

Aprendizaje Automático y Minería de Datos

Clasificador de Pokémons

Diego Baratto Valdivia

Leonor Cuesta Molinero

Contents

[1. Support Vector Machine (SVM): 2](#_Toc30266996)

[1.1. Descripción del Proyecto: 2](#_Toc30266997)

[1.2. Resultados Obtenidos: 2](#_Toc30266998)

[1.2.1. Clasificador de legendarios 2](#_Toc30266999)

[1.3. Conclusiones: 5](#_Toc30267000)

[2. Regresión logística: 6](#_Toc30267001)

[2.1. Descripción del Proyecto: 6](#_Toc30267002)

[2.2. Resultados Obtenidos: 6](#_Toc30267003)

[2.3. Conclusiones: 6](#_Toc30267004)

[3. Redes Neuronales: 6](#_Toc30267005)

[3.1. Descripción del Proyecto: 6](#_Toc30267006)

[3.1.1. Clasificador de Legendarios: 6](#_Toc30267007)

[3.1.2. Predecir el tipo: 7](#_Toc30267008)

[3.2. Resultados Obtenidos: 7](#_Toc30267009)

[3.3. Conclusiones: 7](#_Toc30267010)

# 1. Support Vector Machine (SVM):

## 1.1. Descripción del Proyecto:

Haciendo uso de las SVM, 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.

Además de la clasificación anterior, se ha tratado de realizar clasificación multi-clase de los diferentes tipos de pokémon, entre ellos agua, tierra, roca, entre otros, en base a las diferentes características proporcionadas en el conjunto de datos. Se ha hecho uso de la misma metodología que en el caso anterior, añadiendo c*ross-validation*, elección de los mejores parámetros C y sigma y adición de nuevas características polinomiales. Debido a las complicaciones a la hora de elegir las características más representativas para la clasificación, se ha utilizado el paquete de Python denominado como Boruta, el cual ha ayudado a discriminar aquellos datos más importantes a la hora de la clasificación, mediante los *Random Forest* (EXPLICACION AQUI).

## 1.2. Resultados Obtenidos:

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

### 1.2.1. Clasificador de Legendarios

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):

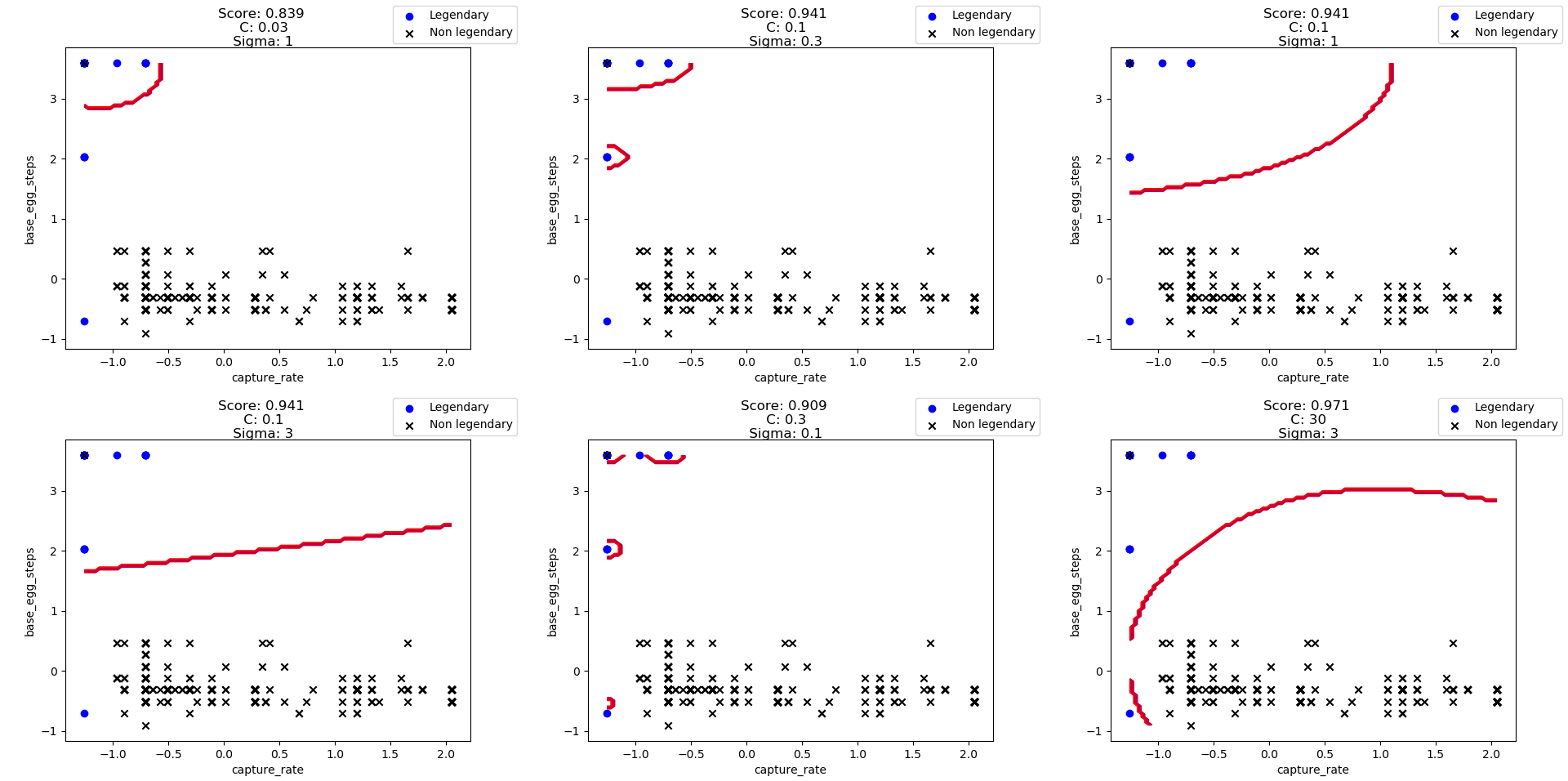
|  |  |
| --- | --- |
|  | Kernel lineal entrenado con los atributos *base egg steps* y *capture rate*. En este caso, al ser atributos diferenciados entre ellos, el kernel lineal funciona de manera aceptable. |
|  | Kernel lineal entrenado con los atributos *base happiness* y *attack*. En este caso, 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*:



