedidos de compra de un cliente), pero también pueden estar desestructurados (como el texto de un libro). Para las aplicaciones a nivel de producción los datos suelen constar de millones e incluso miles de millones de datos.
- **Feature:** características, son uno o más atributos en los datos que sirven como entrada en un algoritmo de ML. Por ejemplo, si queremos aprender sobre el comportamiento de los clientes, podrían ser características del conjunto de datos la edad, el género, los productos, etc.



Introducción (2/2)

Sección 1

- **Label**: etiquetas, es el resultado de los algoritmos de ML basados en clasificación y regresión. Más adelante lo veremos con más claridad. De momento piensa en un atributo sobre el que queremos que se realice una predicción de un algoritmo de ML. Por ejemplo, si es probable que una persona se gaste 100,00€ en compras, ese valor de 100,00€ será una etiqueta para esa predicción.
- **Prediction**: predicción, una vez que el modelo está listo, contiene el algoritmo para producir la etiqueta a partir de las características de entrada proporcionadas. Este proceso de producir una salida utilizando el modelo, se le llama predicción.

Tambien habrás oido hablar con frecuencia sobre: probar, entrenar, evaluar y reentrenar los modelos. Esta terminología la veremos más adelante, justo donde describiremos un flujo de trabajo típico de ML. Pero primero repasemos rápidamente los típicos problemas que suele abordar el ML.



Problemas típicos de ML (1/3)

Sección 1

Existe un debate si en algún momento las máquinas serán tan inteligentes que muchos de los trabajos que conocemos hoy en día no volverán a ser realizados por humanos o que incluso serán capaces de escribir un libro. Sobre este aspecto no voy a profetizar, lo que sí puedo hacer es mostrar qué tipo de problemas ya puede resolver el ML.

- **Binary classification:** clasificación binaria, en esta clase de problemas, un modelo de aprendizaje automático toma alguna entrada y la clasifica en uno de los dos posibles valores. Por ejemplo, un fragmento de texto podría identificarse como un sentimiento negativo o positivo o la decisión de contratación de un candidato podría ser sí o no, etc.
- **Multiclass classification:** clasificación múltiple, es más versátil que la binaria. Este tipo de problemas un algoritmo de ML puede clasificarlos en dos o más categorías. Por ejemplo, podría determinar el riesgo de dar un préstamo a un solicitante en alto, medio o bajo. Se usa mucho para clasificar imágenes, una imagen dispone de muchas etiquetas como color, objetos, animales, etc.



Problemas típicos de ML (2/3)

Sección 1

- **Regression and forecasting:** regresión y pronóstico, en la regresión una etiqueta es valor numérico que el modelo predice a partir de un conjunto de características. Un ejemplo de uso podría ser evaluar el precio justo de una acción o pronosticar la probabilidad de que un vuelo se agoste en los próximos 3 días.
- **Clustering:** agrupación, este es un ejemplo de aprendizaje no supervisado. En esta clase de problemas, no nos volvemos demasiado específicos sobre lo que queremos saber. Se trata de encontrar patrones interesantes en los datos. Por ejemplo, podría investigar los datos de un clic en un anuncio en Facebook y agrupar a los clientes que pertenecen a un determinado grupo demográfico y que, por ejemplo, siempre compran productos similares.
- **Anomaly detection:** detección de anomalías, también es un aprendizaje no supervisado. EL modelo ML intenta encontrar patrones inusuales en los datos. Podría utilizarse para detectar fraudes con tarjetas de crédito, actividad sospechosa en una red informática o un dispositivo IoT estropeado que está enviando datos incorrectos.



Problemas típicos de ML (3/3)

Sección 1

- **Recomendations and rankings:** recomendaciones y clasificación, una función ampliamente utilizada en los comercios electrónicos es mostrar productos recomendados y relacionados. Esto generalmente se implementa con un modelo de recomendación ML que agrupa los datos en entrada y luego proporciona una clasificación de similitud entre cada par de datos. Otro ejemplo de este algoritmo es detectar el plagio clasificando un par de textos en función de que similaridad exista en su contenido.

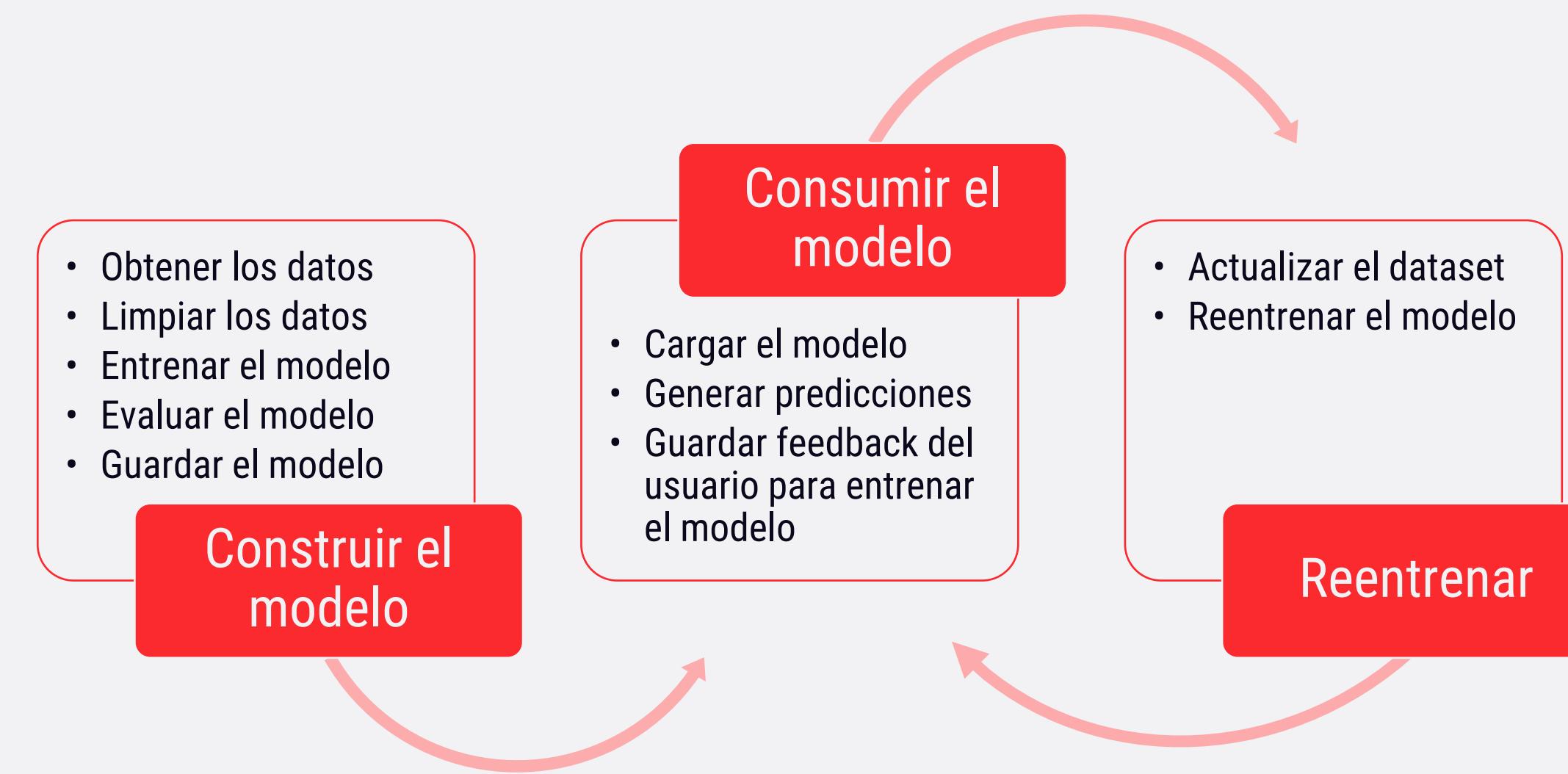


Flujo de trabajo (1/4)

Sección 1

Independientemente de la clase de problemas en los que se esté trabajando, el flujo de trabajo de ML es bastante estándar.

ML tiene dos, a veces, tres fases distintas. Estas fases son:



Flujo de trabajo (2/4)

Sección 1

Si los datos que se utilizan para el modelo permanecen estáticos a largo plazo, no es necesario incluir la fase de reentrenamiento en el flujo de ML.

El reentrenamiento ayuda a mantener el modelo actualizado si los datos de entrenamiento cambian frecuentemente.

Vamos a revisar con detalle cada actividad de cada una de las fases del flujo de ML.

Construir el modelo

La adquisición y limpieza de datos suele ser la parte que más esfuerzo requiere de todo el proceso. Los conjuntos de datos suelen ser enormes y solucionar los problemas de calidad de los datos no es siempre sencillo. Imagínate encontrar todas las variaciones de un mismo nombre escritas de manera diferente, o una dirección para la misma ubicación escrita de dos formas diferentes.



Flujo de trabajo (3/4)

Sección 1

Entrenar el modelo es el proceso de aplicar el algoritmo para producir el modelo como una salida. Por lo general, un modelo se entrena varias veces con algoritmos competidores y luego se elige el mejor modelo mediante el proceso de evaluación.

A efectos de evaluación post-entrenamiento, un subconjunto de datos (normalmente el 20%) se mantiene deliberadamente fuera del proceso de entrenamiento. Una vez construido el modelo, se prueba para ver si sus predicciones en el resto del conjunto de datos son precisas o no. Existen varias medidas matemáticas para determinar la precisión.

Una vez que se elige el mejor modelo, se guarda y se publica para que lo consuman los desarrolladores de la aplicación.

Consumir el modelo

El modelo consta de instrucciones algorítmicas que producen resultados, es decir predicciones, a partir de una entrada proporciona.



Flujo de trabajo (4/4)

Sección 1

El modelo producido en la anterior fase se usa en la aplicación. Algunas aplicaciones tienen mecanismos para almacenar los comentarios de los usuarios. Por ejemplo, si un sitio web usa ML para recomendar un producto puede almacenar un dato que indique si el cliente compró el producto recomendado o no. Estos datos de retroalimentación son extremadamente valiosos para volver a entrenar y mejorar el modelo.

Reentrenamiento del modelo

Utilizando los datos que se han obtenido en los comentarios de los usuarios u otras fuentes, el modelo se puede mejorar y volver a entrenar reiteradamente. Este proceso es muy similar a la fase inicial.

Reentrenar se puede realizar de forma continua (es decir, tan pronto como se reciben datos de comentarios de usuarios) o de forma periódica (por ejemplo, una vez al mes con un nuevo dataset).



Los datos, el nuevo petróleo

Sección 1

¿Facebook es gratis? – No, estás pagando con tus datos

Como desarrolladores de software, estamos familiarizados con el término garbage (basura).

Un modelo de ML es tan bueno como la calidad de sus datos. Podría decirse que es más importante encontrar el conjunto de datos correcto que elegir el algoritmo correcto para resolver el problema de ML.

