

Informe - Laboratorios [1-7]

Minería de datos

Aingeru García Blas

Índice

0. Introducción	3
1. Laboratorio 1: Documentos TV: 'Big data, conviviendo con el algoritmo'	4
1.1. Análisis de las aplicaciones mencionadas	4
1.2. Entrevistados destacados	5
1.3. Reflexiones destacadas	5
1.4. Riesgos y dilemas éticos	5
1.5. Reflexión personal	5
1.6. Recomendaciones para futuros alumnos	6
2. Laboratorio 2: Clasificador del vecino más próximo K-NN con evaluación mediante Hold-Out	7
2.1. Descripción del conjunto de datos	7
2.2. Especificaciones del dataset:	7
2.3. Parámetros del clasificador IBk	8
2.4. Exploración de <i>distanceWeighting</i>	8
2.5. Experimentos y resultados	8
2.6. Reflexión sobre la familia <i>lazy</i> en Weka	9
3. Laboratorio 3: Evaluación de clasificadores: métodos de evaluación	10
3.1. Métodos de evaluación en Weka	10
3.2. Análisis de los resultados	11
4. Laboratorio 4: Preprocesamiento de datos - filtros previos al modelado	12
4.1. Aplicación del filtro <i>Discretize</i> al dataset <i>iris.arff</i>	12
4.2. Aplicación del filtro <i>ReplaceMissingValues</i> al dataset <i>hepatitis.arff</i>	13
4.3. Normalización y Estandarización en <i>iris.arff</i>	14
4.4. Exploración de otros filtros	15
4.5. Uso de <i>Filter - MultiFilter</i>	15
5. Laboratorio 5: Árboles de clasificación – Decision trees	16
5.1. Sobre el dataset <i>strava_actividad.arff</i>	16
5.2. Primer contacto con ambos árboles	17
5.3. Breve comparación entre la versión podada y sin podar	17
5.4. Interpretación de los Números en Hojas del Árbol Podado	18
5.5. Validación del Árbol	18
5.6. Aplicación del Modelo en Producción	19
6. Laboratorio 6: kaggle.com; datasets y competiciones para Machine Learning	20
6.1. Detalles de la Competición	20
7. Laboratorio 7: Selección de variables y PCA	23
7.1. Selección de Variables: Filter vs. Wrapper	23
7.2. Conjunto de datos <i>adult.arff</i>	23
7.3. Ranking de variables predictoras	24
7.4. Relevancia en modelos de clasificación	25
7.5. Clasificador K-NN	25
7.6. Selección de Variables y Redundancia	26
7.7. Correlación en Weka	26
7.8. Wrapper Feature Selection en Weka	27
7.9. Mérito del mejor subconjunto encontrado:	27
7.10. Evaluador de Subconjunto de Atributos:	27
7.11. Feature Selection vs Feature Extraction	28

7.12. Explicación del código en R	30
---	----

0. Introducción



Figura 1: Minería de Datos

Estamos en un punto emocionante donde la tecnología y el conocimiento humano se cruzan; los datos están en el centro de todo. Este proyecto explora cómo convivimos con los algoritmos y cómo diseñamos herramientas para entender grandes cantidades de datos.

El trabajo en los laboratorios nos introduce a las técnicas clave del aprendizaje automático y nos hace pensar en los aspectos éticos de trabajar con inteligencia artificial. Hemos visto cómo los algoritmos impactan nuestra vida diaria y nos hemos enfrentado a retos prácticos en la clasificación y el tratamiento de datos. Por ejemplo, en el Laboratorio 6, nos zambullimos en Kaggle, donde personas de todo el mundo apasionadas por la ciencia de datos resuelven problemas reales. Uno de estos, el de predecir la localización de proteínas en células, me ha atraído especialmente.

Este proyecto no solo trata sobre técnicas y algoritmos, sino que refleja nuestra curiosidad y habilidad para abordar desafíos. A pesar de que aún nos queda muchísimo por ver ya que esto ha sido una mera introducción, lo que hemos aprendido hasta ahora ha sido increíble. Espero que al leer este trabajo, se pueda percibir tanto la pasión como el entusiasmo que he sentido en cada etapa del proceso.

1. Laboratorio 1: Documentos TV: 'Big data, conviviendo con el algoritmo'

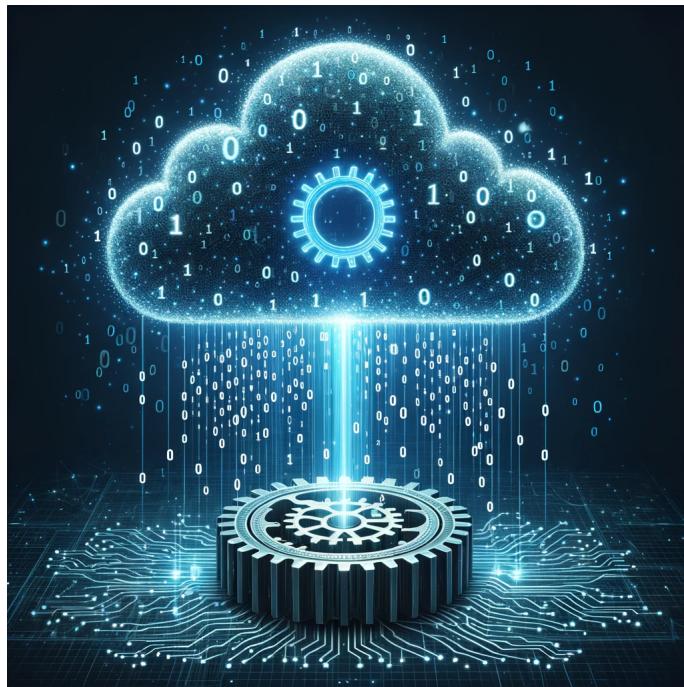


Figura 2: Big data

1.1. Análisis de las aplicaciones mencionadas

Dejando a un lado que todo lo que tenga que ver con videojuegos ya llama mi atención (y en el documental, me ha parecido muy interesante la competición presentada), algo que (**inicialmente***) me ha encantado ha sido **Websays** - tanto por su (posible) utilidad, como por su complejidad. Me ha llevado a investigar un poco sobre sus características y técnicas que utilizan:

<https://websays.com/es/>

Características y técnicas de Websays

1. **Análisis de Sentimiento:** Utiliza técnicas de procesamiento de lenguaje natural (NLP) para determinar el sentimiento de comentarios online.
2. **Monitorización en Tiempo Real:** Permite monitorear menciones y opiniones en tiempo real.
3. **Análisis de Tendencias:** Identifica cambios en la percepción de la marca a lo largo del tiempo.
4. **Cobertura Global:** Analiza datos en varios idiomas.
5. **Alertas:** Configuración de notificaciones para menciones o temas específicos.
6. **Visualización de Datos:** Ofrece dashboards y visualizaciones gráficas para entender los datos.

(*) No obstante, he de decir que me llevé una sensación al ver el documental sobre esto, y al conocer un poco más que es lo que realmente hacen, ésta sensación cambió mucho - en mi opinión semejante tecnología podría estar orientada a otros puntos clave sociales, éticos y menos comerciales. En definitiva, generar estadísticas objetivas sobre puntos de interés y ver, *hacia donde se mueve el mundo*. En una primera instancia, pensaba que era un análisis más a gran escala de temas más interesantes que podrían dar una perspectiva de qué se habla en internet sobre problemas reales o temas de gran impacto. Pero, desgraciadamente, el mundo funciona como funciona y todo tiene un interés económico subyacente.

1.2. Entrevistados destacados

A mi parecer, Elisa Martin (Directora de Tecnología de e Innovación de IBM), ha capturado mi atención tanto con lo que tenía que decir como con el cómo lo ha dicho. Me ha parecido una persona coherente y además, los temas que ha tratado como los sistemas cognitivos y cómo son generados, me resultan de grandísimo interés.

A parte, me he llevado una sorpresa con Chema Alonso, ya que no conocía mucho de él pese a su fama y notoriedad; me ha parecido humilde y más 'normal' de lo que me esperaba.

1.3. Reflexiones destacadas

Me ha parecido muy interesante la reflexión que ha proporcionado Chema Alonso. De manera implícita ha comentado varios elementos relacionados a nuestra huella digital y cómo todo lo que hagamos deja rastro; y cómo - en un futuro - esas huellas pueden influir en decisiones clave que afecten nuestras vidas.

1.4. Riesgos y dilemas éticos

Está claro que la posible falta de privacidad es un riesgo que estará ahí, y más aún al ritmo vertiginoso al que se está desarrollando todo que. Además, como bien sabemos: «*hecha la ley, hecha la trampa*». Por ejemplo, el hecho de que datos tan relevantes como nuestra geolocalización, mensajes privados, bases de datos enormes con nuestras preferencias etc, estén en la nube y se generen estadísticas con ellos, puede resultar en algo bastante peligroso dependiendo de quien lo gestione. Considero ese un punto clave, que no deberíamos de pensar en 'dependiendo de quién lo gestione'. Es un tema muy delicado del que, en el día a día, la mayoría de las personas no somos conscientes.

1.5. Reflexión personal

Es muy complicado el determinar quienes son los que tienen que establecer las reglas para que todo esté regulado y podamos convivir con una inteligencia artificial 'alineada' con los valores éticos del ser humano. Entiendo que, organizaciones internacionales públicas deberían de ser las que validasen o no la publicación o lanzamientos de grandes proyectos de inteligencia artificial ya que, como bien sabemos, si se privatiza, existirá una tendencia a velar por los intereses que favorezcan económicamente.

1.6. Recomendaciones para futuros alumnos

Después de revisar varios contenidos en línea, he encontrado un par de videos muy interesantes:

- **How Far is Too Far? | The Age of A.I.**: Ironman, en un día libre y enmascarado como Robert Downey Jr. explora los límites y posibilidades de la inteligencia artificial en nuestra sociedad actual.

<https://www.youtube.com/watch?v=UwsrzCVZAb8>

- **AlphaGo - The Movie: documentary**: Un recuento del desafío histórico entre el programa de inteligencia artificial AlphaGo y el campeón mundial de Go, Lee Sedol.

<https://www.youtube.com/watch?v=WXuK6gekU1Y>

2. Laboratorio 2: Clasificador del vecino más próximo | K-NN con evaluación mediante Hold-Out

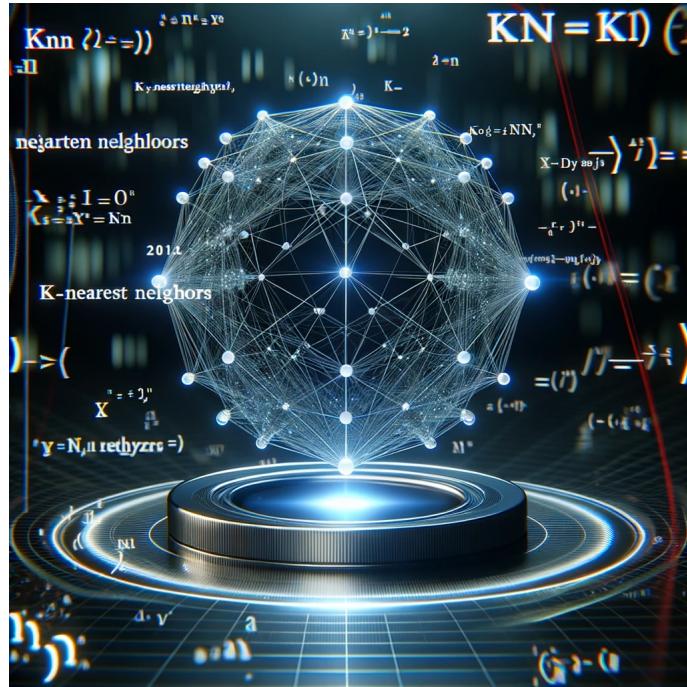


Figura 3: K-Nearest Neighbour

2.1. Descripción del conjunto de datos

Tras analizar el dataset `CoverTypePrediction-Kaggle.arff`, el problema a clasificar es la cobertura forestal o *Cover Type*, determinando qué tipo de árbol se ha de plantar en función de las variables predictoras. Se puede apreciar cómo se aborda un problema que tiene una alta relevancia para la silvicultura y la reforestación, que permite tomar decisiones informadas sobre qué árboles plantar en diferentes condiciones.

Las variables predictoras son 54 en total, incluyendo campos de datos como aspecto, pendiente, distancia horizontal y vertical a fuentes de agua, sombra en distintos momentos del día, entre otros, como muestro a continuación:

2.2. Especificaciones del dataset:

- Número de casos-muestras: **15,120**
- Número de variables: 54
- Naturaleza de las variables: Todas numéricas.
- Número de valores de la variable clase a predecir: 7 (**Spruce/Fir**, **Lodgepole Pine**, **Ponderosa Pine**, **Cottonwood/Willow**, **Aspen**, **Douglas-fir**, **Krummholtz**)

2.3. Parámetros del clasificador IBk

El clasificador IBk tiene varios parámetros, entre ellos:

- **KNN:** Es el número de vecinos más cercanos a considerar en la clasificación.
- **distanceWeighting:** Determina cómo se ponderan las contribuciones de los vecinos basadas en su distancia al punto de consulta.

2.4. Exploración de distanceWeighting

Los valores del parámetro

`distanceWeighting` incluyen:

- **NoDistanceWeighting:** No se considera la distancia; todos los vecinos tienen igual peso en la votación.
- **Weight by 1/distance:** La ponderación es inversamente proporcional a la distancia, por lo que los vecinos más cercanos tienen más influencia en la votación.

2.5. Experimentos y resultados

He realizado pruebas variando los valores de **KNN** y **distanceWeighting**, y se puede observar que con **K=1**, tanto para **1/distance** como para **NoDistanceWeighting**, se obtiene una tasa de acierto del 79.1286 %. Sin embargo, al aumentar **K** a 8, la tasa de acierto disminuye a 76.9111 % para **1/distance** y a 74.3435 % para **NoDistanceWeighting**.

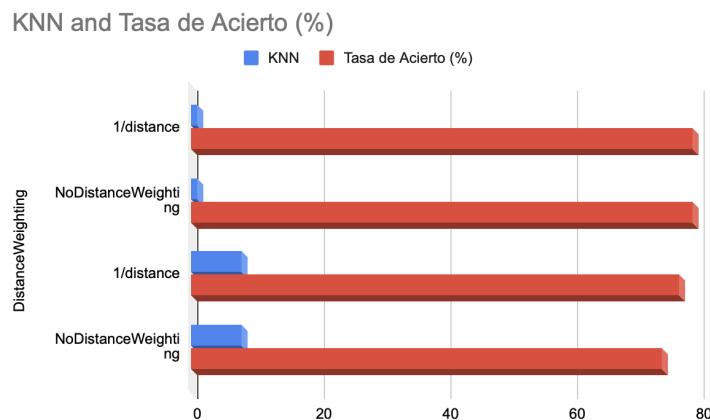


Figura 4: Gráfica mostrando la variación de la tasa de acierto según los parámetros

2.6. Reflexión sobre la familia *lazy* en Weka

Weka clasifica el clasificador IBk dentro de la familia *lazy* porque este tipo de sistema pospone las tareas de clasificación hasta el último momento. Realiza todo el trabajo en tiempo de consulta y no guarda ningún resultado de aprendizaje. Además, basa sus decisiones en instancias específicas (K-instancias) en lugar de generalizar basándose en todo el conjunto de datos y necesita almacenar todo el conjunto de datos para su funcionamiento.

3. Laboratorio 3: Evaluación de clasificadores: métodos de evaluación

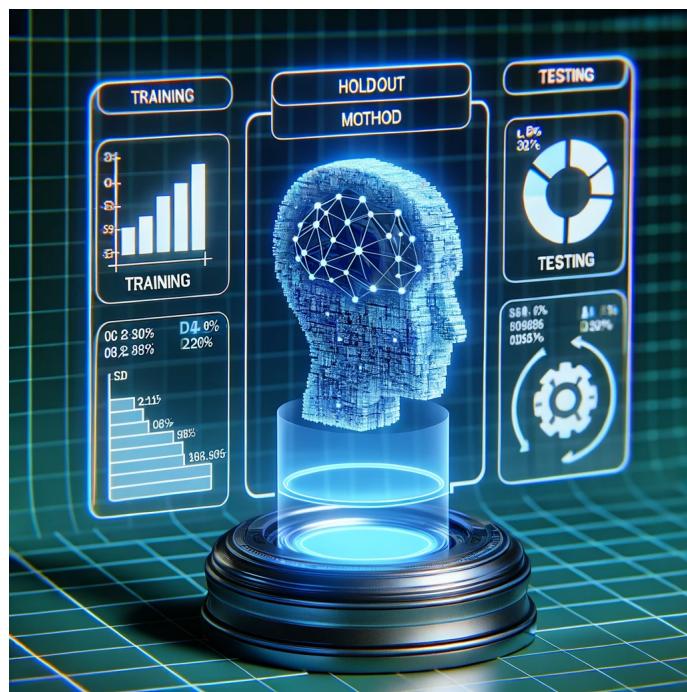


Figura 5: Evaluación de modelos

3.1. Métodos de evaluación en Weka

1. **Método de resustitución - error aparente:** Este método evalúa el rendimiento del modelo en el mismo conjunto de datos que se utilizó para entrenarlo. En Weka, la opción para implementar este método es *use training set*.
2. **Método H, hold-out con 66%:** Este método divide el conjunto de datos en un conjunto de entrenamiento (66 %) y un conjunto de prueba (34 %) para evaluar el rendimiento del modelo.
3. **Método H repetido 5 veces con un promedio de los resultados de validación:** Similar al método H, pero se repite 5 veces con diferentes divisiones aleatorias y se toma el promedio de los resultados.
4. **10-fold cross-validation:** El conjunto de datos se divide en 10 partes o capas, y el modelo se entrena y se prueba 10 veces, cada vez usando una parte diferente como conjunto de prueba y las otras 9 como conjunto de entrenamiento.
5. **5-fold cross-validation:** Similar al 10-fold, pero con solo 5 capas.
6. **Leave-one-out:** Para aplicar este método en Weka, debemos poner el valor igual al número total de instancias en *folds*. Es un caso especial de k-fold cross-validation donde k es igual al número de instancias. En cuanto a los tiempos de cómputo, son altos porque se realiza una validación cruzada para cada instancia en el conjunto de datos.

3.2. Análisis de los resultados

- **¿Devuelven todos el mismo porcentaje de error estimado?**: No, cada método de evaluación puede devolver un porcentaje de error diferente debido a la forma en que se divide el conjunto de datos y se evalúa el modelo:

Método de Evaluación	Error Estimado
Resustitución - Error Aparente	0.0005
Hold-out (Entrenando con 66 %)	0.1356
Estimación basada en 10 rodajas	0.1378
Estimación basada en 5 rodajas	0.1403
Leave-One-Out	0.1352

- **Mejor método en términos de error estimado**: Como era de esperar ya que es un método de evaluación *no honesto*, el que menos error estimado posee es el **Resustitución - Error Aparente**, ya que evalúa los datos con el mismo conjunto con el que entrenó. El modelo no es honesto y ha hecho trampas: *ya había visto los datos*.
- **Estimación más fiable**: La validación cruzada, especialmente con un número mayor de **folds** (siendo el **leave-one-out** el que más capas utiliza), suele considerarse una de las estimaciones más fiables porque el modelo se evalúa en diferentes subconjuntos del conjunto de datos y como se puede comprobar en la tabla, así lo demuestran los resultados.
- **Método que más exige a Weka en términos de cómputo**: Leave-one-out, ya que implica entrenar y evaluar el modelo tantas veces como instancias haya en el conjunto de datos.
- **Número de modelos aprendidos por Weka en 10-fold cross-validation**: Durante el proceso de 10-fold cross-validation en Weka, se entrena y evalúan 10 modelos diferentes, uno para cada partición. Sin embargo, al final del proceso, Weka también entrena un modelo final utilizando todo el conjunto de datos, lo que lleva a un total de 11 modelos: 10 para la validación cruzada y 1 modelo final. Cada uno de los 10 modelos se entrena con el 90 % del conjunto de datos y el modelo final con el 100 %.
- **Posibilidad de que un clasificador sea mejor con una seed-semilla y peor con otra**: Sí, es posible, especialmente en métodos que involucran aleatoriedad en la división del conjunto de datos. Diferentes **seeds** pueden llevar a diferentes divisiones y, por lo tanto, a diferentes resultados.

4. Laboratorio 4: Preprocesamiento de datos - filtros previos al modelado

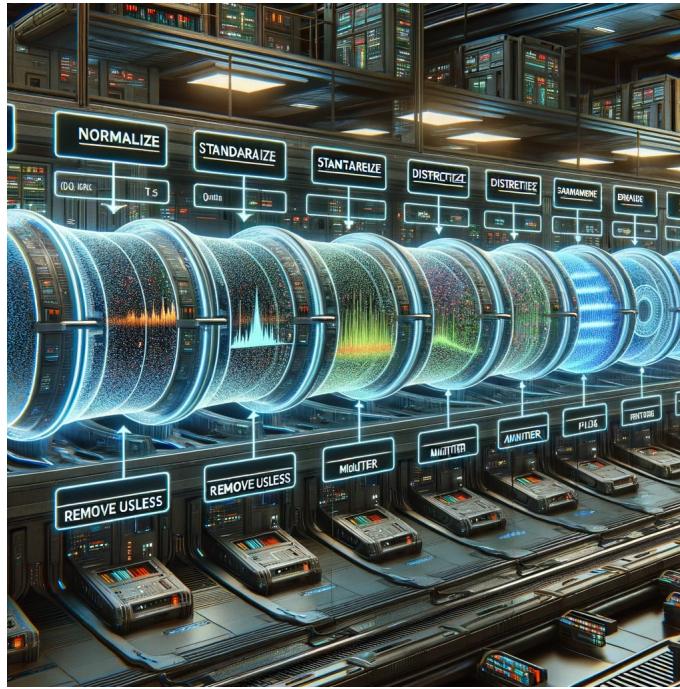


Figura 6: Preprocesamiento de datos

4.1. Aplicación del filtro Discretize al dataset `iris.arff`

- **Aplicación con `useEqualFrequency=True`:** La aplicación de este filtro con la configuración dada discretiza el atributo especificado en `bins` basados en la frecuencia. El filtro *puede o no* actuar sobre las variables *nominales/categóricas* dependiendo de *la naturaleza de las variables* (y si se selecciona esta opción). En general, se podría decir que se centra en convertir atributos numéricos en atributos nominales al dividir el rango de números en intervalos discretos.

Concretamente, para el `useEqualFrequency`, el algoritmo intentará dividir los datos en intervalos de manera que cada intervalo contenga (**aproximadamente**) la misma cantidad de instancias. Para la variable `petallength` ha creado 4 rangos en forma de intervalos, y la ha **convertido en nominal**:

Cuadro 1: `useEqualFrequency=True`

Índice	Intervalo	Count	Weight
1	($-\infty, 1,55]$	37	37
2	(1,55, 4,35]	38	38
3	(4,35, 5,15]	41	41
4	(5,15, $\infty)$	34	34

- **Aplicación con useEqualFrequency=False:** Al aplicar el filtro con esta configuración, se discretiza de nuevo el atributo basándose en el rango de valores, al igual que en el caso anterior ha convertido en nominal a `petallength`, pero en este caso, ha cambiado ya que no ha intentado distribuir las instancias en conjuntos de tamaño igual o similar:

Cuadro 2: useEqualFrequency=False

Índice	Intervalo	Count	Weight
1	($-\infty$, 2,475]	50	50
2	(2,475, 3,95]	11	11
3	(3,95, 5,425]	61	61
4	(5,425, ∞)	28	28

4.2. Aplicación del filtro ReplaceMissingValues al dataset hepatitis.arff

- **Función del filtro:** Identifica valores faltantes en un conjunto de datos y los reemplaza con un método específico: para atributos numéricos, utiliza la media de ese atributo, y para atributos nominales, utiliza el valor modal (el más frecuente). Es una herramienta para gestionar y limpiar datos incompletos.
- **Efecto sobre la variable ALK_PHOSPHATE:**

Al aplicar el filtro `ReplaceMissingValues` en Weka sobre la variable, se pueden observar los siguientes cambios en sus estadísticos descriptivos:

1. **Valores Faltantes (Missing):** Como es evidente, después de aplicar el filtro, no hay valores faltantes en la variable, es decir, los 29 valores faltantes originales han sido reemplazados.
2. **Valores Únicos y Distintos:** Se puede apreciar una reducción en el número de valores únicos y distintos. Esto tiene sentido porque al reemplazar valores faltantes con estadísticas como la media, es probable que se introduzcan valores que ya están presentes en el conjunto de datos.
3. **Media (Mean) y Desviación Estándar (StdDev):** La media se ha reducido ligeramente y la desviación estándar también. Esto sugiere que los valores reemplazados tienden a ser algo menores que la media y *algo más homogéneos*, ya que no parecen estar tan dispersos, al bajar la desviación estandar.

Cuadro 3: Sin ReplaceMissingValues

Estadístico	Valor
Únicos	64
Distinct	83
Missing	29
Minimum	26
Maximum	295
Mean	105.325
StdDev	51.508

Cuadro 4: Con ReplaceMissingValues

Estadístico	Valor
Únicos	55
Distinct	75
Missing	0
Minimum	26
Maximum	295
Mean	103.858
StdDev	44.579

- **Opinión sobre su aplicación en la variable PROTIME:** Bueno, tenemos un **43 %** de valores faltantes para esta variable, entonces, si la variable es muy relevante, me chirriaría mucho, si. Tras investigar un poco sobre el conjunto de datos, veo que representa el tiempo de protrombina, que es una medida de cuánto tiempo tarda la sangre en coagularse - parece bastante relevante, entonces al imputar una variable podrías distorsionar demasiado los resultados.

4.3. Normalización y Estandarización en iris.arff

- **Efecto del filtro Normalize:** La normalización ajusta los valores de las variables numéricas para que estén en una escala común, el estándar suele ser entre 0 y 1. Al aplicar el filtro de *normalize* a un conjunto de datos, todas las variables numéricas han sido reescaladas de modo que sus valores mínimos y máximos sean **0 y 1**, respectivamente. A pesar de este cambio, los valores aún conservan sus proporciones relativas originales. Cabe destacar que este proceso no ha afectado en absoluto a las variables nominales o categóricas, ya que al representar categorías y no magnitudes, no tienen un rango numérico a reescalar.

La fórmula típica de normalización es:

$$valor = \frac{valor - min}{max - min} \quad (1)$$

- **Efecto del filtro Standardize:** centra y reescala los datos numéricos para que tengan una media de 0 y una desviación estándar de 1. Tras investigar un poco, veo que es útil para algoritmos sensibles a la escala de las características, ya que homogeneiza su rango y varianza. No obstante, veo cambios significativos en máximos y mínimos a través de las variables **numéricas**, ya que no afecta a las nominales-categóricas.

La formula es:

$$z = \frac{x - \mu}{\sigma} \quad (2)$$

Donde:

- z es el valor estandarizado.
- x es el valor original.
- μ es la media de la variable.
- σ es su desviación estándar.

4.4. Exploración de otros filtros

- **Filtro seleccionado:** RemoveUseless, ya que para simplificar el dataset, el filtro *RemoveUseless* parece bastante apropiado. Elimina aquellos atributos que no varían o varían en exceso. Para acceder al filtro RemoveUseless en Weka, sigue la ruta:

Weka>Filters>unsupervised>attribute>RemoveUseless

- **Dataset de elección:** hepatitis.arff
- **Efecto del filtro en el dataset:** Pues ha pasado algo bastante interesante - de 20 atributos iniciales que tiene el dataset, tras aplicar el filtro con una varianza máxima del 99 % permitida, siguen estando los 20. Y lo que es más interesante aún, hasta que no se reduce la varianza máxima permitida hasta un 1 %, no se reduce el número de atributos. En éste caso, va de 20 a 7.