Se puede observar como varia la frontera de decisión de la SVM dependiendo de los valores que reciba C y sigma. Se aprecia de manera detallada el sobreajuste que se da en, por ejemplo, con C = 0.3 y sigma = 0.1.

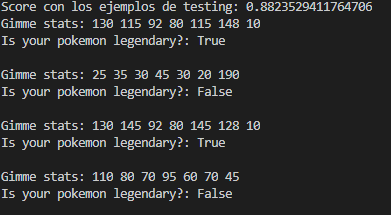
### 1.2.1. Clasificador de tipos

A pesar de haber hecho uso de Boruta para elegir las características más importantes para la clasificación, no se consigue una media armónica mayor a 0.5 en cada tipo. Dentro de los mismos, se observa mucha diferencia de precisión en la predicción entre ellos.

#TODO: meter imagenes

## 1.3. Conclusiones:

La técnica empleada con las SVM funciona de manera precisa y evita el *overfitting* / underfitting correctamente. Tras proporcionarle a los resultados de entrenamiento diferentes pokémon nunca vistos por ella, se logran unos resultados acertados y precisos, además de usar un conjunto de testeo como se ha dicho anteriormente para obtener el *score* del mismo (media armónica), logrando una media de 0.85 en dicho *score*. A continuación, se proporciona una captura de pantalla donde se consulta los datos (*attack*, *defense*, *hp*, *sp\_attack*, *sp\_defense*, *speed*, *capture\_rate*) de cuatro pokémon diferentes de la octava generación (Zacian, Snom, Zamazenta, Flapple), la cual no se encuentra en el *dataset*, siendo el primero y el tercero legendarios, mientras que el segundo y el cuarto no.



**TODO: RELLENAR CON LAS CONCLUSIONES DE LAS SVM PARA LOS TIPOS DE POKÉMON**

# 2. Regresión logística:

## 2.1. Descripción del Proyecto:

## 2.2. Resultados Obtenidos:

## 2.3. Conclusiones:

# 3. Redes Neuronales:

## 3.1. 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.

### 3.1.1. 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.

### 3.1.2. Predecir el tipo:

## 3.2. Resultados Obtenidos:

## 3.3. Conclusiones: