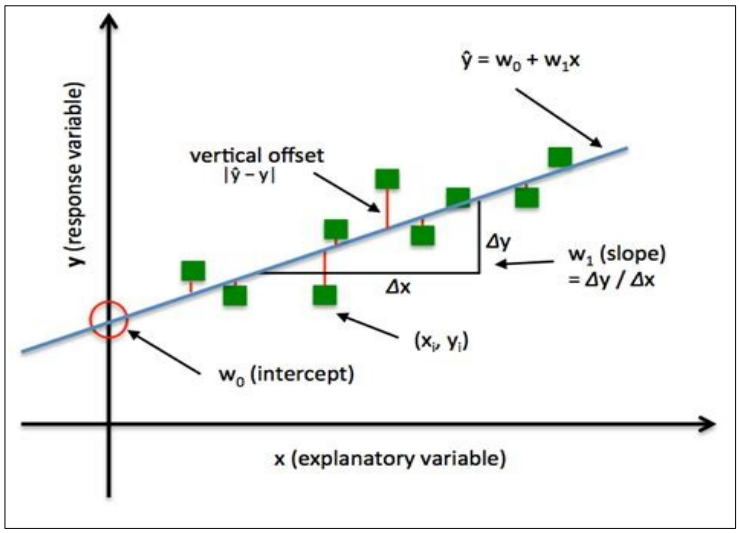
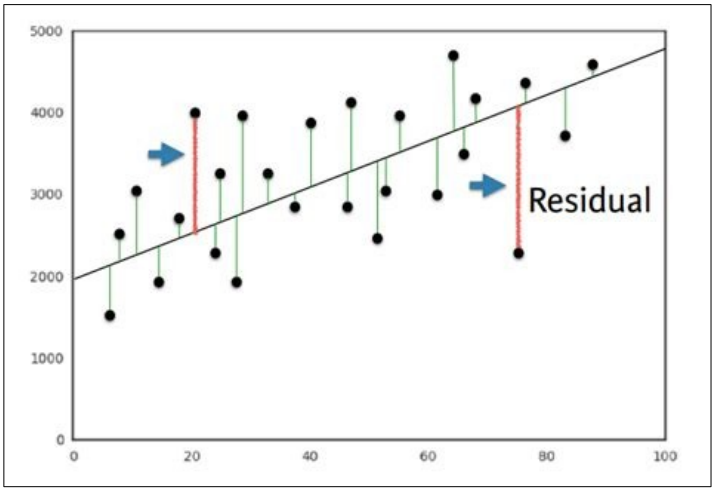
Intro:

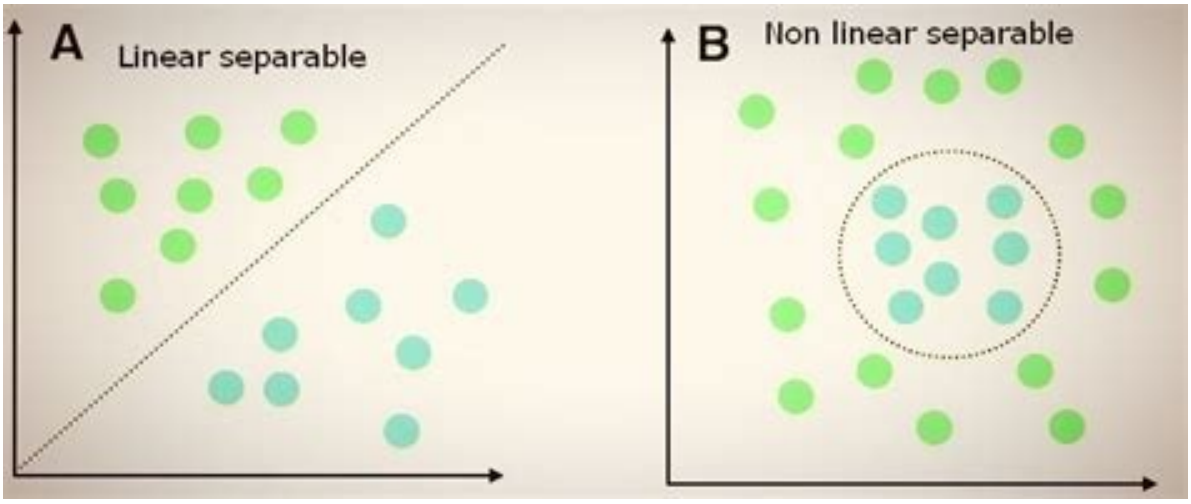
En su forma más simple un modelo predictivo sería algo así:
y = wX+b
Donde w son los pesos y X son las variables de entrada (es por tanto un problema multivariado, I.e., multidimensional).

