Aprendizaje Automático

Introducción al Aprendizaje Automático

Prof. Wladimir Rodríguez

wladimir.rodriguez@outlook.com

Departamento de Computación

Información General

- **Profesor:**
  - Dr. Wladimir Rodriguez
- **Horario:**
  - Lunes de 8:00 a 10:00 y Jueves de 10:00 a 12:00, Salón: Postgrado de Computación
- **Horas de Oficina:**
  - Martes y Miércoles de 10:00 a 12:00, Salón: Postgrado de Computación
- **Material:**
  - https://github.com/wladrod/Aprendizaje-Automatico-2022

Descripción

- El Aprendizaje Automático (o Machine Learning) es una rama de la Inteligencia Artificial que se dedica al estudio de los programas que aprenden a realizar una tarea en base a la experiencia.
- Esta asignatura presenta una visión general del aprendizaje automático. Durante el curso, se estudian los principales métodos, incluyendo aprendizaje supervisado, no supervisado y aprendizaje por refuerzo. Además, se incluyen ejercicios y ejemplos prácticas que permiten afianzar los conocimientos

Programa:

1. Aprendizaje
2. ¿Qué es el Aprendizaje Automático?
3. Inteligencia Artificial, Aprendizaje Automático, Aprendizaje Profundo
4. Aprendizaje Automático y Ciencias de Datos
5. Historia del Aprendizaje Automático
6. Aplicaciones de Aprendizaje Automático
7. Proceso de Aprendizaje Automático

1. Tipos de Aprendizaje Automático
2. Aprendizaje Supervisado
3. Aprendizaje No Supervisado
4. Aprendizaje por Refuerzo
5. Generación del Modelo de Aprendizaje
6. Ambiente para el Aprendizaje Automático y el Aprendizaje Profundo
7. Demo

Evaluación

- Tareas semanales: 55%
- Asistencia y participación en clase: 5%
- Proyecto final: 40%

Aprendizaje:

- Aprendizaje es el proceso a través del cual se adquieren o modifican habilidades, destrezas, conocimientos, conductas o valores como resultado del estudio, la experiencia, la instrucción, el razonamiento y la observación. (Wikipedia)

- Aprendizaje denota cambios en el sistema que son adaptativos en el sentido de que permiten al sistema realizar la tarea o las tareas extraídas de la misma población de manera más eficiente y más eficaz la próxima vez. (H. Simon)

¿Qué es el Aprendizaje Automático?

- El aprendizaje automático o aprendizaje de máquinas (del inglés, *Machine Learning*) es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. (Wikipedia)

- Campo de estudio que da a los computadores la capacidad de aprender sin ser explícitamente programados. (Arthur Samuel, 1959)

- Aprendizaje Automático es predecir el futuro basado en el pasado. (Hal Daume III)

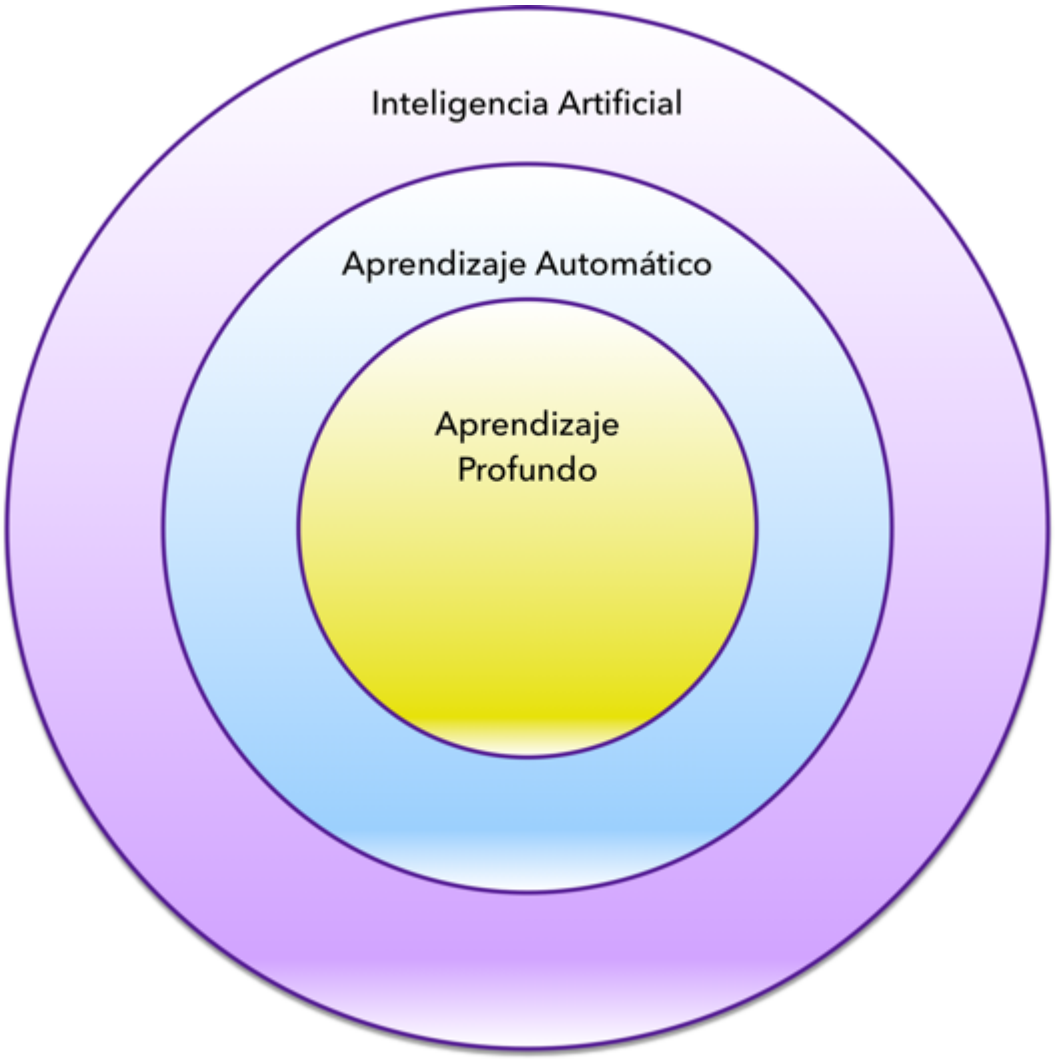
- Aprendizaje Automático es la programación de computadoras para optimizar un criterio de rendimiento utilizando datos de ejemplo o experiencia pasada. (Ethem Alpaydin)

- El objetivo del aprendizaje automático es desarrollar métodos que puedan detectar automáticamente los patrones en los datos y luego utilizar los patrones descubiertos para predecir los datos futuros u otros resultados de interés. (Kevin P. Murphy)

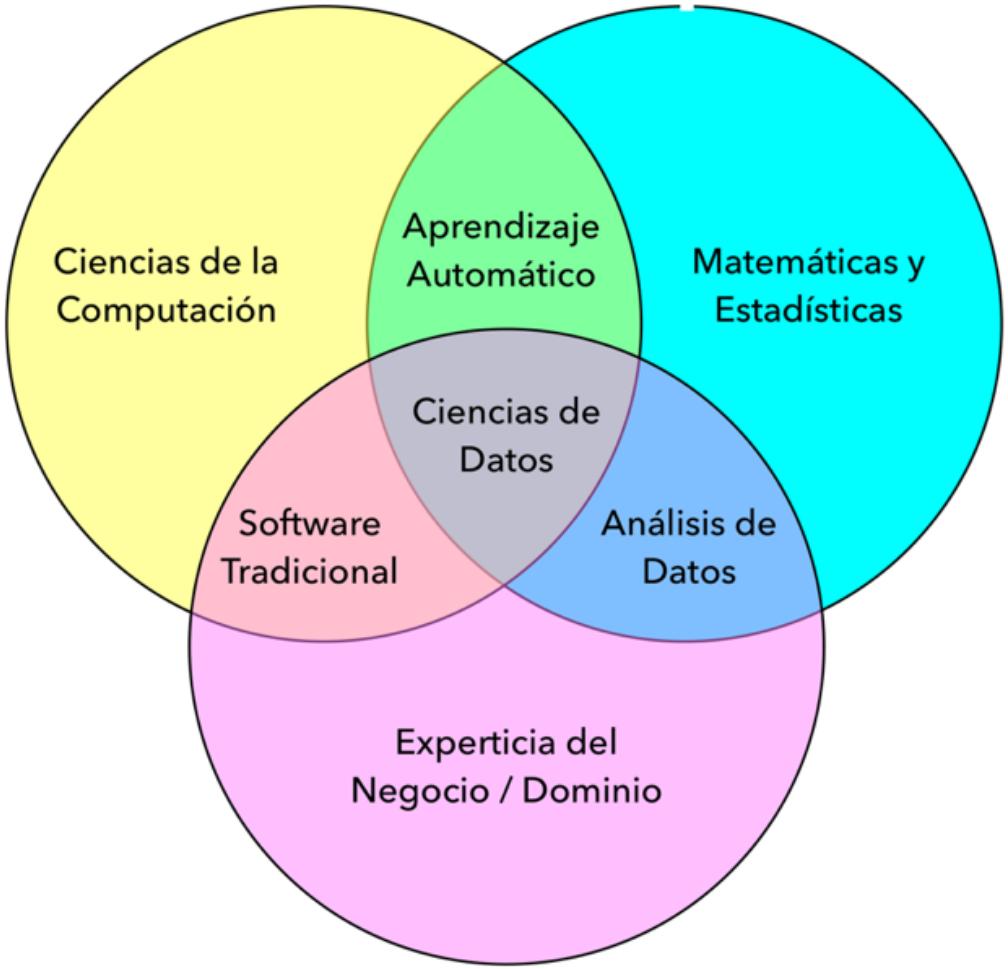
- El campo del reconocimiento de patrones se refiere al descubrimiento automático de regularidades en datos mediante el uso de algoritmos informáticos y con el uso de estas regularidades tomar acciones. (Christopher M. Bishop)

El Mundo del Aprendizaje Automático

Inteligencia Artificial, Aprendizaje Automático, Aprendizaje Profundo



Aprendizaje Automático y Ciencias de Datos



Historia del Aprendizaje Automático

Basado en:
https://roboticsbiz.com/machine-learning-the-complete-history-in-a-timeline/

- **Siglo VIII**: desarrollo de métodos estadísticos: varios conceptos vitales en el aprendizaje automático se derivan de la teoría de la probabilidad y las estadísticas, y se remontan al siglo XVIII. En 1763, el estadístico inglés Thomas Bayes estableció un teorema matemático para la probabilidad condicional, que se conoció como Teorema de Bayes y sigue siendo un concepto central en algunos enfoques modernos del aprendizaje automático.

- **1950** — La prueba de Turing: los artículos del matemático inglés Alan Turing en la década de 1940 estaban llenos de ideas sobre la inteligencia artificial. “¿Pueden pensar las máquinas?” Preguntó. En 1950, sugirió una prueba para la inteligencia de las máquinas, más tarde conocida como la Prueba de Turing, en la que una máquina se llama "inteligente" si sus respuestas a las preguntas pueden convencer a un humano.

- **1952** — Juego de Damas: En 1952, el investigador Arthur Samuel creó una máquina de aprendizaje automatico, capaz de aprender a jugar a las damas. Usó guías anotadas por expertos humanos y jugó contra sí mismo para aprender a distinguir los movimientos correctos de los malos.

- **1956** — El Taller de Dartmouth: El término "inteligencia artificial" nació durante el Taller de Dartmouth en 1956, que es ampliamente considerado como el evento fundacional de la inteligencia artificial como campo. El taller duró de seis a ocho semanas y asistieron matemáticos y científicos, incluido el informático John McCarthy, Marvin Minsky, Nathaniel Rochester y Claude Shannon.

- **1957** — El Perceptrón del célebre psicólogo estadounidense Frank Rosenblatt fue uno de los primeros intentos de crear una red neuronal con el uso de una resistencia rotatoria (potenciómetro) impulsada por un motor eléctrico. La máquina podría tomar una entrada (como píxeles de imágenes) y crear una salida (como etiquetas).

- **1967** — Algoritmo del vecino más cercano: la regla del vecino más cercano es un clásico en el reconocimiento de patrones, que apareció en varios trabajos de investigación en la década de 1960, especialmente en un artículo escrito por T. Cover y P. Hart en 1967. El algoritmo mapeó un ruta para vendedores ambulantes, comenzando en una ciudad aleatoria pero asegurándose de visitar todas las ciudades durante un recorrido corto.

- **1973**: El informe Lighthill y el invierno de la IA: el Consejo de Investigación Científica del Reino Unido publicó el informe Lighthill de James Lighthill en 1973, que presenta un pronóstico muy pesimista en el desarrollo de aspectos centrales en la investigación de la IA. Afirmó que "en ninguna parte del campo los descubrimientos realizados hasta ahora produjeron el gran impacto que se prometió en ese momento". Como resultado, el gobierno británico recortó los fondos para la investigación de IA en todas las universidades excepto en dos. Este período de financiación e interés reducidos se conoce como invierno IA.
- **1979** — Stanford Cart: los estudiantes de la Universidad de Stanford inventaron un robot llamado Cart, conectado por radio a una gran computadora central, que puede sortear obstáculos en una habitación por sí solo. Aunque cruzar toda la habitación tomó cinco horas debido a errores y mapas apenas adecuados, la invención era lo último en tecnología en ese momento.

- **1981** — Aprendizaje basado en explicaciones: Gerald Dejong introdujo el concepto de aprendizaje basado en explicaciones, que analiza datos y crea una regla general que puede seguir descartando datos sin importancia.
- **1985** — NetTalk: El profesor Terry Sejnowski inventó NetTalk, un programa que aprende a pronunciar un texto escrito en inglés mostrándole el texto como entrada y haciendo coincidir las transcripciones fonéticas para comparar. La intención era construir modelos simplificados que pudieran arrojar luz sobre el aprendizaje humano.
- **1986** — Procesamiento distribuido en paralelo y modelos de redes neuronales: David Rumelhart y James McClelland publicaron Procesamiento distribuido en paralelo, que avanzó en el uso de modelos de redes neuronales para el aprendizaje automático.

- **1992** — Jugando al backgammon: el investigador Gerald Tesauro creó un programa basado en una red neuronal artificial, que era capaz de jugar al backgammon con habilidades que igualaban a los mejores jugadores humanos.
- **1997** — Deep Blue: Deep Blue de IBM se convirtió en el primer sistema informático de juego de ajedrez en vencer al actual campeón mundial de ajedrez. Deep Blue usó el poder de la computación en la década de 1990 para realizar búsquedas a gran escala de movimientos potenciales y seleccionar el mejor movimiento.

- **2006** — Aprendizaje profundo: Geoffrey Hinton creó el término "aprendizaje profundo" para explicar los nuevos algoritmos que ayudan a las computadoras a distinguir objetos y texto en imágenes y videos.
- **2010** — Kinect: Microsoft desarrolló el dispositivo de entrada de detección de movimiento llamado Kinect que puede rastrear 20 características humanas a una velocidad de 30 veces por segundo. Permitió a las personas interactuar con la computadora a través de movimientos y gestos.
- **2011** — Watson y Google Brain: Watson de IBM ganó un juego del concurso estadounidense Jeopardy contra dos de sus campeones. En el mismo año, Google Brain desarrolló su red neuronal profunda que podía descubrir y categorizar objetos de la forma en que lo hace un gato.

- **2012**: Clasificación de ImageNet y visión por computadora: el año vio la publicación de un influyente artículo de investigación de Alex Krizhevsky, Geoffrey Hinton e Ilya Sutskever, que describe un modelo que puede reducir drásticamente la tasa de error en los sistemas de reconocimiento de imágenes. Mientras tanto, X Lab de Google desarrolló un algoritmo de aprendizaje automático capaz de navegar de forma autónoma en videos de YouTube para identificar los videos que contienen gatos.
- **2014** — DeepFace: Facebook desarrolló un algoritmo de software DeepFace, que puede reconocer y verificar personas en fotos con una precisión humana.
- **2015** — Amazon Machine Learning: Andy Jassy de AWS lanzó sus servicios administrados de Machine Learning que analizan los datos históricos de los usuarios para buscar patrones e implementar modelos predictivos. En el mismo año, Microsoft creó el kit de herramientas de aprendizaje automático distribuido, que permite la distribución eficiente de problemas de aprendizaje automático en varias computadoras.

- **2016**: AlphaGo: creado por investigadores de Google DeepMind para jugar el antiguo juego chino de Go, ganó cuatro de cinco partidos contra Lee Sedol, quien ha sido el mejor jugador de Go del mundo durante más de una década.
- **2017** — Libratus y Deepstack: Investigadores de la Universidad Carnegie Mellon crearon un sistema llamado Libratus, y derrotó a cuatro de los mejores jugadores en No Limit Texas Hold 'em, luego de 20 días de juego en 2017. Investigadores de la Universidad de Alberta también reportaron un éxito similar con su sistema, Deepstack.
- **2017** - AlphaZero es un programa informático desarrollado por la empresa de investigación de inteligencia artificial DeepMind para dominar los juegos de ajedrez, shogi y go. Este algoritmo utiliza un enfoque similar a AlphaGo Zero.

- **2020**: GPT-3. OpenAI GPT-3 se presentó por primera vez en mayo de 2020 y las pruebas beta comenzaron en junio de 2020. OpenAI GPT-3 es un modelo de lenguaje que genera texto mediante la adopción de algoritmos preentrenados.
- **2021**: AlphaFold es un programa de inteligencia artificial (IA) desarrollado por DeepMind de Alphabet/Google que realiza predicciones de la estructura de las proteínas. El programa está diseñado como un sistema de aprendizaje profundo.

Aplicaciones del Aprendizaje Automático

Asistentes personales virtuales

Alexa, Bixby, Cortana, Google Assistant, Siri son algunos de los ejemplos populares de asistentes personales virtuales. Como su nombre indica, ayudan a encontrar información cuando se les pregunta por voz. Todo lo que necesita hacer es activarlos y preguntar "¿Cuál es mi horario para hoy?", "¿Cuáles son los vuelos de Alemania a Londres?"

Atención al cliente en línea

Actualmente, varios sitios web ofrecen la opción de chatear con un representante de atención al cliente mientras navegan dentro del sitio. Sin embargo, no todos los sitios web tienen un ejecutivo en vivo para responder sus consultas. En la mayoría de los casos, hablas con un *chatbot*. Estos bots tienden a extraer información del sitio web y presentarla a los clientes. Mientras tanto, los *chatbots* avanzan con el tiempo. Tienden a comprender mejor las consultas de los usuarios y les brindan mejores respuestas, lo cual es posible gracias a sus algoritmos de aprendizaje automático.

Recomendaciones de productos

Usted compró un producto en línea hace unos días y luego sigue recibiendo correos electrónicos para sugerencias de compras. Si no es así, entonces habrá notado que el sitio web de compras o la aplicación le recomienda algunos artículos que de alguna manera coinciden con su gusto. Ciertamente, esto refina la experiencia de compra. En función de su comportamiento con el sitio web o la aplicación, las compras anteriores, los artículos que le gustaron o añadieron al carrito, las preferencias de marca, etc., se realizan las recomendaciones del producto.

Detección de fraude en línea

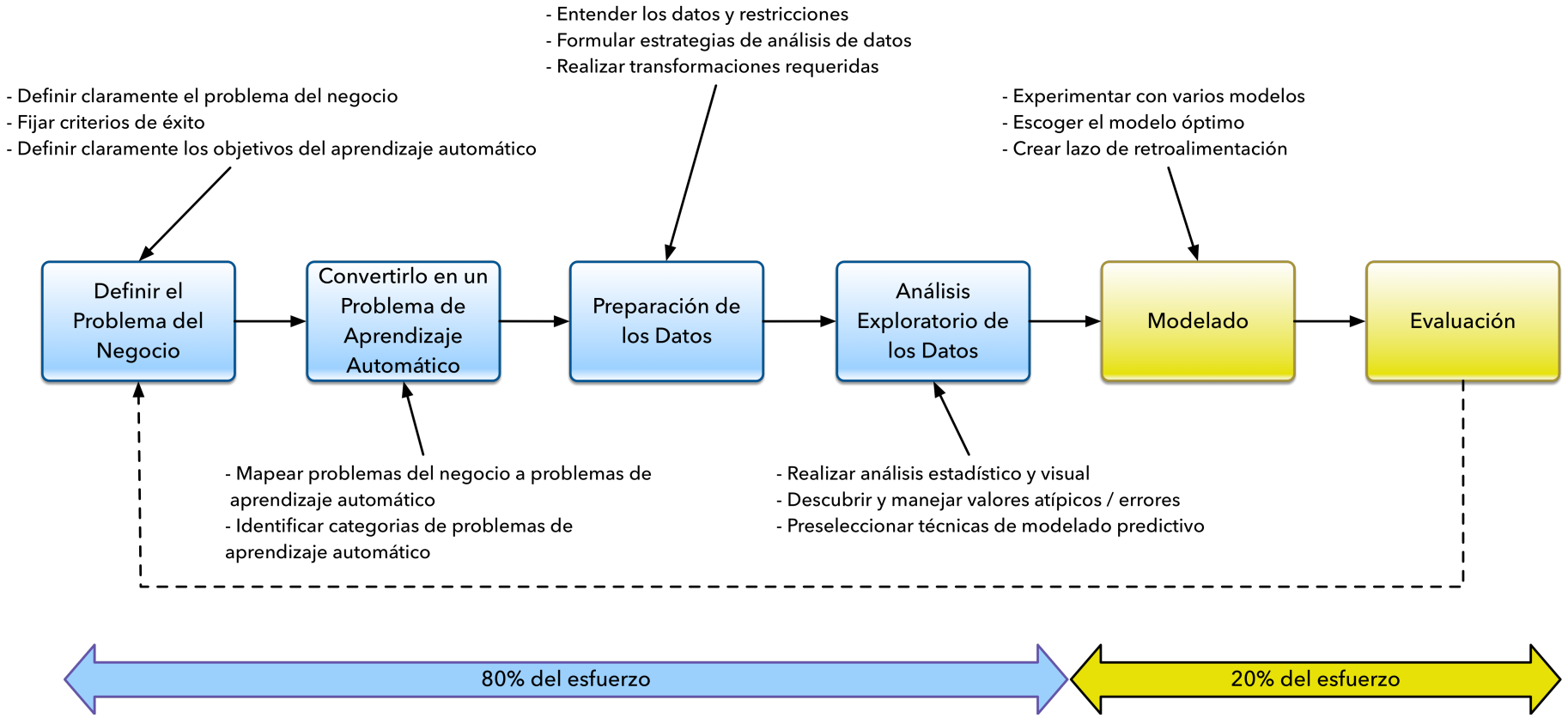
El aprendizaje automático está demostrando su potencial para hacer del ciberespacio un lugar seguro y el seguimiento de fraudes monetarios en línea es uno de sus ejemplos. Por ejemplo: *Paypal* está usando Aprendizaje Automático para la protección contra el lavado de dinero. La empresa utiliza un conjunto de herramientas que les ayuda a comparar millones de transacciones que se realizan y a distinguir entre transacciones legítimas o ilegítimas que tienen lugar entre compradores y vendedores.

Servicios de redes sociales

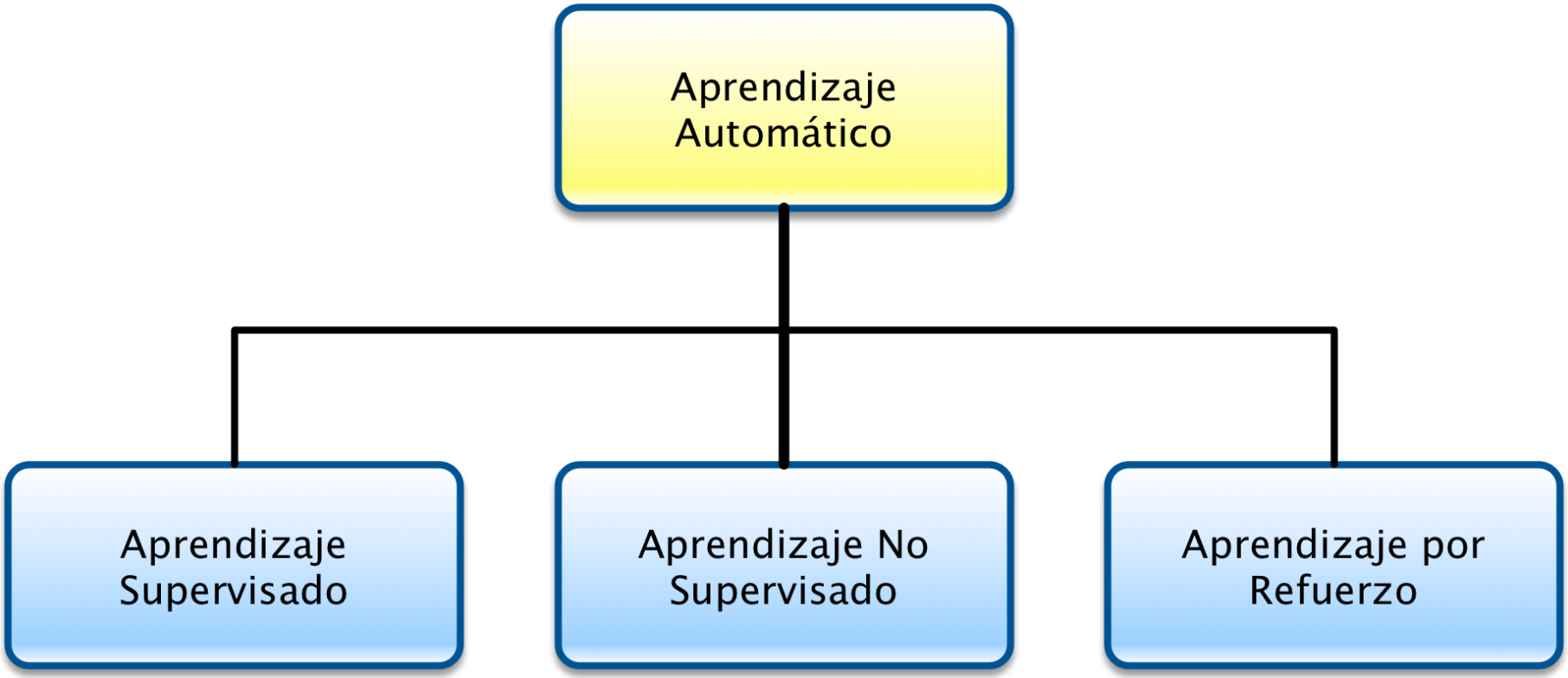
Desde la personalización de su servicio de noticias a la mejor orientación de anuncios, las plataformas de medios sociales están utilizando el aprendizaje automático para sus propios beneficios y para el usuario. Aquí hay algunos ejemplos que debe notar, usar y amar en sus cuentas de redes sociales, sin darse cuenta de que estas maravillosas características no son más que aplicaciones de Aprendizaje Automático.

- Personas que puedes conocer
- Reconocimiento de rostros

Proceso del Aprendizaje Automático



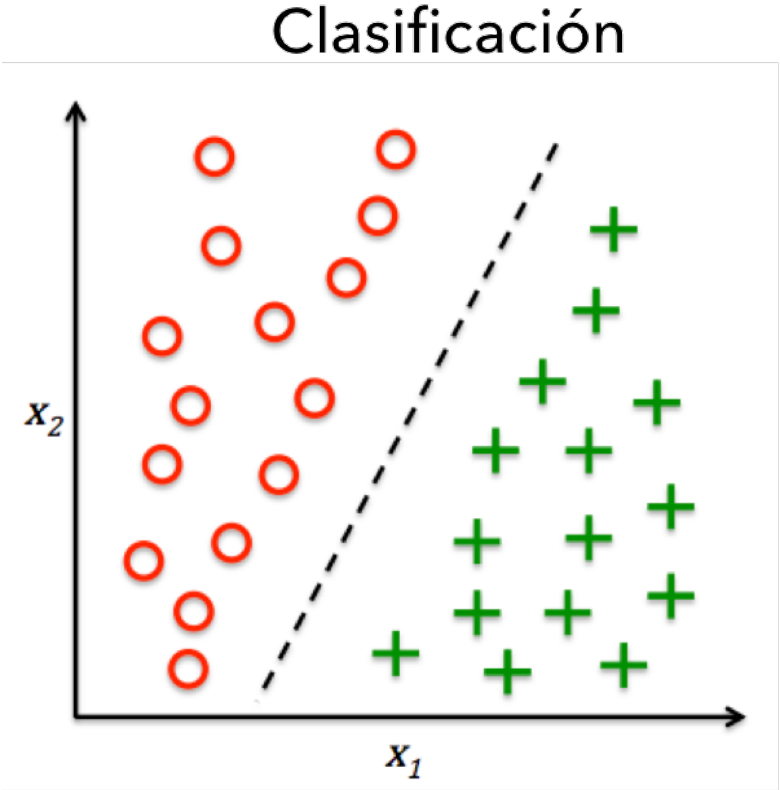
Tipos de Aprendizaje Automático

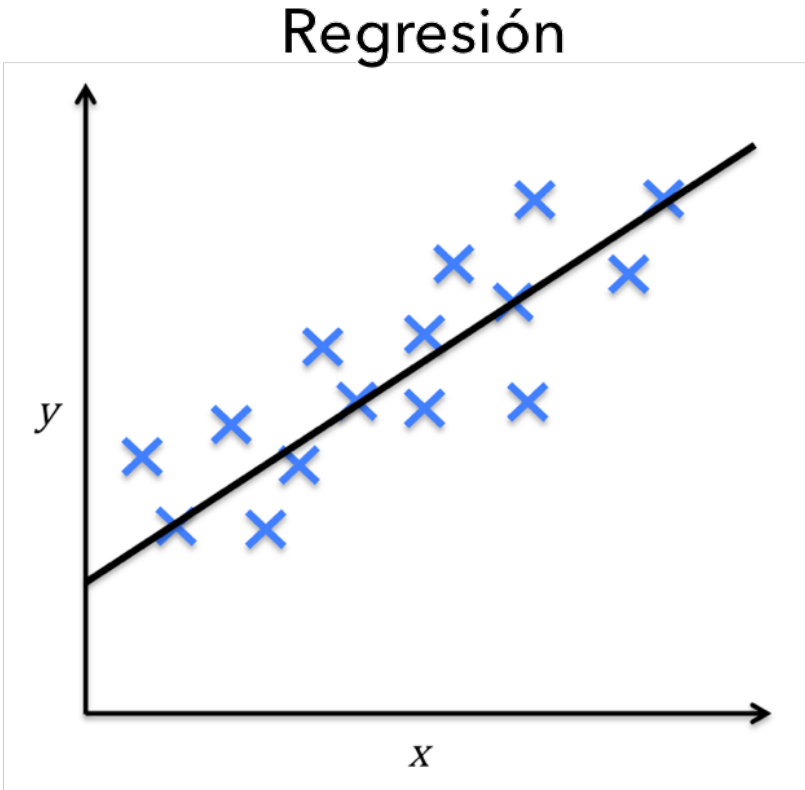


Aprendizaje Supervisado

Es una técnica para deducir una función a partir de datos de entrenamiento. Los datos de entrenamiento consisten de pares de objetos (normalmente vectores): un componente del par son los datos de entrada y el otro, los resultados deseados. La salida de la función puede ser un valor numérico (como en los problemas de regresión) o una etiqueta de clase (como en los de clasificación). El objetivo del aprendizaje supervisado es el de crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada válido después de haber visto una serie de ejemplos, los datos de entrenamiento. Para ello, tiene que generalizar a partir de los datos presentados a las situaciones no vistas previamente. (Wikipedia)

Clasificación y Regresión

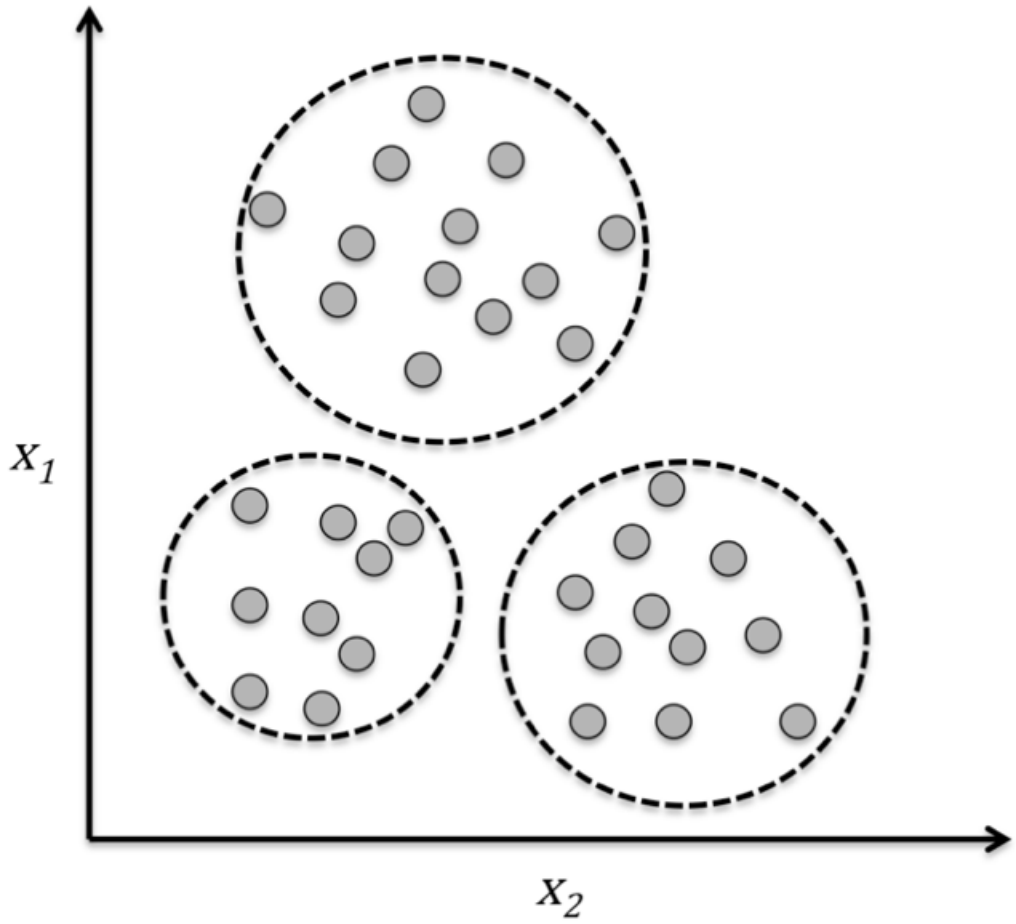




Aprendizaje No Supervisado

Es un método de Aprendizaje Automático donde un modelo es ajustado a las observaciones. Se distingue del Aprendizaje Supervisado por el hecho de que no hay un conocimiento a priori. En el aprendizaje no supervisado, un conjunto de datos de objetos de entrada es tratado. Así, el aprendizaje no supervisado típicamente trata los objetos de entrada como un conjunto de variables aleatorias, siendo construido un modelo de densidad para el conjunto de datos. (Wikipedia)

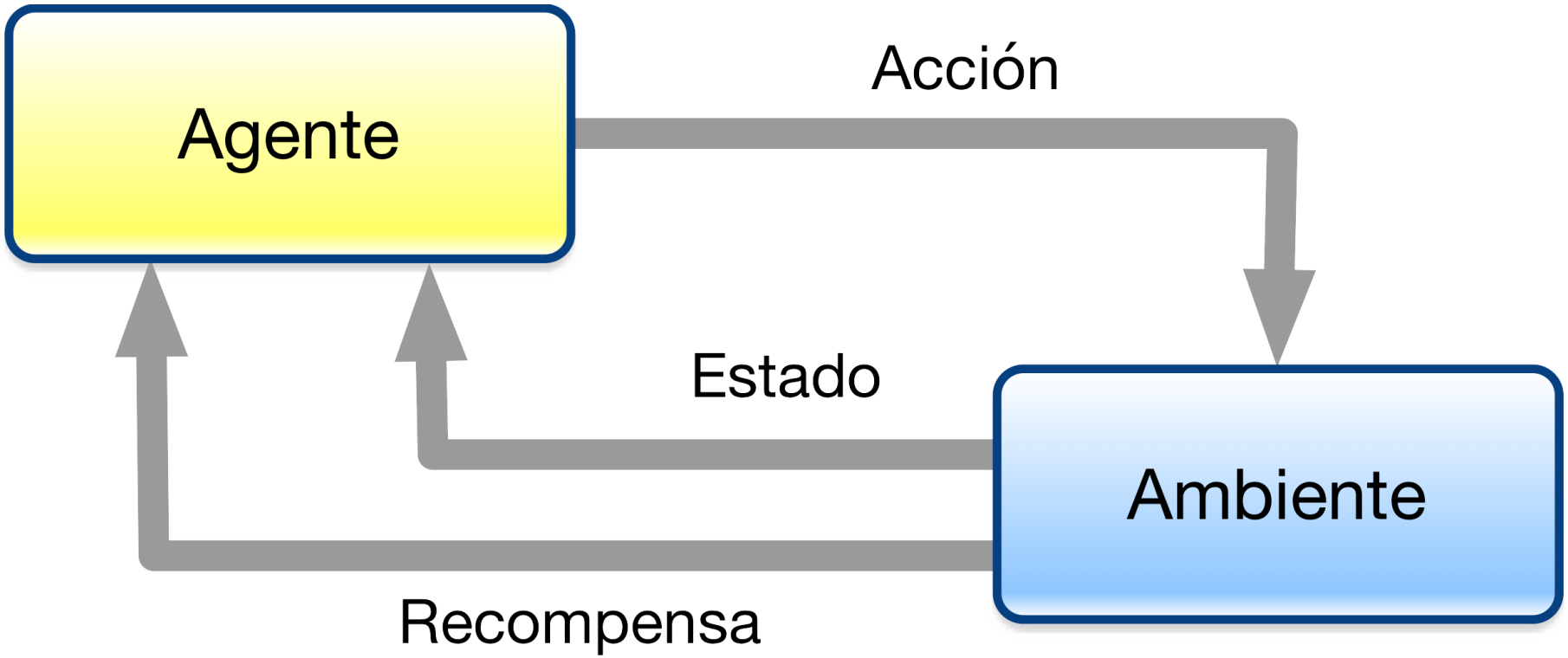
Agrupamiento



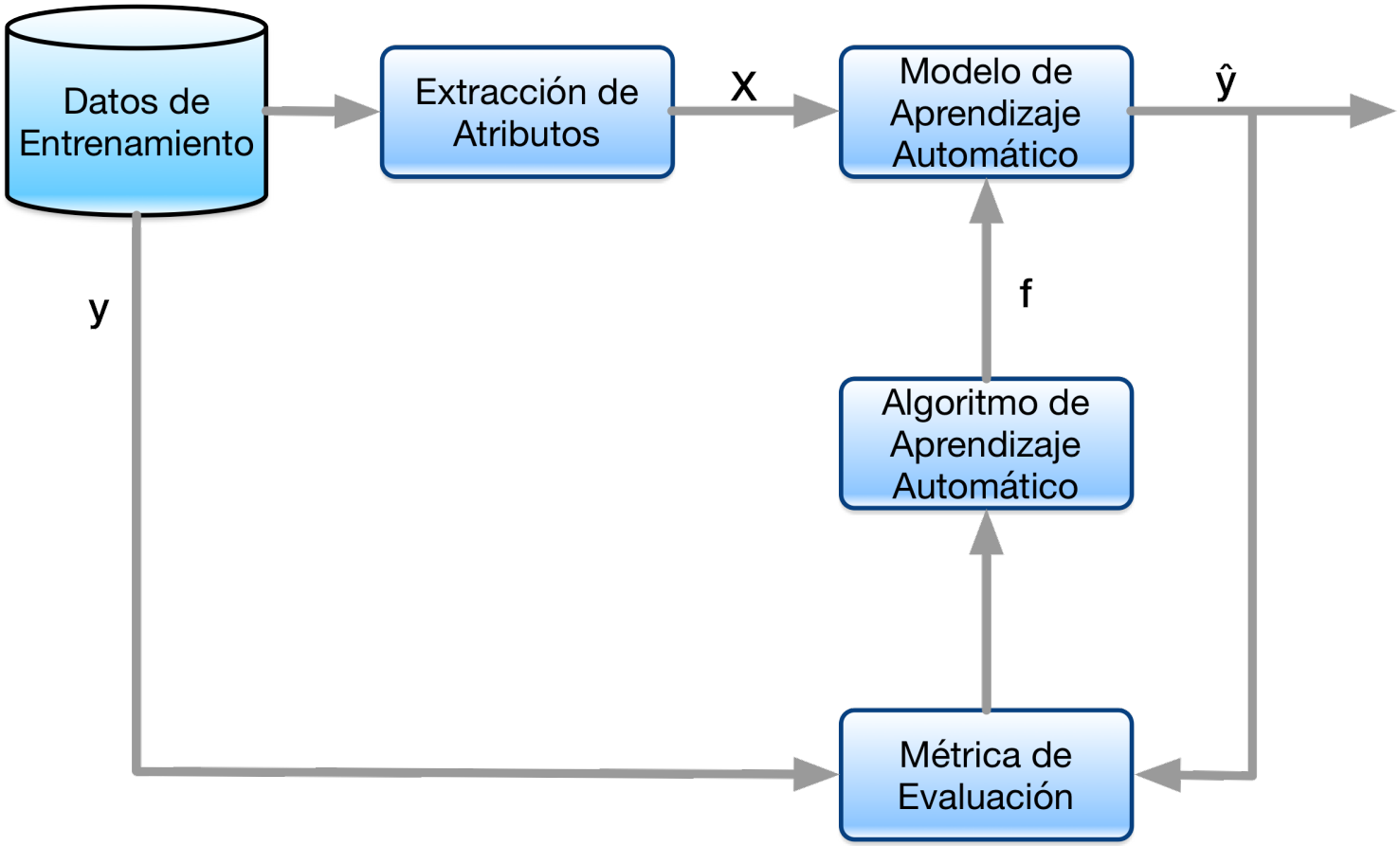
Aprendizaje por Refuerzo

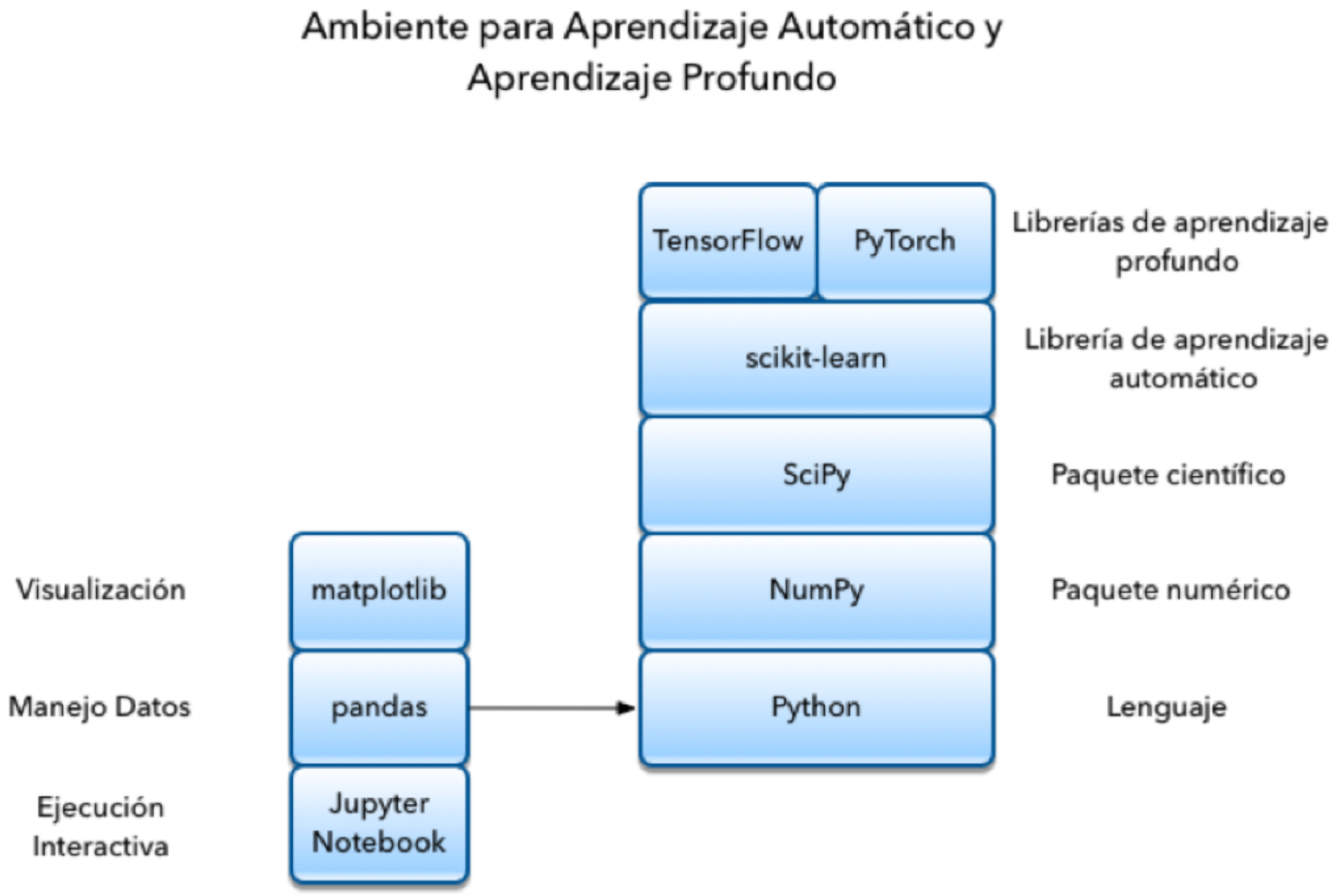
Es un área del aprendizaje automático inspirada en la psicología conductista, cuya ocupación es determinar qué acciones debe escoger un agente de software en un entorno dado con el fin de maximizar alguna noción de "recompensa" o premio acumulado.

Agente - Ambiente



Generación del Modelo de Aprendizaje





Demostración

Ejemplo de Inicio a Fin de un Problema de Aprendizaje Automático

Agenda

- Introducción
- Descripción de datos
- Exploración de datos
- Preparación de datos
- Entrenamiento y evaluación del modelo de aprendizaje automático
- Interpretando el modelo de aprendizaje automático
- Guardar el modelo
- Hacer predicciones con el modelo

Introducción

Usaremos Aprendizaje Automático para predecir si una persona tiene diabetes o no, a partir de la información sobre el paciente, como la presión arterial, el índice de masa corporal (IMC), la edad, etc.

Los datos fueron recopilados y puestos a disposición por el "Instituto Nacional de Diabetes y Enfermedades Digestivas y Renales" como parte de la base de datos de diabetes de los Pima Indians.

Utilizaremos `Python` y algunos de sus populares librerías relacionados con la ciencia de datos. En primer lugar, importaremos `pandas` para leer nuestros datos de un archivo CSV y manipularlos para su uso posterior. También utilizaremos `Numpy` para convertir datos en un formato adecuado para alimentar nuestro modelo de clasificación. Usaremos `seaborn` y `matplotlib` para las visualizaciones. A continuación, importaremos el algoritmo de Regresión Logística de `sklearn`. Este algoritmo nos ayudará a construir nuestro modelo de clasificación. Por último, utilizaremos la librería `joblib` para guardar nuestro modelo para uso futuro.

In [31]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.linear_model import LogisticRegression
import joblib
```

Descripción de los Datos

Tenemos nuestros datos guardados en un archivo CSV llamado `diabetes.csv`. Primero leemos nuestro conjunto de datos a un *dataframe* de `pandas` llamado `diabetes`, y luego usamos la función `head()` para mostrar los primeros cinco registros de nuestro conjunto de datos.

In [32]:
```
diabetes = pd.read_csv('../datos/diabetes.csv')
diabetes.head(10)
```

Out[32]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

- **Pregnancies**: Número de veces embarazada
- **Glucose**: concentración plasmática de glucosa durante 2 horas en una prueba oral de tolerancia a la glucosa
- **BloodPressure**: presión arterial diastólica (mm Hg)
- **SkinThickness**: espesor del pliegue de la piel del tríceps (mm)
- **Insulin**: insulina sérica de 2 horas (mu U / ml)
- **BMI**: índice de masa corporal (peso en kg/(altura en m)^2)
- **DiabetesPedigreeFunction**: función de pedigrí de la diabetes (una función que puntúa la probabilidad de diabetes según el historial familiar)
- **Age**: Edad (años)
- **Outcome**: variable de clase (0 si no es diabético, 1 si es diabético)

También asegurémonos de que nuestros datos estén limpios (no tiene valores nulos, etc.).

In [15]:
```
diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Exploración de datos

Exploremos ahora nuestro conjunto de datos para tener una idea de cómo se ve y obtener algunas ideas al respecto.

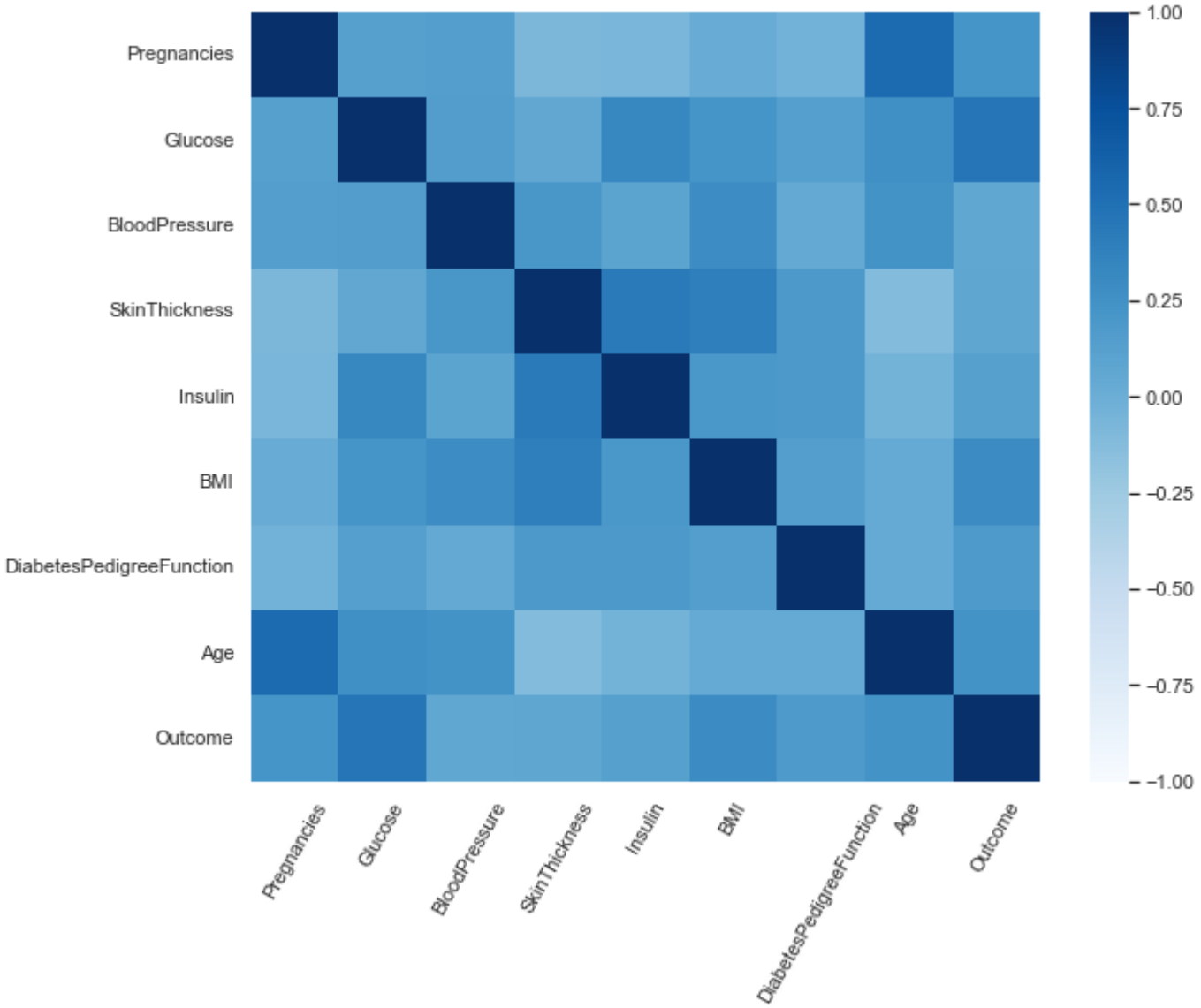
Comencemos por encontrar la correlación de cada par de atributos (y la variable de resultado), y visualicemos las correlaciones usando un mapa de calor.

In [33]:
```
corr = diabetes.corr()
corr
```

Out[33]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

In [18]:
```
sns.set(rc={"figure.figsize": (10, 8)})
ax = sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, cmap='Blues', vmin=-1, vmax=1)
ax.set_xticklabels(ax.get_xticklabels(),rotation = 60);
```

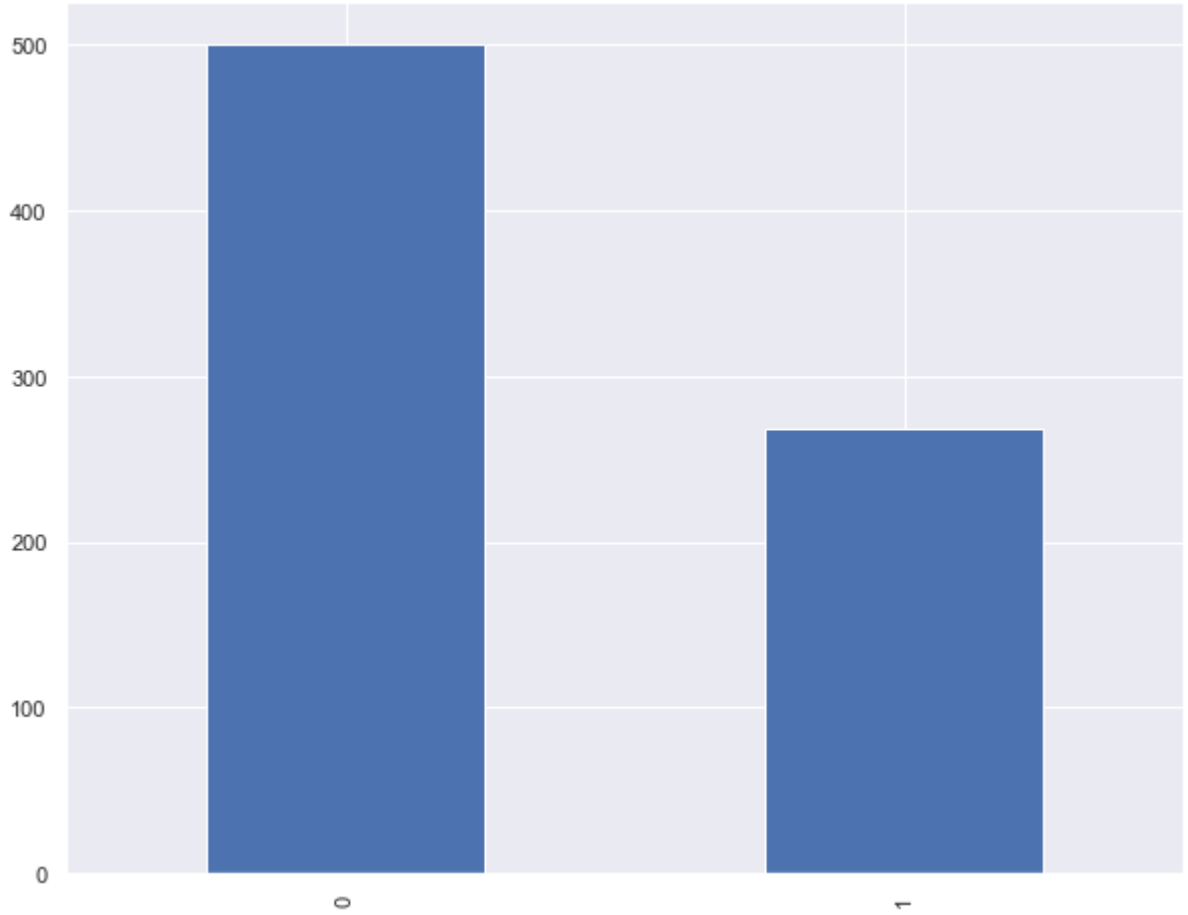


En el mapa de calor anterior, los colores más oscuros indican una mayor correlación. Como podemos ver en la tabla y en el mapa de calor, los niveles de glucosa, la edad, el IMC y el número de embarazos tienen una correlación significativa con la variable de resultado. También observe la correlación entre los pares de atributos, como la edad y los embarazos, o la insulina y el grosor de la piel.

Observemos también cuántas personas en el conjunto de datos son diabéticas y cuántas no. A continuación se muestra un gráfico de barras:

In [19]:
```
diabetes.Outcome.value_counts().plot.bar()
```

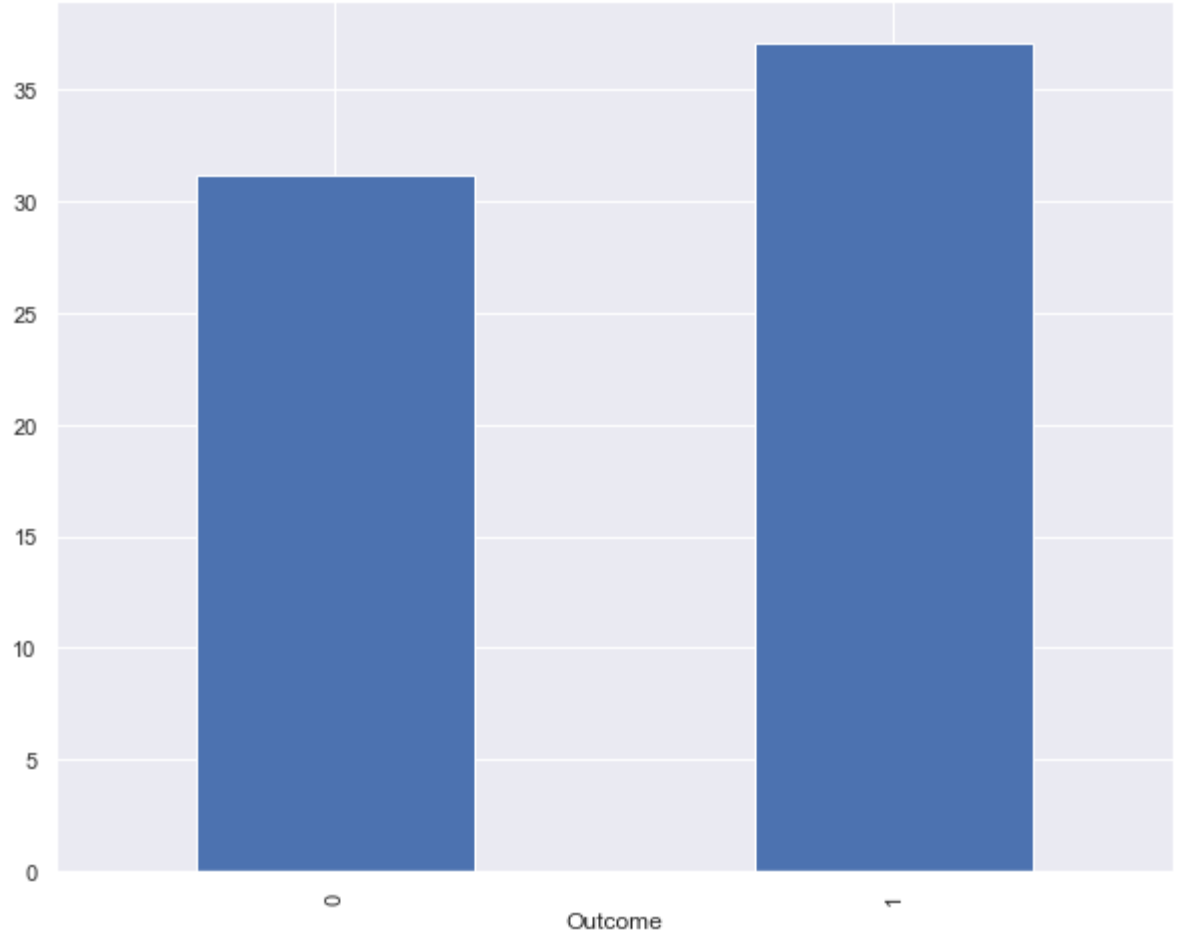
Out[19]: <AxesSubplot:>



También es útil visualizar las relaciones entre una sola variable y el resultado. A continuación, veremos la relación entre `Age` y `Outcome`.

In [7]:
```
diabetes.groupby("Outcome")["Age"].mean().plot.bar()
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1a25ef4bd0>



Preparación del Conjunto de Datos

Al usar algoritmos de aprendizaje automático, siempre debemos dividir nuestros datos en un conjunto de entrenamiento y un conjunto de prueba. Usaremos la función `train_test_split` de sklearn para dividir el conjunto de datos. Dividiremos el conjunto datos en 80% para el conjunto de entrenamiento y 20% para el conjunto de prueba.

Primero separaemos los atributos y la salida del conjunto de datos.

In [20]:
```
from sklearn.model_selection import train_test_split
salida = diabetes.Outcome
atributos = diabetes.drop('Outcome', 1)
X_entrenamiento, X_prueba, y_entrenamiento, y_prueba = train_test_split(atributos, salida,
                                                                        test_size = 0.2,
                                                                        random_state=42)

X_entrenamiento.head()
```

```
C:\Users\wladi\AppData\Local\Temp\ipykernel_22404\233928204.py:3: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
  atributos = diabetes.drop('Outcome', 1)
```

Out[20]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
60	2	84	0	0	0	0.0	0.304	21
618	9	112	82	24	0	28.2	1.282	50
346	1	139	46	19	83	28.7	0.654	22
294	0	161	50	0	0	21.9	0.254	65
231	6	134	80	37	370	46.2	0.238	46

Como paso final antes de usar el aprendizaje automático, normalizaremos nuestras entradas. Los modelos de Aprendizaje Automático a menudo se benefician sustancialmente de la normalización de las entradas. También nos facilita la comprensión de la importancia de cada atributo más adelante, cuando observemos los pesos del modelo. Normalizaremos los datos de modo que cada atributo tenga media 0 y desviación estándar de 1.

Usaremos `StandardScaler` de sklearn para normalizar los datos.

In [21]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(X_entrenamiento)
X_entrenamiento = sc.transform(X_entrenamiento)
X_prueba = sc.transform(X_prueba)
```

Entrenamiento y Evaluación del Modelo de Aprendizaje Automático

Ahora podemos entrenar nuestro modelo de clasificación. Usaremos un modelo de aprendizaje automático simple llamado regresión logística. Dado que el modelo está disponible en `sklearn`, el proceso de entrenamiento es bastante fácil y podemos hacerlo en pocas líneas de código. Primero, creamos una instancia llamada `Modelo_Diabetes` y luego usamos la función `fit` para entrenar el modelo.

In [22]:
```
Modelo_Diabetes = LogisticRegression()
Modelo_Diabetes.fit(X_entrenamiento, y_entrenamiento)
```

Out[22]:
```
LogisticRegression()
```

A continuación, utilizaremos nuestros datos de prueba para conocer la exactitud del modelo.

In [23]:
```
exactitud = Modelo_Diabetes.score(X_prueba, y_prueba)
print("exactitud = ", exactitud * 100, "%")
```

```
exactitud =  75.32467532467533 %
```

Interpretación del Modelo

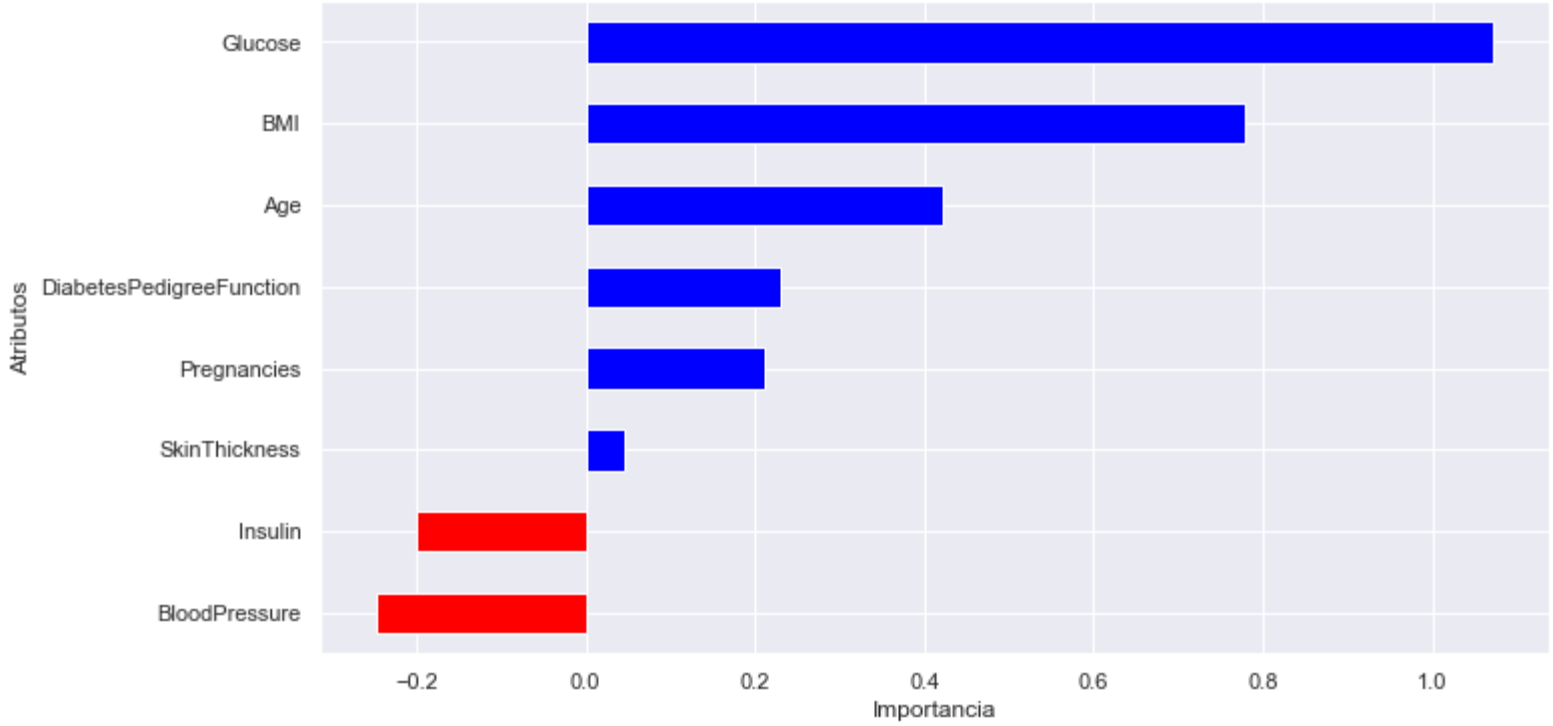
Para tener una mejor idea de lo que está sucediendo dentro del modelo de regresión logística, podemos visualizar cómo nuestro modelo usa los diferentes atributos y qué atributos tienen un mayor efecto.

In [24]:
```
coeficientes = list(Modelo_Diabetes.coef_[0])
etiquetas = list(atributos.columns)
importancia = pd.DataFrame()
importancia['Atributos'] = etiquetas
importancia['importancia'] = coeficientes
importancia.sort_values(by=['importancia'], ascending=True, inplace=True)
importancia['positiva'] = importancia['importancia'] > 0
importancia.set_index('Atributos', inplace=True)
importancia
```

Out[24]:

Atributos	importancia	positiva
BloodPressure	-0.247842	False
Insulin	-0.200827	False
SkinThickness	0.045697	True
Pregnancies	0.212558	True
DiabetesPedigreeFunction	0.230607	True
Age	0.421002	True
BMI	0.778152	True
Glucose	1.071132	True

In [25]:
```
importancia.importancia.plot(kind='barh', figsize=(11, 6),color = importancia.positiva.map({True: 'blue', False: 'red'}))
plt.xlabel('Importancia');
```



Algunas Conclusiones:

- El nivel de glucosa, el BMI, la edad y la función de pedigrí de la diabetes tienen una influencia significativa en el modelo, especialmente el nivel de glucosa y el BMC. ¡Es bueno ver que nuestro modelo de aprendizaje automático coincida con lo que hemos estado escuchando a los médicos durante toda nuestra vida!
- La presión arterial tiene una influencia negativa en la predicción, es decir, una mayor presión sanguínea se correlaciona con una persona que no es diabética.
- Aunque la edad estuvo más correlacionada que el BMC con la variable de salida (como vimos durante la exploración de datos), el modelo se basa más en el BMC. Esto puede suceder por varias razones, incluido el hecho de que la correlación capturada por edad también es captada por alguna otra variable, mientras que la información capturada por BMC no es capturada por otras variables.

Guardar el Modelo

Ahora guardaremos nuestro modelo entrenado para uso futuro usando `joblib`

In [26]:
```
joblib.dump(Modelo_Diabetes, 'Modelo_Diabetes.pkl')
```

Out[26]:
```
['Modelo_Diabetes.pkl']
```

Para verificar si hemos guardado el modelo correctamente o no, usaremos nuestros datos de prueba para verificar la precisión de nuestro modelo guardado

In [27]:
```
Modelo = joblib.load('Modelo_Diabetes.pkl')
exactitud_modelo = Modelo.score(X_prueba, y_prueba)
print("exactitud = ", exactitud_modelo * 100,"%")
```

```
exactitud =  75.32467532467533 %
```

A continuación mostraremos tres ejemplos:

- Predicción correcta (no diabetes)
- Predicción correcta (diabetes)
- Predicción incorrecta

In [28]:
```
muestra = X_prueba[0,:].reshape(1, -1)
probabilidad_prediccion  = Modelo.predict_proba(muestra)
prediccion = Modelo.predict(muestra)
print('Probabilidad:', probabilidad_prediccion)
print('Prediction:', prediccion)
print('Salida Real:', y_prueba.iloc[0])
```

```
Probabilidad: [[0.72266096 0.27733904]]
Prediction: [0]
Salida Real: 0
```

In [29]:
```
muestra = X_prueba[18,:].reshape(1, -1)
probabilidad_prediccion  = Modelo.predict_proba(muestra)
prediccion = Modelo.predict(muestra)
print('Probabilidad:', probabilidad_prediccion)
print('Prediction:', prediccion)
print('Salida Real:', y_prueba.iloc[18])
```

```
Probabilidad: [[0.14221802 0.85778198]]
Prediction: [1]
Salida Real: 1
```

In [30]:
```
muestra = X_prueba[15,:].reshape(1, -1)
probabilidad_prediccion  = Modelo.predict_proba(muestra)
prediccion = Modelo.predict(muestra)
print('Probabilidad:', probabilidad_prediccion)
print('Prediction:', prediccion)
print('Salida Real:', y_prueba.iloc[15])
```

```
Probabilidad: [[0.61368987 0.38631013]]
Prediction: [0]
Salida Real: 1
```

Referencias:

End-to-End Example: Using Logistic Regression for predicting Diabetes