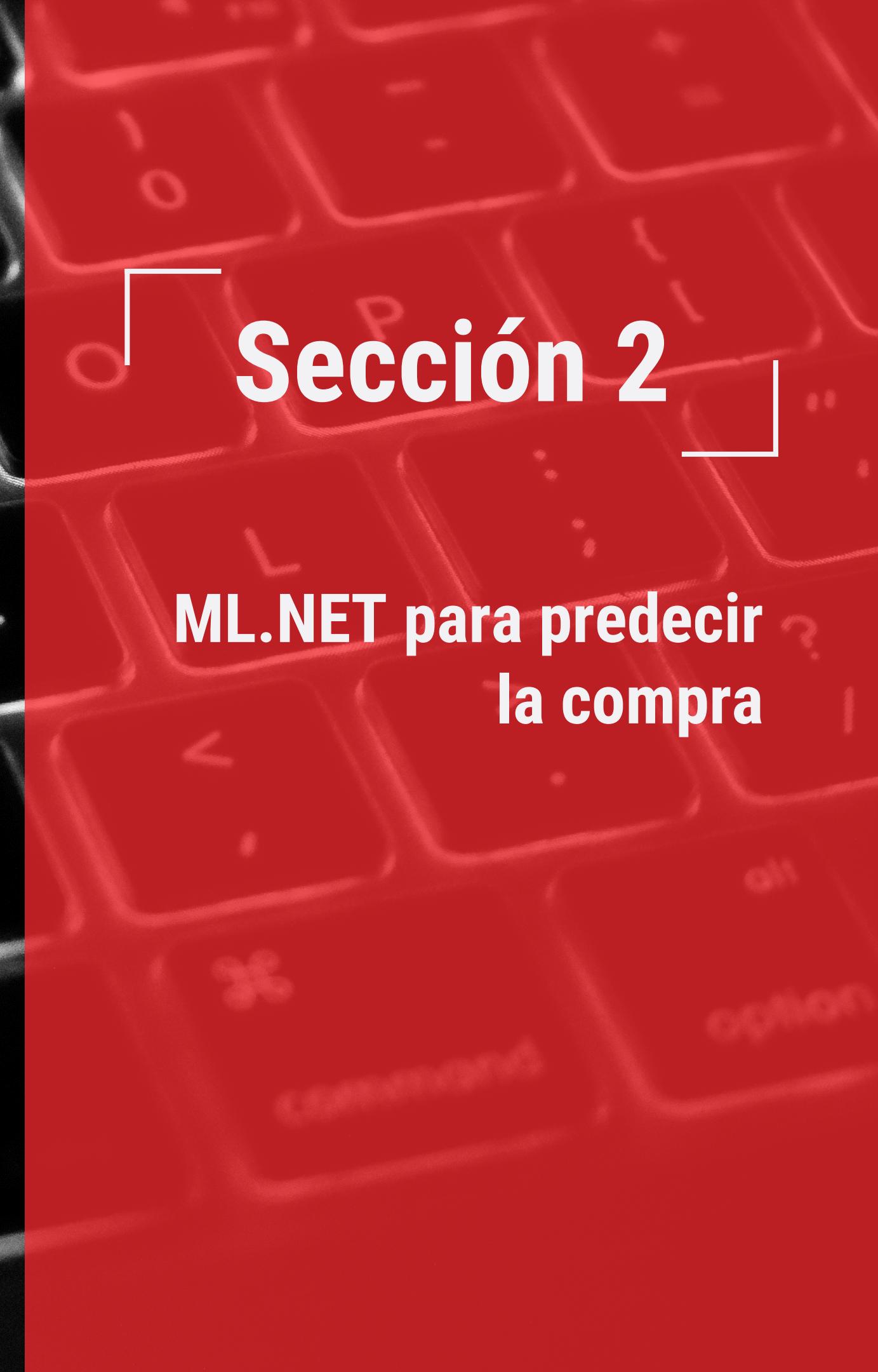
Actualmente los datos son el nuevo petróleo.

Entonces, ¿de dónde obtenemos los datos? Echemos un vistazo a algunos recursos útiles que podemos usar para recopilar datos de muestra que son útiles tanto para principiantes como para expertos.

Existen muchos sitios donde obtener datos y bajo distintos términos de licencias, pero personalmente estos son los que considero más útiles para experimentar:

- <https://www.kaggle.com/>: Kaggle una empresa de ciencia de datos perteneciente a de Google.
- <https://archive.ics.uci.edu/ml/index.php>, el repositorio de ML de University of California Irvine.





Introducción a ML.NET

Sección 2

Manos a la obra – Comenzando con lo básico

ML.NET es una librería en .NET Estándar que Microsoft nos ofrece a los desarrolladores para poder crear de forma fácil y sencilla soluciones basadas en el aprendizaje automático.

Proporciona un API para todas las funcionalidades habituales de ML: adquisición de datos, limpieza, entrenamiento de modelos, evaluación e implementación.

Todos los tipos de problemas principales están cubiertos con una gran cantidad de algoritmos de ML para cada uno de ellos.

La librería es extensible, incluso admite el consumo de modelos como Tensorflow, ONNX, etc. Esta es una gran ventaja, ya que permite a los devs de .NET trabajar en colaboración con científicos de datos que normalmente prefieren las tecnologías anteriormente mencionadas.

Vamos a usar unas funciones de ML.NET: ML.NET [Model Builder](#) y [AutoML](#).

AutoML es parte del API de ML.NET que automatiza la tarea de entrenar, evaluar y elegir el mejor algoritmo para un conjunto de datos concreto. Usa el CLI.

ML.NET Model Builder es la extensión exclusiva de Visual Studio que consume de forma gráfica AutoML.



Obteniendo el dataset

Sección 2

Vamos a descargar un fichero llamado `mall_customer.csv` de:

<https://www.kaggle.com/shwetabh123/mall-customers>

Este dataset es de dominio publico y se puede usar, modificar, etc. No incumpliremos ningún acuerdo de licencia.

Consta de un set 200 líneas y 5 columnas. El CustomerID no nos va a interesar. Usaremos Gender, Age y AnnualIncome para entrenar nuestro modelo para predecir SpendingScore.

El valor AnnualIncome esta expresado en miles de \$ por tanto \$15, serán \$15,000.



El problema

Sección 2

Estoy en la pagina 251

Tras hacer el programa en línea de comandos vamos a llevarlo a una function como un servicio, idea principal.

<https://docs.microsoft.com/es-es/dotnet/machine-learning/>



El problema

Sección 2

Voy a ir directamente a usar un problema de regresión, ya que estamos presidiendo un valor numérico (probabilidad de gasto) basado en las características de un conjunto de datos (edad, sexo e ingresos anuales).

¿Podrían pensar en otros problemas donde este conjunto de datos puede sernos útil?

Por ejemplo, podremos usar el resultado del modelo en una función que añada una nueva columna. De este modo podemos publicar una tabla completa indicando que para esa edad, sexo e ingresos podrás o no acceder a un préstamo. Es decir, podemos usar la puntuación, establecer un umbral y todos aquellos individuos que saquen más de 75% podrán acceder al préstamo... y si añadimos al modelo más variables como el gasto de tarjeta de crédito podremos ser más específicos con ese modelo y esa puntuación.

En resumen, este modelo nos permitirá crear un servicio de puntuación de umbral de préstamo, por ejemplo: un formulario donde metan las variables y nosotros le indicamos si puede o no acceder al préstamo.

Ya puestos podemos usar ese conjunto de datos para un problema de clustering. Es decir, usamos ML no supervisado para encontrar grupos de clientes interesantes, patrones y anomalías en función de las features disponibles.

Pero para aprender a usar ML.NET lo mejor es no complicarse y aprender sobre una base sencilla, menos, es más.

En nuestro ejercicio nos vamos a saltar la fase de reentrenamiento ya que nuestros datos son estáticos, recuerda las explicaciones anteriores.



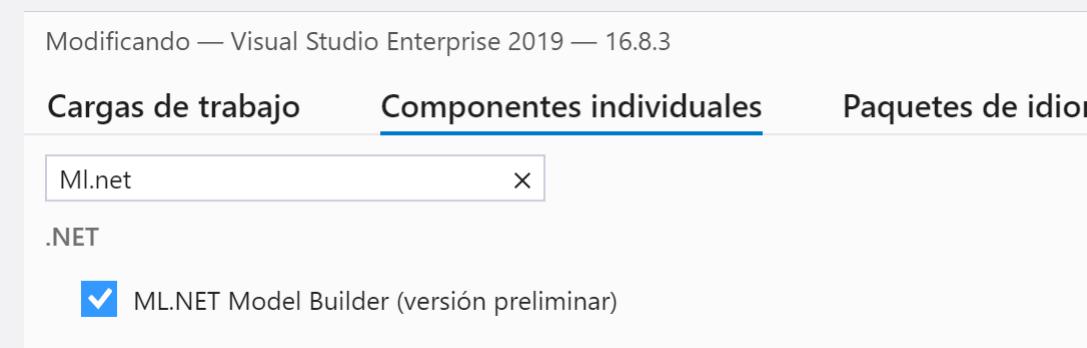
Comenzando con ML.NET Model Builder (1/2)

Sección 2

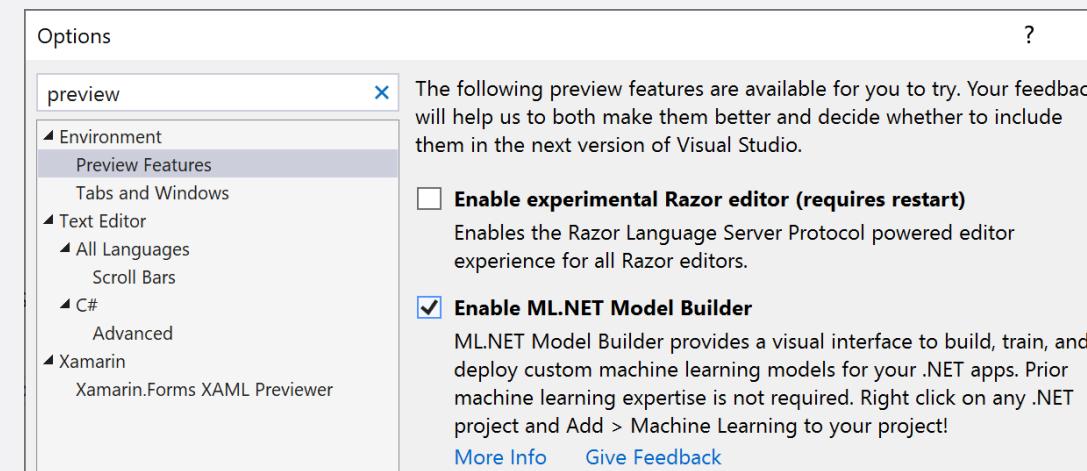
Visual Studio 2019 – VS Code ya dispone de una versión preview a la fecha de este curso

Vamos a lanzar la funcionalidad en VS2019.

1. Ejecuta en instalador de VS y selecciona buscar.
2. Escribe “ML.NET Model Builder” y marca la opción encontrada. A fecha de este curso esta en preview.