En el caso más simple, la ecuación de la recta es la recta que mejor se ajusta a la distribución de los datos, por lo que conociendo un nuevo punto xi, podemos calcular la efectividad, posicionándolo en el valor correspondiente a su recta de ajuste. Ejemplo de regresión simple:

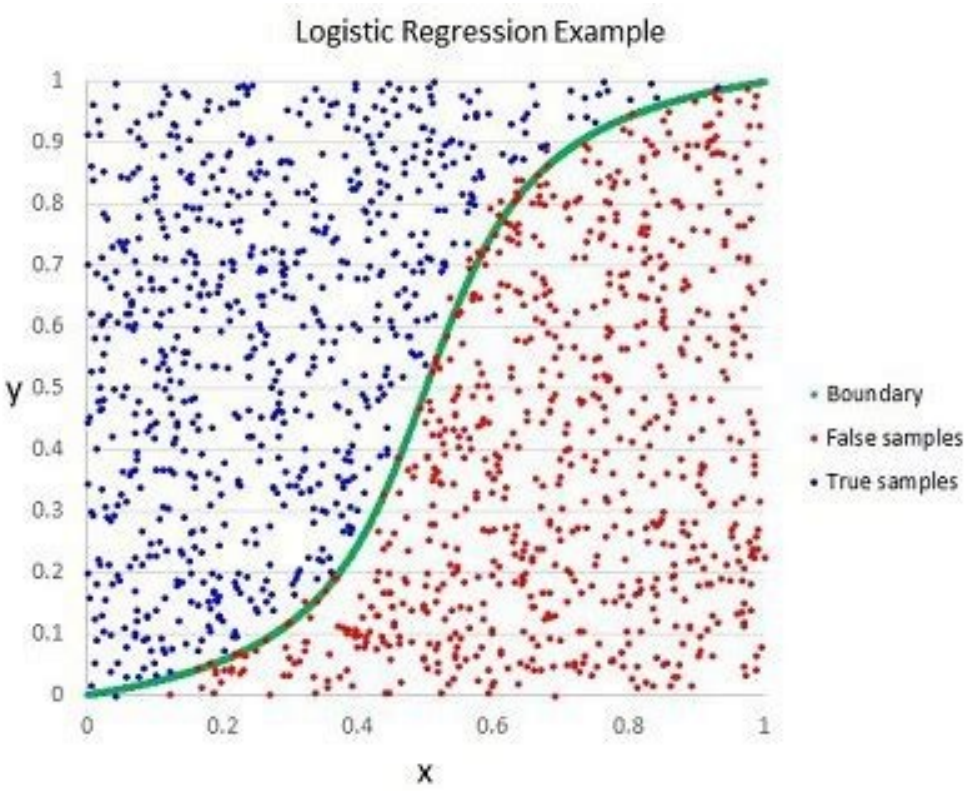




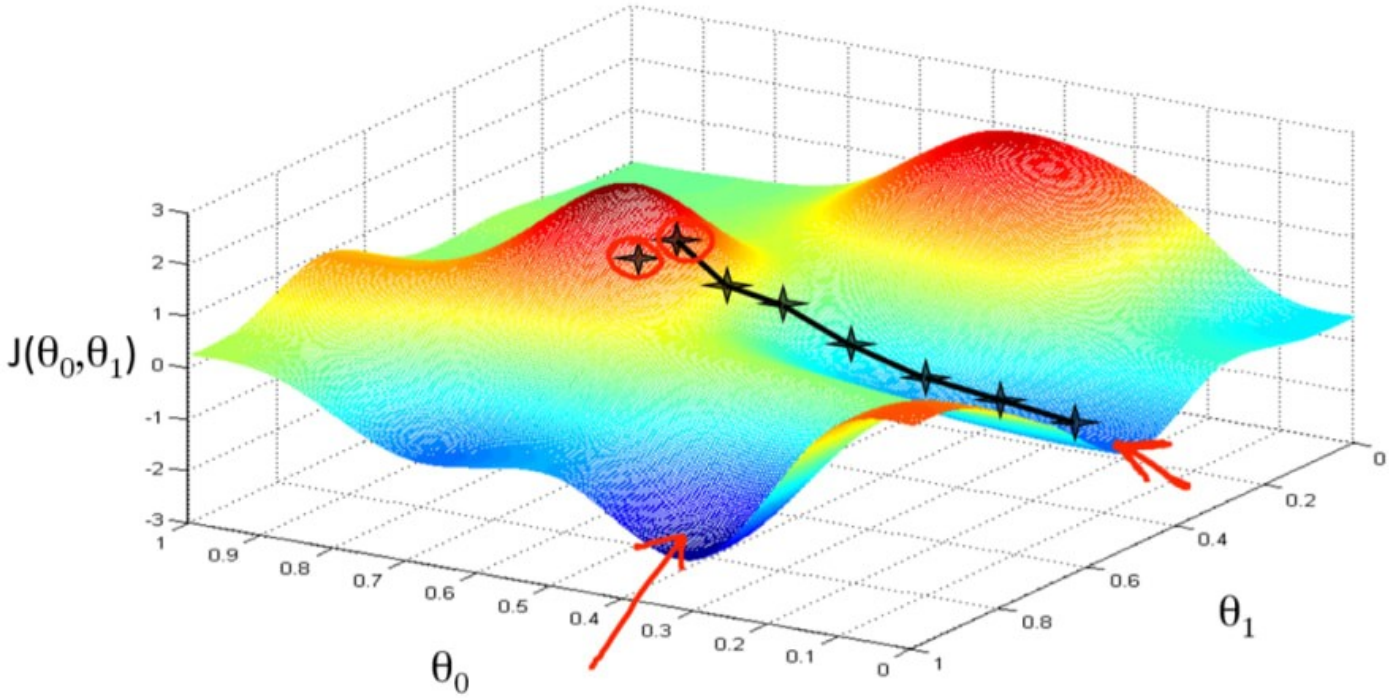
Para el caso de clasificación, el algoritmo a utilizar puede ser el mismo, pero la función de error o coste ha de cambiar. Ejemplo:



Para el caso de clasificación binaria, si los datos no fueran linealmente separables, habría que recurrir a modelos no lineales:



Todo esto es ML de 1980, pero actualmente hay modelos más sofisticados capaces de adaptar un hiperplano no lineal a los datos. En resumen buscamos un modelo que se adapte a la realidad de las variables de entrada. Esto es parecido a intentar aproximar un terreno con un conjunto de funciones matemáticas. De esta forma teniendo un nuevo punto **X**, podemos calcular la **y** de la forma más precisa. I.e., menor error posible → global mínimum.



Trabajos realizados:

1. Generación de dataset:

Manteniendo los valores de las X, hemos filtrado la efectividad para E<=1, E<=2, E<=4, generando tres datasets, donde la X se mantiene, pero la variable a predecir varía según el caso.

Variable a predecir (también llamada Target Variable (y): **Efectividad**

Variables de entrada (también llamadas Labels or Predictor Variables) (X):

1. 1º o 2º saque,
2. Lado(1:Iguales;0:Ventaja),
3. DIRECCIÓN:1 abierto;2 al cuerpo;3 a la T,
4. V(km/h),
5. [YA],
6. ZA,
7. Znet,
8. TIME,
9. difV,
10. &(deg),
11. ANG. IN,
12. dLinea

2. Entrenamiento de los 3 modelos: E<=1, E<=2, E<=4.

2.1. ML/Deep Learning Framework:

Utilizando AutoGluon hemos entrenado los tres modelos mencionados. La ventaja de AutoGluon, es que permite entrenar de forma automática múltiples modelos y además se encarga de preprocesar los datos. Esto tiene un coste, que es el de la flexibilidad. Por ejemplo, autogluon no permite la implementación de diferentes funciones de error (Loss functions). Pero en este caso las ventajas superan a las desventajas. Pros y Cons de **Autogluon**:

PROS:
1. Preprocesado de datos de gran calidad
2. utilización de múltiples modelos de entrenamiento
3. Flexibilidad para utilizar diferentes métricas de Evaluación de resultados.
4. Permite utilizar modelos explicativos de ML, que nos proporcionan los valores de las variables que favorecen la efectividad.

CONS:
1. Falta de Flexibilidad: Funciones de error

Conclusión: AutoGluon es por tanto un framework End2End muy potente para un entrenamiendo de ML de gran nivel, aunque para un proyecto púramente de AI puede quedarse corto en cuanto a flexibilidad. Sin embargo, para proyectos en los que la AI es una herramienta y no el objetivo de estudio es una herramienta muy interesante.

2.1. Entrenamiento (falta **completar** explicando cada apartado)**:**
2.1.1. Entrenamiento de un modelo de Clasificación.
2.1.2. Selección de métricas de Evaluación que mejor resultados generan (hyperparámetro principal en este caso).
2.1.2. Ćálculo de Feature Importance.
2.1.3. Análisis de reglas de decisión xAI (Explainable AI).

Resultados:

E<=1 → ver email

```
AutoGluon training complete, total runtime = 69.28s ... Best model: "WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor = TabularPredictor.load("AutogluonModels/ag-20220823_093725/")
> ------------------------------
> FIGS-Fast Interpretable Greedy-Tree Sums:
>       Predictions are made by summing the "Val" reached by traversing each tree
> ------------------------------
dLinea <= 0.296 (Tree #0 root)
        V(km/h) <= 196.650 (split)
                &(deg) <= 7.008 (split)
                        Val: 0.260 (leaf)
                        Znet <= 1.142 (split)
                                Val: 0.785 (leaf)
                                Val: 0.335 (leaf)
                Val: 0.638 (leaf)
        dLinea <= 0.539 (split)
                &(deg) <= 5.986 (split)
                        Val: 0.082 (leaf)
                        TIME <= 0.427 (split)
                                Val: 0.402 (leaf)
                                Val: 0.121 (leaf)
                Val: 0.026 (leaf)

        +
difV <= 4.556 (Tree #1 root)
        Val: 0.834 (leaf)
        dLinea <= 0.150 (split)
                Val: 0.120 (leaf)
                Val: -0.008 (leaf)

        +
difV <= 16.446 (Tree #2 root)
        Val: -0.002 (leaf)
        Val: 0.744 (leaf)
```

```
> ------------------------------
> BoostedRules:
>       Rule →predicted probability (final prediction is weighted sum of all predictions)
> ------------------------------
  If dLinea <= 0.2915 →0.00 (weight: 1.23)
  If dLinea > 0.2915 →0.23 (weight: 0.68)
  If dLinea <= 0.5395 →0.34 (weight: 0.25)
  If dLinea > 0.5395 →0.12 (weight: 0.14)
  If &(deg) <= 5.35964 →1.00 (weight: 0.12)
```

```
:feature_importance
Computing feature importance via permutation shuffling
        3.85s   = Expected runtime (0.77s per shuffle
        1.27s   = Actual runtime (Completed 5 of 5 shu
                                                importance
dLinea                                            0.210129
&(deg)                                            0.033071
V(km/h)                                           0.003238
TIME                                              0.003115
difV                                              0.000053
1º o 2º saque                                     0.000000
Lado(1:Iguales;0:Ventaja)                         0.000000
DIRECCIÓN:1 abierto;2 al cuerpo;3 a la T          0.000000
[YA]                                              0.000000
ZA                                                0.000000
Znet                                              0.000000
ANG. IN                                           0.000000
```

```
:feature_importance
Computing feature importance via permutation shuffling for 12 fe
        3.68s   = Expected runtime (0.74s per shuffle set)
        1.83s   = Actual runtime (Completed 5 of 5 shuffle sets)
                                              importance    stddev
dLinea                                          0.043893  0.001227
difV                                            0.013404  0.001153
&(deg)                                          0.011711  0.001594
V(km/h)                                         0.009627  0.000929
Znet                                            0.005889  0.001953
TIME                                            0.003669  0.000925
DIRECCIÓN:1 abierto;2 al cuerpo;3 a la T        0.001178  0.000892
1º o 2º saque                                   0.000000  0.000000
Lado(1:Iguales;0:Ventaja)                       0.000000  0.000000
[YA]                                            0.000000  0.000000
ANG. IN                                         0.000000  0.000000
ZA                                             -0.000082  0.000112

Evaluation:
Evaluation: f1_weighted on test data: 0.9300566672671776
Evaluations on test data:
{
    "f1_weighted": 0.9300566672671776,
    "accuracy": 0.9352236616964067,
    "balanced_accuracy": 0.7255665955566718,
    "mcc": 0.5300949415278825,
    "roc_auc": 0.885454585828947,
    "f1": 0.5546218487394958,
    "precision": 0.6707317073170732,
    "recall": 0.47277936962750716
}
```

EVALUACIÓN:

	precision	recall	f1-score	support	accuracy	*balanced_accuracy*
0	0.76	0.90	0.82	3077	0.90	0.72
1	0.29	0.12	0.17	1014		
0	0.96	0.93	0.94	3784	0.90	***recall***
1	0.37	0.47	0.41	307		0.47
0	0.94	0.99	0.96	3737	0.93	***recall_weighted***
1	0.71	0.37	0.49	354		0.93
0	0.94	0.98	0.96	3762	0.93	** **F1** **
1	0.59	0.34	0.43	329		0.43
0	0.95	0.98	0.97	3742	0.94	** **f1_weighted** **
1	0.67	0.47	0.55	349		xxx (missing data)

F1_weighted es la mejor métrica, por presentar el mejor equilibrio entre precision, recall, F1.score y Accuracy

E<=2:

	precision	recall	f1-score	support	accuracy	f1_weighted
0.0	0.74	0.99	0.84	3011	0.73	0.63
1.0	0.38	0.02	0.04	1080		
0.0	0.74	1.00	0.85	3035	0.74	f1
1.0	0.14	0.00	0.00	1056		0.004
0.0	0.74	1.00	0.85	3035	0.74	balanced_accuracy
1.0	0.50	0.00	0.00	1056		0.50
0.0	0.74	0.99	0.85	3042	0.74	recall
1.0	0.30	0.01	0.02	1049		0.01
0.0	0.74	1.00	0.85	3012	0.74	recall_weighted
1.0	0.00	0.00	0.00	1079		0.74
0.0	0.75	1.00	0.85	3052	0.75	accuracy
1.0	0.43	0.00	0.01	1039		

```
:feature_importance
Computing feature importance via permutation shuffling for 12 features using 4091 rows with 5 shuffle sets...
        0.9s    = Expected runtime (0.18s per shuffle set)
        0.48s   = Actual runtime (Completed 5 of 5 shuffle sets)
                                          importance    stddev   p_value  n  p99_high   p99_low
TIME                                        0.004989  0.002573  0.006147  5  0.010287 -0.000309
V(km/h)                                     0.004683  0.001155  0.000410  5  0.007061  0.002305
dLinea                                      0.000591  0.000279  0.004550  5  0.001166  0.000016
1º o 2º saque                               0.000000  0.000000  0.500000  5  0.000000  0.000000
Lado(1:Iguales;0:Ventaja)                   0.000000  0.000000  0.500000  5  0.000000  0.000000
DIRECCIÓN:1 abierto;2 al cuerpo;3 a la T     0.000000  0.000000  0.500000  5  0.000000  0.000000
[YA]                                        0.000000  0.000000  0.500000  5  0.000000  0.000000
ZA                                          0.000000  0.000000  0.500000  5  0.000000  0.000000
Znet                                        0.000000  0.000000  0.500000  5  0.000000  0.000000
ANG. IN                                    -0.000818  0.000289  0.998404  5 -0.000223 -0.001413
difV                                       -0.004149  0.003957  0.960530  5  0.003997 -0.012296
&(deg)                                     -0.004513  0.001627  0.998280  5 -0.001162 -0.007864
```

```
TabularPredictor saved. To load, use: predictor = TabularPredictor.load("AutogluonModels/ag-20220824_094511/")
> ----------------------------
> FIGS-Fast Interpretable Greedy-Tree Sums:
>       Predictions are made by summing the "Val" reached by traversing each tree
> ----------------------------
V(km/h) <= 182.610 (Tree #0 root)
        V(km/h) <= 144.018 (split)
                Val: 0.153 (leaf)
                Val: 0.237 (leaf)
        dLinea <= 0.177 (split)
                Val: 0.156 (leaf)
                difV <= 13.119 (split)
                        Val: 0.311 (leaf)
                        Val: 0.100 (leaf)

        +
difV <= 3.625 (Tree #1 root)
        Val: -0.254 (leaf)
        [YA] <= 1.021 (split)
                Val: 0.032 (leaf)
                Val: -0.005 (leaf)
```

