**Práctica 2 de Aprendizaje Computacional (Modelos de clasificación)**

Adrià Orozco Lorente 1490952 y Arnau Cruz Gargallo 1494996

**Resumen**

Durante el desarrollo de esta práctica se observa la aplicación de modelos de clasificación haciendo énfasis en la aplicación de diferentes clasificadores utilizando la regresión logística i las máquinas de vectores de soporte, aunque en nuestro caso también usaremos el knn explicado posteriormente en momento de su utilización de manera que haciendo uso de estos tres clasificadores se entiendan las mejoras que producen los kernels. También se busca evaluar correctamente los errores del modelo visualizando los datos i por supuesto el modelo resultante.

Por otra parte se busca la capacidad de aplicar técnicas de clasificación en casos reales, validando los resultados en datos reales i fomentando la capacidad para presentar resultados técnicos de aprendizaje computacional de forma adecuada delante de otras personas

**Palabras clave**

Clasificador, regresión logística, svm, knn, kernel, modelo, algoritmo

**Abstract en ingles**

**Palabras clave en ingles**

**Introducción**

Durante el desarrollo de esta práctica y bien como se ha explicado en el resumen se busca como principal objetivo la aplicacions de diferentes modelos de clasificación por tal de evaluar su error i visualizar los datos y el modelo resultante. Por otra parte se procede a realizar estas técnicas de clasificación en casos reales valiadndo además los resultados a partir de los datos de estos propios casos.

Esto se dividirá en dos apartados (B y A), donde en el primero de ellos se realizara una comparativa de modelos de una base de datos preestablecida de sklearn donde se aplicaran varios modelos de clasificación y se mostraran los resultados por tal de realizar una comparativa de modelos y de los resultados con la aplicación de funciones kernel.

Para el segundo apartado, se realizara un estudio completo de una base de datos asignada (en nuestro caso una base de datos referetnte a predicciones de lluvia en Australia) a partir de la cual se debe realizr un exhaustivo estudio donde se analicen todos los datos, estableciendo una variable objetivo y unas variables predictoras, para posteriormente preparar estos datos para completar un problema de clasificación.

**Apartado B**

Para el apartado B se escoge la base de datos breast\_cancer de sklearn para posteriormente realizar varios modelos de clasificación sobre ella y evaluar el comportamiento de estos, además del valor sobre el error que obtenemos en cada caso por tal de realizar un análisis exhaustivo de cada modelo de clasificación.

Primeramente se crea archivo jupyter notebook que cargara dicha base de datos juntamente a las librerías necesarias para aplicar los algoritmos y donde se procederá a realizar el estudio.

Así, de esta manera previamente a la carga de la base de datos comenzamos importando las siguientes librerías:

**Sklearn:** Esta librería sirve para realizar el aprendizaje automático, y nos ayudara a realizar el análisis predictivo. Incluye varios algoritmos de clasificación, regresión y análisis de grupos.

**Numpy:** Esta librería nos da soporte para crear vectores y matrices de una dimensión considerable y multidimensional, está creada precisamente para procesar grandes cantidades de datos de la forma más óptima posible y contiene una larga colección de funciones matemáticas de alto nivel para operar con ellas.

**Pandas:** Esta librería es una extensión de la librería de numpy, hecha para realizar la manipulación y el análisis de datos en python, ofreciéndonos así estructuras de datos y operaciones para manipular tablas numéricas y series temporales, de manera que con esta librería podremos importar de forma sencilla nuestra base de datos.

**Matplotlib:** Esta librería consigue la generación de gráficos a partir de datos contenidos en listas, los cuales habrán sido importados en nuestro caso previamente con uso de las funciones de la librería pandas, de manera que podamos representar los datos deseados de manera gráfica usando varios tipos de técnicas.

**Scipy:** Esta librería nos será útil para aplicar módulos de optimización, algebra lineal, e integración de datos entre otras, cabe destacar que es parte del conjunto de la biblioteca numpy y extiende bibliotecas de computación científica.

**Seaborn:** Esta librería consiste en ofrecer herramientas para la visualización de datos para Python desarrollada sobre matplotlib ofreciendo una interfaz de alto nivel para la creación de gráficas para la visualización del resultado.

Con todo esto ya se puede cargar la base de datos y realizar un análisis de datos básicos para conocer la estructura y sus características

Observando las características de la base de datos cargada, podemos observar que la variable objetivo es un valor binario (0 si es maligno, 1 si es benigno) por el cual se intentaran clasificar cada muestra, de manera que se intenta predecir si un elemento x, pertenece a la clase 0 1.

Para la realización de los modelos de clasificación se realizaran la regresión logística, svc y knn.

La regresión logística es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica (una variable que puede adoptar un número limitado de categorías) en función de las variables independientes o predictoras.

En el caso de svc para sklearn es equivalente a una máquina de vector de soporte son un conjunto de algoritmos de aprendizaje automático supervisado que producen varias predicciones en base a cada modelo y se juntan para obtener una única predicción. Hay que tener en cuenta que se pueden usar varios tipos de función kernel con objetivo de representar los datos en dimensiones superiores a la original de forma óptima y sin necesidad de tener que calcular los datos en un espacio superior, de forma que se realizan también comparaciones por los resultados con diversos tipos de funciones kernel.

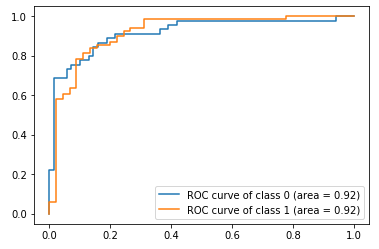
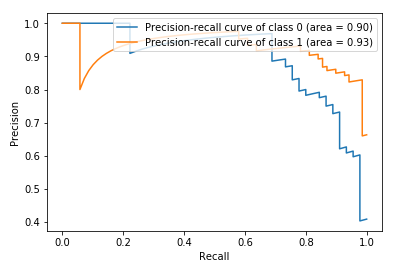
El knn es un método de clasificación supervisado que sirve para estimar la función de densidad F(x/Cj) de las predictoras x por cada clase Cj. Este es un método de clasificación no paramétrico, que estima el valor de la función de densidad de probabilidad o directamente la probabilidad a posteriori de que un elemento x pertenezca a la clase Cj a partir de la información proporcionada por el conjunto de prototipos.

Conociendo esto y los atributos que forman nuestra base de datos comenzamos aplicando una regresión logística y un algoritmo de clasificación de vectores de soporte con kernel rbf por varias particiones de los datos de entrada y observamos la correcta clasificación de estos por tal de hacer la primera comparación. Destacar antes que el kernel rbf es el kernel predeterminado utilizado dentro de las máquinas de vectores de soporte y es el más popular entre ellos.

Como primeras observaciones se debe comentar que se están utilizando pequeños porcentajes de la estructura de datos y se están teniendo en cuenta los dos primeros atributos para realizar el estudio por lo tanto puede afectar tanto al desempeño del kernel como a la evaluación del resultado causante de los datos obtenidos para el entrenamiento.

Con esto obtenemos que el mayor porcentaje de acierto lo obtiene el regresor logístico por un 0,5% de datos consiguiendo así un 89% de acierto con diferencia de 5% sobre SVM, a medida que aumentamos el porcentaje se ve la afectación en los resultados mínimamente debido a la inclusión ya sea de muestras difícilmente clasificables, outliers e incluso que estas muestras sean más fácilmente separables en un espacio dimensional superior al que ya teníamos cosa que consigue realizar SVM mediante la función kernel. Así que para el 0,7% de los datos los resultados en ambos casos empeoran cerca de un 3-5, pero añadiendo una 0,1 por ciento extra a los datos para el modelo, la máquina de vectores de soporte es capaz de mejorar su rendimiento igualando a la regresión logística.

Para continuar con la comparativa de modelos podemos considerar todas las clases en conjunto en una sola curva (micro-averaging) por tal de visualizar gráficamente los resultados entre ambos modelos, además de poder representar su curva ROC la cual nos da la representación de la razón o proporción de verdaderos frente a la razón o proporción de falsos positivos según se varía un umbral de discriminación (valor a partir del cual decidimos que un caso es un positivo).



Observando estos casos podemos ver la compensación entre la precisión y exhaustividad para diferentes umbrales, de manera que para el primero se observa hasta un 0.7 de exhaustividad aproximadamente un alto grado de precisión y exhaustividad relacionado así con la tasa baja de falsos positivos y una tasa baja de falsos negativos. A partir de este valor hacia arriba se observa sobretodo un incremento de falsos negativos.

Para la curva ROC podemos observar una correcta proporción de verdaderos y falsos positivos indicándonos que nuestro modelo tiene una alta precisión para los valores de entrada que ha obtenido siendo esta curva proporcional al número de datos registrado, ya que a medida que aumentan los falsos negativos siguen estando los valores de verdaderos positivos muy altos para ambos modelos (Regresor logístico y svm).

Con este análisis de ambas gráficas, se procede a la realización del modelo svm con varios tipos de funciones kernel y variaciones entre las variables slack.

Para explicar la afectación del kernel en las SVM Se tiene que tener en cuenta que en muchos casos los grupos a clasificar no serán linealmente separables en el espacio original, por lo tanto una solución que ofrece la Maquina de vectores de soporte es aumentar la dimensión de los datos, la cual se puede transformar combinando o modificando cualquiera de sus dimensiones. Para hacerlo se utiliza el kernel que se una función que devuelve el resultado del producto entre dos vectores realizado en un nuevo espacio dimensional diferente al espacio original en el que se encontraban.

De forma que nos permite operar en el espacio de características original sin calcular las coordenadas de los datos en un espacio de mayor dimensión, ofreciéndonos en esencia una forma más eficiente y menos costosa de transformar los datos en dimensiones más altas.

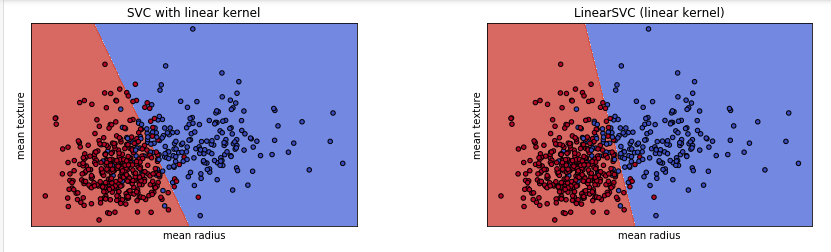
Además el uso de variables slack nos sirve en casos en los que los datos del mundo real están desordenados y casi siempre habrá algún caso que el clasificador no puede acertar, puesto que realizar una separación perfecto no siempre se posible, y en el supuesto de que lo sea, el resultado del modelo puede no ser generalizado por otros datos (overfitting). Por lo que para solucionar este problema y permitir cierta flexibilidad las svm utilizan un parámetro C que controla la compensación entre errores de entrenamiento y los márgenes rígidos creando así un soft-margin que permite algunos errores en la clasificación a la vez que los penaliza.

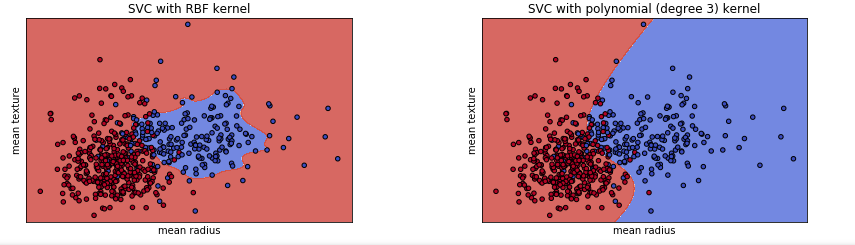
Cuando esta C es pequeña, los errores de clasificación tienen menos importancia y el enfoque se encuentra en maximizar el margen, mientras que cuando C es grande, el enfoque se encuentra en evitar la clasificación errónea a expensas de mantener el margen pequeño.

Trata de un compromiso, obtener un mejor clasificador y más robusto a expensas de un margen amplio.

Conociendo esto se aplica el svc con kernel lineal, con rbf y polinomial con grado 3 ademas de un svc lineal con el mismo kernel, sobre el conjunto total de datos teniendo en cuenta las mismas variables que en los modelos anteriores.

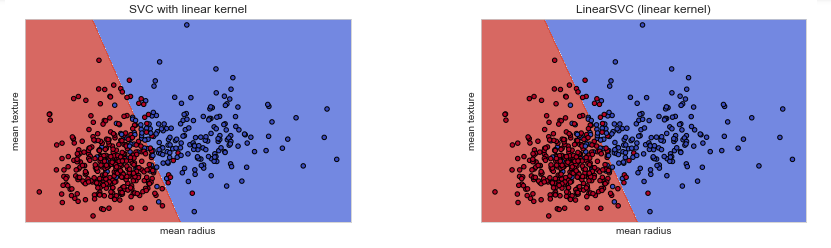
Como primer paso establecemos 0,1 como valor para la variable slack haciendo asi que los errores tengan menos importancia y el enfoque se encuentre en maximizar el margen de manera que obtenemos los siguientes resultados:

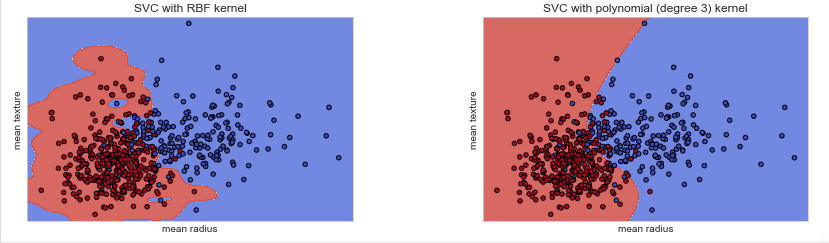




Para comenzar el primer análisis entre modelos cabe destacar que svc con kernel lineal y svc lineal con el mismo kernel producen resultados similares aunque en este caso parece que linearSVC nos ajusta un poco más el límite de decisión de manera que permite menos errores y los datos están mejor clasificados. En caso del kernel RBF el cual es el más popular y más usado, en general vemos que con un valor de la variable slack bajo obtenemos unos límites de decisión muy amplios de manera que los puntos más dispares en el grafico no son correctamente clasificados. Y para el SVC con kernel polinomial vemos que la curva de decisión se ajusta correctamente a los datos a clasificar permitiendo algunos errores ya que todavía el margen es muy amplio.

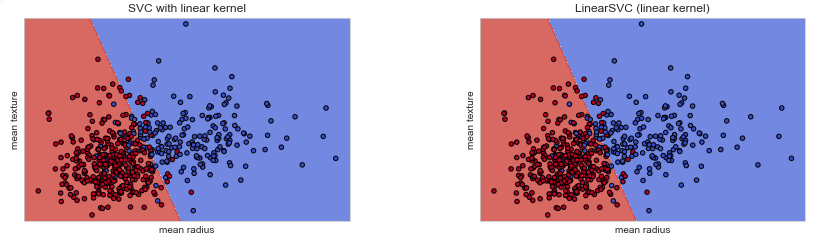
Ahora veremos qué es lo que ocurre si nos concentramos en minimizar el margen y evitar la clasificación errónea de los datos optando por un valor de C elevado, en este caso 5.

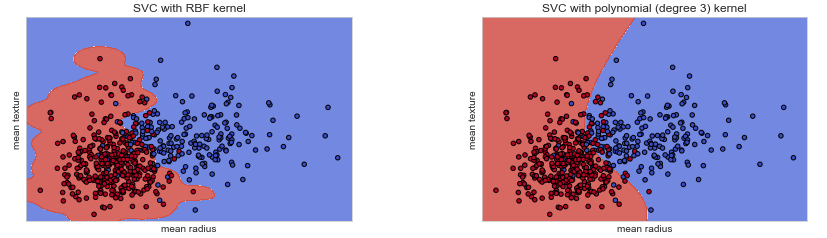




Ahora, se debe tener en cuenta que se busca evitar una clasificación incorrecta a costa de mantener márgenes todavía más estrictos, (cabe destacar que en estos casos puede ocurrir el overfitting es decir que el algoritmo utilizado no generalice correctamente haciendo casi una memorización de la solución incluso modelando el ruido de esta).

En este caso y comparando los resultados gráficamente entre los modelos similares con la varianza del coeficiente C, observamos que para el modelo SVC con kernel lineal no apreciamos diferencias significativas, mientras que para el LinearSVC que previamente parecía ligeramente una mejor opción, ahora clasifica mejor la parte de datos que están más condensados y es más difícil realizar una correcta clasificación pero eso ocurre a expensas de producir algún error más en valores que antes predecía bien, la solución ante esto sería equivalente a usar una función polinomial ya que una recta no es capaz de modelar la curva natural de los datos y cómo podemos observar al aumentar este coeficiente y usar el kernel polinomial, aquellos valores que se encontraban en los límites de decisión están ahora correctamente clasificados y además se producen mayores aciertos en los valores que son más difíciles de clasificar aumentando ligeramente la precisión de este modelo. Por ultimo con el kernel RBF el cual como ya se ha comentado es el más popular entre todos, obtenemos una gran mejora del algoritmo consiguiendo muy pocos errores y siendo capaz de modelar incluso parte de los outliers de los datos, cosa que puede indicar overfitting. Para corregir esto vamos a probar con un valor mas bajo de la variable slack de manera que no clasifique los outliers y que sea capaz de modelar correctamente los demás datos, y gracias a esto obtenemos que con un valor de C equivalente a 0,9. Donde vemos que seguimos manteniendo en cada uno de los modelos los menores errores de clasificación.





**APARTADO A**

En este apartado se muestra como se realiza un modelo de clasificación a partir de un conjunto real de datos equivalente a datos sobre la lluvia en Australia los cuales se analizan de forma exhaustiva y se tratan por tal de realizar un algoritmo lo más preciso y completo posible

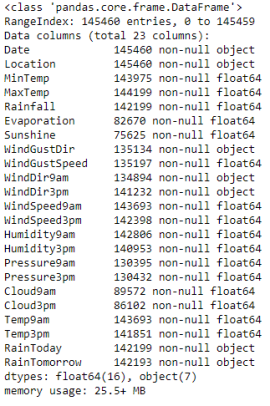
**EDA:**

Para comenzar con nuestro modelo se debe examinar y entender como esta estructurada nuestra base de datos, para ello podemos utilizar funciones de la librería pandas con objetivo de ver los atributos que esta contiene, las relaciones entre ellos, el tipo de variables que existen y así poder comenzar a realizar las primeras hipótesis sobre las variables que debemos incluir en el estudio.

Primeramente se carga la base de datos descargada y localizada en la carpeta Data/ApartadoA mediante un csv delimitado por comas, en una variable la cual contendrá toda esta información. A partir de aquí podemos examinar cuantos datos contiene, y como se almacenan estos.

