Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

Ingeniería Informática
Especialidad: Computación
Cuarto curso. Primer cuatrimestre

Introducción a la Minería de Datos.

Práctica 2: Minería de datos supervisada: Clasificación.

Victoria Pradas Laguna
github.com_victoriapl01_MineriaDatos
i92prlav@uco.es

Curso académico 2022-2023
Córdoba, 2 de febrero de 2023

Índice

Índice de figuras IV

1. Práctica 2.1 **1**
1.1. Ficheros de datos 1
1.2. Árbol de decisión, el vecino más cercano y máquinas de vectores soporte. 5
1.3. Validación cruzada de 10 particiones y entrenamiento con diferentes clasificadores. 6
1.3.1. Validación cruzada de 10 particiones 6
1.4. Test de Wilcoxon de comparación de dos algoritmos sobre N problemas y aplíquelo a dos de los algoritmos anteriores. Obtenga el rango de Friedman para cada clasificador y configuración y represente gráficamente los resultados. Aplique el test de Iman-Davenport sobre los tres clasificadores 12
1.5. Compare el mejor método según el rango medio de Friedman con el resto de métodos usando el procedimiento de Holm. . . 13
1.6. Compración de los métodos por parejas usando el procedimiento de Bonferroni-Dunn 14
1.7. Validación de hiperparámetros con *grid search* 15

2. Práctica 2.2 **17**
2.1. Ficheros de datos 17
2.2. Método RIPPER del módulo *wittgenstein*[9]. 17
2.3. Clasificador SVM, kernel lineal y valor fijo c =1 19
2.4. Método *GridSearchCV* para la obtención de hiperparámetros con valores determinados 20

3. Práctica 2.3 **21**
3.1. Selección de métodos base: árbol de decisión y una máquina de vectores soporte 21
3.2. Para los dos métodos de clasifiación utilice los siguientes pasos usando validación cruzada de 10 particiones. 22
3.2.1. Aplicar los métodos bases y anotación de resultados obtenidos. 22
3.2.2. Aplicar el método de combinación de clasificadores Bagging a cada uno de los conjuntos 22
3.2.3. Dos algoritmos Boosting 23
3.2.4. Diferencias significativas utilizando el test Iman-Davenport. 24
3.3. Conclusiones del estudio 25

4. Práctica 2.4 **26**
4.1. Selección de algoritmo de clasificación capaz de resolver problemas de más de dos clases. 26
4.2. Aplicación de clasificador base a cada conjunto 30
4.3. Métodos múlticlase one-vs-one, one-vs-all, y error correcting output codes. 31
4.3.1. Método One vs. One (OVO) 31
4.3.2. Método One vs All (OVA) 31
4.3.3. Error Correcting Output codes (ECOC) 33
4.4. Diferencias significativas usando test Iman-Davenport y aplicación de Wilcoxon. 34

Índice de figuras

1. Iris Dataset . . . 1
2. Car Dataset . . . 2
3. Wine Dataset . . . 2
4. Diabetes Dataset . . . 3
5. Glass Dataset . . . 3
6. Cancer Dataset . . . 4
7. Titanic Dataset . . . 4
8. Vote Dataset . . . 4
9. Segment Dataset . . . 5
10. Zoo Dataset . . . 5
11. Validación cruzada Iris . . . 7
12. Validación cruzada Car . . . 7
13. Validación cruzada Wine . . . 8
14. Validación cruzada Diabetes . . . 8
15. Validación cruzada Glass . . . 9
16. Validación cruzada Cancer . . . 9
17. Validación cruzada Titanic . . . 10
18. Validación cruzada Vote . . . 10
19. Validación cruzada Segment . . . 11
20. Validación cruzada Zoo . . . 11
21. Comparación de diferentes modelos a partir de validación cruzada . . . 12
22. Bonferroni DTree vs KNN . . . 14
23. Bonferroni DTree vs SVM . . . 15
24. Bonferroni KNN vs SVM . . . 15
25. Comparación SVM parámetros fijos y árbol de decisión . . . 19
26. Comparación SVM vs SVM gridSearch . . . 21
27. DTree vs SVM . . . 22
28. Comparación de métodos aplicando Bagging . . . 23
29. Comparación de métodos aplicando Boosting . . . 24
30. Iris Dataset . . . 26
31. Car Dataset . . . 27
32. Wine Dataset . . . 27
33. Glass Dataset . . . 28
34. Segment Dataset . . . 28
35. Abalone Dataset . . . 29
36. Leaf Dataset . . . 29
37. Soybean Dataset . . . 29

38. Zoo Dataset 30
39. SVM Multiclase 30
40. SVM OVO vs SVM Method 31
41. SVM OVA vs SVM Method 32
42. SVM ECOC vs SVM Method 33

1. Práctica 2.1

En esta práctica se llevará a cabo diferentes ejercicios usando el módulo Pandas así como cualquier módulo adicional que sea necesario para llevar a cabo ejercicios de minería de datos supervisada, concretamente clasificación. Para evaluar el efecto de las operaciones de preprocesado se considera un árbol de decisión (Decision Tree) y un clasificador de vecino más cercano (KNN) y máquina de vectores soporte (SVM). Estos clasificadores tienen diferente naturaleza, y por ello se hará uso de ellos, para realizar distintas comparaciones.

1.1. Ficheros de datos

Los ficheros de datos son obtenidos de UCI Machine Learning Repository[1] y Weka Data Sets[2], repositorios con una gran variedad de Data Sets. Entre ellos, se ha realizado una selección:

- **Dataset Iris**. Conjunto de datos que contiene tres clases de cincuenta instancias cada una, donde cada clase se refiere a un tipo de planta de iris. Los atributos contienen la siguiente información: longitud y anchura del sépalo y longitud y anchura del pétalo. Todos los atributos son de tipo *float64* y la clase de tipo *object*.En la figura 30 se muestra las cinco primeras instancias.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Figura 1: Iris Dataset

- **Dataset Car Evaluation**. Conjunto de datos con diferentes características propias del vehículo. Este conjunto de datos deriva de un modelo de decisión jerárquica simple pero será utilizada sin tener en cuenta dicha jerarquía. Tiene mil setecientos veintiocho instancias con seis atributos: precio del vehículo, precio del mantenimiento, número

de puertas, número de asientos, tamaño del maletero y seguridad. Tienen cuatro clases indicando de menor a mayor valor del vehículo. En la figura 31 se muestra las cinco primeras instancias.

	buying	maint	doors	persons	lug_boot	safety	class
0	4	4	2	2	1	1	unacc
1	4	4	2	2	1	2	unacc
2	4	4	2	2	1	3	unacc
3	4	4	2	2	2	1	unacc
4	4	4	2	2	2	2	unacc

Figura 2: Car Dataset

En este conjunto de datos, se han realizado modificaciones en los valores de los atributos para un mejor estudio de los datos. Los datos eran de tipo categórico y se han cambiado a tipo numérico,ratio, entero a partir de la función *map*.

- **Dataset Vino**. Conjunto de datos que se dedican a determinar el origen de los vinos mediante análisis químicos. Los atributos son alcohol, ácido málico, ash, alcalinidad de ash, magnesio, fenoles totales, flavanoides, fenoles no flavanoides, proantocianinas, intensidad del color, tonalidad, OD280/OD315 de vinos diluidos y prolina. En la figura 32 se muestra las cinco primeras instancias.

	class	alcohol	malic acid	ash	alcalinityOfAsh	Magnesium	Total Phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color Intensity	Hue	OD280	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

Figura 3: Wine Dataset

- **Dataset Diabetes** Conjunto de datos que se obtiene a partir de restricciones de una base de datos más grandes. En particular, todos los pacientes(instancias) son mujeres de al menos veintiún años. Los atributos a tener en cuenta son el número de veces embarazadas, la concentración de glucosa en plasma a las dos horas en una prueba de tolerancia oral a la glucosa, presión arterial diastólica(mm Hg), grosor del pliegue

cutáneo del trícpes (mm), insulina sérica de 2 horas (mu U/ml), índice de masa corporal (peso kg / (altura en m)$\hat{2}$), función de pedigrí de diabetes y edad. Existen 768 instancias y las clases determinan si han sido positivas o negativas en las pruebas de diabetes. En la figura 4 se muestra las cinco primeras instancias.

	Times Pregnant	Glucose tolerance	Blood pressure	Triceps skin fold	Insulin	Body mass	Diabetes pedigree	Age	Class
0	6	148	72	35	0	33.6	0.627	50	tested_positive
1	1	85	66	29	0	26.6	0.351	31	tested_negative
2	8	183	64	0	0	23.3	0.672	32	tested_positive
3	1	89	66	23	94	28.1	0.167	21	tested_negative
4	0	137	40	35	168	43.1	2.288	33	tested_positive

Figura 4: Diabetes Dataset

- **Dataset Glass**. Este conjunto de datos determina el tipo de vidrio, por tanto tiene 7 clases entre ellas ventana flotante o no flotante, o ventanilla de coche flotante. Esta conjunto de datos nace tras un estudio de clasificación de tipos de vidrio que fue motivado por una investigación criminológica. En el que el vidrio puede utilizarse como prueba si se identifica correctamente. En la figura 33 se muestra las cinco primeras instancias.

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.51793	12.79	3.50	1.12	73.03	0.64	8.77	0.0	0.00	'build wind float'
1	1.51643	12.16	3.52	1.35	72.89	0.57	8.53	0.0	0.00	'vehic wind float'
2	1.51793	13.21	3.48	1.41	72.64	0.59	8.43	0.0	0.00	'build wind float'
3	1.51299	14.40	1.74	1.54	74.55	0.00	7.59	0.0	0.00	tableware
4	1.53393	12.30	0.00	1.00	70.16	0.12	16.19	0.0	0.24	'build wind non-float'

Figura 5: Glass Dataset

- **Dataset Breast Cancer Wisconsin**. Las características de este conjunto de datos son calculadas a partir de una imagen digitalizada de una aspiración con aguja fina (FNA) de una masa mamaria. Describen características de los núcleos celulares presentes en la imagen. Los atributos son de tipo numérico, ya que son números reales que determinan el radio, textura, perímetro, área, suavidad, compacidad, concavidad, puntos cóncavos, simetría y dimensión fractal. Por tanto las clases correspondientes serán dos, maligno o benigno. En la figura 6 se muestra las cinco primeras instancias.

	id	Clump Thickness	Cell Size	Cell Shape	Marginal Adhesion	Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

Figura 6: Cancer Dataset

- **Dataset Titanic**. Este conjunto de datos, determinan los datos de los pasajeros del crucero Titanic como el identificador, nombre, sexo, ticket o cabina. En nuestro caso, ha sido adaptada seleccionando los atributos que se consideran más adecuados para nuestro estudio. La clase que se determina es si la persona sobrevivió o no consiguió sobrevivir. En la figura 7 se muestra las cinco primeras instancias.

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0	3	0	22.0	1	0	7.2500
1	1	1	1	38.0	1	0	71.2833
2	1	3	1	26.0	0	0	7.9250
3	1	1	1	35.0	1	0	53.1000
4	0	3	0	35.0	0	0	8.0500

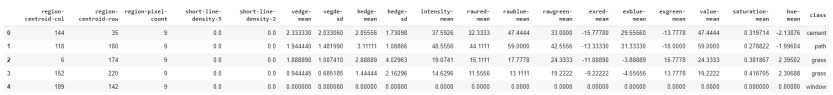
Figura 7: Titanic Dataset

- **Dataset Congressional Voting Records** Este conjunto de datos obtiene los registros de votación del Congreso de los Estados Unidos de 1984, clasificando las instancias como republicano o demócrata. Los atributos, en este caso determinan características propias de las personas como si tienen hijos discapacitados, costo compartido de agua, si apoyan la inmigración o la exportación libres de impuesto entre muchos de ellos. En este caso, al tener la base de datos atributos categóricos, 'y' ó 'n' , se han modificado a '0' y '1'. En la figura 8 se muestra las cinco primeras instancias.

	handicapped-infants	water cost sharing	adoption the budget resolution	physician fee freezen	god aid	religous groups	anti satellite test	aid to nicaraguan contras	mx-missil	inmigration	recortes	gasto de educacion	demandas	crimen	exportacion libre impuestos	administracion sur africa	class
0	1	0	1	0	0	0	1	1	1	0	2	0	0	0	1	0	republican
1	1	0	1	0	0	0	1	1	1	1	1	0	0	0	1	2	republican
2	2	0	0	2	0	0	1	1	1	1	0	1	0	0	1	1	democrat
3	1	0	0	1	2	0	1	1	1	1	0	1	0	1	1	0	democrat
4	0	0	0	1	0	0	1	1	1	1	0	2	0	0	0	0	democrat

Figura 8: Vote Dataset

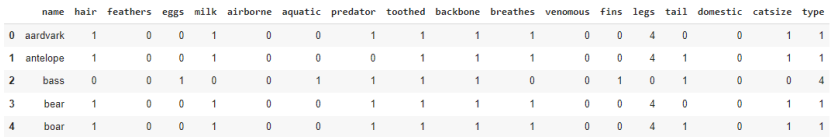
- **Dataset Segment challenge**. Este conjunto de datos es un subconjunto de los datos de segmentación, es decir un subconjunto de los datos originales de entrenamiento y prueba, obteniendo las instancias al azar. La distribución de clase serán cielo, follaje, cemento, ventana camino, hierba o brickface. En la figura 34 se muestra las cinco primeras instancias.



	region-centroid-col	region-centroid-row	region-pixel-count	short-line-density-5	short-line-density-2	vedge-mean	vedge-sd	hedge-mean	hedge-sd	intensity-mean	rawred-mean	rawblue-mean	rawgreen-mean	exred-mean	exblue-mean	exgreen-mean	value-mean	saturation-mean	hue-mean	class
0	144	35	9	0.0	0.0	2.333330	2.033060	2.05556	1.73098	37.5926	32.3333	47.4444	33.0000	-15.77780	29.55560	-13.7778	47.4444	0.319714	-2.13876	cement
1	118	180	9	0.0	0.0	1.944440	1.481990	3.11111	1.08866	40.5556	44.1111	59.0000	42.5556	-13.33330	31.33330	-18.0000	59.0000	0.278822	-1.99604	path
2	6	174	9	0.0	0.0	1.888890	1.007410	2.88889	4.02963	19.0741	15.1111	17.7778	24.3333	-11.88890	-3.88889	15.7778	24.3333	0.381867	2.39592	grass
3	152	220	9	0.0	0.0	0.944445	0.685185	1.44444	2.16296	14.6296	11.5556	13.1111	19.2222	-9.22222	-4.55556	13.7778	19.2222	0.416705	2.30688	grass
4	189	142	9	0.0	0.0	0.000000	0.000000	0.00000	0.00000	0.0000	0.0000	0.0000	0.0000	0.00000	0.00000	0.0000	0.0000	0.000000	0.00000	window

Figura 9: Segment Dataset

- **Dataset Zoo**. Una base de datos simple que contiene 17 atributos con valores booleanos, siendo el atributo 'tipo' el atributo de clase en las que cada tipo, muestra un conjunto de animales propios de es tipo. Los atributos a tener en cuenta serán el pelo, huevos, leche, patas, cola, domésticos, entre muchos de ellos. En la figura 38 se muestra las cinco primeras instancias.



	name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	type
0	aardvark	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
1	antelope	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
2	bass	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
3	bear	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
4	boar	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1

Figura 10: Zoo Dataset

1.2. Árbol de decisión, el vecino más cercano y máquinas de vectores soporte.

A partir de los conjuntos de datos del apartado anterior, se evaluará el árbol de decisión (Decision Tree), algoritmo de vecino más cercano (KNN) y máquinas de vector soporte (SVC).

Se hace uso de la biblioteca *scikit-learn* [3] en la que se encuentras diferentes funciones para aplicar los métodos de clasificación mencionados anteriormente, [4] [5] [6].

1.3. Validación cruzada de 10 particiones y entrenamiento con diferentes clasificadores.

Para realizar comparaciones entre los diferentes métodos en cada una de los conjuntos de datos, se hará uso de una medida de evaluación, como puede ser precisión, recall o f1-score o tasa de error. En este caso, se pide utilizar dos métricas estudiadas en teoría y por tanto han sido elegidas la precisión (*accuracy*) y *f1_ macro*.

Tras tener cargado los datos separados por características y clases, se procede a dividir los datos en conjunto de entrenamiento y prueba haciendo uso de la función *train_ test_ split*, para posteriormente entrenar un árbol de decisión, un modelo K-NN y una máquina de vectores soporte utilizando los datos de entrenamiento.

1.3.1. Validación cruzada de 10 particiones

En este apartado se hará uso del método *cross_ val_ score*[7] que su trabajo es evaluar una puntuación mediante validación, siendo esta puntuación pasada por parámetro: *scoring*. En modo explicativo, esta función es utilizada para realizar validación cruzada para encontrar los mejores hiperparámetros para un modelo de aprendizaje. El parámetro *scoring* le indica a la función la métrica para evaluar el rendimiento del modelo en los datos de prueba en cada iteración de la validación cruzada. Por tanto, cada una de las particiones, genera una medida que será almacenada en un array con las puntuaciones lo cual devolverá dicha función.

En nuestro caso, para conocer una métrica general del modelo, se calcula la media del array devuelto por la función *cross_ val_ score*.

Primero se realizará un estudio de los resultados individuales por cada dataset, haciendo una comparación entre el uso de la validación cruzada con el modelo árbol de decisión y teniendo en cuenta las métricas tanto de precisión como F1 macro. También la realización de validación cruzada con el modelo KNN y teniendo en cuenta las métricas anteriores y por último igual pero con el modelo de máquinas de vector soporte. Finalmente, se realizará una gráfica de forma resumen de cada uno de los modelos con los diferentes modelos.

En el caso del dataset iris, se observa que el mejor modelo ha sido SVM con ambas métricas, seguido posteriormente del modelo KNN y finalmente

el árbol de decisión. Figura 11

A de tenerse en cuenta, que para los hiperparámetros de la máquina se vector soporte, se ha realizado un estudio a partir de *grid_ search* y hemos obtenido los valores que mayor rendimiento tienen siendo estos *'kernel'=lineal, 'C'=1.*

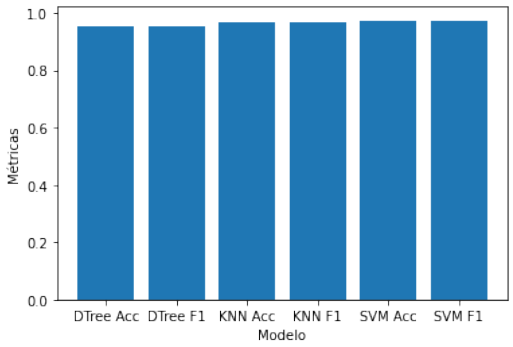


Figura 11: Validación cruzada Iris

El caso del dataset de car evaluation, podemos destacar que la métrica de F1Macro es más estricta y por tanto da un valor bastante más bajo en todos los modelos. Figura 12 Pero a pesar de ello, si observamos la precisión observamos que SVM vuelve a ser el mejor modelo a utilizar, debido al estudio de hiperparámetros a partir de *grid_ search*, seguido de el árbol de decisión que trabajan prácticamente de forma muy óptima ambos.

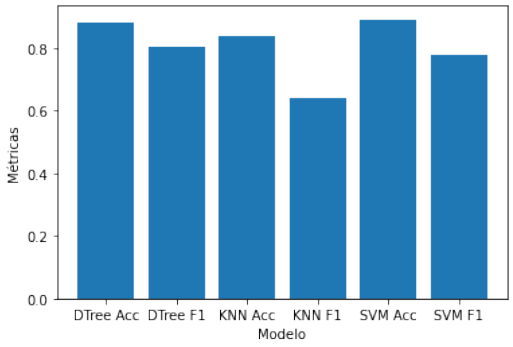


Figura 12: Validación cruzada Car

Dataset vino, muestra la peculiaridad de que las diferentes métricas evalúan de forma muy similar en este caso. La máquina de vector soporte hace uso de los hiperparámetros *'kernel'=linear, 'C'=0.1* siendo el modelo más eficiente como se observa en la Figura 13.

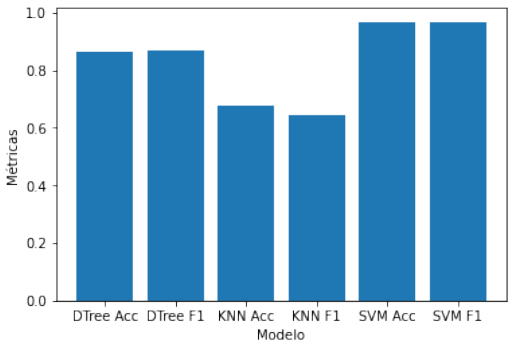


Figura 13: Validación cruzada Wine

Una vez más, en el conjunto de datos diabetes, la métrica F1 Macro es más estricta y por ellos puntúa menor que la métrica de precisión. De todas formas a pesar de ello, independientemente de la métrica que se utilice el orden de los modelos por su rendimiento es igual. Figura 14.

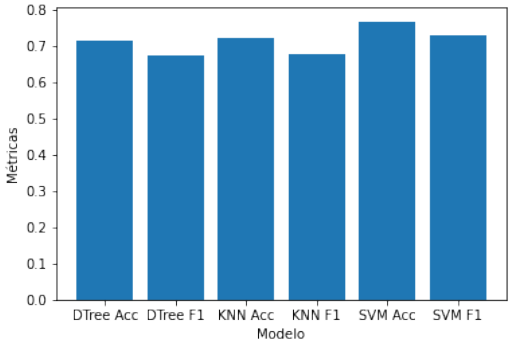


Figura 14: Validación cruzada Diabetes

En el caso del conjunto de datos Glass, sigue siendo similar al anterior y se mantiene el ranking independientemente de la métrica. En este caso el árbol de decisión trabaja de forma más eficiente y esto es debido a que este modelo trabaja muy bien con una gran cantidad características como es el caso. Figura 15

En la siguiente base de datos, se observa una gran precisión con todos los modelos. Figura 16. El caso de los hiperparámetros han sido elegidos para la máquina de vector *'kernel'=rbf, 'C'=0.1*

Titanic dataset, muestra una precisión algo más baja que las anteriores concretamente con el modelo KNN independientemente una vez más de la

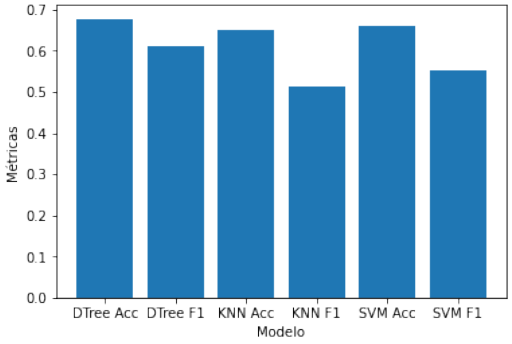


Figura 15: Validación cruzada Glass

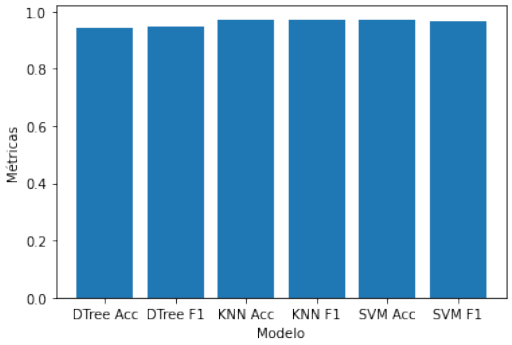


Figura 16: Validación cruzada Cancer

métrica utilizada. 17 Los hiperparámetros, obtenidos a partir de *grid_ search*, determinan que los mejores para un mejor rendimiento del modelo de SVM serían *'kernel'=linear, 'C'=1*

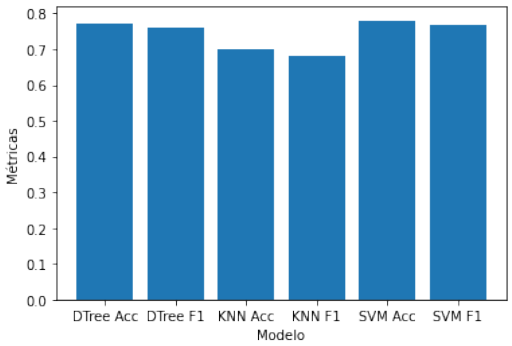


Figura 17: Validación cruzada Titanic

Los siguientes conjuntos de datos, tanto vote como segment, se puede observar una gráfica similar 18, 19 donde el modelo KNN es el que menor rendimiento tiene y peor es entrenado, y las métricas trabajan y puntuan de forma bastante semejante.

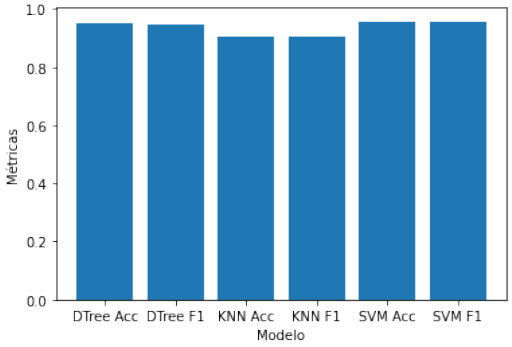


Figura 18: Validación cruzada Vote

Por último, dataset zoo tiene un valor muy bajo con la métrica F1Macro y esto puede ser ya que en algunos casos, la clase menos popular tiene menos miembros de 10 que es lo que se ha fijado en el enunciado y por tanto no pueden calcularse en esos casos, y hace la media de los anteriores obteniendo menos resultados. De todas formas, la precisión del modelo Máquina de Vectores soporte trabaja con un 0.95 de precisión. Figura 20.

De forma general, pudiendo observar una gráfica común con todos los conjuntos de datos y observando cada uno de los clasificadores podemos

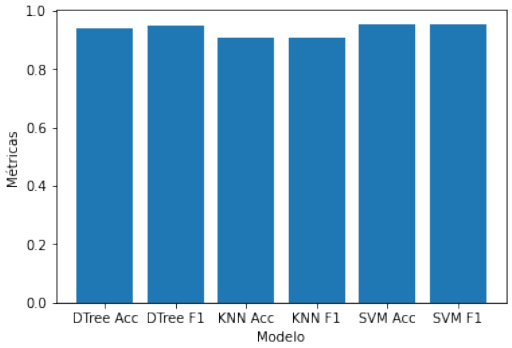


Figura 19: Validación cruzada Segment

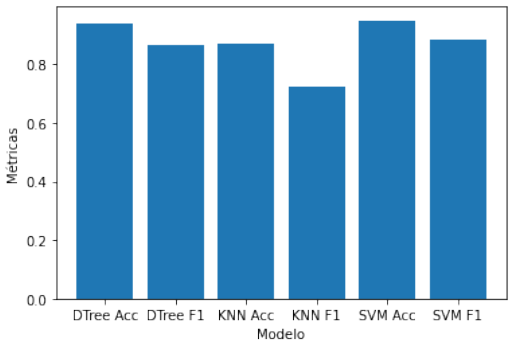


Figura 20: Validación cruzada Zoo

sacar conclusiones, Figura 21. Se observa que en la mayoría de conjuntos el mejor modelo es SVM y una causa es el uso de *gris_ search* para encontrar los hiperparámetros con mejor rendimiento del modelo. De forma contraria, el peor modelo es KNN debido a su algoritmo que por lo generar no es el mejor clasificador.

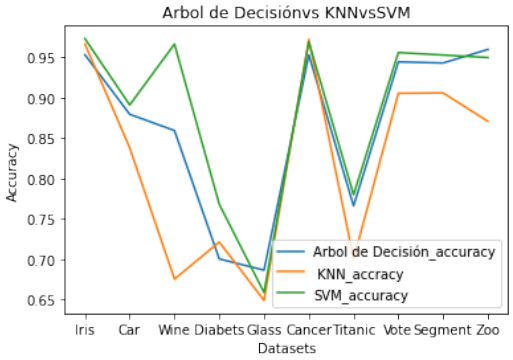


Figura 21: Comparación de diferentes modelos a partir de validación cruzada

Podría existir una mejora de estos, aplicando la función *grid_ search* a cada uno de los modelos para encontrar los mejores hiperparámetros para cada uno de los conjunto de datos.