E<=4:

	precision	recall	f1-score	support	accuracy	balanced_accuracy
0.0	0.79	1.00	0.88	3212	0.78	0.50
1.0	0.00	0.00	0.00	879		
0.0	0.78	1.00	0.88	3206	accuracy	f1_weighted
1.0	0.00	0.00	0.00	885	0.78	0.69
0.0	0.78	1.00	0.88	3195	accuracy	recall
1.0	0.00	0.00	0.00	896	0.78	0.00
0.0	0.78	1.00	0.88	3193	accuracy	recall_weighted
1.0	0.00	0.00	0.00	898	0.78	0.78

```
AutoGluon training complete, total runtime = 93.9s ... Best model: "WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor = TabularPredictor.load("AutogluonModels/ag-20220824_104402/")
> ----------------------------
> FIGS-Fast Interpretable Greedy-Tree Sums:
>       Predictions are made by summing the "Val" reached by traversing each tree
> ----------------------------
dLinea <= 0.241 (Tree #0 root)
        Val: 0.141 (leaf)
        dLinea <= 1.951 (split)
                TIME <= 0.405 (split)
                        dLinea <= 1.594 (split)
                                Val: 0.221 (leaf)
                                Val: 0.282 (leaf)
                        Val: 0.206 (leaf)
                Val: 0.276 (leaf)

        +
difV <= 3.536 (Tree #1 root)
        Val: -0.211 (leaf)
        [YA] <= 0.481 (split)
                Val: -0.069 (leaf)
                Val: 0.003 (leaf)
```

```
:feature_importance
Computing feature importance via permutation shuffling for 12 features using 4091 rows with 5 shuffle sets...
        1.47s   = Expected runtime (0.29s per shuffle set)
        0.56s   = Actual runtime (Completed 5 of 5 shuffle sets)
                                          importance  stddev  p_value  n  p99_high  p99_low
1º o 2º saque                                    0.0     0.0      0.5  5       0.0      0.0
Lado(1:Iguales;0:Ventaja)                        0.0     0.0      0.5  5       0.0      0.0
DIRECCIÓN:1 abierto;2 al cuerpo;3 a la T         0.0     0.0      0.5  5       0.0      0.0
V(km/h)                                          0.0     0.0      0.5  5       0.0      0.0
[YA]                                             0.0     0.0      0.5  5       0.0      0.0
ZA                                               0.0     0.0      0.5  5       0.0      0.0
Znet                                             0.0     0.0      0.5  5       0.0      0.0
TIME                                             0.0     0.0      0.5  5       0.0      0.0
difV                                             0.0     0.0      0.5  5       0.0      0.0
&(deg)                                           0.0     0.0      0.5  5       0.0      0.0
ANG. IN                                          0.0     0.0      0.5  5       0.0      0.0
dLinea                                           0.0     0.0      0.5  5       0.0      0.0
```

Conclusión:
… incompleto …

E<=2 y sobre todo E<=3 no son tan representativos para análisis como E<=1. De hecho, al ser el Recall = 0 para Efectividad = 1, el cálculo de feature importance no es informativo (todas dan 0.0)

… incompleto …