3. Cuando la instalación se complete, ejecuta VS2019 y ve a Tools | Options.
4. Busca “preview”, dentro de esta opción verás la posibilidad de marcar “Enable ML.NET Model Builder”, selecciónala. Esta opción nos autogenerará código (un poco de trabajo ahorrado nunca está de más).



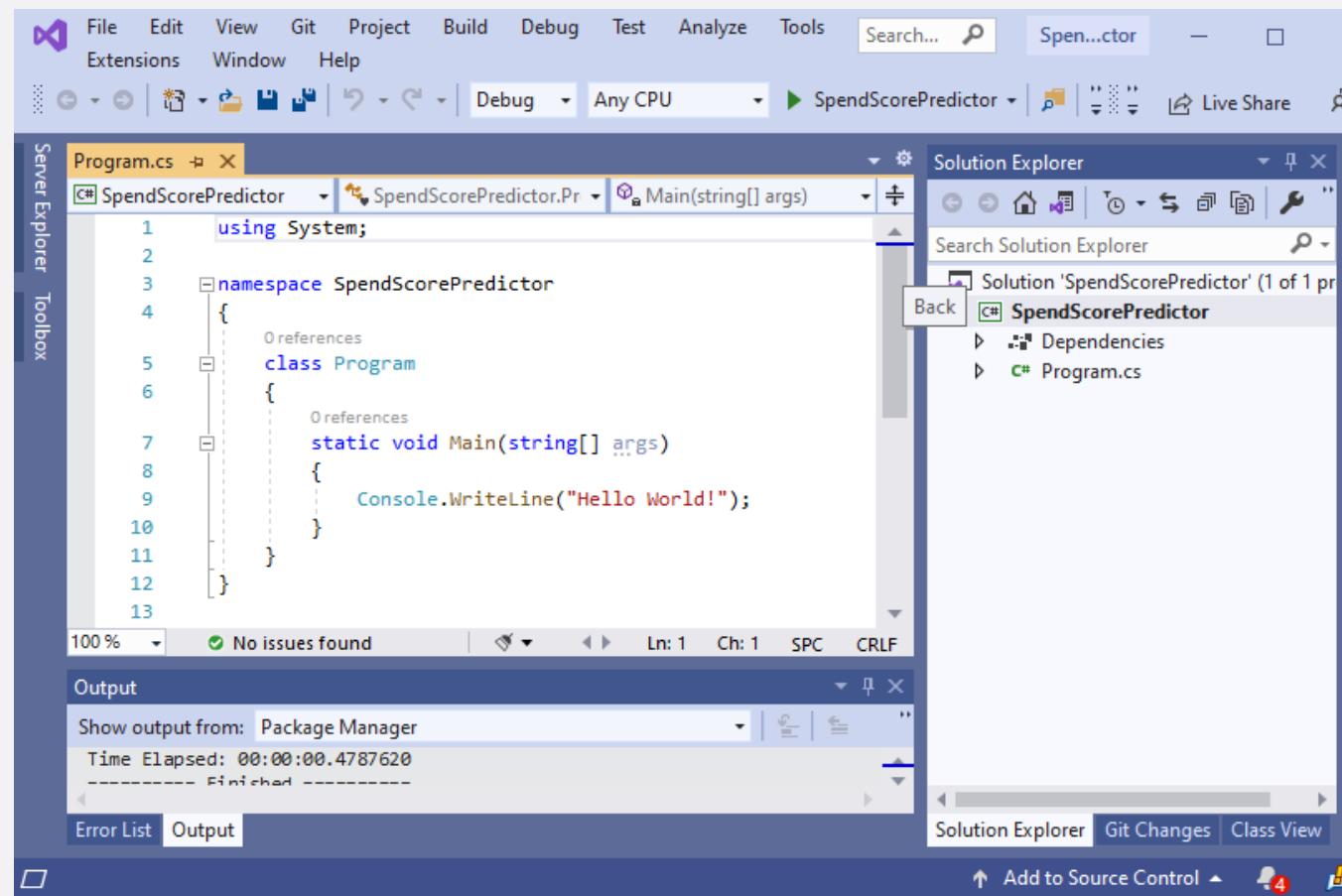
Crear la solución

Sección 2

Vamos a crear una solución, aunque asumo que muchos de ustedes están muy versados en esto, quiero que la presentación abarque diversos niveles.

1. Ejecuta VS y Create New Project.
2. Selecciona Console App (.NET Core).
3. Crear la solución con el siguiente nombre: SpendScorePredictor

La aplicación de consola que hemos creado es donde vamos a utilizar ML.NET Model Builder con el modelo que nos hemos descargado.



Entrenar el modelo (1/7)

Sección 2

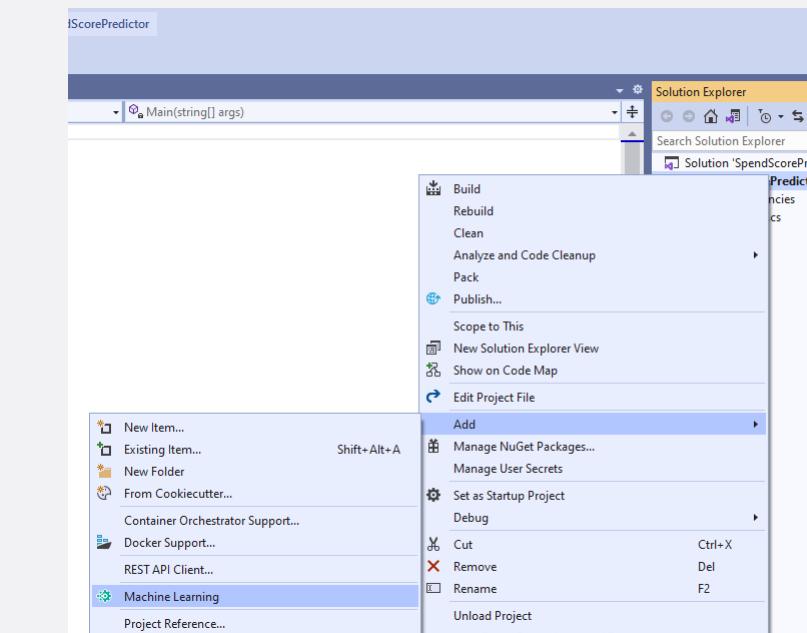
Los científicos de datos y estadísticos han desarrollado una serie de algoritmos a lo largo de los últimos para resolver problemas de regresión. Cada uno de estos algoritmos tiene varios parámetros que se pueden cambiar para ajustar la precisión de los resultados.

Con experiencia y un poco de conocimiento más profundo en de estos algoritmos, podemos decidir cual de ellos usar para nuestro dataset. Por tanto, los científicos de datos suelen experimentar con algoritmos candidatos variando los parámetros para elegir el mejor algoritmo para el problema planteado vs dataset.

Afortunadamente, para aquellos que no les gusta la estadística, Model Builder puede automatizar en gran medida este proceso.

Sigamos estos pasos:

Paso 1. Sobre la solución, clic derecho, Add | Machine Learning:



Entrenar el modelo (2/7)

Sección 2

Paso 2. Seleccionamos el escenario Value Prediction desde el Wizard.

1. Scenario Select a scenario

2. Environment

3. Data

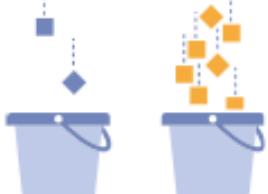
4. Train

5. Evaluate

6. Code

Train with your data

The following scenarios use Automated ML to train and pick the best model for your data. [Learn more about training with your own data in Model Builder.](#)

 Text classification
Classify text data into 2+ categories, e.g. predict if comments are positive or negative sentiments.
[Local ML](#)

 Value prediction
Predict a numeric value from your data (regression), e.g. predict the price of a house based on features like size, location, etc.
[Local ML](#)

 Image classification
Classify images, e.g. predict if an image shows a dog or a cat.
[Azure ML](#)

Limited scenarios

The following scenarios are not yet supported by Automated ML, so walkthroughs with an example dataset and [Learn more about examples in Model Builder.](#)



Entrenar el modelo (3/7)

Sección 2

Paso 3. En el siguiente paso elegimos en environment. Para el escenario en cuestión, Model Builder únicamente soporta entrenamiento en local. Para escenarios donde se necesite más poder de computación, como clasificación de imágenes, podremos usar la nube. Y presta atención al siguiente enlace, puede ayudarte a ahorrar tiempo:

<https://docs.microsoft.com/es-es/dotnet/machine-learning/how-to-guides/install-gpu-model-builder>

Paso 4. Sencillamente añade el fichero mall_customer.csv.

The screenshot shows the 'Add data' step in the Microsoft Model Builder wizard. The left sidebar lists steps 1 through 6: Scenario, Environment, Data (selected), Train, Evaluate, and Code. The main area is titled 'Add data' with the sub-instruction 'In order to build a model, you must add data and choose your column to predict.' Below this is a 'How do I get sample datasets and learn more?' link. The 'Input' section has a dropdown menu set to 'File' and a 'Select a file:' input field with a browse button. It also specifies supported formats: '.csv', '.tsv' or '.txt'. Below this are fields for 'Column to predict (Label)' and 'Input Columns (Features)', both with 'Select column' dropdowns. At the bottom, there's a 'Data Preview' section with a note to 'Select data to see the preview.', a 'Next step: train your model' note, and a 'Train' button.



Entrenar el modelo (4/7)

Sección 2

Paso 5. Elegimos las Features (columnas de entrada) y el Label (columna a predecir):

✓ 1. Scenario
✓ 2. Environment
✓ 3. Data
4. Train
5. Evaluate
6. Code

Add data

In order to build a model, you must add data and choose your column to predict.
[How do I get sample datasets and learn more?](#)

Input

Choose input data source from either SQL Server or File:

File

Select a file: C:\Projects\MyRepos\MLdotNET\

Supported file formats: .csv, .tsv or .txt.

Column to predict (Label):

Input Columns (Features):

Data Preview

10 of 201 rows and 4 of 5 columns. (1 Label, 3 Features).

Spending_Score_1_100 (Label)	Genre	Age	Annual_Income_k
39	Male	19	15
81	Male	21	15
6	Female	20	16
77	Female	23	16
40	Female	31	17
76	Female	22	17
6	Female	35	18
94	Female	23	18
3	Male	64	19
72	Female	30	19

Next step: train your model

Train



Entrenar el modelo (5/7)

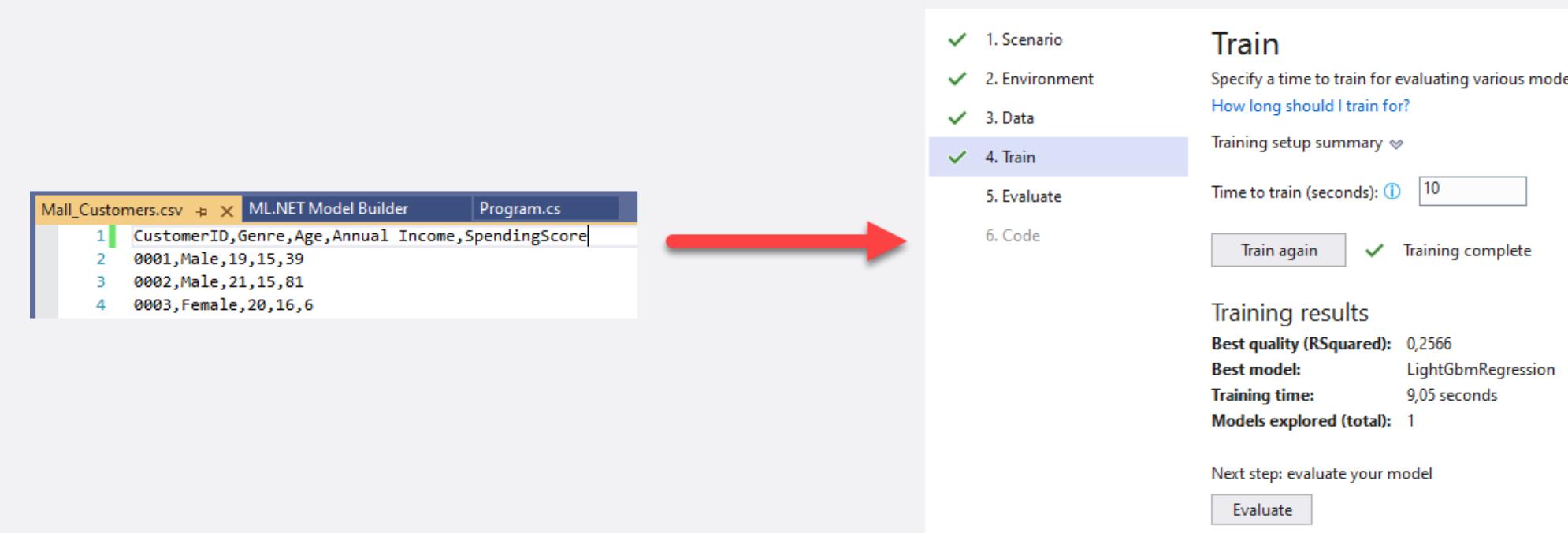
Sección 2

Paso 7. Seleccionamos la opción de Train y automáticamente el sistema ya hace la separación 80-20, una práctica común en el mundo ML y que ya os he explicado anteriormente (es reservar el 20% de los datos para verificar el algoritmo).

Paso 8. Para dataset pequeños como el nuestro el tiempo establecer 10 segundos para entrenar el algoritmo es suficiente, pero para dataset más grandes el tiempo detrás ampliarlo, aquí una pequeña explicación:

<https://docs.microsoft.com/es-es/dotnet/machine-learning/automate-training-with-model-builder#how-long-should-i-train-for>

Paso 9. Os dará error y esto se debe al nombre de las columnas, cambiarlo y volver al Paso 4. Intencionadamente lo he dejado así, para que observeis el log de errores y aprendáis a depurar esa salida.



Entrenar el modelo (6/7)

Sección 2

Paso 10. Revisar la salida .

The screenshot shows the ML.NET Model Builder interface with the following details:

- Train** tab selected.
- Time to train (seconds):** 10
- Training results:**
 - Best quality (RSquared):** 0,2566
 - Best model:** LightGbmRegression
 - Training time:** 9,05 seconds
 - Models explored (total):** 1
- Output** pane:
 - Shows the "Top 5 models explored" table:

Trainer	RSquared	Absolute-loss
LightGbmRegression	0,2566	14,03
FastForestRegression	0,0751	21,23
SdcaRegression	0,0022	23,77
FastTreeTweedieRegression	-0,0387	15,89
OlsRegression	-0,0448	16,52
 - Shows the "Top 2 models explored" table:

Trainer	RSquared	Absolute-loss
LightGbmRegression	0,2566	14,03
SdcaRegression	-0,0268	16,16
 - Shows the "Code Generated" section.

Annotations in red callouts:

- "Si tienes una GPU NVidia, instala los driver"
- "RSquare es "Coeficiente de determinación"."



Entrenar el modelo (7/7)

Sección 2

En este paso voy a destacar 2 cosas muy importantes:

- Si tienes una CPU o un equipo más potente, realizará pruebas con distintos **modelos**. En la captura central solo aparecen 2 debido a que el equipo no tiene GPU. Y en la sección marcada en negro (para que se vea bien) tienes 5 modelos distintos, esto se debe a que este equipo **si** tiene GPU.
- Por otro lado, he marcado el valor **RSquare** o “coeficiente de determinación” o R2. Este valor estadístico es el que va a determinar que modelo es mejor. Aquí ML.NET no se complica nada, solamente escoge el modelo que mejor resultado tiene y nos los propone. En este caso no dice que el mismo tanto para una maquina como para otra (dependiendo de tu equipo, los valores podrán ser diferentes). Nos da la LightGbmRegresion, que para aquellos que sean más inquietos aquí tienen más información:
 - https://en.wikipedia.org/wiki/Gradient_boosting
 - <https://github.com/Microsoft/LightGBM>



Entrenar el modelo (7/7)

Sección 2

En este paso voy a destacar 2 cosas muy importantes:

- Si tienes una CPU o un equipo más potente, realizará pruebas con distintos **modelos**. En la captura central solo aparecen 2 debido a que el equipo no tiene GPU. Y en la sección marcada en negro (para que se vea bien) tienes 5 modelos distintos, esto se debe a que este equipo **si** tiene GPU.
- Por otro lado, he marcado el valor **RSquare** o “coeficiente de determinación” o R2. Este valor estadístico es el que va a determinar que modelo es mejor. Aquí ML.NET no se complica nada, solamente escoge el modelo que mejor resultado tiene y nos los propone. En este caso no dice que el mismo tanto para una maquina como para otra (dependiendo de tu equipo, los valores podrán ser diferentes). Nos da la LightGbmRegresion, que para aquellos que sean más inquietos aquí tienen más información:
 - https://en.wikipedia.org/wiki/Gradient_boosting
 - <https://github.com/Microsoft/LightGBM>



Evaluar el modelo

Sección 2

En el mismo wizard tenemos la opción de evaluar nuestro modelo. Utiliza valores aleatorios para ver como se comporta. ¿Podrías jugar con las variables y ver cual de ellas es la que mayor peso tiene en el algoritmo?

The image displays two side-by-side screenshots of a "Try your model" interface from a machine learning wizard. Both screenshots show a "Sample data" section with pre-filled fields for "Genre" (Male), "Age" (19 or 30), and "AnnualIncome" (15). The "Results" section on the right shows the "SpendingScore". In the first screenshot, the "Age" field contains "19" and the "SpendingScore" is 67,21. In the second screenshot, the "Age" field has been changed to "30" and the "SpendingScore" has increased to 75,27. The "Age" field in the second screenshot is highlighted with a red box.

Si los resultados de un escenario real se comportan de forma extraña, puedes volver a la sección anterior y elegir y probar con otros algoritmos, este wizard te lo pone fácil.

Y si has jugado habrá observado que el genero y la edad a un mismo salario nos da que los hombres de entre 20/30 años son los que más puntuación sacan, es decir, mayor probabilidad de consumir.



Generar código automáticamente

Sección 2

Solo nos queda una opción en el wizard generar el código.

Se han incluido 2 proyectos nuevos:

- **SpendScorePredictorML.Model**: contiene las clases de entrada y salidas del modelo generado. Un fichero llamado `MLModel.zip` con el modelo. Y una clase llamada `ConsumeModel` para generar un motor de predicción usado por el archivo del modelo que podremos usar para crear predicciones de las entradas.
- **SpendScorePredictorML.ConsoleApp**: una aplicación de consola con la clase `ModelBuilder.cs`, que contiene código para entrenar y generar el modelo a partir de `Mall_Customers.csv`. No necesitaremos esta clase ya que ML.NET Model Builder ya nos ha generado un fichero ZIP para nuestro proyecto. Este proyecto referencia a `SpendScorePredictorML.Model` y desde `Program.cs` podremos consumir nuestro modelo.

Tomate tu tiempo para revisar ambos proyectos.



Prueba la aplicación (1/2)

Sección 2

Si has revisado ambos proyectos, habrá visto que deberás establecer como proyecto de inicio a: [SpendScorePredictorML.ConsoleApp](#) y con cambiar los valores del objeto sampleData podrás realizar tus pruebas.

```
1 reference
class Program
{
    0 references
    static void Main(string[] args)
    {
        // Create single instance of sample data from first line of dataset for model input
        ModelInput sampleData = new ModelInput()
        {
            Genre = @"Male",
            Age = 19F,
            AnnualIncome = 15F,
        };

        // Make a single prediction on the sample data and print results
        var predictionResult = ConsumeModel.Predict(sampleData);

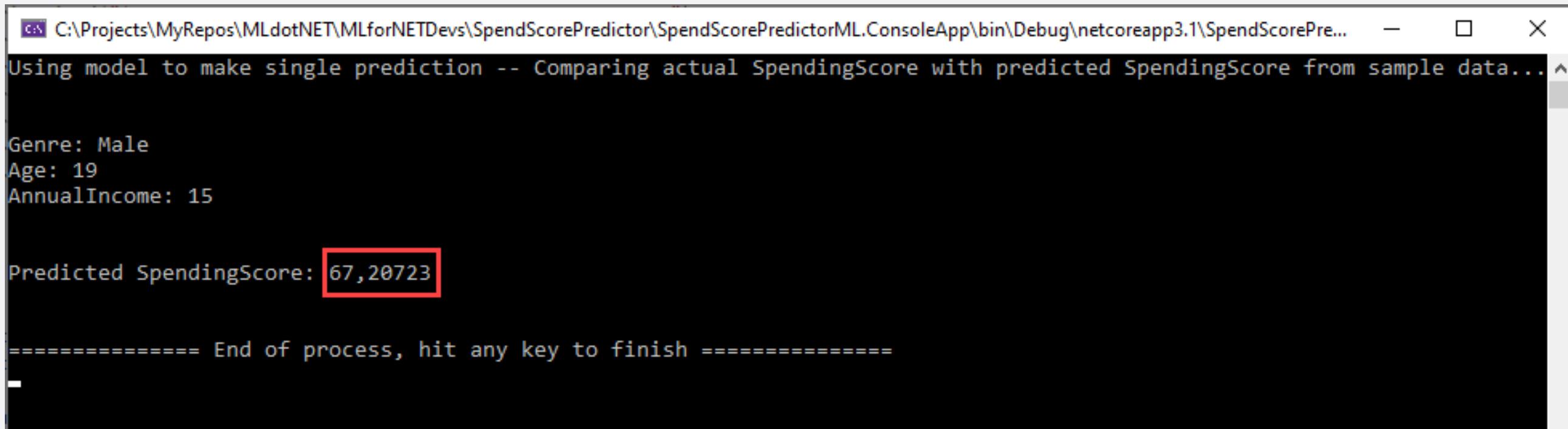
        Console.WriteLine("Using model to make single prediction -- Comparing actual SpendingScore with predicted SpendingScore from sample data...\n\n");
        Console.WriteLine($"Genre: {sampleData.Genre}");
        Console.WriteLine($"Age: {sampleData.Age}");
        Console.WriteLine($"AnnualIncome: {sampleData.AnnualIncome}");
        Console.WriteLine($"{Environment.NewLine}Predicted SpendingScore: {predictionResult.Score}{Environment.NewLine}");
        Console.WriteLine("===== End of process, hit any key to finish =====");
        Console.ReadKey();
    }
}
```

Establece este proyecto como inicio y juega con estos datos.



Prueba la aplicación (2/2)

Sección 2

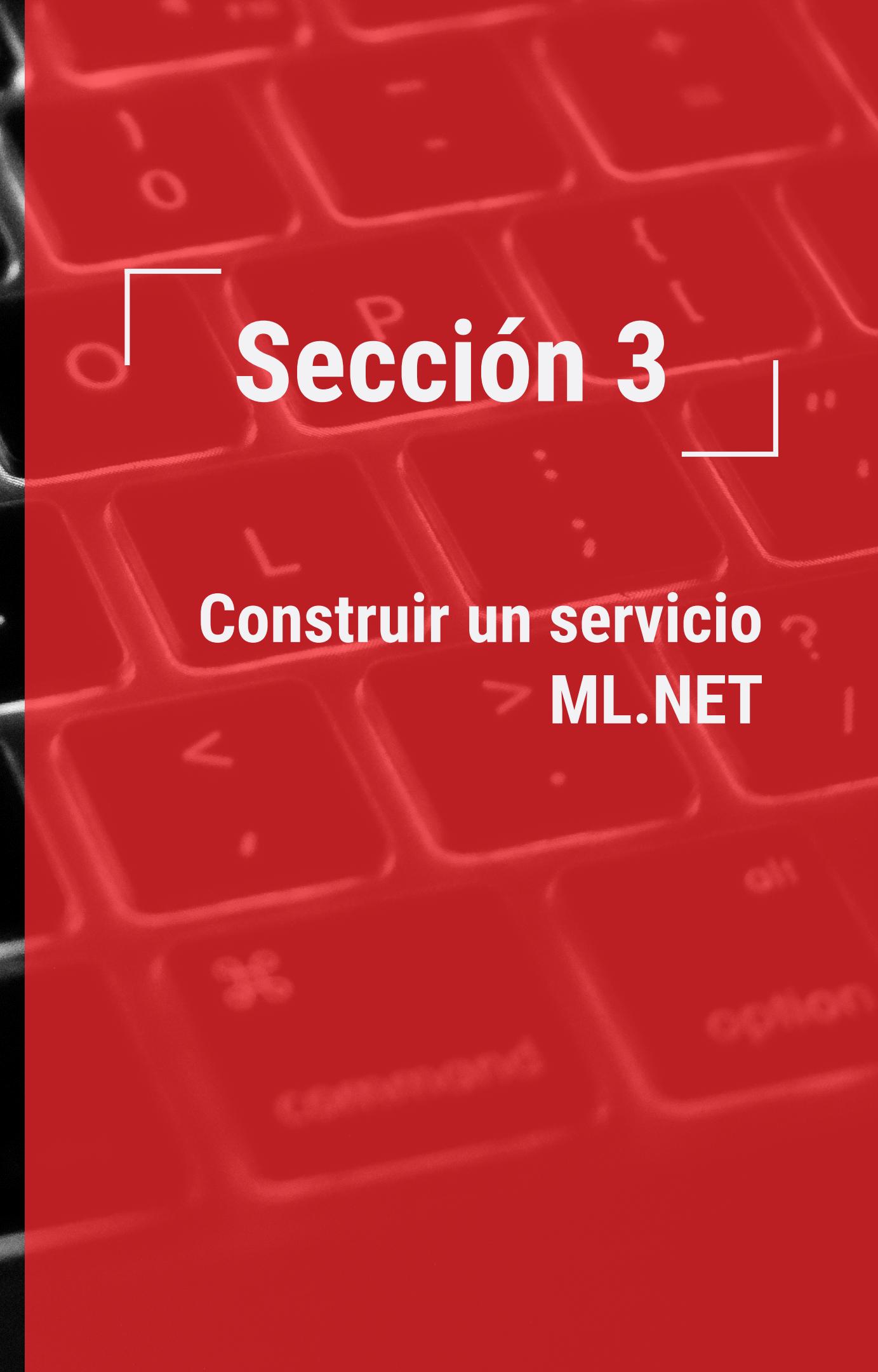


```
C:\Projects\MyRepos\MLdotNET\MLforNETDevs\SpendScorePredictor\SpendScorePredictorML.ConsoleApp\bin\Debug\netcoreapp3.1\SpendScorePre...
Using model to make single prediction -- Comparing actual SpendingScore with predicted SpendingScore from sample data...
Genre: Male
Age: 19
AnnualIncome: 15

Predicted SpendingScore: 67,20723

===== End of process, hit any key to finish =====
```





Azure Functions que usan modelos ML.NET (1/5)

Sección 2

Manos a la obra – Un proyecto más realista

Cuando ya tenemos nuestro modelo de ML completado, lo habitual es que deba ser consumido desde una aplicación .NET, ya sea WPF, web, API o desde microservicios.

En esta ocasión vamos a crear un modelo de análisis de sentimiento y luego lo integraremos en una Azure Function.

Los datos de este ejemplo vienen de:

<https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences>

Y cuyos datos son 3 ficheros con reseñas positivas y negativas, 500 de cada una, sacadas de Amazon, IMDb y Yelp.

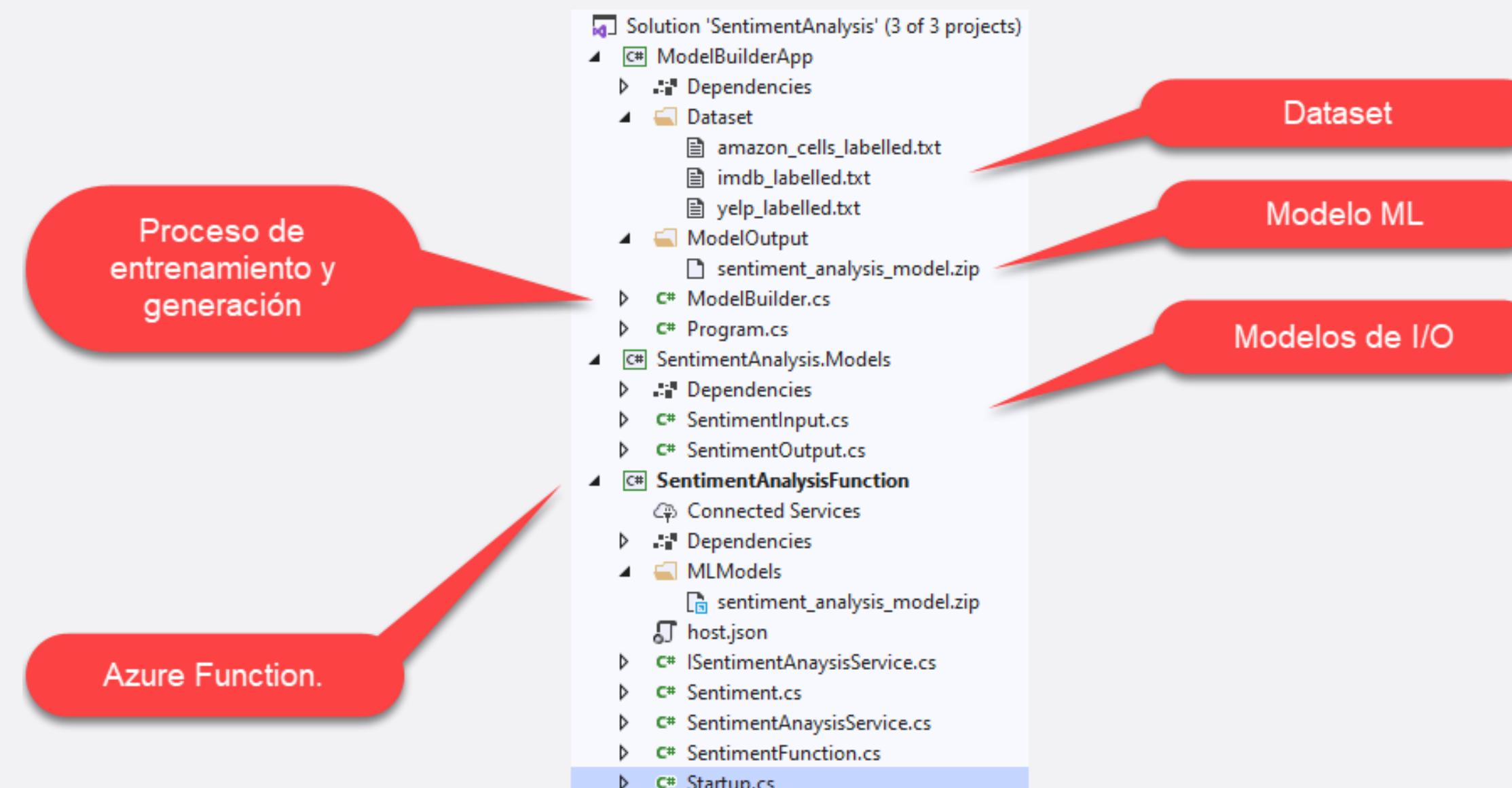
El valor del sentimiento es 0 para negativo y 1 para positivo. La muestra, conscientemente, esta seleccionada para que no existan sentimientos neutros (en la sección también os mostraré una técnica que ayuda a determinar un sentimiento neutro dentro de un texto). La muestra esta en inglés.



Azure Functions que usan modelos ML.NET (2/5)

Sección 2

La estructura del proyecto es la siguiente:



Azure Functions que usan modelos ML.NET (3/5)

Sección 2

Un detalle importante en este ejemplo es que, a partir de tres sets de datos, vamos a generar un dataset:

```
1 reference
public static void CreateModel()
{
    // create loader
    TextLoader textLoader = mlContext.Data.CreateTextLoader<SentimentInput>(separatorChar: '\t', hasHeader: false);

    // Load Data
    IDataView trainingDataView = textLoader.Load(TRAINING_DATA_FILES);
```

3 ficheros

```
// Build training pipeline
IEstimator<ITransformer> trainingPipeline = BuildTrainingPipeline(mlContext);
```

Entrenamiento

```
// Train Model
ITransformer mlModel = TrainModel(mlContext, trainingDataView, trainingPipeline);
```

Entrenamos

```
// Evaluate quality of Model
Evaluate(mlContext, trainingDataView, trainingPipeline);
```

```
// Save model
SaveModel(mlContext, mlModel, MODEL_FILEPATH, trainingDataView.Schema);
}
```

Generamos el Modelo



Azure Functions que usan modelos ML.NET (4/5)

Sección 2

Para probar nuestro servicio, lanzamos la function y hacemos uso de Postman (por ejemplo):

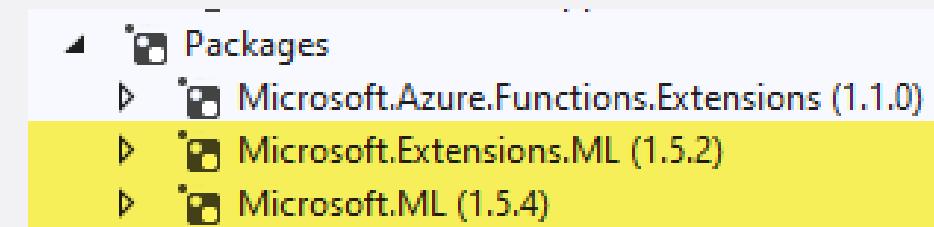
The screenshot shows the Postman application interface. On the left, the sidebar displays 'Functions' with a single entry: 'GetSentiment: [GET] http://localhost:7071/api/GetSentiment'. Below this, there's a note: 'For detailed output, run func with --verbose flag.' and a timestamp: '[2021-02-15T14:36:44.247Z] Host lock lease acquired by instance ID '000000000000000000000000AB85A630''. The main workspace shows a 'ML.NET - Demo / GetSentiment' collection. A 'GET' request is selected with the URL 'http://localhost:7071/api/GetSentiment'. The 'Body' tab is active, containing the text '1 It's good'. The 'Test Results' section at the bottom shows a single test step with status '1 1'. The overall status bar indicates 'Status: 200 OK'.



Azure Functions que usan modelos ML.NET (5/5)

Sección 2

Si observáis al código de la function, solamente necesitaremos el zip del modelo ML.NET y referenciar, para que todo funcione, a:



Y tal como os adelantaba al principio, solo quedaría añadir un pequeño toque al servicio para introducir la neutralidad:

The screenshot shows the Azure Functions developer tools interface. On the left, there's a preview pane for a POST request to `http://localhost:7071/api/GetSentiment`. The 'Body' tab is selected, showing the input text: `1 It's good bad`. On the right, the function code is displayed in a code editor:

```
var prediction = predictionEnginePool.Predict(modelName: "SentimentAnalysisModel", example: input);

var confidence = prediction.Prediction == "0" ? prediction.Score[0] : prediction.Score[1];
if (confidence < 0.95)
    return Sentiment.Neutral;
```

A red arrow points from the input text in the preview pane to the line `if (confidence < 0.95)`. Another red arrow points from the same line to the corresponding line in the code editor: `if (conf < 0.95)`.





Usando Tensorflow en ML.NET (1/7)

Sección 4

Tensorflow – <https://www.tensorflow.org/> la plataforma de Google

Vamos a usar un modelo previamente entrenado de Tensorflow, en concreto el modelo Inception, integrarlo y aplicarlo en un proyecto.

La idea es coger un modelo previamente enterando y aplicaremos el aprendizaje por transferencia (la idea es superar el aprendizaje aislado y utilizar el conocimiento adquirido por una tarea para resolver problemas relacionados, https://www.tensorflow.org/tutorials/images/transfer_learning), añadiendo algunas imágenes de comida y masas de agua. Una vez realizado el aprendizaje por transferencia, el usuario podrá usar sus propias imágenes.

La atención es que adquiera una comprensión de lo necesario para integrar un modelo de Tensorflow en una aplicación ML.NET.

¿Qué es el modelo Inception?

<https://cloud.google.com/tpu/docs/inception-v3-advanced?hl=es>



Usando Tensorflow en ML.NET (2/7)

Sección 4

Antes de comenzar el proyecto indicaros que las imágenes las he bajado de:

<https://cocodataset.org/#download>

The screenshot shows the COCO dataset website. At the top, there's a navigation bar with the COCO logo, a search bar, and links for Home, People, Dataset, Tasks, and Evaluate. Below the navigation, there's a "News" section with a list of recent announcements. Under "What is COCO?", there are icons for various objects and a brief description of the dataset's features. A sidebar on the left contains a red hexagon icon.

News

- We are pleased to announce the COCO 2020 Detection, Keypoint, Panoptic, and DensePose Challenges.
- The new rules and awards for this year challenges encourage innovative methods.
- Results to be announced at the Joint COCO and LVIS Recognition ECCV workshop.

What is COCO?

COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- ✓ Object segmentation
- ✓ Recognition in context
- ✓ Superpixel stuff segmentation
- ✓ 330K images (>200K labeled)
- ✓ 1.5 million object instances
- ✓ 80 object categories
- ✓ 91 stuff categories
- ✓ 5 captions per image
- ✓ 250,000 people with keypoints

Collaborators

Tsung-Yi Lin Google Brain
Genevieve Patterson MSR, Trash TV
Matteo R. Ronchi Caltech
Yin Cui Google
Michael Maire TTI-Chicago
Serge Belongie Cornell Tech
Lubomir Bourdev WaveOne, Inc.
Ross Girshick FAIR
James Hays Georgia Tech
Pietro Perona Caltech
Deva Ramanan CMU
Larry Zitnick FAIR
Piotr Dollár FAIR

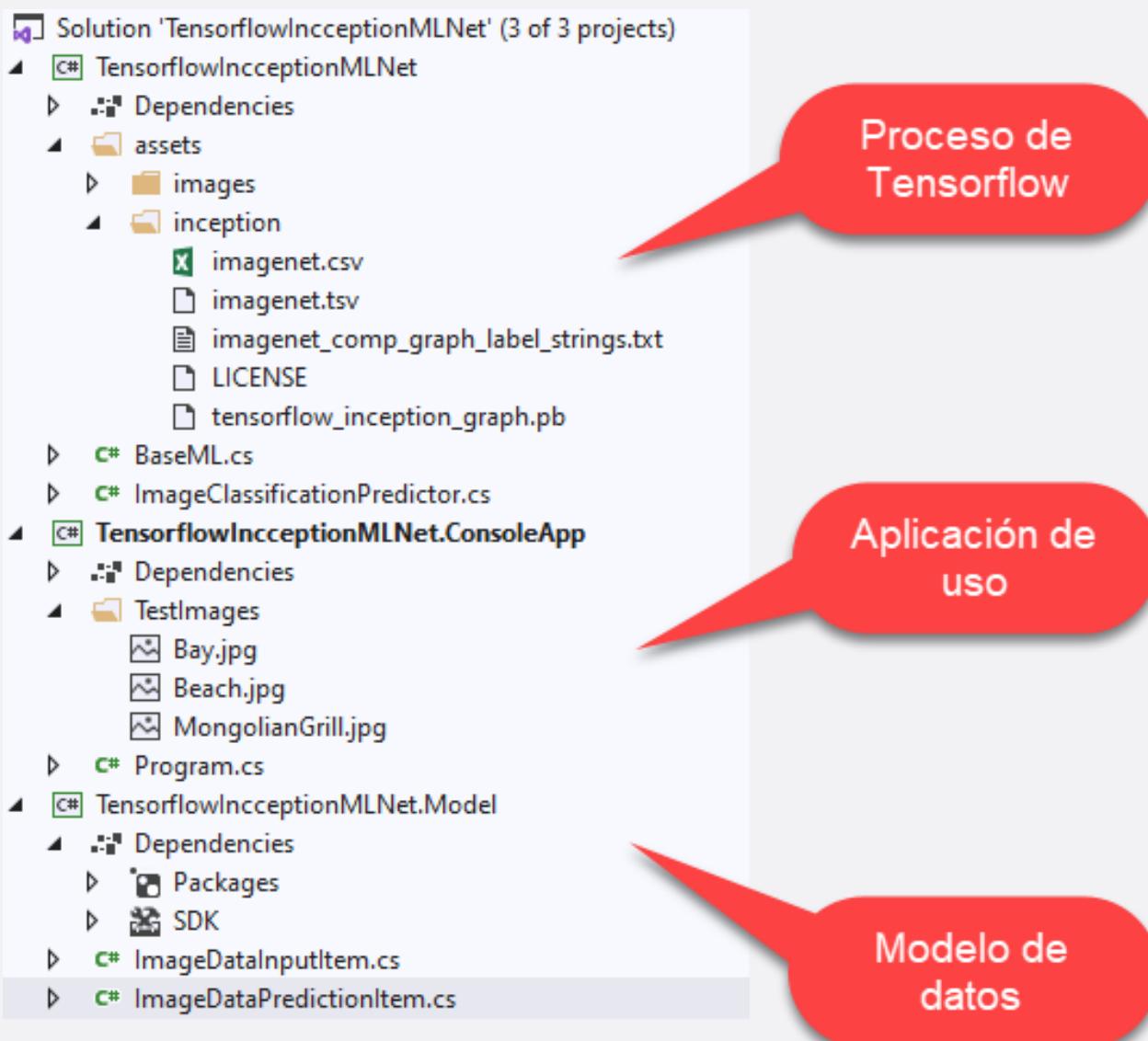
Sponsors

CVDF
 Microsoft
 facebook
 Mighty Ai

Usando Tensorflow en ML.NET (3/7)

Sección 4

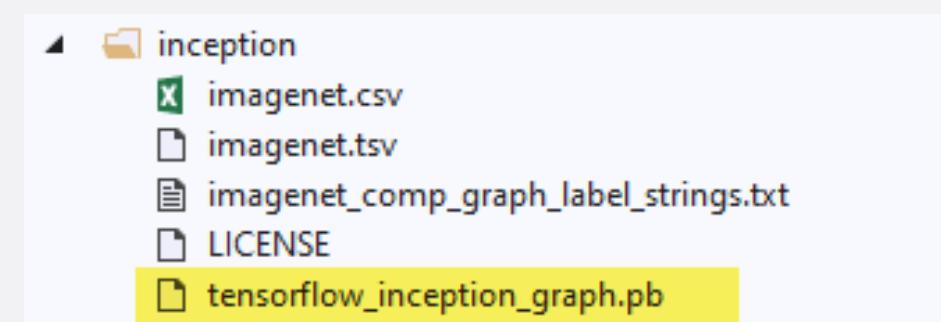
Nuestro proyecto volverá a ser un ejemplo en CLI, ya no tiene sentido hacerlo de otra forma, en el capítulo 3 has aprendido a como montar un servicio, por tanto, vamos a ser concretos con nuestros ejemplos:



Usando Tensorflow en ML.NET (4/7)

Sección 4

La parte más importante es tener en el proyecto el modelo de Tensorflow:



Y una vez dentro, lo más complejo es usarlo para generar un modelo compatible:

```
private static readonly string ML_NET_MODEL = Path.Combine(_assetsPath, "TensorflowModel.mdl");
```

Para ello vamos a usar una transformación para generar el fichero mdl.



Usando Tensorflow en ML.NET (5/7)

Sección 4

```
if (File.Exists(ML_NET_MODEL))
{
    _model = MlContext.Model.Load(ML_NET_MODEL, out DataViewSchema modelSchema);

    return (true, string.Empty);
}

IEstimator<ITransformer> pipeline = MlContext.Transforms.LoadImages(outputColumnName: INPUT, imageFolder: _imagesFolder, inputColumnName: nameof(ImageDataInputItem.ImagePath))
    .Append(MlContext.Transforms.ResizeImages(outputColumnName: INPUT, imageWidth: InceptionSettings.ImageWidth, imageHeight: InceptionSettings.ImageHeight, inputColumnName: INPUT))
    .Append(MlContext.Transforms.ExtractPixels(outputColumnName: INPUT, interleavePixelColors: InceptionSettings.ChannelsLast, offsetImage: InceptionSettings.Mean))
    .Append(MlContext.Model.LoadTensorFlowModel(_inceptionTensorFlowModel)
        .ScoreTensorFlowModel(outputColumnNames: new[] { TF_SOFTMAX }, inputColumnNames: new[] { INPUT }, addBatchDimensionInput: true))
    .Append(MlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "LabelKey", inputColumnName: nameof(ImageDataPredictionItem.Label)))
    .Append(MlContext.MulticlassClassification.Trainers.LbfgsMaximumEntropy(labelColumnName: "LabelKey", featureColumnName: TF_SOFTMAX))
    .Append(MlContext.Transforms.Conversion.MapKeyToValue(nameof(ImageDataPredictionItem.PredictedLabelValue), "PredictedLabel"))
    .AppendCacheCheckpoint(MlContext);

IDataView trainingData = MlContext.Data.LoadFromTextFile<ImageDataInputItem>(path: _trainTagsTsv, hasHeader: false);

_model = pipeline.Fit(trainingData);

MlContext.Model.Save(_model, trainingData.Schema, ML_NET_MODEL);

return (true, string.Empty);
```



Usando Tensorflow en ML.NET (6/7)

Sección 4

Una vez generado, nos podemos guardar el fichero mdl y hacer una nueva aplicación que utilice este modelo.

En nuestro ejemplo, somo menos ambiciosos y hacemos algo sencillo para aprender de forma ordenada:

```
namespace TensorflowInceptionMLNet.ConsoleApp
{
    // References
    class Program
    {
        // References
        static void Main(string[] args)
        {
            string _assetsPath = Path.Combine(Environment.CurrentDirectory, "TestImages");
            var imagePath = Path.Combine(_assetsPath, "Bay.jpg");

            var prediction = new ImageClassificationPredictor();
            prediction.Initialize();

            var result = prediction.Predict(imagePath);

            Console.WriteLine($"Image ({imagePath}) is a picture of {result.PredictedLabelValue} with a confidence of {result.Score[1]:P2}");
        }
    }
}
```



Usando Tensorflow en ML.NET (7/7)

Sección 4

Y para la imagen el resultado que obtenemos con un intervalo de confianza (es curioso, desde que termine estadística he recurrido a los IC en diversas ocasiones y en un proyecto actual también he vuelto a ello: *la verdad estadística*).



```
Microsoft Visual Studio Debug Console
2021-02-16 14:30:16.382722: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
Image (C:\Projects\MyRepos\MLdotNET\MLforNETDevs\TensorflowInceptionMLNet\TensorflowInceptionMLNet.ConsoleApp\bin\Debug\netcoreapp3.1\TestImages\Bay.jpg) is a picture of [water with a confidence of 95,90 %]
```





ONNX en ML.NET (1/4)

Sección 5

¿Qué es ONNX? – Open Neural Network Exchange

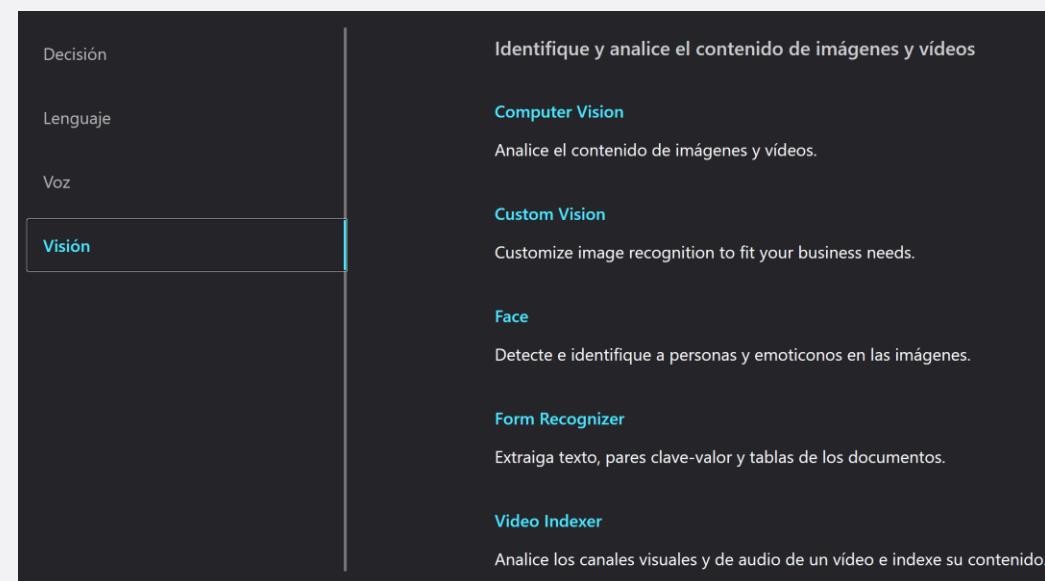
Nuevamente vamos a realizar una aplicación de consola que detecta objetos dentro de una imagen mediante un modelo de aprendizaje profundo (**Deep Learning**) de ONNX previamente entrenado.

<https://onnx.ai/>

¿Qué es la detección de objetos? Se trata de un problema de visión informática. Y aunque este relacionado con la clasificación de imágenes, la detección de objeto realiza la clasificación de imágenes a una escala más granular. La detección ubica y categoriza entidades dentro de las imágenes.

A continuación os recomiendo revisar los servicios cognitivos de Azure, darle un vistazo:

<https://azure.microsoft.com/es-es/services/cognitive-services/>



ONNX en ML.NET (2/4)

Sección 5

En resumen, la detección de objetos y la clasificación se puede explicar con esta imagen:

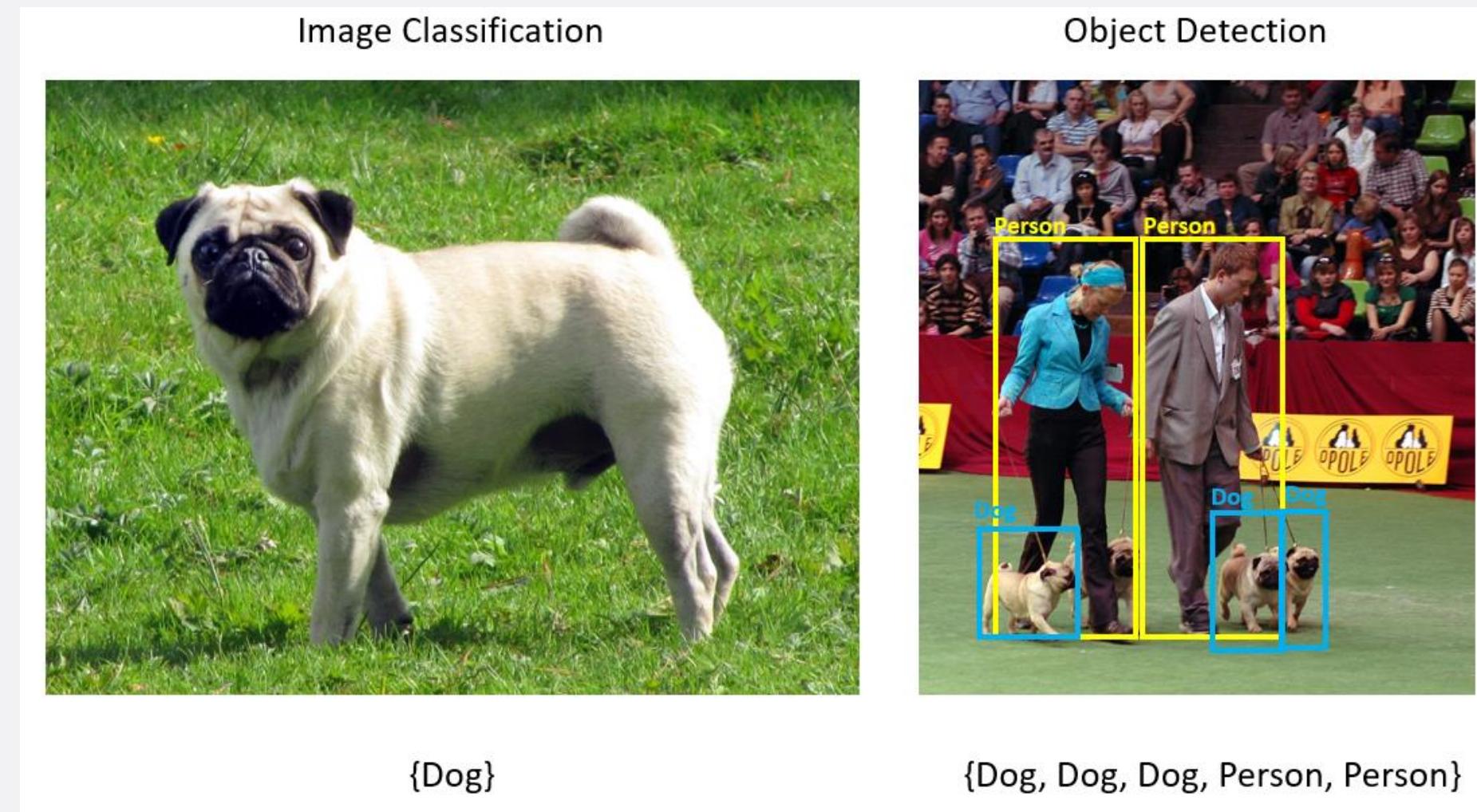


Imagen: <https://docs.microsoft.com/es-es/dotnet/machine-learning/tutorials/object-detection-onnx>



ONNX en ML.NET (3/4)

Sección 5

Y como en:

<https://docs.microsoft.com/es-es/dotnet/machine-learning/tutorials/object-detection-onnx>

Esta excelentemente explicado el funcionamiento, no voy a repetirlo aquí.

Los que si voy a cuento es que ONNX es un formato open source para modelos de IA que nos ofrece interoperatividad entre marcos, es decir, que puedes entrenar en los distintos marcos de aprendizaje más populares, convertirlo a ONNX y consumirlo en ML.NET, nuestro objetivo.



ONNX en ML.NET (4/4)

Sección 5

Y nuestro ejemplo – A continuación, vamos a realizar la aplicación de consola
Los primero es bajar el modelo desde:

<https://gallery.azure.ai/Model/Tiny-YOLOv2>

Pero en este caso os voy a remitir a otro sitio de interés:

https://github.com/dotnet/machinelearning-samples/tree/master/samples/csharp/getting-started/DeepLearning_ObjectDetection_Onnx

Debes estudiar el code walkthrough y seguir los pasos que nos indican.





Bonus

¿Podemos predecir el
precio del aceite de
oliva en España?

Un ejercicio para que practiques

Bonus

La respuesta es por supuesto, pero tal y como decía en esta presentación: tan buena es tu predicción como tan buenos son tus datos que generan el modelo. **Los datos es el quid de toda cuestión en ML.** Los datos que tan gratuitamente damos en las redes sociales ;)

Por tanto, lo complicado de este ejercicio es preparar el dataset. Os digo como tendríamos que prepararlo:

- Ir a la UE y obtener el fichero de evolución de precios:
<https://data.europa.eu/euodp/es/data/dataset/olive-oil-market-prices>
- Ir al INE, a la sección de datos abiertos y obtener la evolución del IPC, la renta per capita.
- Ir al instituto Meteorológico y obtener datos de higrometría, temperatura y precipitaciones.
- Con estas variables ya podríamos comenzar a montar nuestro csv, en resumen, nuestro dataset. Se pueden añadir muchas más variables asociadas a la producción, precio de la mano de obra, calidad de la tierra, etc. Etc. A partir de cierto punto añadir una variable más, no supondría mejoría en el modelo (lo único que incurría sería en lentitud del cálculo).

Y lo más importante, ¿qué algoritmo debemos usar para predecir los precios? Muy sencillo un modelo de regresión nos sirve para predecir precios.

Conclusión, ¿qué es lo más complejo? Preparar el dataset.



Otras cosas que quizá deberías conocer (1/3)

Bonus

Azure Machine Learning Studio – Ecosistema

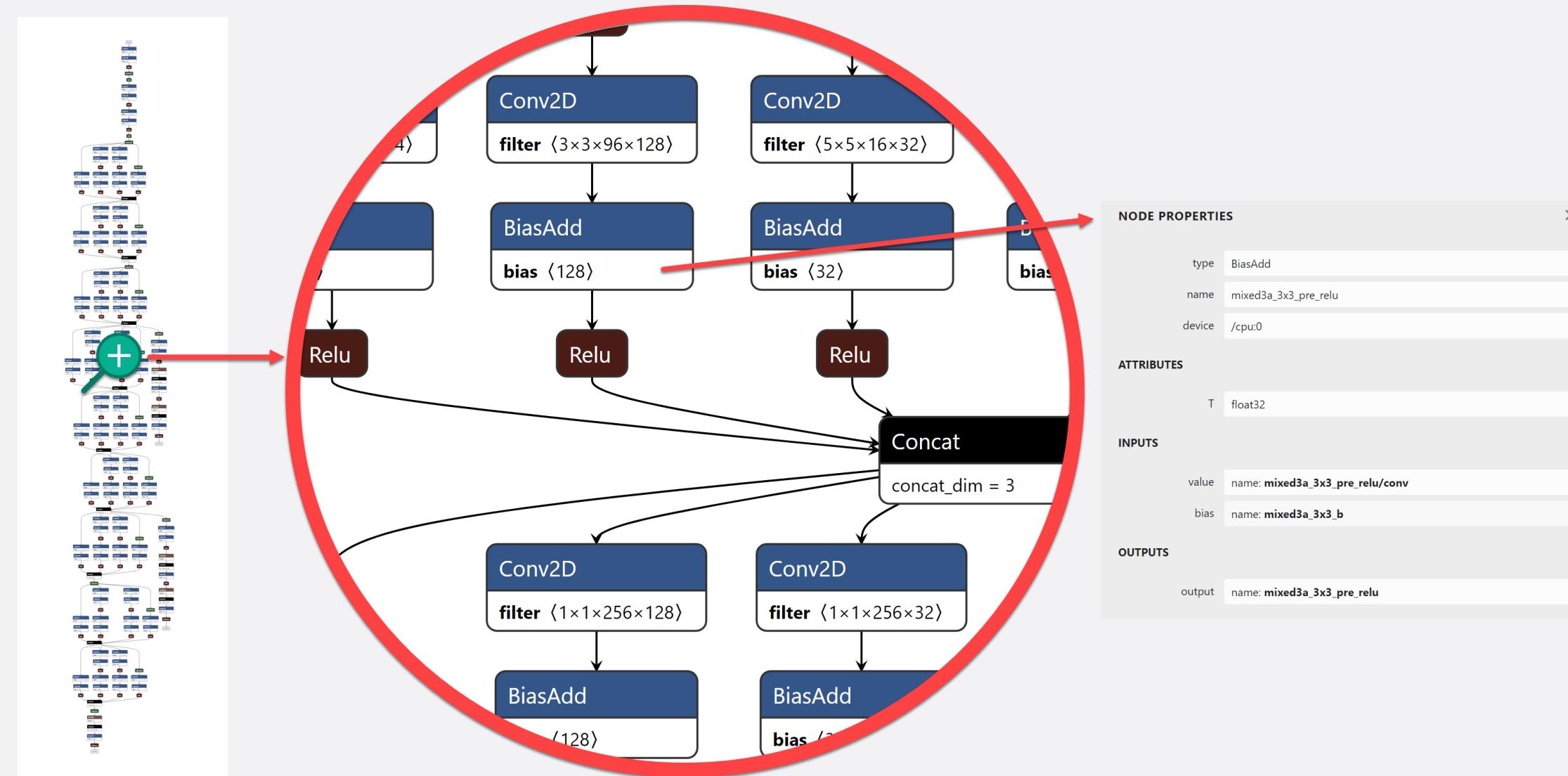
The image displays two screenshots of the Azure Machine Learning Studio interface. The left screenshot shows the main homepage with sections for 'Nuevo' (New), 'Inicio' (Home), 'Notebooks', 'ML automatizado' (Automated ML), and 'Diseñador' (Designer). It also features a 'Crear nuevo' (Create new) button, 'Notebooks' and 'ML automatizado' cards, a 'Tutoriales' (Tutorials) section with links to various Azure Machine Learning resources, and a 'Vínculos' (Links) section for 'Blog' and 'Documentación'. The right screenshot shows the 'Diseñador' (Designer) workspace for a 'Regression - Automobile Price Prediction (Compare algorithms)' project. A red arrow points from the 'Diseñador' link on the homepage to the 'Diseñador' icon in the workspace sidebar. The workspace interface includes a search bar, a toolbar with various icons, a list of available components like Datasets, Sample datasets, Data Input and Output, Data Transformation, Feature Selection, Statistical Functions, Machine Learning Algorithms, Model Training, Model Scoring & Evaluation, Python Language, R Language, Text Analytics, Computer Vision, Recommendation, Anomaly Detection, and Web Service, and a visual workflow diagram for comparing regression algorithms.

Otras cosas que quizá deberías conocer (2/3)

Bonus

Netron.App – Herramientas

<https://netron.app/>, para poder visualizar los modelos. En este caso he cargado el de la Sección 3:



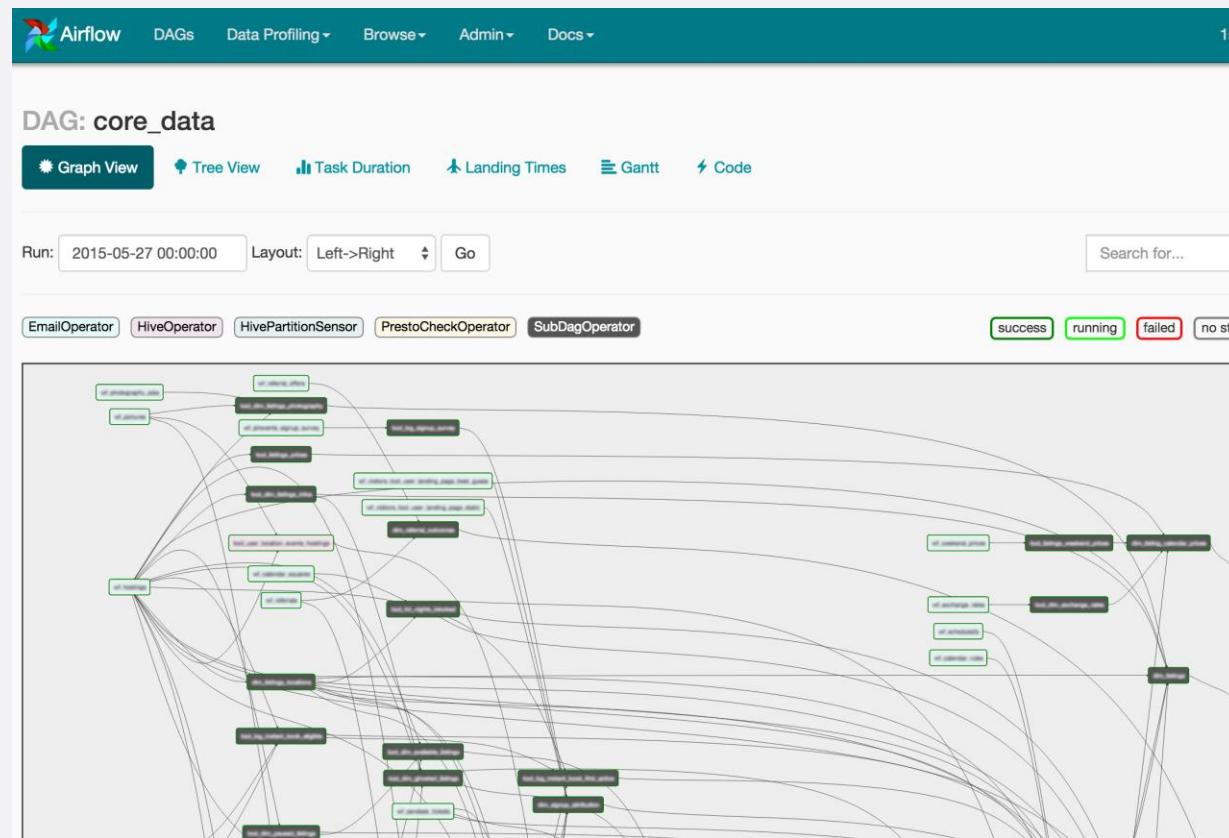
Otras cosas que quizá deberías conocer (3/3)

Bonus

Tools – Herramientas

Otra y no es un ETL propiamente dicho, gratuita y para hacer pipelines para Spark es [Apache Airflow](https://airflow.apache.org/) (<https://airflow.apache.org/>).

Usala con docker para revisar su funcionamiento (<https://hub.docker.com/r/puckel/docker-airflow>). El ecosistema de Apache es muy potente ([Apache Spark](#)) y gratuito, nunca esta de más saberlo, aunque estemos hablando de .NET lo comento debido a que existen bindings por parte de Microsoft o bien usando Azure HDInsight si deseas estar en Azure.



Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
28	count at NativeMethodAccessorImpl.java -2	2017/04/12 20:28:12	0.2 s	1/1	1/1
27	collect at utils.scala:52	2017/04/12 19:47:32	43 ms	1/1	2/2
26	count at NativeMethodAccessorImpl.java -2	2017/04/12 19:47:32	0.2 s	2/2	3/3
25	collect at utils.scala:195	2017/04/12 19:40:41	7 ms	1/1	1/1





¡Gracias!

Puedes encontrarme buscando por **jmfloreszazo** en