1.4. Test de Wilcoxon de comparación de dos algoritmos sobre N problemas y aplíquelo a dos de los algoritmos anteriores. Obtenga el rango de Friedman para cada clasificador y configuración y represente gráficamente los resultados. Aplique el test de Iman-Davenport sobre los tres clasificadores

El resultado del test de Wilcoxon te indicará si existe una diferencia significativa entre los dos grupos de datos, dando un valor de 'p-value' para indicar la probabilidad de que los datos sean iguales. Cuando el p-value tiene un valor menor a 0.05 indica que la diferencia entre los grupos es estadísticamente significativa, es decir que es poco probable que los datos sean iguales por casualidad. De forma contraria, si es mayor a 0.05 no hay evidencia para afirmar que existe una diferencia significativa entre los grupos de datos.

Tras aplicar el test Wilcoxon a los métodos de árbol de decisión y knn obtenemos una estadística de prueba de 6 y p-value 0.027. El primer valor

representa la diferencia entre los dos grupos de datos, pero nos proporciona más información la probabilidad asociada a ese valor, siendo ésta el valor de p (*p-value*).

El rango de Friedman nos proporciona como resultlado una lista de rangos para cada método, que se puede usar para comparar los métodos y determinal cuál es el mejor. Por ejemplo, si el rango medio del árbol de decisión es mayor que el de knn y SVM, podemos concluir que el árbol de decisión es el mejor método en términos de rendimiento.

Rango medio de árbol de decisión: [8. 5. 4. 1. 2. 9. 3. 7. 6. 10.] Rango medio de KNN: [9. 5. 2. 4. 1. 10. 3. 7. 8. 6.] Rango medio de SVM: [10. 4. 8. 2. 1. 9. 3. 7. 6. 5.]

1.5. Compare el mejor método según el rango medio de Friedman con el resto de métodos usando el procedimiento de Holm.

Para comparar el mejor método según el rango medio de Friedman con el resto de métodos utilizando el procedimiento de Holm, se deben seguir los siguientes pasos:

Calcular el rango medio para cada método; se asigna un rango a cada valor de rendimiento (en este caso, la exactitud) y se suman los rangos para cada método. El rango medio para cada método se calcula dividiendo la suma de los rangos por el número de datos.

Aplicar el test de Friedman: se aplica el test de Friedman sobre los rangos medios de los métodos para determinar si existe alguna diferencia significativa entre los métodos. Si el valor p obtenido es menor que el nivel de significancia (0.05), se concluye que existen diferencias significativas entre los métodos.

Posteriormente se debe aplicar el procedimiento de Holm: se ordenan los métodos de acuerdo a su rango medio y se aplica el procedimiento de Holm para determinar qué método es el mejor y cuáles son significativamente diferentes del mejor. El procedimiento de Holm consiste en comparar cada método con el mejor método y calcular el valor p correspondiente. Si el valor p es menor que el nivel de significancia dividido por el número de comparaciones (0.05/3=0.0167), se concluye que el método es significativamente diferente del mejor.

Llevando tal teoría a la práctica, determinamos el test de Friedman de los

tres métodos (árbol de decisión, knn, y máquina de vector soporte), siendo un valor de p-value bastante bajo, 0.014 y siendo menor a 0.05 por lo que nos indica que existen diferencias significativas entre dichos métodos.

1.6. Compración de los métodos por parejas usando el procedimiento de Bonferroni-Dunn

El método Bonferroni-Dunn es un método post-hoc para comparar múltiples algoritmos de aprendizaje después de realizar una prueba estadística global como se han realizado en los apartados anteriores. Este procedimiento consiste en aplicar una corrección de multiplicidad al valor p para cada par de comparaciones y seleccionar el mejor algoritmo basado en el menor pvalue corregido. Esto permite controlar el riesgo de error de falsos positivos y es útil para evitar conclusiones erróneas cuando se comparan muchos algoritmos a la vez.

La librería Scikit-posthocs [8] incluye el procedimiento bonferroni, para la comparación de estos datos, por lo que mostraremos los resultados en la práctica.

Antes de ello, debemos tener en cuenta que nos da como resultado una matriz de comparaciones por parejas, en las que la diagonal de la matriz representa la comparación del método consigo mismo, por lo que su valor siempre será 1.0. Y el resto la comparación entre ambos métodos, representando el rendimiento entre ellos.

Primero, realizaremos la comparación entre árbol de decisión y vecino más cercano obteniendo se refleja en la Figura 22. En las que se puede observar que el valor entre los métodos es de 0.364346 siendo mejor el árbol de decisión en comparación con kNN con un nivel de confianza del 64.3 %.

	1	2
1	1.000000	0.364346
2	0.364346	1.000000

Figura 22: Bonferroni DTree vs KNN

Otra comparación, sería entre el árbol de decisión y la máquina de vector soporte que nos muestra un resultado similar, que el anterior, pero siendo ahora máquina de vector soporte más eficiente que el árbol de decisión. 23

	1	2
1	1.000000	0.364346
2	0.364346	1.000000

Figura 23: Bonferroni DTree vs SVM

Y por último, la diferencia o matriz de comparación por parejas entre KNN y SVM, que por lógica de los casos anteriores, si KNN es peor que el arbol de decisión, y SVM mejora con respecto al árbol de decisión, podemos concluir que la diferencia entre máquinas de vector soporte y el vecino más cercano indicaran un mayor rendimiento del método SVM, como se observa en la Figura 24, ccon un nivel de confianza del 79 % aproximadamente.

	1	2
1	1.000000	0.212122
2	0.212122	1.000000

Figura 24: Bonferroni KNN vs SVM

1.7. Validación de hiperparámetros con *grid search*

En la sección anterior 1.3.1, ya se ha realizado este apartado sin previo conocimiento de este apartado. Por lo que se analizará el resultado de cada uno de los resultados de hiperparámetros.

```
from sklearn.model_selection import GridSearchCV
```

```
def hiperparametosSVM(data_X, data_y):
  accuracy_svm=[]
  param_grid= {'kernel': ['linear','rbf','sigmoid'], 'C': [0.1, 1, 3]}
  svm = SVC()
  grid = GridSearchCV(svm, param_grid, cv=5)
  grid_search = grid.fit(data_X, data_y)
  print('Mejores hiperparámetros: ', grid_search.best_params_)
  svm= SVC(kernel=grid_search.best_params_['kernel'],
    C=grid_search.best_params_['C'])
  return svm
```

Como se observa en el código mostrado, se ha realiza el estudio de los mejores hiperparámetros en función de las variables de cada uno de los conjuntos de datos.

Los hiperparámetros estudiados han sido C, que es un parámetro de regularización, siendo inversamente proporcional a la fuerza de regularización. Se debe tener encuenta que debe ser positiva, y por ello se ha realizado con la posibilidad de valores *[0.1, 1, 3]*. Por otra parte, se ha especificado también el kernel; parámetro que especifica el tipo de núcleo que se utilizará en el algoritmo, pudiendo ser *linear, poly, rbf, sigmoide o precomputd*, y que para el estudio se analizará únicamente pudiendo ser *linear o rbf*. En cada uno de los conjuntos de datos, se analiza la posibildad de los mejores hiperparámetros mostrando especificamente los resultados:

- **Iris Dataset**. Mejores componentes principales: 'C': 1, 'kernel': 'linear'
- **Car Dataset**. Mejores componentes principales: 'C': 3, 'kernel': 'rbf'
- **Wine Dataset**. Mejores componentes principales: 'C': 0.1, 'kernel': 'linear'
- **Diabetes Dataset** Mejores componentes principales: 'C': 3, 'kernel': 'linear'
- **Glass Dataset** Mejores componentes principales: 'C': 3, 'kernel': 'linear'
- **Breast Cancer Wisconsin Dataset** Mejores componentes principales: 'C': 0.1, 'kernel': 'rbf'

- **Titanic Dataset** Mejores componentes principales: 'C': 1, 'kernel': 'linear'
- **Vote Dataset** Mejores componentes principales: 'C': 0.1, 'kernel': 'linear'
- **Segment Challeng Dataset** Mejores componentes principales: 'C': 1, 'kernel': 'linear'
- **Zoo Dataset** Mejores componentes principales: 'C': 3, 'kernel': 'rbf'

2. Práctica 2.2

En esta práctica se llevará a cabo diferentes ejercicios usando el módulo scikit-learn así como cualquier módulo adicional para introducir el uso de métodos de clasificación estudiados previamente en teoría.

2.1. Ficheros de datos

En esta sección, se elegirán tres conjuntos de datos que tengan 2 clases, que haciendo uso de los dataset anteriores, se utilizaran: Diabetes Dataset 1.1, Vote Dataset 1.1 y Titanic Datasest 1.1 determinando si la persona sobrevivió o no.

2.2. Método RIPPER del módulo *wittgenstein*[9].

El algoritmo RIPPER, *Repeated Incremental Pruning to Produce Error Reduction*: Poda incremental repetida para producir reducción de errores, un modelo de clasifiación binaria y multiclase, el cual en este apartado haremos uso como binaria ya que hemos elegido dataset de dos clases. Es un tipo de algoritmo de reglas de decisión, lo que significa que produce reglas de decisión legibles y comprensibles para el ser humano en lugar de un modelo matemático complejo.

RIPPER es eficiente y rápido en términos de tiempo de entrenamiento y espacio de memoria, y además es capaz de se eficiente ante atributos irrelevantes o redundantes. Dependiendo de la base de datos, puede ser más o

menos preciso que algunos algoritmos más complejos como árboles de decisión y/o redes neuronales.

A partir del conjunto de datos divididos en entrenamiento y prueba, se lleva a cabo la creación y entrenamiento del modelo RIPPER determinando el dataset, el atributo clase, y la clase positiva. A partir del método *ripper_ clf.score* determinaremos el rendimiento del modelo y la eficiencia de clasifiCación de éste. Además podemos mostrar las reglas obtenidas de cada uno de los conjuntos de datos.

- **Diabetes Dataset**. Precisión: 0.7598425196850394 Conjunto de reglas:

```
[[Glucosetolerance=>167.0] V
[Age=42.6-51.0 ^ Insulin=>210.0] V
[Glucosetolerance=147.0-167.0 ^ Diabetespedigree=0.37-0.45] V
[Age=29.0-33.0 ^ Insulin=150.0-210.0] V
[Tricepsskinfold=<8.2 ^ Glucosetolerance=134.0-147.0 ^
    Bloodpressure=<54.0] V
[Bodymass=>41.5 ^ Bloodpressure=>88.0] V
[Age=38.0-42.6 ^ Glucosetolerance=109.0-117.0]]
```

  El árbol de precisión obtenía una precisión del 70.43 %, obteniendo una mejora con el conjunto de reglas.

- **Vote Dataset**. Precisión: 0.9444444444444444 Conjunto de reglas:

```
[[physicianfeefreezen=1 ^ adoptionthebudgetresolution=0] V
[physicianfeefreezen=1] V
[physicianfeefreezen=2 ^ watercostsharing=0] V
[recortes=0 ^ mx-misil=0] V
[recortes=0 ^ adoptionthebudgetresolution=0 ^ antisatellitetest=1] V
[adminitracionsurafrica=2 ^ adoptionthebudgetresolution=0 ^
    antisatellitetest=1]]
```

  El árbol de precisión obtenía una precisión del 90.57 %, obteniendo una mejora con el conjunto de reglas.

- **Titanic Dataset**. Precisión: 0.7704918032786885 Conjunto de reglas:

```
[[Sex=1] V
[Age=<17.2]]
```

El árbol de precisión obtenía una precisión del 77.60 %, obteniendo una mejora con el conjunto de reglas.

En el caso de la base de datos de votaciones, es donde mejor funciona el conjunto de datos. Además una observación a tener en cuenta es que hace una selección de los mejores atributos para cada conjunto de reglas.

En los dos primeros casos, el método basado en reglas obtiene una mejor precisión que el árbol de decisión esto puede ser debido a la obtención de un mayor número de reglas que el tercero que es capaz de concluir dos únicas reglas.

2.3. Clasificador SVM, kernel lineal y valor fijo c =1

Los modelos utilizados en esta sección han sido el árbol de decisión y máquina de vectores soporte haciendo uso de un kernel lineal y un parámetro C igual a uno. Por ello, se lleva a cabo la gráfica para la interpretación de los datos y comparación de los modelos en los distintos conjuntos de datos. Figura 25.

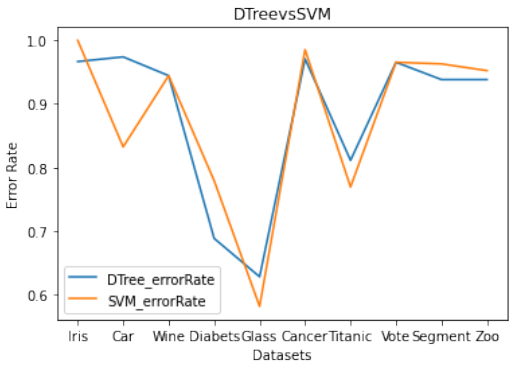


Figura 25: Comparación SVM parámetros fijos y árbol de decisión

Al tener hiperparámetros fijos, no se determina la precisión óptima del modelo si no una posible aproximación. En el caso del conjunto de datos iris o cáncer, puede ser la forma más adecuada, ya que muestra una precisión

muy alta. En muchos otros casos, como el dataset car no es la forma más adecuada y puede mejorar mucho ya que se ve como el árbol de decisión supera de forma llamativa al modelo SVM.

2.4. Método *GridSearchCV* para la obtención de hiperparámetros con valores determinados

En este apartado se utilizará el método *GridSearchCV* para la obtención de los mejores hiperparámetros. Para ello, se realizará el estudio con diferentes parámetros por validación cruzada:

```
# Set the parameters by cross-validation parameters =
    [  {"kernel": ["rbf"],
    "gamma": [0.01, 0.1, 1.0], "C": [1, 10, 100, 1000]},
    {"kernel": ["linear"], "C": [1, 10, 100, 1000]}, ]
```

Como se observa, para el kernel rbf, se aplicará un gamma y un C determinado pudiendo ser los valores propuestos en el código. De forma contraria, si el kernel es lineal, únicamente se determinará el hiperparámetro C con posibles valores 1, 10, 100 o 1000. Se muestran los siguientes valores que han obtenido los conjuntos de datos:

- **Iris Dataset**. Mejores hiperparámetros: 'C': 1, 'gamma': 0.1, 'kernel': 'rbf'
- **Car Dataset**. Mejores hiperparámetros: 'C': 10, 'gamma': 0.1, 'kernel': 'rbf'
- **Wine Dataset**Mejores hiperparámetros: 'C': 1, 'kernel': 'linear'
- **Diabetes Dataset** Mejores hiperparámetros: 'C': 100, 'kernel': 'linear'
- **Glass Dataset**. Mejores hiperparámetros: 'C': 10, 'gamma': 1.0, 'kernel': 'rbf'
- **Cancer Dataset**. Mejores hiperparámetros: 'C': 1, 'kernel': 'linear'
- **Titanic Dataset** Mejores hiperparámetros: 'C': 1000, 'kernel': 'linear'
- **Vote Dataset** Mejores hiperparámetros: 'C': 10, 'gamma': 0.01, 'kernel': 'rbf'

- **Segment Dataset**. Mejores hiperparámetros: 'C': 100, 'kernel': 'linear'
- **Zoo Dataset**. Mejores hiperparámetros: 'C': 10, 'gamma': 0.1, 'kernel': 'rbf'

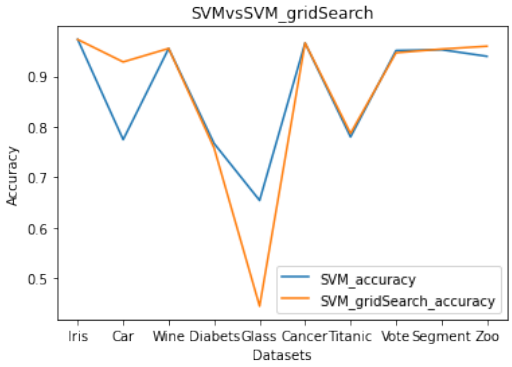


Figura 26: Comparación SVM vs SVM gridSearch

En la Figura 26, se puede observar que en todos los casos exceptuando el conjunto de base de datos glass mejora o se iguala ya que han sido estudiado diferentes mejoras con el método gridSearch. La comparación se ha realizado a partir de la validación cruzada teniendo en cuenta la precisión de los modelos.

3. Práctica 2.3

En esta práctica, se llevará a cabo conceptos de clasificación usando agrupaciones (*ensembles*) de clasificadores. Para ello, se aplicarán dos clasificadores diferentes para ver cómo se comportan como métodos base de un *ensemble*.

3.1. Selección de métodos base: árbol de decisión y una máquina de vectores soporte

Los métodos base serán el árbol de decisión con los hiperparámetros por defecto, siendo esto *criterion='gini', splitter='best, min_samples=2, max_depth='None'*. Por otro lado, la máquina de vector soporte usando un *kernel='linear', C=1*.

	Iris	Car	Wine	Diabetes	Glass	Cancer	Titanic	Vote	Segment
DTree	0.96	0.8784	0.8650	0.7018	0.6870	0.9547	0.7789	0.9446	0.9444
SVM	0.9733	0.7747	0.9555	0.7669	0.6541	0.9664	0.7799	0.9514	0.9531

Tabla 1: Comparación árbol de decisión con máquina de vector soporte

```
#Modelos a utilizar en esta práctica
tree=DecisionTreeClassifier()
svm=SVC(kernel='linear', C=1)
```

3.2. Para los dos métodos de clasifiación utilice los siguientes pasos usando validación cruzada de 10 particiones.

3.2.1. Aplicar los métodos bases y anotación de resultados obtenidos.

En la Tabla 1, se muestran los valores comparativos de cada uno de los modelos. También se muestra de forma gráfica en la Figura 27, siendo más representativo, y observandose de mejor forma que el modelo de máquina de soporte trabaja de forma más eficiente.

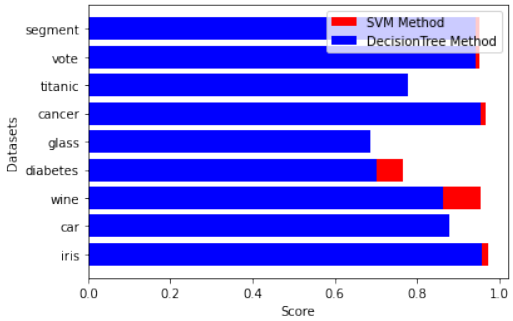


Figura 27: DTree vs SVM

3.2.2. Aplicar el método de combinación de clasificadores Bagging a cada uno de los conjuntos

Bagging es un método de ensamblaje de varios clasificadores que se basa en la selección aleatoria de muestras con reemplazo de los datos de entrenamiento para construir varios modelos. Cada modelo se entrena con un

	Iris	Car	Wine	Diabetes	Glass	Cancer	Titanic	Vote	Segment
DTree	0.96	0.8784	0.8650	0.7018	0.6870	0.9547	0.7789	0.9446	0.9444
Bagging_DTree	0.9466	0.8819	0.9388	0.7304	0.7287	0.9591	0.7996	0.9539	0.9506
SVM	0.9733	0.7747	0.9555	0.7669	0.6541	0.9664	0.7799	0.9514	0.9531
Bagging_SVM	0.9866	0.7799	0.9607	0.7734	0.607	0.9693	0.7800	0.9516	0.9592

Tabla 2: Métodos base y métodos aplicando bagging

conjunto diferentes de datos y luego se combinan para obtener una mejor precisión, obteniendo el resultado final por ejemplo mediante la votación de los clasificadores individuales.

Como se observa en la Tabla 2, el método Bagging se aplica correctamente y cumple su función puesto que mejora la precisión en la mayoría de casos. En estos casos, al ser conjunto de datos con pocas instancias no se refleja de gran forma, pero para aquellos que tienen una gran cantidad de datos se puede observar una clara mejora.

En la Figura 28, se observa, que el método de máquina de vector soporte es mejor que el del árbol aún aplicando el método Bagging.

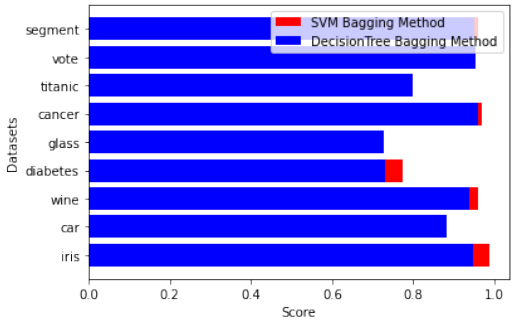


Figura 28: Comparación de métodos aplicando Bagging

3.2.3. Dos algoritmos Boosting

Los algoritmos elegidos han sido AdaBoost y Gradient Boosting, por lo que se aplicaran de forma independiente.

AdaBoost (Adaptative Boosting) es un algoritmo de aprendizaje automáatico en el que se combinan varios clasificadores débiles para producir uno fuerte. El algoritmo comienza asignando un peso igual a todas las muestras y luego ajusta los pesos de las muestras mal clasificadas en cada iteración. Para eset algoritmo se hará uso de la biblioteca *AdaBoostClassifier*.

	Iris	Car	Wine	Diabetes	Glass	Cancer	Titanic	Vote	Segment
AdaBoost	0.9533	0.8622	0.8833	0.7552	0.4675	0.9547	0.8039	0.9631	0.4506
GradBoost	0.96	0.8969	0.9160	0.7604	0.7807	0.9620	0.8194	0.9561	0.9530

Tabla 3: Resultado Adaptative Boosting y Gradient Boosting

Por otro lado, Gradient Boosting es un algoritmo que combina varios árboles de decisión débiles para producir uno fuerte. El algoritmo comienza con un árbol de decisión simple y luego ajusta los pesos de las muestras mal clasificadas en cada iteración. Se hace uso de la biblioteca *GradienntBoostingClassifier*

El resultado de ambos, queda plasmado en la siguiente tabla 3. Podemos ver como el segundo algoritmo es más eficiente en general en todos los conjutnos de datos, ya que en el caso de AdaBoost es bastante bajo tanto en Glass Ddataset como sen segment Dataset.

Otra forma de visualizar los resultados, sería con una gráfica como se muestra en la Figura 29 que como se observa AdaBoost penaliza mucho la precisión en algunos conjuntos de datos.

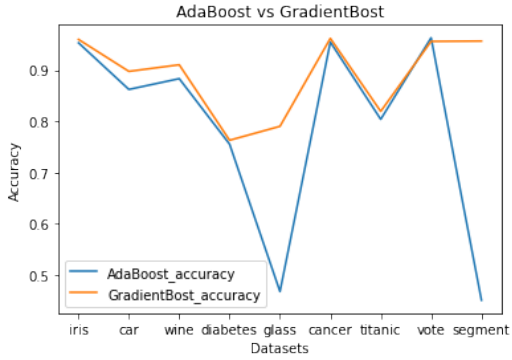


Figura 29: Comparación de métodos aplicando Boosting

3.2.4. Diferencias significativas utilizando el test Iman-Davenport.

En este caso, se lleva el caso el test Iman-Davenport, y si este consigue un valor menor de 0.5 significa que existen diferencias significativas y se procede a aplicar wilcoxon entre cada método de agrupaci´no con el clasificador base.

En los modelos Bagging, aplicando el test Iman-Davenport se rechaza la hipotesis nula del test de Iman-Davenport, y por tanto se lleva a cabo el test de Wilcoxon.

```
MODELOS BAGGING
Se rechaza la hipotesis nula del test de Iman-Davenport
Test de wilcoxon arbol vs bagging:
    WilcoxonResult(statistic=6.0, pvalue=0.0546875)
Test de wilcoxon svc vs bagging:
    WilcoxonResult(statistic=15.0, pvalue=0.42578125)
```

Posteriormente, aplicando los modelos boosting, se vuelve a aplicar el test de Iman-Davenport que volverá a rechazar la hipotesis nula y por tanto se vuelve a aplciar Wilcoxon.

```
MODELOS BOOSTING
Se rechaza la hipotesis nula del test de Iman-Davenport
Test de wilcoxon adabost vs arbol:
    WilcoxonResult(statistic=22.0, pvalue=1.0)
Test de wilcoxon adabost vs svc:
    WilcoxonResult(statistic=14.0, pvalue=0.359375)
Test de Wilcoxon gradientbost vs arbol:
    WilcoxonResult(statistic=0.0, pvalue=0.011718685599768628)
Test de Wilcoxon gradientbost vs svc:
    WilcoxonResult(statistic=17.0, pvalue=0.5703125)
```

3.3. Conclusiones del estudio

Utilizar un clasificador de árbol de decisión o un clasificador basado en el algoritmo de vectores de soporte (SVM) puede tener diferentes ventajas y desventajas en función del conjunto de datos y el problema específico. Sin embargo, cuando se utilizan técnicas de Boosting o Bagging, se pueden mejorar los resultados de estos clasificadores.

Boosting consiste en combinar varios clasificadores débiles para producir uno fuerte. Al combinar varios modelos, se pueden reducir los errores cometidos por cada modelo individual y mejorar la precisión global del modelo. Además, Boosting puede ser especialmente efectivo para manejar conjuntos de datos con un gran número de características o con un gran desequilibrio en la distribución de las clases.

Por otro lado, Bagging consiste en entrenar varios modelos independientes con diferentes subconjuntos de los datos de entrenamiento y luego combinarlos para producir un modelo final. Al entrenar varios modelos independientes,

se pueden reducir los errores cometidos por un modelo debido a un conjunto de datos específico y mejorar la robustez del modelo.

En resumen, utilizar técnicas de Boosting o Bagging con un clasificador de árbol de decisión o SVM puede mejorar el rendimiento de estos clasificadores. Sin embargo, es importante evaluar el rendimiento del modelo en función del conjunto de datos y del problema específico antes de decidir qué técnica utilizar.

4. Práctica 2.4

El objetivo de esta práctica es introducir los conceptos de clasificación usando métodos multiclase y comparar con los clasificadores usando más de dos clases de forma directa.

4.1. Selección de algoritmo de clasificación capaz de resolver problemas de más de dos clases.

- **Dataset Iris**. Conjunto de datos que contiene tres clases de cincuenta instancias cada una, donde cada clase se refiere a un tipo de planta de iris. Los atributos contienen la siguiente información: longitud y anchura del sépalo y longitud y anchura del pétalo. Todos los atributos son de tipo *float64* y la clase de tipo *object*.En la figura 30 se muestra las cinco primeras instancias. Las clases de este conjunto de datos son: Iris-Setosa, Iris-versicolor e Iris-virginica.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Figura 30: Iris Dataset

- **Dataset Car Evaluation**. Conjunto de datos con diferentes características propias del vehículo. Este conjunto de datos deriva de un modelo de decisión jerárquica simple pero será utilizada sin tener en

cuenta dicha jerarquía. Tiene mil setecientos veintiocho instancias con seis atributos: precio del vehículo, precio del mantenimiento, número de puertas, número de asientos, tamaño del maletero y seguridad. Tienen cuatro clases indicando de menor a mayor valor del vehículo. En la figura 31 se muestra las cinco primeras instancias. Las clases de este conjunto de datos son *unacc, acc, vgood, good.*

	buying	maint	doors	persons	lug_boot	safety	class
0	4	4	2	2	1	1	unacc
1	4	4	2	2	1	2	unacc
2	4	4	2	2	1	3	unacc
3	4	4	2	2	2	1	unacc
4	4	4	2	2	2	2	unacc

Figura 31: Car Dataset

En este conjunto de datos, se han realizado modificaciones en los valores de los atributos para un mejor estudio de los datos. Los datos eran de tipo categórico y se han cambiado a tipo numérico,ratio, entero a partir de la función *map*.

- **Dataset Vino**. Conjunto de datos que se dedican a determinar el origen de los vinos mediante análisis químicos. Los atributos son alcohol, ácido málico, ash, alcalinidad de ash, magnesio, fenoles totales, flavanoides, fenoles no flavanoides, proantocianinas, intensidad del color, tonalidad, OD280/OD315 de vinos diluidos y prolina. En la figura 32 se muestra las cinco primeras instancias. Las clases de este conjunto de datos son *1,2 y 3* según el tipo de vino.

	class	alcohol	malic acid	ash	alcalinityOfAsh	Magnesium	Total Phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color Intensity	Hue	OD280	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

Figura 32: Wine Dataset

- **Dataset Glass**. Este conjunto de datos determina el tipo de vidrio, por tanto tiene 7 clases entre ellas ventana flotante o no flotante, o ventanilla de coche flotante. Esta conjunto de datos nace tras un estudio de

clasificación de tipos de vidrio que fue motivado por una investigación criminológica. En el que el vidrio puede utilizarse como prueba si se identifica correctamente. En la figura 33 se muestra las cinco primeras instancias. Las clases de este conjunto de datos son: *build wind float, vehic wind float, tableware, build wind non-float, headlamps, containers* en función del tipo de vidrio y sus correspondientes componentes.

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.51793	12.79	3.50	1.12	73.03	0.64	8.77	0.0	0.00	'build wind float'
1	1.51643	12.16	3.52	1.35	72.89	0.57	8.53	0.0	0.00	'vehic wind float'
2	1.51793	13.21	3.48	1.41	72.64	0.59	8.43	0.0	0.00	'build wind float'
3	1.51299	14.40	1.74	1.54	74.55	0.00	7.59	0.0	0.00	tableware
4	1.53393	12.30	0.00	1.00	70.16	0.12	16.19	0.0	0.24	'build wind non-float'

Figura 33: Glass Dataset

- **Dataset Segment challenge**. Este conjunto de datos es un subconjunto de los datos de segmentación, es decir un subconjunto de los datos originales de entrenamiento y prueba, obteniendo las instancias al azar. La distribución de clase serán cielo, follaje, cemento, ventana camino, hierba o brickface. En la figura 34 se muestra las cinco primeras instancias. Este conjunto de datos tiene las siguientes clases: *cement, path, grass, window, foliage, brickface, sky.*

	region-centroid-col	region-centroid-row	region-pixel-count	short-line-density-5	short-line-density-2	vedge-mean	vedge-sd	hedge-mean	hedge-sd	intensity-mean	rawred-mean	rawblue-mean	rawgreen-mean	exred-mean	exblue-mean	exgreen-mean	value-mean	saturation-mean	hue-mean	class
0	144	35	9	0.0	0.0	2.333330	2.033060	2.05556	1.73098	37.5926	32.3333	47.4444	33.0000	-15.77780	29.55560	-13.7778	47.4444	0.319714	-2.13876	cement
1	118	180	9	0.0	0.0	1.944440	1.481990	3.11111	1.08866	40.5556	44.1111	59.0000	42.5556	-13.33330	31.33330	-18.0000	59.0000	0.278822	-1.99604	path
2	6	174	9	0.0	0.0	1.888890	1.007410	2.88889	4.02963	19.0741	15.1111	17.7778	24.3333	-11.88890	-3.88889	15.7778	24.3333	0.381867	2.39592	grass
3	152	220	9	0.0	0.0	0.944445	0.685185	1.44444	2.16296	14.6296	11.5556	13.1111	19.2222	-9.22222	-4.65556	13.7778	19.2222	0.416705	2.30688	grass
4	189	142	9	0.0	0.0	0.000000	0.000000	0.00000	0.00000	0.0000	0.0000	0.0000	0.0000	0.00000	0.00000	0.0000	0.0000	0.000000	0.00000	window

Figura 34: Segment Dataset

- **Abalone Dataset**. Este dataset predice la edad del abulón a partir de mediciones físicas. La edad del abulón se determina cortando la concha a través del cono, tiñéndola y contando el número de anillos a través de un microscopio. Existen otras medidas más fáciles de obtener que sirven para la predicción de edad, como patrones climáticos, ubicación, etc. Esta base de datos ha sido adaptada, ya que se han eliminado los valores faltantes y los rangos de valoers continuos se escalaron para su uso con una ANN (dividiendo por 200). Además, para el estudio más sencillo de la base de datos y clasificación se han modificado las variables categóricas a variable numérica. Las primeras instancias se

pueden observar en la figura 35. En este caso, existen distintas clases del 1 al 24.

	Sex	Lenght	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	1	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

Figura 35: Abalone Dataset

- **Leaf Dataset**. Este conjunto de datos consiste en una colección de características de forma y textura extraídas de imágenes digitales de especímenes de hojas provenientes de un total de 40 especies de plantas diferentes. En este caso, este dataset es una reducción de la original y solo se tienen en cuenta 30 tipos de hojas que serán las clases estudiadas. Las primeras cinco instancias están reflejadas en la figura 36.

	class	Specimen Nº	Eccenticity	Aspect Ratio	Elongation	Solidity	Stochastic Convexity	Isoperimetric Factor	Maximal Indentation Depth	Lobedness	Intensity	Contrast	Smoothness	Thids moment	Uniformity	Entropy
0	1	2	0.74173	1.5257	0.36116	0.98152	0.99825	0.79867	0.005242	0.005002	0.024160	0.090476	0.008119	0.002708	0.000075	0.69659
1	1	3	0.76722	1.5725	0.38998	0.97755	1.00000	0.80812	0.007457	0.010121	0.011897	0.057445	0.003289	0.000521	0.000038	0.44348
2	1	4	0.73797	1.4597	0.35376	0.97566	1.00000	0.81697	0.006877	0.008607	0.015950	0.065491	0.004271	0.001154	0.000066	0.58785
3	1	5	0.82301	1.7707	0.44462	0.97698	1.00000	0.75493	0.007428	0.010042	0.007938	0.045339	0.002051	0.000560	0.000024	0.34214
4	1	6	0.72997	1.4892	0.34284	0.98755	1.00000	0.84482	0.004945	0.004451	0.010487	0.058528	0.003414	0.001125	0.000025	0.34068

Figura 36: Leaf Dataset

- **Soybean dataset**. Esta es la famosa base de datos de enfermedades de la soja de Michalski. Consta de 19 clases el dataset original, de las cuales solo cuatro son utilizadas para esta muestra. En la Figura 37 se muestran las primeras instancias.

	date	plant-stand	precip	temp	hail	corp-hist	area-damaged	severity	seed-tmt	germination	...	sclerotia	fruit-pods	fruit spots	seed	mold-growth	seed-discolor	seed-size	shriveling	roots	class
0	4	0	2	1	1	1	0	1	0	2	...	0	0	4	0	0	0	0	0	0	D1
1	5	0	2	1	0	3	1	1	1	2	...	0	0	4	0	0	0	0	0	0	D1
2	3	0	2	1	0	2	0	2	1	1	...	0	0	4	0	0	0	0	0	0	D1
3	6	0	2	1	0	1	1	1	0	0	...	0	0	4	0	0	0	0	0	0	D1
4	4	0	2	1	0	3	0	2	0	2	...	0	0	4	0	0	0	0	0	0	D1

Figura 37: Soybean Dataset

- **Dataset Zoo**. Una base de datos simple que contiene 17 atributos con valores booleanos, siendo el atributo 'tipo' el atributo de clase en las que cada tipo, muestra un conjunto de animales propios de es tipo. Los atributos a tener en cuenta serán el pelo, huevos, leche, patas, cola, domésticos, entre muchos de ellos. En la figura 38 se muestra las cinco primeras instancias y debe tenerse en cuenta que tiene 6 clases, siendo los tipos de animal correspondiente.

	name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	type
0	aardvark	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
1	antelope	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
2	bass	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
3	bear	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
4	boar	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1

Figura 38: Zoo Dataset

4.2. Aplicación de clasificador base a cada conjunto

Para este apartado, debe elegirse un modelo capaz de clasificar un conjunto de datos multiclase y no simplemente binario. Por ello, la elección del clasificador base ha sido elegir una máquina de vectores soporte, obteniendo los resultados gráficamente39. Además haciendo uso de funciones realizadas para los casos anteriores, se aplicará el algoritmo *GridSearchCV* para determinar los hiperparámetros más eficientes y así encontrar un mejor rendimiento del problema y una mejor precisión.

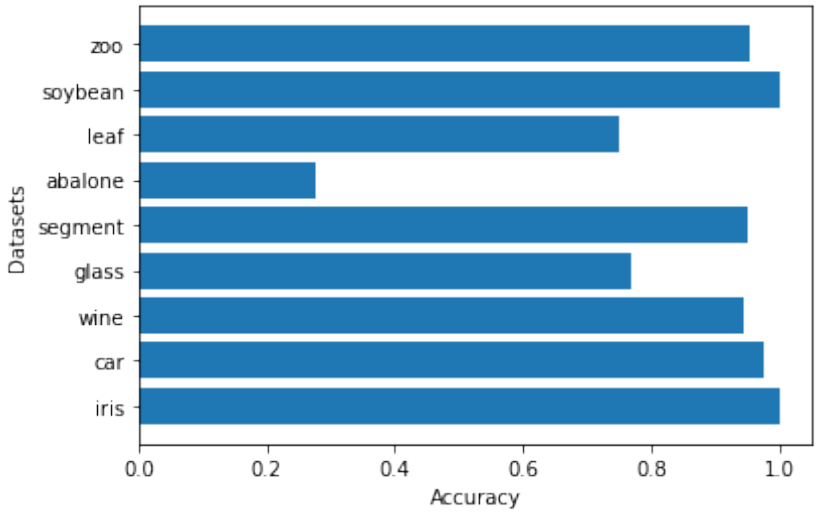


Figura 39: SVM Multiclase

	Iris	Car	Wine	Glass	Segment	Abalone	Leaf	SoyBean	Zoo
SVM	1.0	0.9740	0.9444	0.7674	0.9506	0.2763	0.75	1.0	0.9524
OVO	1.0	0.9711	0.9444	0.6047	0.9506	0.2752	0.7059	1.0	0.9524

Tabla 4: Resultados SVM, One vs One

4.3. Métodos múlticlase one-vs-one, one-vs-all, y error correcting output codes.

4.3.1. Método One vs. One (OVO)

Tanto en forma de Tabla 4 como gráficamente 40, se observa que en algunos métodos la aplicación del método OVO no provoca buenos resultados como se ve en el conjunto de datos leaf, glass o car. Algunos casos, se observa como no mejora la precisión o simplemente, queda con una precisión muy cercana como se ve en la Tabla 4 .

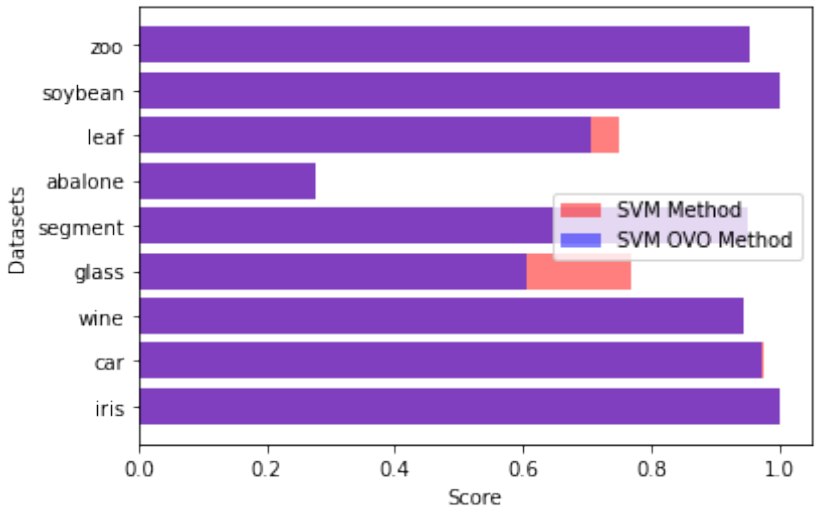


Figura 40: SVM OVO vs SVM Method

4.3.2. Método One vs All (OVA)

El siguiente método es uno contra todos, que al igual que el anterior, se observa que el método base obtiene resultados iguales o mejores en muchos casos. El motivo está explicado posteriormente al finalizar dicho apartado. En este caso los resultados son de la Tabla 5 y gráfica 41.

	Iris	Car	Wine	Glass	Segment	Abalone	Leaf	SoyBean	Zoo
SVM	1.0	0.9740	0.9444	0.7674	0.9506	0.2763	0.75	1.0	0.9524
OVO	1.0	0.9711	0.9444	0.6047	0.9506	0.2752	0.7059	1.0	0.9524
OVA	0.8667	0.9740	0.9444	0.4884	0.9012	0.1962	0.6618	1.0	0.9524

Tabla 5: Resultados comparativa, SVM base, One vs One, One vs All

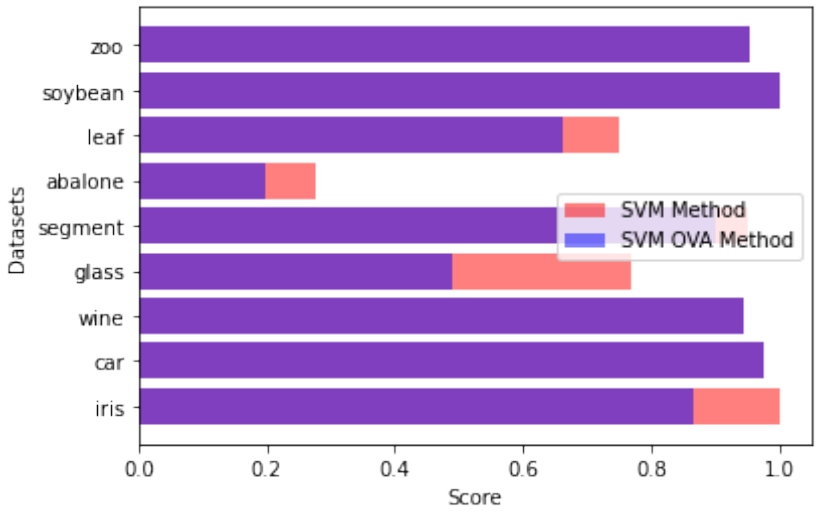


Figura 41: SVM OVA vs SVM Method

	Iris	Car	Wine	Glass	Segment	Abalone	Leaf	SoyBean	Zoo
SVM	1.0	0.9740	0.9444	0.7674	0.9506	0.2763	0.75	1.0	0.9524
OVO	1.0	0.9711	0.9444	0.6047	0.9506	0.2752	0.7059	1.0	0.9524
OVA	0.8667	0.9740	0.9444	0.4884	0.9012	0.1962	0.6618	1.0	0.9524
ECOC	0.5667	0.9335	0.9444	0.5349	0.7901	0.2666	0.3823	1.0	1.0

Tabla 6: Resultados SVM, OVO, OVA y ECOC

4.3.3. Error Correcting Output codes (ECOC)

Como en los casos anteriores, vuelve a darnos resultados menos satisfactorios observables en la Tabla 6 y Figura 42

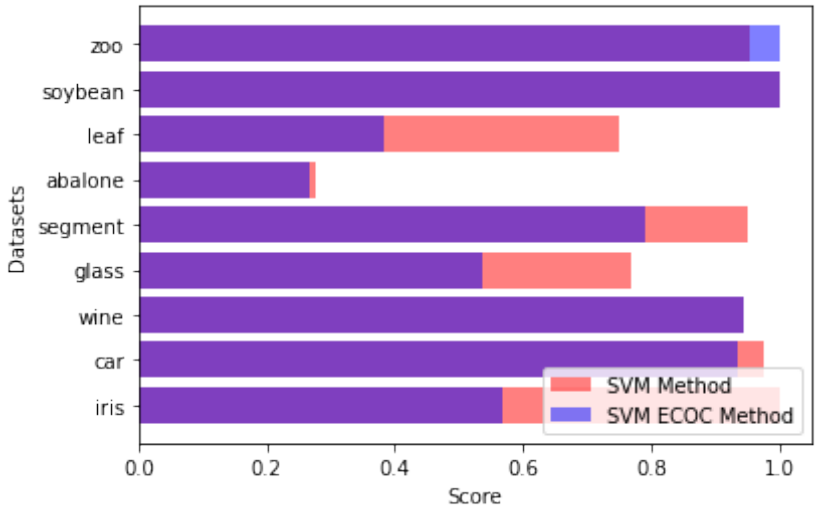


Figura 42: SVM ECOC vs SVM Method

La precisión puede ser menor en estos casos (OVO, OVA, ECOC) debido a la complejidad que se agrega al modelo y la posibilidad de errores de votación en la etapa de decisión. Además, la división de las clases en múltiples clasificadores binarios puede resultar en una pérdida de información y una menor capacidad de generalización del modelo. Por tanto, podemos concluir que por ello el modelo base utilizado, SVM original es más preciso en la mayoría de conjunto de datos debido a la simplicidad y la capacidad de generalización más alta de su enfoque de un solo clasificador para todas las clases.

4.4. Diferencias significativas usando test Iman-Davenport y aplicación de Wilcoxon.

El test Iman-Davenport es test estadístico para determinar si la diferencia en los valores de precisión entre los métodos son estadísticamente significativas, cuando esto ocurre se utiliza el método Wilcoxon para comparar cada método multiclase con el clasificador base y las diferencias entre ellos.

En este caso, el p-value es mayor a 0.05 por lo que se puede concluir que no existen diferencias significativas entre los métodos.

Referencias

[1] Marcos Bermejo. *UCI Machine Learning Repository*. URL: `https://archive.ics.uci.edu/ml/datasets.php`. [En Línea. Última consulta: 23-01-2023].

[2] *Weka Data Sets*. URL: `https://storm.cis.fordham.edu/~gweiss/data-mining/datasets.html`. [En Línea. Última consulta: 23-01-2023].

[3] *Scikit-learn*. URL: `https://scikit-learn.org/stable/index.html`. [En Línea. Última consulta: 25-01-2023].

[4] *Decision Trees*. URL: `https://scikit-learn.org/stable/modules/tree.html`. [En Línea. Última consulta: 25-01-2023].

[5] *KNeighborsClassifier*. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`. [En Línea. Última consulta: 25-01-2023].

[6] *Máquinas Vectores Soporte*. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`. [En Línea. Última consulta: 25-01-2023].

[7] *Crosvalidación Cruzada*. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html`. [En Línea. Última consulta: 25-01-2023].

[8] *Scikit Posthocs*. URL: `https://scikit-posthocs.readthedocs.io/en/latest/tutorial/`. [En Línea. Última consulta: 25-01-2023].

[9] *Wittgenstein*. URL: `https://pypi.org/project/wittgenstein/`. [En Línea. Última consulta: 26-01-2023].