Con ello deduzco que los atributos de este conjunto de datos tienen una **variabilidad muy baja y son de gran calidad**.

4.5. Uso de Filter - MultiFilter

Curiosidad

1. ¿Qué hace la acción Filter - multifilter en Weka?

El filtro `multifilter` en Weka permite aplicar múltiples filtros de manera sucesiva a un conjunto de datos.

2. ¿Sabrías utilizarlo ajustando sus parámetros?

En principio si, se selecciona `multifilter`, se puede configurar la lista de filtros al hacer click en `filters`, así como ajustar también sus parámetros.

5. Laboratorio 5: Árboles de clasificación – Decision trees

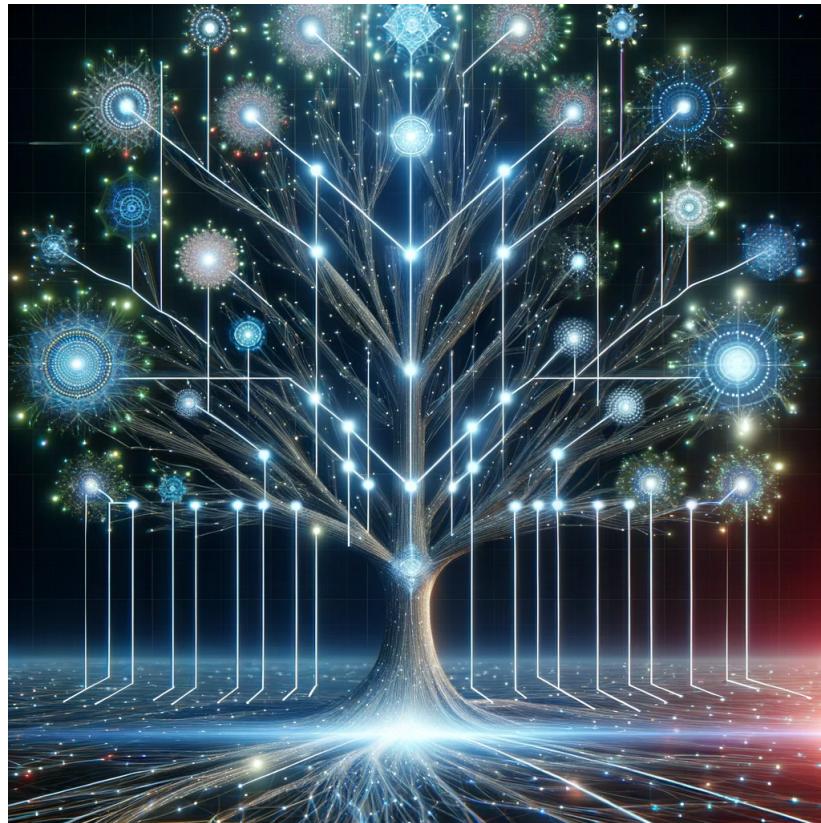


Figura 7: Decisional trees

5.1. Sobre el dataset strava_actividad.arff

Este conjunto de datos, denominado **strava_actividad.arff**, proviene de actividades deportivas recopiladas de la plataforma **Strava** por Unai Sainz de la Maza Gamboa. Estos datos han sido grabados con dispositivos GPS confiables (*Garmin Fenix 6, Suunto Ambit 2 y Garmin Edge 530*), lo que garantiza su precisión y autenticidad. Con un total de **1045** instancias y **5** atributos, este dataset ofrece una visión detallada de diferentes métricas asociadas a cada actividad.

- **Variables originales en árboles finales:**
 - **type:** Es una variable nominal que especifica el tipo de actividad realizada, ya sea una carrera (*Run*) o un paseo en bicicleta (*Ride*).
 - **distance:** Esta variable numérica representa la distancia total recorrida durante la actividad y se mide en metros.
 - **moving_time:** Indica el tiempo total en movimiento durante la actividad, es una variable numérica y se mide en segundos.
 - **elapsed_time:** Es otra variable numérica que mide el tiempo total, en segundos, desde el inicio hasta el final de la actividad, incluidos los descansos.
 - **total_elevation_gain:** Representa el desnivel total durante la actividad, es decir, cuánto se ha ascendido en total. Es una variable numérica y se mide en metros.

5.2. Primer contacto con ambos árboles

Los árboles de decisión generados por Weka, proporcionan una salida en formato ASCII que ofrece una visión textual y menos visual del modelo. Entre los diversos datos proporcionados, destacan dos valores: **Number of Leaves** y **Size of the tree**. Explico primero, el significado de cada uno de estos términos para luego exponer los datos concretos para cada uno de los árboles:

- **Number of Leaves (Número de hojas)**: Representa el número total de hojas en el árbol de decisión. Las hojas son los nodos finales del árbol donde se toma una decisión final sobre la instancia.
- **Size of the tree (Tamaño del árbol)**: Indica el número total de nodos en el árbol, incluidos tanto los nodos hoja como los nodos internos (decisiones, y como bien se ha explicado al comienzo del enunciado - basadas en la **Información Mutua** de las variables con respecto a la clase a predecir).
- **Árbol sin podar**:
 - *Number of Leaves*: 16 - Indica que el árbol sin podar tiene un total de 16 hojas o decisiones finales.
 - *Size of the tree*: 31 - Muestra que el árbol sin podar consta de 31 nodos en total, combinando nodos hoja y nodos de decisión.
- **Árbol podado**:
 - *Number of Leaves*: 12 - Refleja que el árbol, una vez podado, tiene 12 hojas o decisiones finales.
 - *Size of the tree*: 23 - El árbol podado tiene un total de 23 nodos, incluyendo tanto hojas como nodos de decisión.

5.3. Breve comparación entre la versión podada y sin podar

A partir de los árboles proporcionados, tanto el árbol sin podar como el podado, podemos realizar las siguientes observaciones:

Observaciones

- **¿Son los dos árboles distintos?** Sí, los dos árboles son claramente diferentes en su estructura y en las decisiones que toman.
- **¿Hasta qué punto?** El árbol sin podar tiene ramas y nodos adicionales en comparación con el árbol podado. Ejemplo de ello es, que en el árbol sin podar, hay decisiones basadas en **elapsed_time** que no aparecen en el árbol podado, lo que sugiere que el proceso de poda ha eliminado algunos de los nodos que podrían considerarse menos cruciales para la clasificación.
- **¿En el número de variables intermedias?** Sí, hay diferencias en el número de veces que se utilizan ciertas variables. El árbol sin podar utiliza más a menudo variables como **elapsed_time** y **moving_time** en comparación con el árbol podado.
- **¿En el número de hojas?** Si, hay una diferencia en el número de hojas entre los dos árboles. El árbol sin podar tiene 16 hojas - el árbol podado tiene 12 hojas.

5.4. Interpretación de los Números en Hojas del Árbol Podado

- **Significado de los números en la izquierda:** Representan el *número total de instancias correctamente clasificadas* que han llegado a esa hoja específica. Cuando sumamos todos estos números (sin considerar los decimales) de las hojas del árbol podado, obtenemos el número total de instancias que fueron clasificadas correctamente por el árbol, que es: 991.
- **Significado de los números en la derecha (cuando están presentes):** Representan el *número de errores* o las instancias que fueron clasificadas incorrectamente por esa hoja. En otras palabras, de las instancias que llegaron a esa hoja, este número indica cuántas de ellas no pertenecen a la clase predicha por la hoja. Y ocurre lo mismo, si sumamos todos, nos dará el número total de errores: 54.

5.5. Validación del Árbol

Cuadro 5: Output en Weka

Clase	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Run	0.983	0.120	0.942	0.983	0.962	0.883	0.934	0.949
Ride	0.880	0.017	0.963	0.880	0.919	0.883	0.934	0.906

Cuadro 6: Matriz de Confusión

		Clasificado como	
		Run	Ride
Real	Run	683	12
	Ride	42	308

- **Valoración y elección del árbol:** Considerando la estructura del árbol y los resultados obtenidos, la versión podada parece ser la opción más adecuada ya que generaliza mejor y evita el sobreajuste/overfitting.
- **Valores de TPR y FPR:** Para la clase Run como positiva, el TPR es del 98.3% y el FPR es del 12.0%.
- **Definiciones** tal y como yo las entiendo:
 - **TPR (True Positive Rate):** el TPR nos muestra qué porcentaje de las veces nuestro modelo acierta cuando se trata de instancias positivas. Es decir, de todas las veces que debería haber dicho 'sí', ¿cuántas veces lo hizo correctamente?.
 - **FPR (False Positive Rate):** Por otro lado, el FPR nos indica cuántas veces nuestro modelo se equivocó al clasificar instancias negativas como positivas. De todas las veces que debería haber dicho 'no', ¿en qué porcentaje se confundió y dijo 'sí'?

5.6. Aplicación del Modelo en Producción

Instancia	Actual	Predicted	Confianza (%)
1	?	Run	94.7
2	?	Run	100
3	?	Ride	98.7
4	?	Run	96.4
5	?	Ride	98.7

Cuadro 7: Predicciones para las instancias no etiquetadas.

- **Creación del segundo dataset strava_tests.arff:** He creado un mini test-set con las 5 instancias que el modelo clasificará en base a lo entrenado. Veamos su contenido (obviando el resto de cabeceras):

Type	distance (m)	moving_time (s)	elapsed_time (s)	total_elevation_gain (m)
?	10,000.0	2,600	2,600	100.0
?	20,500.3	6,600	6,630	570.0
?	31,000.5	4,500	4,500	320.0
?	15,500.0	4,900	4,950	480.0
?	34,000.5	4,700	4,750	630.0

Cuadro 8: Instancias creadas para la predicción.

- **Interpretación de <Predictions on test set>:** Weka presenta las predicciones para cada instancia sin etiquetar en función del modelo J48 previamente entrenado. Esto me ha parecido muy interesante; para cada predicción, se ofrece una métrica de confianza que indica cuánta certeza tiene el modelo en su predicción.

-
- Definición de cada columna:

- **actual:** Etiqueta real de la instancia. En este caso, todas las instancias tienen un valor desconocido ? ya que no estaban etiquetadas. Obviamente serán desconocidas.
- **predicted:** Es la etiqueta que el modelo predice para la instancia basándose en los datos de entrada y el conocimiento adquirido durante el entrenamiento, es decir, en base al modelo generado (he usado el árbol podado).

Comentario general

- Es **impresionante** cómo el modelo puede hacer predicciones precisas basadas simplemente en el entrenamiento previo y sin conocer la etiqueta real. No obstante, pese a la confianza del modelo, siempre tenemos que tener un grado de escepticismo ya que pocas veces podremos estar seguros al 100 %.

6. Laboratorio 6: kaggle.com; datasets y competiciones para Machine Learning

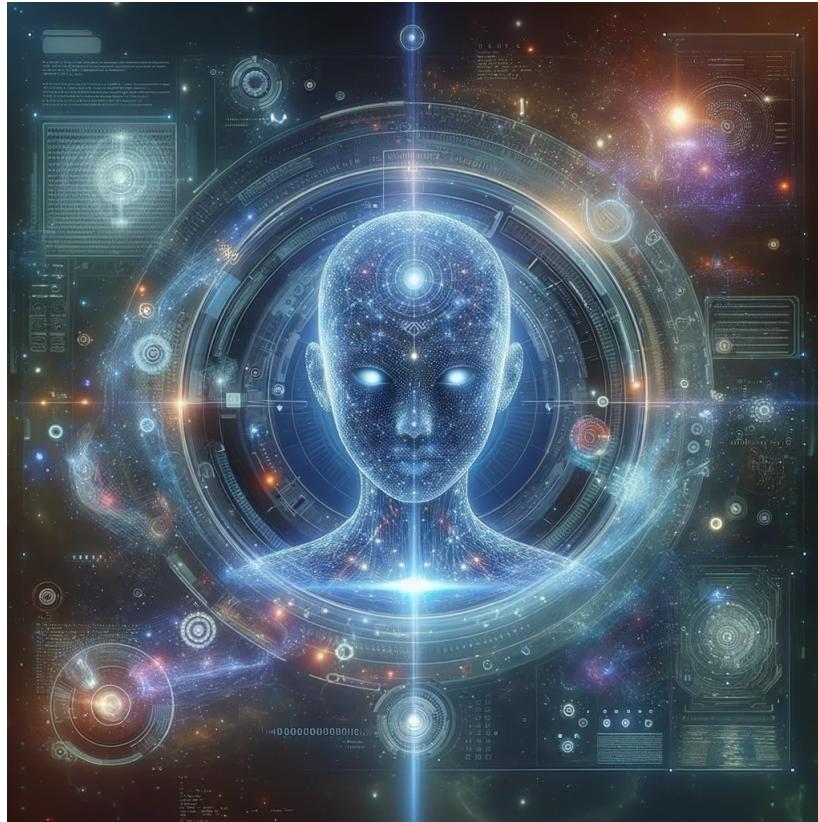


Figura 8: Machine Learning

6.1. Detalles de la Competición

- **Nombre de la competición:** [Human Protein Atlas - Single Cell Classification](#).
- **Propósito y problema de clasificación:** El propósito es identificar y localizar proteínas a nivel de célula individual dentro de imágenes. Los participantes deben predecir las localizaciones de orgánulos proteicos para cada célula en la imagen, incluidas las células en los bordes cuando haya suficiente información para determinar las etiquetas.
- **Patrocinador/owner:** Human Protein Atlas y Kaggle.
- **Tamaño del training set y test set:** Los conjuntos de datos contienen una mezcla de imágenes PNG de diferentes resoluciones (1728x1728, 2048x2048 y 3072x3072). Sin embargo, el número exacto de imágenes no se especifica. He intentado descargarme el fichero en su directorio, pero requiere que me registre en la competición.
- **Problema de clasificación supervisada:** El desafío consiste en una clasificación multietiqueta en el nivel celular basado en la localización subcelular de proteínas en imágenes.

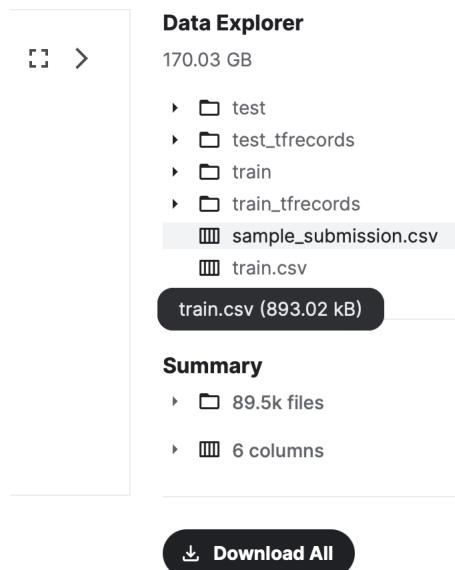


Figura 9: Directorio de datasets: descarga de `train.csv` requiere registro en competición.

- **Variable clase y sus valores:** Se deben predecir las etiquetas de localización orgánulo proteico para cada célula. Hay un total de 19 etiquetas diferentes, que incluyen 18 etiquetas para ubicaciones específicas y una etiqueta para señales negativas y no específicas. Estas etiquetas van desde Nucleoplasm (0) hasta Negative (18).
- **Desbalanceo de clases:** No se especifica en la descripción proporcionada si el conjunto de datos está desbalanceado. Para ello tendría que poder mirar el `train.csv`.
- **Número de variables predictoras:** Cada muestra de imagen está representada por cuatro filtros (verde, azul, rojo, amarillo) que actúan como variables predictoras. La imagen en el filtro verde representa la proteína de interés, mientras que las otras imágenes actúan como referencias.
- **Métrica de evaluación:** La métrica utilizada para evaluar las predicciones es [mAP], con el promedio tomado de las 19 clases segmentables del desafío.
- **Secciones de la competición:**
 - **Data:** Sección que proporciona los conjuntos de datos para entrenamiento y pruebas.
 - **Code:** Espacio para compartir y discutir códigos relacionados con la competición.
 - **Models:** Donde los competidores pueden compartir modelos pre-entrenados o soluciones propuestas.
 - **Leaderboard:** Tabla de clasificación que muestra el rendimiento de los participantes según su precisión en el set de prueba.

- **Variables predictoras relevantes:**

1. **Filtro Verde (Proteína de interés):** Representa la proteína que se está intentando localizar y es fundamental para la predicción.
2. **Filtro Azul (Núcleo):** Proporciona una referencia para la posición y estructura del núcleo celular.
3. **Filtro Rojo (Microtúbulos):** Ayuda a identificar la estructura y organización de la célula.
4. **Filtro Amarillo (Retículo Endoplásmico):** Otro marcador importante que puede influir en la localización de la proteína de interés.

- **Puntos de interés:** La verdad es que me ha llamado la atención cómo un desafío tan crucial en la biología puede abordarse utilizando técnicas de Machine Learning. Además, tras investigar un poco, he descubierto un nuevo concepto; la *naturaleza débil* de las etiquetas de entrenamiento - donde las etiquetas se proporcionan a nivel de imagen pero se deben predecir a nivel celular, añade una capa adicional de complejidad al problema por la relación.

Videojuegos

- **Otras competiciones consideradas:** El Machine Learning, me intriga y apasiona en cualquier área en la que se pueda aplicar, pero en este caso, me habría encantado encontrar una competición relacionada a **Videojuegos**; me fascinan; agitan mi curiosidad tanto en el juego en sí (como usuario, sobre todo las aventuras inmersivas) como en su ingeniería y la inteligencia artificial subyacente.

Sin embargo, la mayoría si no todas se basaban en la creación de *bots* con unas características, de manera que se evalúan los mismos siguiendo unos criterios dados y se observa su evolución en el tiempo. Por lo que he visto, se podría haber llegado a clasificar de alguna manera el *output* de los mismos para predecir si va a ganar o perder, pero sería 'tocar' mucho el problema y creo que no es la finalidad del laboratorio. He encontrado varios, pero éste me ha parecido bastante interesante: <https://www.kaggle.com/competitions/halite>.

No obstante, mi decisión final tras centrarme exclusivamente en problemas de clasificación ha resultado ser igual o incluso más interesante, aunque mucho más complejo.

7. Laboratorio 7: Selección de variables y PCA

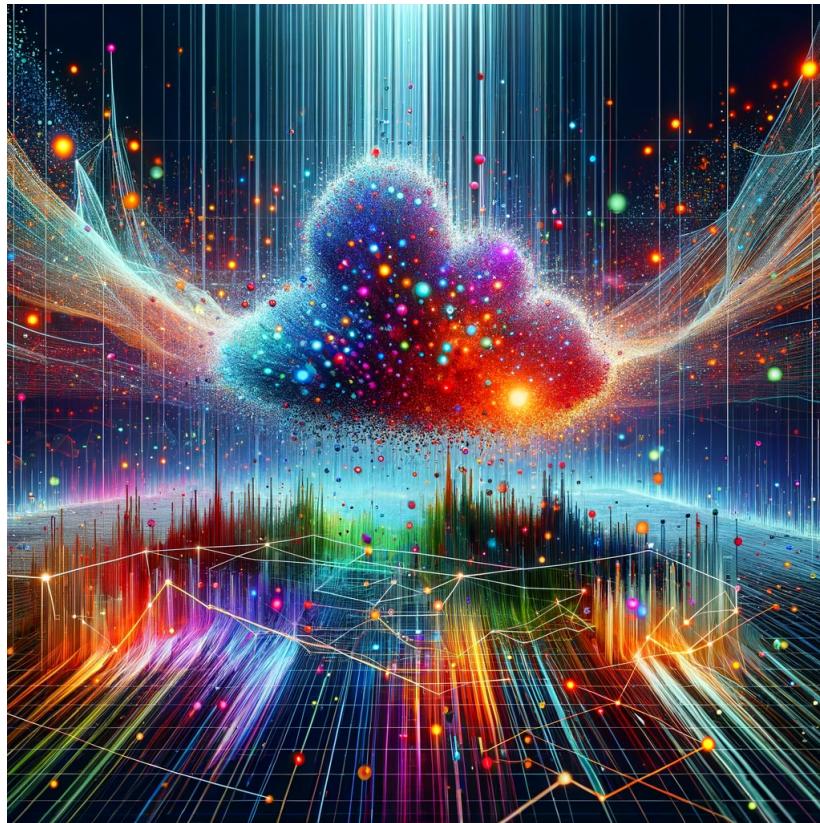


Figura 10: Principal Component Analysis

7.1. Selección de Variables: Filter vs. Wrapper

- Tal y como los entiendo, cada uno tiene sus características y propiedades. Sin embargo, lo que parece resaltar como diferencia principal es el hecho de que en el **filter** las variables no miden la correlación entre sí, si no que lo hacen con respecto de la variable a predecir - y, sin embargo, en el **Wrapper** éste factor si se tiene en cuenta al evaluar los subconjuntos (busca combinaciones de características que maximizan el rendimiento del modelo), aunque el costo computacional sea mucho más elevado.

7.2. Conjunto de datos adult.arff

Recuerdo este dataset del año pasado, '*cacharreamos*' con él en la asignatura de **Métodos estadísticos de la Ingeniería**. El conjunto de datos **Adult** se utiliza para predecir si el ingreso anual de una persona supera o no los 50.000 dólares basándose en información del censo de EE.UU. de 1994. La variable de clase es binaria y se define tal que:

- **Variable Clase:** Ingreso anual.
- **Valores Posibles:**
 - >50K: Indica ingresos anuales superiores a 50.000 dólares.
 - <=50K: Indica ingresos anuales de 50.000 dólares o menos.

Algunas de las predictoras relevantes son:

- **Edad:** Valor numérico que indica la edad del individuo.
- **Clase de trabajo:** Categórica, incluye clases como *Private*, *Federal-gov*, etc.
- **Educación:** Nivel educativo alcanzado, con categorías como *Bachelors*, *HS-grad*, etc.
- **Estado civil:** Categórica, indica el estado civil actual.

El problema de clasificación consiste en utilizar estas y otras predictoras para estimar de manera efectiva la variable de clase.

7.3. Ranking de variables predictoras

- Veamos un ranking con una métrica **filter**, para a continuación mostrar la fórmula utilizada para el cálculo de la **correlación**:

He escogido: `weka.attributeSelection.InfoGainAttributeEval`, y la ganancia de información se calcula como hemos ido haciendo en la asignatura hasta ahora:

$$\text{InfoGain}(\text{Class}, \text{Attribute}) = H(\text{Class}) - H(\text{Class} | \text{Attribute}). \quad (3)$$

Ranking

==== Attribute Selection on all input data ====

Search Method:

- Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 15 class):

- Information Gain Ranking Filter

Ranked attributes:

Score	Attribute Number	Attribute Name
0.16542	8	relationship
0.15700	6	marital-status
0.11663	11	capital-gain
0.09687	1	age
0.09207	4	education
0.09186	5	education-num
0.08241	7	occupation
0.05751	13	hours-per-week
0.05187	12	capital-loss
0.03669	10	sex
0.01562	2	workclass
0.00819	9	race
0.00797	14	native-country
0.00000	3	fnlwgt

- **Variables con correlación nula:** `fnlwgt`, de esto se trata este tema precisamente - teniendo una correlación de 0 implica que ésta variable no nos aporta nada con respecto a la clase, de hecho es una buena candidata a eliminar en el caso de que deseemos realizar una **reducción de dimensiones**.

7.4. Relevancia en modelos de clasificación

- Vamos a comprobar la relevancia de las variables *top-ranked* en un árbol de clasificación J48 y en las reglas inducidas por el clasificador Jrip.
- **J48 Tree:** Si observamos el nodo raíz y los primeros nodos que subsecuentemente se tienen en cuenta, se puede apreciar que efectivamente hay una relación con el *ranking* de correlaciones previo:

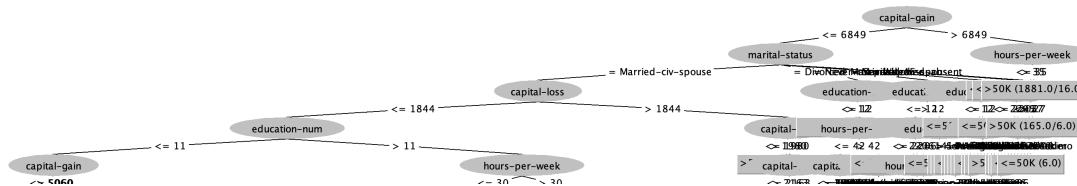


Figura 11: J48

- **Clasificador Jrip:** Al parecer, el clasificador Jrip ha priorizado la característica **marital-status**, seguida de **education-num**, **capital-gain**, **age**, **hours-per-week**, **occupation**, y **workclass**. Si se compara esto con la tabla previamente expuesta de correlaciones; se puede apreciar que ver que **marital-status** y **education-num** están entre las características más importantes.

Necesidad de K-fold

- **¿Era realmente necesario?**: No era necesario, ya que dada la correlación intrínseca, es altamente probable que el clasificador produzca un modelo que describa de manera inherente la importancia y, eso es totalmente independiente de la técnica de evaluación que se utilice. Con lo que, para abaratar el costo computacional podríamos haber utilizado prácticamente cualquier otro método de evaluación, siendo el **Holdout** una buena opción, a modo de ejemplo.

7.5. Clasificador K-NN

- Vamos a construir un clasificador K-NN usando solo las top-5 predictoras y a estimar los porcentajes de acierto con 5-fold cross-validation.
 - **Con las top-5 predictoras:**
 - ◊ La tasa de instancias correctamente clasificadas no se proporciona directamente en Weka.
 - ◊ Los altos valores de error absoluto y relativo sugieren un rendimiento inferior.
 - **Con todas las predictoras:**
 - ◊ Tasa de instancias correctamente clasificadas: 40001 de 48842 (81,8988 %).
 - ◊ El desempeño es significativamente mejor que el clasificador con las top-5 predictoras.
 - **Conclusión:**
 - ◊ El clasificador **K-NN** con **todas las predictoras** tiene un **mejor porcentaje de acierto** que el que utiliza solo las *top-5* predictoras.
 - ◊ Las variables adicionales fuera de las *top-5* contribuyen de manera significativa a la capacidad predictiva del modelo.

- Repito el proceso con las *top-10* predictoras y comparo los resultados:

Tras construir un clasificador k-NN con las *top-10* predictoras seleccionadas del ranking aprendido y utilizando un esquema de validación cruzada de 5 particiones, así como manteniendo los parámetros del clasificador con $k = 3$, he obtenido:

- Correlación de Pearson: 0.4021
- Error absoluto medio: 8.2014
- Raíz del error cuadrático medio: 11.7337
- Error absoluto relativo: 108.0059 %
- Raíz del error cuadrático medio relativo: 94.6904 %
- Número total de instancias: 48842

Comparando estos resultados con el clasificador que incluye todas las predictoras, podemos observar que el clasificador con las *top-10* predictoras no supera en rendimiento al clasificador completo. La inclusión de más variables predictoras en el modelo, parece contribuir positivamente al porcentaje de acierto del clasificador.

7.6. Selección de Variables y Redundancia

Reflexionando sobre si la selección de *top-k* variables garantiza la ausencia de redundancia entre ellas en relación con la variable clase, se puede llegar a la conclusión de que esta técnica puede identificar qué variables no aportan información, pero en principio, si deseamos detectar redundancia, tendremos que utilizar técnicas de reducción de dimensión como PCA, que consideran la correlación entre variables. En ese caso, si tenemos dos variables estrechamente relacionadas y, una de ellas, tiene mucha correlación con la clase - la otra será redundante porque *tiene poco que decir, ya que la otra variable lo ha dicho ya*.

7.7. Correlación en Weka

- Tras investigar, veo que la **correlación de punto biserial** que utiliza Weka es una técnica utilizada para medir la relación entre una variable numérica y una variable de clase nominal binaria, y se lleva a cabo siguiendo los siguientes pasos:

1. Se divide el conjunto de datos en dos grupos basados en las categorías de la variable de clase nominal.
2. Se calcula la media de la variable numérica para cada grupo.
3. Se calcula la desviación estándar de la variable numérica para todos los datos, independientemente de la división de clase.
4. Se utiliza la proporción de casos en cada categoría de la variable de clase para ajustar el cálculo de la correlación.

La fórmula toma en cuenta estas estadísticas y proporciona un coeficiente que indica la fuerza y la dirección de la asociación lineal entre las variables.

7.8. Wrapper Feature Selection en Weka

Utilizando el método de búsqueda *Greedy Stepwise* con evaluación hacia adelante, y con un clasificador basado en árboles de decisión *J48* en Weka. Se empezó con un conjunto vacío de atributos, añadiendo progresivamente aquellos que mejoraban la métrica de *mérito*.

7.9. Mérito del mejor subconjunto encontrado:

- Valor de mérito: **0.861**

7.10. Evaluador de Subconjunto de Atributos:

- Método de clasificación: `weka.classifiers.trees.J48`
- Opciones del esquema: `-C 0.25 -M 2`
- Evaluación del subconjunto: precisión de clasificación
- Número de folds para la estimación de la precisión: 5

Atributos Seleccionados:

- Número total de atributos seleccionados: 7
- Atributos específicos:

-
- 1. Edad (*age*)
 - 2. Número de años de educación (*education-num*)
 - 3. Estado civil (*marital-status*)
 - 4. Relación (*relationship*)
 - 5. Ganancia de capital (*capital-gain*)
 - 6. Pérdida de capital (*capital-loss*)
 - 7. Horas de trabajo por semana (*hours-per-week*)

Implicaciones:

- Los atributos seleccionados son probablemente los más relevantes para la tarea de clasificación.
- La reducción del conjunto de atributos a 7 puede llevar a un modelo más simple y generalizable.
- La validación cruzada aporta robustez a la evaluación de la precisión y reduce la susceptibilidad al sobreajuste.

GreedyStepWise

La búsqueda heurística Greedy no explora todo el espacio de búsqueda, sino que se mueve en la dirección de la mejora inmediata, luego puede quedarse atascada en **óptimos locales**. No tiene garantía de encontrar el mejor óptimo global porque no evalúa todas las combinaciones posibles. Sobre estas ideas trabajamos el año pasado hacia el final del curso en la asignatura de **Investigación Operativa**.

7.11. Feature Selection vs Feature Extraction

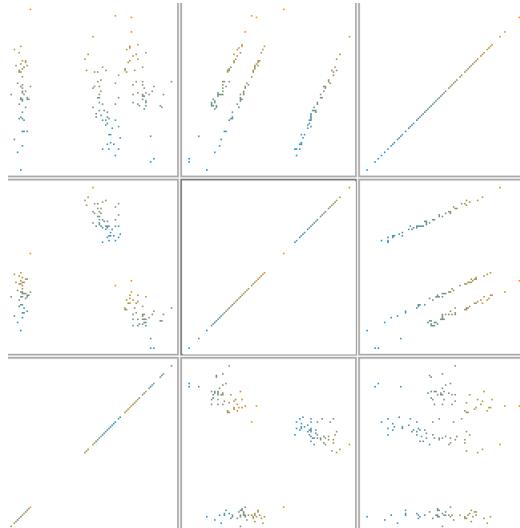


Figura 12: PCA (Iris dataset)

Ejecutando PCA sobre el conjunto de datos `Iris.arff`, se observa que las nuevas dimensiones (componentes principales) son **combinaciones lineales** de las variables originales. Los dos primeros componentes principales explican un 87,248 % de la varianza, lo que indica una fuerte reducción de dimensionalidad al pasar de cuatro dimensiones a dos con una retención considerable de la información original.

La matriz de correlación en PCA y los vectores propios son los siguientes:

Correlation matrix =	$\begin{bmatrix} 1 & -0,11 & 0,87 & 0,82 & -0,72 & 0,08 & 0,64 \\ -0,11 & 1 & -0,42 & -0,36 & 0,6 & -0,46 & -0,13 \\ 0,87 & -0,42 & 1 & 0,96 & -0,92 & 0,2 & 0,72 \\ 0,82 & -0,36 & 0,96 & 1 & -0,89 & 0,12 & 0,77 \\ -0,72 & 0,6 & -0,92 & -0,89 & 1 & -0,5 & -0,5 \\ 0,08 & -0,46 & 0,2 & 0,12 & -0,5 & 1 & -0,5 \\ 0,64 & -0,13 & 0,72 & 0,77 & -0,5 & -0,5 & 1 \end{bmatrix}$
Eigenvalues (V1, V2, V3) =	$\begin{bmatrix} 0,4107 & -0,1575 & -0,4409 \\ -0,2308 & -0,4511 & -0,7501 \\ 0,4755 & -0,0219 & -0,0733 \\ 0,4652 & -0,0853 & -0,0384 \\ -0,451 & -0,2292 & 0,0351 \\ 0,1022 & 0,6999 & -0,3598 \\ 0,3488 & -0,4707 & 0,3247 \end{bmatrix}$

Las nuevas variables que forman los dos ejes de la visualización de PCA son las combinaciones lineales de las variables originales con los mayores valores propios. En este caso, los dos primeros componentes principales:

-
- La primera componente principal **V1** es una combinación lineal de las características originales, definida como:

$$\begin{aligned} V1 = & 0,476 \cdot \text{Petallength} + 0,465 \cdot \text{Petalwidth} - 0,451 \cdot \text{Iris-setosa} \\ & + 0,411 \cdot \text{Sepallength} + 0,349 \cdot \text{Iris-virginica} \end{aligned} \quad (4)$$

- La segunda componente principal **V2** es otra combinación lineal, expresada por la ecuación:

$$\begin{aligned} V2 = & 0,700 \cdot \text{Iris-versicolor} - 0,471 \cdot \text{Iris-virginica} - 0,451 \cdot \text{Sepalwidth} - 0,229 \cdot \text{Iris-setosa} - \\ & 0,158 \cdot \text{Sepallength} \end{aligned} \quad (5)$$

Conclusión

La principal diferencia entre **feature selection** y **feature extraction** como PCA es que el primero selecciona un subconjunto de variables originales manteniendo su interpretación original, mientras que el segundo crea nuevas variables a partir de combinaciones lineales de las originales.

7.12. Explicación del código en R

Resumen del conjunto de datos iris

```
1 summary(iris)
```

Proporciona un resumen estadístico de cada variable en el conjunto de datos `iris`.

Análisis de componentes principales (PCA)

```
1 pca_iris <- prcomp(iris[1:4], scale=TRUE)
```

Realiza un PCA en las primeras 4 columnas de `iris`, estandarizando las variables.

Resumen del objeto PCA

```
1 summary(pca_iris)
```

Muestra un resumen de los resultados del PCA, incluyendo la varianza.

```
> summary(iris)
   Sepal.Length   Sepal.Width    Petal.Length   Petal.Width      Species
Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa   :50
1st Qu.:5.100  1st Qu.:2.800  1st Qu.:1.600  1st Qu.:0.300  versicolor:50
Median :5.800  Median :3.000  Median :4.350  Median :1.300  virginica:50
Mean   :5.843  Mean   :3.057  Mean   :3.758  Mean   :1.199
3rd Qu.:6.400  3rd Qu.:3.300  3rd Qu.:5.100  3rd Qu.:1.800
Max.   :7.900  Max.   :4.400  Max.   :6.900  Max.   :2.500
> pca_iris <- prcomp(iris[1:4],scale=TRUE)
> summary(pca_iris)
Importance of components:
              PC1       PC2       PC3       PC4
Standard deviation 1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion 0.7296 0.9581 0.99482 1.00000
```

Figura 13: `summary(pca_iris)`

- La **primera** componente principal (PC1) explica el 72.96 % de la varianza.
- La **segunda** componente principal (PC2) explica el 22.85 % de la varianza.

Biplot del PCA

```
1 biplot(pca_iris, scale=0, col=c("blue", "red"))
```

Genera un biplot de los resultados del **PCA** con colores específicos para los puntos y vectores.

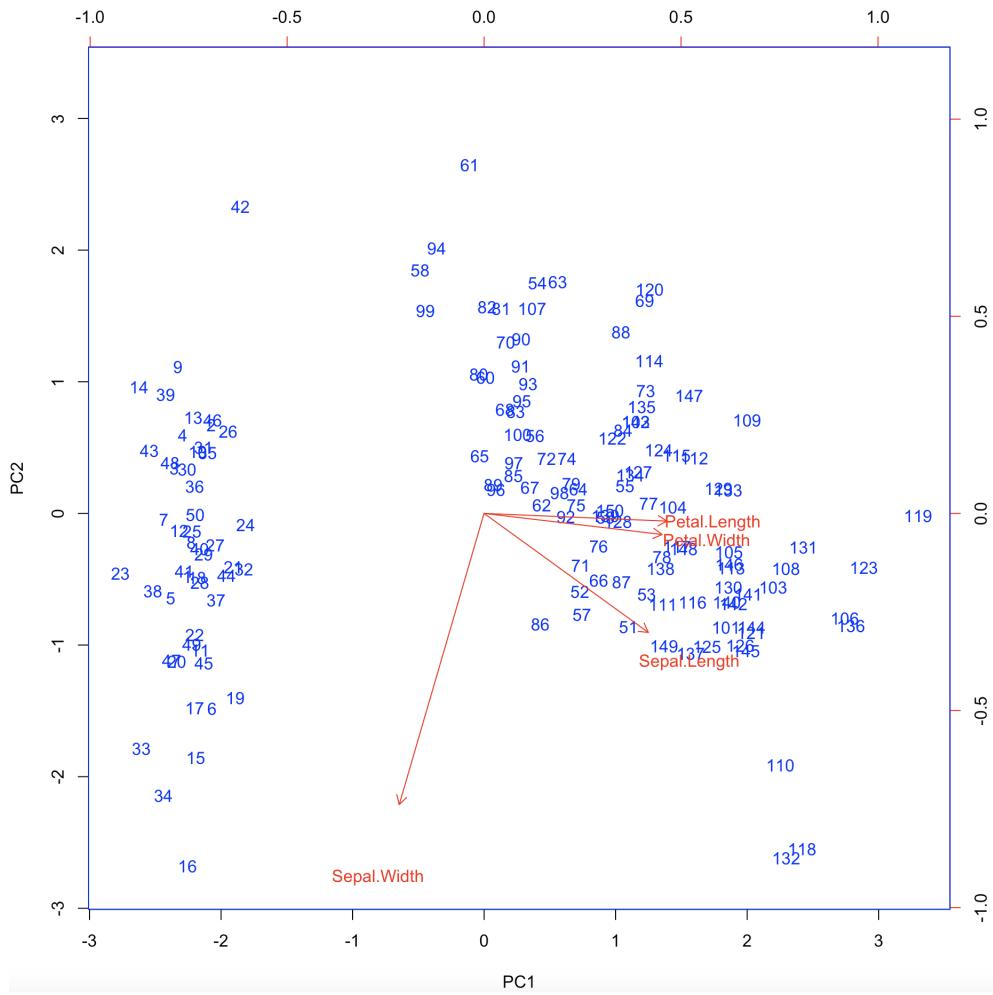


Figura 14: Biplot

Tras investigar un poco:

- **Los números azules** indican la contribución de cada variable a las componentes principales. Cuanto más cerca esté un número de una variable de un componente principal, mayor será su influencia en esa componente.
- **Los dos ejes** de la visualización representan las dos primeras componentes principales (PC1 y PC2). Estas dos componentes principales son combinaciones lineales de las variables originales que explican la mayor parte de la variabilidad en los datos.
- **Las direcciones rojas** en la visualización representan las proyecciones de las variables originales en el espacio de las dos primeras componentes principales (PC1 y PC2). Indican la dirección y magnitud de la contribución de cada variable a las componentes principales.

Instalación y carga de paquetes ggplot2 y ggfortify

```
1 install.packages("ggfortify")
2 library(ggfortify)
3 install.packages("ggplot2")
4 library(ggplot2)
```

Instala y carga los paquetes `ggfortify` y `ggplot2`.

Visualizaciones con autoplot

```
1 autoplot(pca_iris)
```

Crea una visualización básica de los resultados del **PCA**:

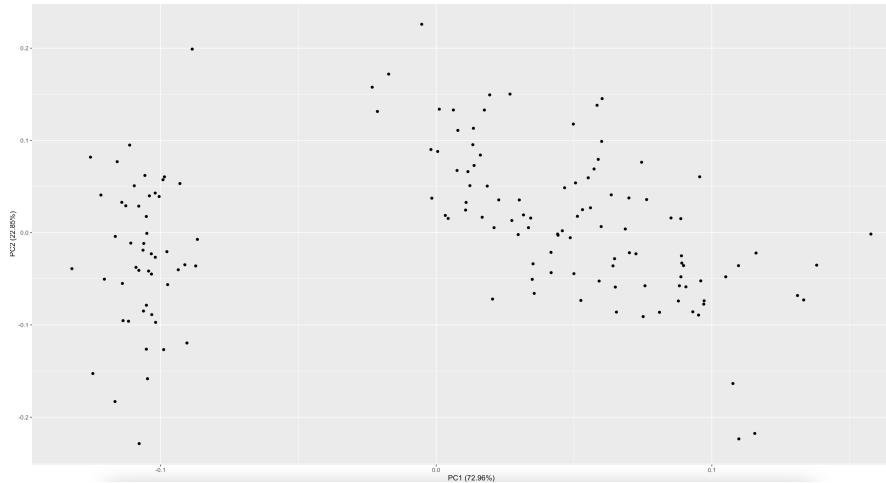


Figura 15: Autoplot

```
1 autoplot(pca_iris, data = iris, colour = 'Species')
```

Visualiza los resultados del **PCA** con colores para diferenciar las especies del conjunto de datos `iris`:

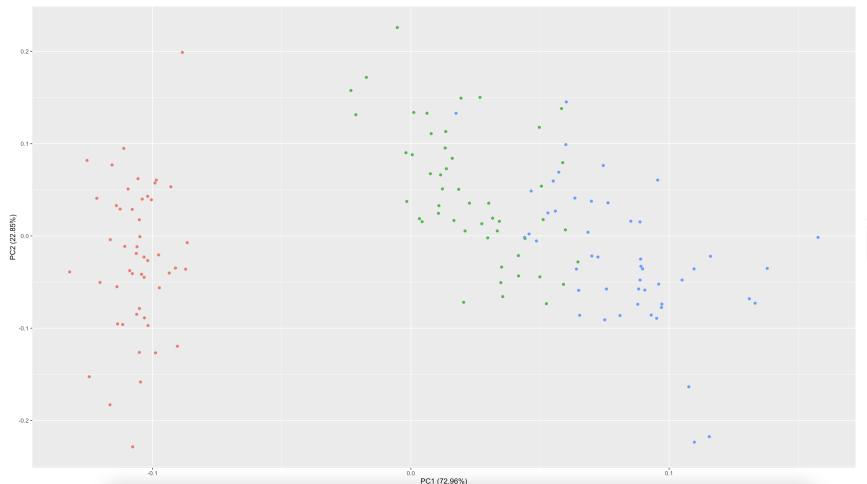


Figura 16: Autoplot coloreando las especies

```
1 autoplot(pca_iris, data = iris, colour = 'Species', shape = FALSE, label.size = 3)
```

Visualiza el **PCA** con ajustes en la forma de los puntos y el tamaño de las etiquetas.

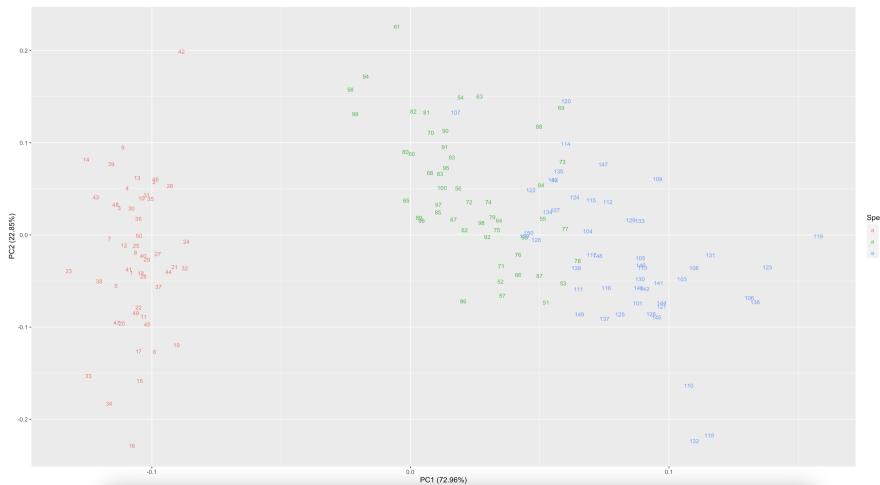


Figura 17: Autoplot coloreando las especies

- **La primera visualización** generada por `autoplot(pca_iris)` muestra un gráfico de dispersión que representa las observaciones en el espacio de las dos primeras componentes principales (PC1 y PC2). Cada observación se representa como un punto en el gráfico.
- **La segunda visualización**, `autoplot(pca_iris, data = iris, colour = 'Species')`, es similar a la primera, pero agrega color a los puntos en función de la variable `Species`. Cada especie (`setosa`, `versicolor` y `virginica`) se representa con un color diferente. Esto permite visualizar cómo se distribuyen las observaciones de diferentes especies en el espacio de las componentes principales.
- **La tercera visualización**, `autoplot(pca_iris, data = iris, colour = 'Species', shape = FALSE, label.size = 3)`, es similar a la segunda, pero se desactivan los símbolos de forma (`shape`) y se aumenta el tamaño de las etiquetas (`label.size`) de manera que los puntos en el gráfico se representan únicamente mediante etiquetas de texto. Cada punto lleva una etiqueta que indica a qué especie pertenece.

Conclusión

Estas tres visualizaciones son útiles para explorar las relaciones entre las observaciones y las componentes principales, así como para identificar patrones en función de la variable `Species`. Cada una de ellas aporta un nivel adicional de información en comparación con la anterior, y las diferencias radican en la presentación de los datos.

«En el corazón de la inteligencia artificial hay una llama de creatividad que ilumina el camino hacia un mundo de posibilidades infinitas»

–Gary Kasparov.

3 de Noviembre de 2023