Mediante la función shape() podemos ver el número de filas, correspondiente al número de elementos de nuestro dataset y el número de columnas**, correspondiente al número de atributos que tiene nuestra base de datos que en nuestro caso es igual a 23 atributos.**

Para saber que atributos tenemos exactamente y como estos están almacenados, podemos recurrir a varios tipos de funciones de la librería pandas, en nuestro caso hacemos uso de la función head() para observar las primeros 5 filas del dataset y así poder realizar un primer vistazo y tener una pequeña referencia sobre cada elemento, gracias al que ya primeramente podemos observar que existe algún valor nulo en cada fila los cuales tendremos que procesar mas tarde en siguientes fases, además de ver datos tanto numéricos como categóricos. Para saber bien que tipos de atributos tenemos y como están representados podemos usar la función info() que nos muestra el tipo de dato almacenado en cada uno de nuestro atributos los cuales representamos seguidamente, respondiendo asi a la pregunta: **¿Qué tipos de atributos tenemos?**



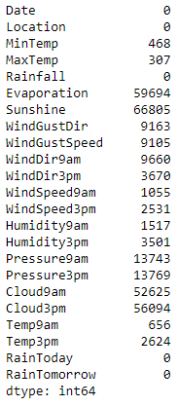
Estos son los principales datos que tenemos en nuestra base de datos, los cuales están divididos en object y float64, siendo los primeros tipos de datos categóricos y los segundos tipos de datos numéricos. Algunos de los datos categóricos son ordinales de más de 2 categorías lo cual indica que los tendremos que convertir en valores numéricos para el posterior estudio y así observar si hay posibles correlaciones entre ellos, en cambio otros son tan solo de dos categorías, los cuales pueden ser transformados en valores binarios (0, 1) y puede ser un indicativo también de que alguno de estos atributos sea una variable objetivo del problema de clasificación.

Estos atributos que acabamos de describir son RainToday y RainTomorrow. Entre estos dos se decide que el atributo objetivo será RainTomorrow, ya que nos parece más interesante estudiar la probabilidad de que llueva al día siguiente teniendo en cuenta las condiciones climáticas del día anterior según diversos factores, que no la probabilidad de que llueva hoy la cual podemos saber de forma más sencilla.

Con esto **se responde a la pregunta de cómo es el target y que categorías diferentes existen** Sabiendo que RainTomorrow es una variable ordinal de 2 categorías (Yes, No), aunque posteriormente se cambiaran esos valores por binarios para el estudio de correlaciones.

Para **poder ver si existe alguna correlación entre X e y**, convertimos las variables RainToday y RainTomorrow en variables cardinales de manera que podamos ver mediante una matriz de correlación si existe algún tipo de relación. Destacar que es difícil cuantificar este tipo de datos y extraer conclusiones a partir de una matriz de correlación. Por lo que en este caso no observamos ningún tipo de correlación en valores numéricos. Por lo que posteriormente se procede a realizar gráficos entre variables para observar de forma visual como se distribuyen los datos tratando outliers y valores nulos para ver si nos puede dar algún indicador de que existan ciertas correlaciones y volver a hacer la matriz de correlación.

Para observar las correlaciones y relaciones entre variables primeramente comprobamos cuantos valores nulos existen por categoría, haciendo uso de la función isnull().sum() de manera que obtenemos el número de nulos que existen por cada atributo.



Como podemos observar, existe una gran cantidad de valores nulos, cosa que afectara al estudio de las variables, correlaciones y relaciones entre variables. Así que para realizar un análisis exhaustivo de los datos trataremos estos valores nulos en el siguiente apartado de preprocessing, el cual comenzaremos con este tratamiento y el análisis exhaustivo que dejamos pendiente hasta ese instante.

Para responder a la pregunta de si **están balanceadas las etiquetas** podemos observar que estas toman valores distintos por cada atributo de manera que existen fechas, valores decimales, variables almacenadas en strings, y valores enteros, por lo que es difícil categorizar todo de forma uniforme y esto **afectará a la clasificación y distribución** debido a que se deben tomar decisiones sobre cómo se representaran estos datos y como se transformarán para realizar el estudio y poder comparar-los