

Aprendizaje Automático y Minería de Datos

Clasificador de Pokémons

Diego Baratto Valdivia

Leonor Cuesta Molinero

Contents

[SVM: 2](#_Toc29732102)

[Descripción del Proyecto: 2](#_Toc29732103)

[Resultados Obtenidos: 2](#_Toc29732104)

[Conclusiones: 2](#_Toc29732105)

[Regresión logística: 2](#_Toc29732106)

[Descripción del Proyecto: 2](#_Toc29732107)

[Resultados Obtenidos: 2](#_Toc29732108)

[Conclusiones: 2](#_Toc29732109)

[Redes Neuronales: 2](#_Toc29732110)

[Descripción del Proyecto: 2](#_Toc29732111)

[Clasificador de Legendarios: 2](#_Toc29732112)

[Predecir el tipo: 2](#_Toc29732113)

[Resultados Obtenidos: 2](#_Toc29732114)

[Conclusiones: 3](#_Toc29732115)

# SVM:

## Descripción del Proyecto:

Haciendo uso de las SVM (Support Vector Machine), se ha creado un clasificador de pokémon legendarios en función de las características elegidas por el usuario.

Se ha tomado como referencia la práctica realizada en clase sobre SVM, modificando aquellos fragmentos de códigos necesarios para lograr el objetivo, por ejemplo, dividir los datos en tres grupos diferentes (entrenamiento, validación y testeo) para realizar *cross-validation* (evitando así el sobreajuste) y elección de los parámetros C y sigma.

Se ha añadido también soporte de introducción de datos del usuario, es decir, tras el entrenamiento de la SVM, es posible testear la solución incluyendo por consola nuevos pokémon que no se encuentren en el grupo de los datos utilizados para entrenar.

## Resultados Obtenidos:

A continuación se incluyen los diferentes resultados obtenidos, así como imágenes de apoyo y explicaciones al pie de las mismas.

De las primeras pruebas que se realizaron fue con kernel lineal, útil si el número de atributos (n) es grande comparado con el número de ejemplos (m):

|  |  |
| --- | --- |
|  | Kernel lineal entrenado con los atributos *base egg steps* y *capture rate*. En este caso, al ser atributos diferenciados entre ellos, el kernel lineal funciona de manera aceptable. |
|  | Kernel lineal entrenado con los atributos *base happiness* y *attack*. En este caso, al ser atributos linealmente no diferenciables entre ellos, el kernel lineal no funciona de manera aceptable. |
| C:\Users\Usuario\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lineal3.png | Kernel lineal entrenado con los atributos *speed* y *weight\_kg*. En este caso, al ser atributos linealmente no diferenciables, el kernel lineal no funciona de manera aceptable. |

En todos los casos con más de un atributo, el kernel lineal no funciona (*score* entre 0.4 y 0.6) debido a que el número de casos de entrenamiento es mayor que el número de atributos.

Tras estas pruebas, se decidió usar el kernel Gaussiano, apto para un número de atributos pequeño y un número de casos de ejemplo intermedio.

|  |  |
| --- | --- |
|  | Kernel Gaussiano entrenado con los atributos *base egg steps* y *capture rate*. |
|  | Kernel Gaussiano entrenado con los atributos *base happiness* y *attack*. |
|  | Kernel Gaussiano entrenado con los atributos *speed* y *weight\_kg*. |
|  | Kernel Gaussiano entrenado con los atributos *attack* y *defense*. |

Gracias a la visualización de estas gráficas, se puede observar cuales son los atributos que mejor definen a los pokémon legendarios y cuales ensucian el aprendizaje automático de la svm. A continuación se muestra la elección de los parámetros C y sigma con los atributos que mejor definen a los legendarios, *capture rate* y *base egg steps*:

## Conclusiones:

# Regresión logística:

## Descripción del Proyecto:

## Resultados Obtenidos:

## Conclusiones:

# Redes Neuronales:

## Descripción del Proyecto:

Utilizando redes neuronales, hemos creado tanto un clasificador de legendarios en función de sus otras características como un predictor del tipo de Pokémon que es en función de su relación con otros tipos y características.

### Clasificador de Legendarios:

Tomando como referencia distintas características de entre *[attack, base\_egg\_steps, base\_happiness, base\_total, capture\_rate, defense, experience\_growth, height\_m, hp, percentage\_male, sp\_attack, sp\_defense, weight\_kg]*.

Se ha utilizado como referencia la práctica de laboratorio basada en redes neuronales, así como el módulo de Python “Keras” con “Tensorflow”.

En este clasificador, entrena una red neuronal dividida en grupos de entrenamiento, validación y testeo, cogiendo ejemplos del grupo total de forma aleatoria y barajándose para evitar que los legendarios y no legendarios se agrupen al principio o final de los grupos.

Para tratar de asegurar el mejor *score* de entrenamiento, se repite el proceso un número establecido de veces y guarda el mejor.

### Predecir el tipo:

## Resultados Obtenidos:

## Conclusiones: