Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Ingeniería Informática
UPV/EHU

Laboratorios de Minería de Datos

Mikel Molina

Minería de Datos
1 de Octubre de 2023

Primer Laboratorio (13-14 de septiembre)

En este laboratorio se nos pide ver un vídeo sobre algoritmos y big data, y responder las siguientes preguntas:

1. **De todas las aplicaciones que se mencionan, ¿cuál es la que más te ha llamado la atención? ¿por qué? ¿la ves "factible" - "creíble"?**

La que más me ha llamado ha sido una de las primeras: la app de movilización. Esto es porque en Donostia es la primera vez que vi la existencia de bicicletas por alquiler, y no me pareció del todo factible. Esto se debe a que uno, —quizá esperando lo peor de la humanidad—, supondría que acabarían siendo robadas. Más tarde comprendí que si se da un buen servicio, la gente no tiene por qué tener que recurrir a otros medios más extremos.

2. **¿Qué persona entrevistada es la que más te ha llamado la atención? ¿Por qué?**

El experto en ciberseguridad Chema Alonso, ha sido, sin duda alguna, el que más atención me ha llamado. Esto es porque antes de ver el vídeo ya le había visto en alguna que otra parte, dado que el mundo de la ciberseguridad es un sector que me interesa, y uno no investiga sobre este tema sin dar con él. En definitiva, me ha parecido que han acertado al invitarle, porque es una persona bastante capacitada para hablar sobre violaciones de privacidad y lo que agentes maliciosos con capaces de hacer con nuestros datos. Además de que lleva bastante tiempo metido en el sector, y es capaz de hablar de todos los temas que el entrevistador le plantea.

3. **¿Cuál es la idea-reflexión que más te ha llamado la atención? ¿Por qué?**

La idea que más me ha llamado la atención es la discriminación sobre una persona según los datos que recogemos. En el vídeo defienden de una forma algo cuestionable el hecho de que si una persona es más proclive a generar accidentes, a usar menos el coche, etc. Se le cobrará más en el seguro. Dicha tésis se defiende, si no he entendido mal, diciendo que " el propio usuario es el que se lo ha buscado", cosa que me parece mal; si lo extrapolamos a casos más generales, estaríamos perpetuando una gran discriminación entre grupos.

4. **De entre los riesgos y dudas-peligros éticos que plantean los entrevistados, ¿cuál es el que te ha llamado la atención?**

Esta respuesta tiene que ver con lo previamente respondido. Pero para añadir más, lo que una mujer del programa plantea, que es (parafraseando un poco) "detenerse un momento para pensar en la ética, en vez de crear", me parece necesario. Por ejemplo, no estaría de más marcar una línea para saber si deberíamos clasificar a la gente o no según qué casos, ¿acaso deberíamos detener a la gente preventivamente porque un algoritmo nos lo ha indicado así?

5. **Coméntame al menos un párrafo, una reflexión más tuya, propia, más de "tu cosecha".**

Hay algo que en el documental me ha estado preocupando o molestando un poco, y se trata en que enseñan algoritmos para mantener al usuario más pegado al móvil, consuma más, y gaste más dinero. Esto, en varios casos, lo pintan como algo fascinante, pero lo veo peligroso. El caso más preocupante es en el de los videjuegos: el grupo estaba ideando un algoritmo para que los usuarios gastaran más dinero para comprar diferentes monstruos y ser mejores en el juego. ¿Acaso han tenido en cuenta el hecho de que puede que niños jueguen el videojuego? Posiblemente sepan que con este tipo de algoritmos están empujando a los jugadores a la ludopatía. La cultura de las microtransacciones es algo muy criticado, e incluso suele dar mala imagen a la empresa. En mi caso, no habría asistido a dicho concurso, dado que no me habría encontrado cómodo ayudando en dicho fin.

Segundo Laboratorio (20-21 de septiembre)

En esta tarea se nos ofrece un dataset llamado CoverTypePrediction.arff. Tenemos que analizar su fin y aprender a aplicar los conceptos aprendidos en clases teóricas.

1. **Entiende el problema a clasificar.**

Esta base de datos se encarga de predecir el tipo de árboles que se encuentra en un espacio de terreno de 30 metros por 30 metros. Los datos se han obtenido del Bosque Nacional de Roosevelt, situado en el norte de Colorado. Para ello, toma en cuenta unas variables clave para predecir el tipo de árbol:

- La elevación en metros del terreno
- El aspecto del terreno
- La pendiente
- La distancia horizontal/vertical
- La distancia horizontal hacia la carretera más cercana
- La cantidad de sombra que hay a las nueve de la mañana, al mediodía y a las tres de la tarde
- La distancia horizontal hacia la hoguera o punto caluroso más cercano
- El bioma.
- El tipo de tierra.
- El tipo de árbol que se encuentra en dicho terreno.

2. **Especifica para este dataset: número de casos-muestras, número de variables, naturaleza de las variables (¿numéricas? ¿categóricas?), número de valores de la variable clase a predecir.**

El propio WEKA nos dice cuantas instancias se encuentran en el fichero: 15120. También nos dice el número de variables: 56. A continuación, vamos a enumerar cada variable y su tipo.

- ID: Numérico.
- Elevation: Numérico.
- Aspect: Numérico.
- Slope: Numérico.
- Horizontal distance to hidrology: Numérico.
- Vertical distance to hidrology: Numérico.
- Horizontal distance to roadway: Numérico.
- Hillshade (a todas horas): Numérico.
- Horizontal distance to to fire points: Numérico.
- Tipo de tierra salvaje, categórico. Consta de cuatro tipos distintos..
- Tipo de tierra, categórico. Consta de 43 tipos distintos

Con el tipo de tierra salvaje, para indicar si el terreno se trata de uno o de otro, cada objeto constará de cuatro columnas binarias, asignando a cada columna un tipo distinto. Lo mismo ocurre con el tipo de tierra, solo que en esta vez, al haber registrado cuarenta tipos distintos, se empleará ese número de columnas binarias, asignando a cada terreno una columna.
Finalmente tenemos la variable a predecir, que se trata del tipo de árbol, para esta variable tenemos siete tipos de árboles, esta vez, en vez de añadir otras siete columnas binarias, asignamos un número del uno al siete a cada tipo de árbol.

3. **Explica los siguientes parámetros del clasificador "IBk", y relaciónalo con lo visto en clase.**

El IBK se trate del clasificador K-NN que hemos estado viendo en clase. Curiosamente, se encuentra dentro del conjunto de clasificadores "vagos" (lazy). Los parámetros con los que consta son los aprendidos en clase:

- **KNN:** En este apartado podemos escoger nuestro K preferido, esto puede influir a la hora de clasificar. Por ejemplo, si estuvieramos haciendo uso del método nearest neighbour, si pusiéramos en dicho apartado 1, tan solo estaría teniendo en cuenta la variable más cercana.

- **distanceWeighting**: Se trata sobre si decidimos ponderar las variables que se encuentran más cerca o lejos.

4. **Basándote en los apuntes de la teoría y tu intuición, explica los siguientes valores del parámetro "distanceWeighting": "NoDistanceWeighting", "Weight by 1/distance".**

Como hemos dicho, distance weighting es una parámetro que marcamos si queremos ponderar los casos. Por defecto, tenemos marcado que no lo haga, es decir "NoDistanceWeighting". Si queremos que ponderación, tenemos Weight by 1/distance, que tal como dice, no solo calcula la distancia euclídea, sino que calcula su inversa también. De esta forma, las variables que más cerca se encuentren, — y por tanto, tengan una distancia menor —, tendrán un valor mayor, y viceversa.

5. **Haz pruebas de manera informal, cambiando los valores de los parámetros "KNN" y "distanceWeighting": y viendo cómo cambia la tasa de acierto, estimando ésta con el método "Hold-out"**

En la figura 1 (sin ponderación) podemos ver cómo con k = 2, obtenemos una mayor tasa de aciertos (TPR). Se ha probado hasta con K = 10, y podemos ver que conforme más aumentemos el número, la tasa de aciertos disminuye. Cabe destacar que con K = 4, TPR aumenta más que con k = 5 y k = 3. Para asegurarnos de que no ha sido una coincidencia, se ha cambiado la semilla, y esta vez ha dado una tasa de acierto de 0.729, mayor que antes incluso. Parece ser que k = 2 y k = 4 son las k's más optimas.
Algo curioso ocurre con la figura 2 (con ponderación), y es que con k = 10, hemos obtenido la mayor tasa de aciertos. Como la última vez, nos hemos asegurado de que no haya sido gracias al factor suerte cambiando la semilla. Pero obtenemos que con k = 10 hemos obtenido un TPR de 0.645 esta vez, una vez más nos da 0.643. Parece ser que ésta vez sí que ha sido gracias a la semilla que con k = 10 se han obtenido mejores resultante. La que sí parece consistente es con k = 1, que tras probar con distintas semillas no da una tasa de acierto parecida, sino una mayor.

6. **¿Por qué crees WEKA agrupa en la familia "lazy" a este tipo de clasificador?**

Esto es porque no hay ninguna fase de entrenamiento. No crea ningún modelo a partir de dataset, sino que los memoriza y predice la clase de la variable en función de lo memorizado. Esto implica que el algoritmo del vecino más cercano es más costoso que otros clasificadores, esta es una de las muchas razones por las que simplemente aumentando el número de K en WEKA el tiempo para construir la matriz de confusión es más grande.

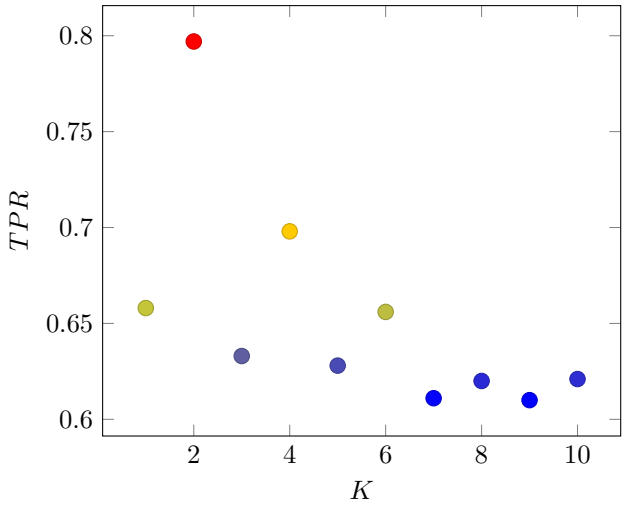


Figure 1: Sin ponderación de datos.

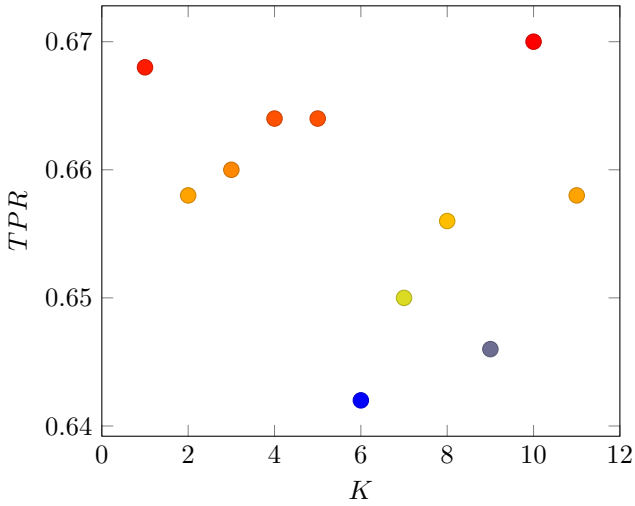


Figure 2: Con ponderación de datos.

Tercer Laboratorio (27-28 de septiembre)

Este laboratorio trata sobre la validación de modelos. Para ello, vamos a usar las siguientes técnicas de validación:

- Método de resustitución o método no honesto de estimación.
- Método H, holdout con 66 %
- Método H 5 veces, haciendo una media de todos los resultados.
- Método de estimación 10-fold cross-validation.
- Método de estimación 5-fold cross-validation.
- Método de estimación, leave-one-out.

1. **¿Devuelven todos el mismo porcentaje de error estimado? ¿Por qué?**

No, no devuelven el mismo porcentaje de error. Esto se debe a que estamos haciendo uso de varias técnicas de validación. Por ejemplo, el método no honesto tiene el menos porcentaje de error.

2. **¿Cuál de los métodos anteriores devuelve el mejor porcentaje de error estimado? ¿Era esperable? ¿Por qué?**

Como es obvio, el método que devuelve el mejor porcentaje de error estimado es el primero, en el que usamos el mismo conjunto de entrenamiento para validar nuestro algoritmo. Esto es porque nuestro modelo se basa en dicha base de datos; ha aprendido sobre ella, así que como es de esperar, va a etiquetarlos correctamente. Claro que, como consecuencia tampoco sabremos cuán preciso es el modelo.

3. **¿Cuál te parece la estimación más fiable de las 6 anteriores? Ten en cuenta que el error real únicamente se puede conocer teniendo infinitos casos: y ésto no es posible.**

El más fiable es el caso de leave-one-out, porque estamos tomando el número mayor de casos posible para validar nuestro algoritmo. Si bien no es posible obtener infinitos casos, podemos sacar el máximo provecho de los datos con los que contamos.

4. **¿Cuál de los 6 métodos anteriores de estimación crees que le ha exigido más trabajo de cómputo a WEKA?**

El último es el más costoso. Al fin y al cabo, estamos forzando WEKA que por cada instancia de la base de datos construya un modelo, y lo verifique con todas las instancias de la base de datos.

5. **Cuando le pedimos a WEKA que estime el error mediante 10-fold cross-validation, ¿cuántos modelos aprende WEKA en total? ¿Por qué no son 10? ¿Con qué porcentaje de casos del total aprende cada uno de esos modelos?**

En total, WEKA aprende 11 modelos, (si es que a lo creado mediante K-NN se le puede llamar modelo). Aprende los 10 modelos desde los diez subconjuntos particionados del conjunto principal, y otro modelo más desde el conjunto principal, que es el que buscamos etiquetar y usar para predecir en la vida real. Por lo tanto, un modelo de uno de los subconjuntos aprende sobre el 90 % del total de casos, y el modelo principal, sobre el 100 %

6. **¿Puede darse el caso que para una "seed-semilla" y método de validación concreto, en una ejecución el clasificadorX "sea mejor" que el "clasificadorY", y que para otra "seed-semilla" ocurra justamente lo contrario? ¿Por qué crees que se puede dar esta situación?**

Sí lo veo posible. Esto es porque en función del input un modelo puede llegar a ser más eficiente que otro en algunos aspectos. El clasificador K-NN es bueno a la hora de clasificar instancias que se encuentren cerca de un grupo de variables que pertenezcan a la misma clase. Sin embargo, si tocara una semilla que ofrece unos datos no tan cerca de variables guardadas en la base de datos, esto no puede ser tan fácil a primera vista, dando pie a un gran error, que, quizá otro clasificador sabría manejar correctamente, e incluso mejor que el anterior.

Cuarto Laboratorio (27-28 de septiembre)

En este laboratorio vamos a ver el preprocesamiento de datos. Para ello, se nos pide abrir iris.arff. Y aplicarle los filtros: attributeIndices = 3, bins = 4 y useEqualFrequency = TRUE.

1. **¿Qué hemos conseguido aplicando los siguientes filtros? Descríbelo. ¿Actúa este filtro sobre las variables nominales-categóricas? ¿Por qué?**

Vamos a ir explicando los parámetros que hemos cambiado:

- **attributeIndices:** Tras poner el número 3, estamos indicando que vamos a aplicar el filtro sobre únicamente el tercer atributo
- **bins:** El número de contenedores que vamos a usar al aplicar el filtro.
- **useEqualFrequency:** Indicamos que queremos que las variables sean divididas en dichos contenedores de una forma equitativa, con un número de instancias parecido en cada uno.

No.	Label	Count	Weight
1	'(-inf-1.55]'	37	37
2	'(1.55-4.35]'	38	38
3	'(4.35-5.15]'	41	41
4	'(5.15-inf)'	34	34

Figure 3: useEqualFrequency = True y attributeIndices = 3

No.	Label	Count	Weight
1	'(-inf-2.475]'	50	50
2	'(2.475-3.95]'	11	11
3	'(3.95-5.425]'	61	61
4	'(5.425-inf)'	28	28

Figure 4: useEqualFrequency = False y attributeIndices = 3

Por lo tanto, con este filtro hemos conseguido discretizar la varialble continua de la largura del pétalo (porque estamos indicando attributeIndices = 3). Para ser más específicos, hemos decidido crear 4 intervalos, (tal y como hemos indicado en bins), y hemos contado cuántas instancias se encontraban dentro de cada intervalo. El criterio para decidir los intervalos, (es decir, cuando acaban o cuando empiezan), se ha basado en tratar de que en cada intervalo se encontrasen el mismo número de instancias que en los demás.
No va a funcionar sobre las variables categóricas. Para empezar, las variables ya están discretizadas; no hay necesidad. Además de que no podríamos crear ningún tipo de intervalo, ya que eso implicaría que tendríamos que poner un color de flor superior a otro, cosa que no tendría sentido.

2. **Cambia useEqualFrequency a FALSE, ¿qué diferencia existe respecto al previo filtro?**

Como podemos observar en ambas capturas, y como bien nos dice WEKA. Cuando cambiamos la variable a False, el programa se limitará a crear intervalos que tengan una misma distancia (salvo el primero y el último). En nuestro caso, la distancia de los intervalos es de 1'475. Mientras tanto, cuando equalFrequency es True, priorizamos que los intervalos tengan aproximadamente el mismo número de instancias en cada uno, haciendo que algunos intervalos sean más largos que otros.
A continuación, vamos a cargar el dataset "hepatitis.arff", que reúne los datos de pacientes que tienen hepatitis, para luego determinar si va a morir o vivir.

3. **Usa el filtro replaceMissingValue, ¿Qué hace este filtro? ¿Cuál es su efecto sobre 'ALK_PHOSPHATE'? ¿Hay algún problema con 'PROTIME'?**

El filtro, según WEKA, trata de predecir los valores desconocidos con el modelo obtenido mediante el dataset. Por lo tanto, podemos observar cómo una vez aplicado el filtro, añadimos 29 instancias en las que el valor de la variables ALK_PHOSPHATE no había sido establecido. Éstos valores se han añadido

—segun la documentación de la función—, con la moda y el valor medio obtenido en el training set. Con PROTIME ocurre un problema, y es que tenemos casi la mitad de datos sin registrar, por lo que al predecir mediante el filtro su valor, estamos aportando demasiados datos "inventados". Esto podría llevar a errores parecidos al método no honesto de estimación.

4. **Usa el filtro "Normalize" sobre iris.arff. ¿Qué efecto ha tenido?**

Ahora el valor de toda variable numérica está acotada entre los valores [0,1]; es decir, ha normalizado las variables numéricas. Obviamente, no tendría sentido hacer esto con las variables nominales, esto queda claro una vez vista la fórmula que se emplea para normalizar un valor cualquiera:

$$x_{\text{normalizado}} = \frac{x - \text{valor máximo}}{\text{valor máximo - valor mínimo}} \tag{1}$$

5. **¿Qué diferencia tiene con respecto al filtro "standardize"?**

Con el filtro standardize —como podemos ver en el dataset resultante—, estamos estableciando la media a 0 y la desviación a 1, de tal modo que el valor de las variables numéricas no estará acotado. Se hace uso de la siguiente fórmula —y por ende no tiene sentido estandarizar variables nominales—:

$$x_{\text{estandarizado}} = \frac{x - \text{media}}{\text{desviación estándar}} \tag{2}$$

6. **Vamos a escoger el filtro "Center" y analizarlo.**

He escogido el filtro center porque tiene que ver con standardize y normalize, ya que suele ir de la mano con éstos. Y es que Center tan solo intenta establecer la media a 0. Es crucial a la hora de interpretar regresión linear. La fórmula es la más sencilla:

$$x_{\text{Center}} = x - \text{media} \tag{3}$$

Una vez aplicado, podemos ver que la media de cada valor es 0.

7. **¿Qué hace la acción "Filter - MultiFilter"? ¿Sabrías utilizarlo, jugando con sus parámetros?**

MultiFilter, tal y como su nombre indica, nos permite usar múltiples filtros. Cabe destacar que aplica éstos iterativamente, es decir, uno tras otro, y no simultáneamente (de la otra forma no tendría sentido). Su objetivo es mejorar la comodidad del usuario, ya que podemos guardar distintos filtros según nuestro gusto, en vez de ir uno a uno escogiéndolos. Un uso que le podemos dar que tenga que ver con lo que hemos hecho este laboratorio sería el de añadir valores y luego normalizarlos. Para ello clickamos en el nombre en negrita de MultiFilter - Hacemos click en la casilla de filters - Pinchamos "Choose", escogemos el filtro ReplaceMissingValues, y añadimos mediante add. Hacemos lo mismo para Normalize. Cerramos las ventanas, aplicamos el filtro y veremos cómo WEKA nos ha aplicado ambos filtros. De esta forma, la próxima vez que queramos hacer lo mismo con otro dataset, tan solo tendremos que aplicar MultiFilter.

Quinto Laboratorio (4-5 de octubre)

En este laboratorio tratamos la estructura de los árboles de decisión, y técnicas de podado. Para ello, se nos ha ofrecido una base de datos llamada "strava-actividad", una colección de datos pequeña que nos permite ver el árbol generado para analizarlo. Antes de empezar, expliquemos las variables y qué variable tratamos de establecer como predictora. Strava-Actividad no es más que un dataset que podemos usar para clasificar el tipo de la variable que nosotros decidamos. Entre ellos, se encuentran los siguientes tipos:

- **type:** El tipo de actividad realizado, solo hay dos tipos: Run o Ride, nominal.
- **distance:** Es la distancia en metros realizada en la actividad, numeric.
- **moving-time:** El tiempo que se ha estado en movimiento realizando dicha actividad, numeric.
- **elapsed-time:** Tiempo total que se ha estado realizando la actividad, incluyendo los momentos en los que no ha habido movimiento, numeric.
- **total-elevation-gain:** Desnivel total realizado en toda la actividad, numeric.

1. **¿Son los dos árboles distintos? ¿Hasta qué punto? ¿En el número de variables intermedias? ¿En el número de hojas?**

Ambos árboles (figura 5 y figura 6), se asemejan hasta cierto punto. Por ejemplo, la rama derecha de la raíz está intacta, es decir, que no ha habido ninguna necesidad de podarla. La parte izquierda es la que más cambios ha sufrido debido al proceso de podado. Podemos ver cómo mientras en el árbol no podado, (figura 5) hay una rama que se extiende demasiado (es la más larga), en el árbol podado esta rama es totalmente podada desde moving-time. Posiblemente sea porque en este caso el árbol se estaría acomodando demasiado a la base de datos.

2. **¿Aparecen todas las variables originales en los árboles finales? ¿Tienen que aparecer obligatoriamente (recuerda cómo es el método de construcción de árboles de clasificación)?**

En los árboles finales salen todas las variables originales. Sin embargo, esto se debe a que las mismas han sido escogidas de una manera sensata, sabiendo que afectarían a la hora de clasificar una variable, por lo tanto dando una información mutua mayor que 0. Si en esta base de datos hubieran añadido una variable insignificante, así como el color de ojos del sujeto, acabaría no siendo implementado en el árbol.

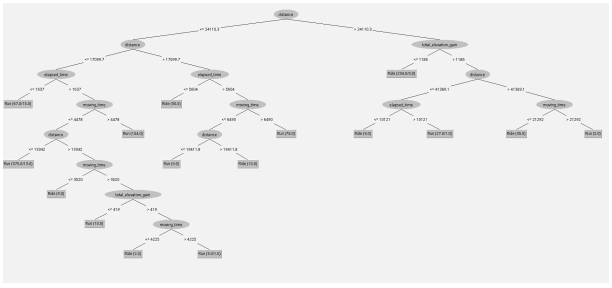


Figure 5: Árbol sin podar.

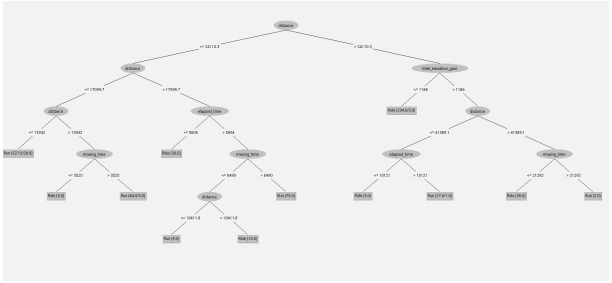


Figure 6: Árbol podado.

3. **¿Qué significan los número de la izquierda y de la derecha de cada hoja final?**

Si sumamos los valores que es encuentran a la izquierda de cada hoja de cada árbol, acabaremos con el número 1045, que vendría a ser el número de instancias que se encuentran en el training set, (WEKA mismo nos lo dice después de generar el árbol). La cifra derecha se contribuye al número de variables que no son de la clase que la hoja predice (Run o Ride). Por ejemplo, en el árbol podado, en la hoja que se encuentra más a la izquierda, tenemos Run(67/15), esto quiere decir que de las 67 instancias que han acabado en esa hoja, y por lo tanto clasificadas como Run, 15 son de la clase Ride. Es decir, cuanto mayor sea el número de la derecha, más "impura" será, (por así decirlo), será nuestra hoja.

4. **Muestra los valores estimados TPR y FPR de un árbol que elijas (con una 10-fold cross-validation))**

Escogemos el árbol podado para evaluar el valor estimado, y la clase Run como positiva. Los resultados son los siguientes:

- TPR = 0.981
- FPR = 0.117

Como podemos ver, hemos obtenido una gran tasa de acierto. TPR se refiere a True Positive Rate, es decir, la tasa de veces que nuestro modelo ha acertado correctamente la clase de una variable, siendo ésta. Mientras tanto, tenemos FPR, que es justamente lo contrario; la tasa de veces que nuestro modelo ha fallado acertando nuestra clase positiva "Run".

5. **Ahora vamos a crear 5 instancias pero con la clase como "?". A continuación, vamos a pasarle dichos datos al clasificador.**

Las cinco instancias que hemos decidido añadir son las siguientes (vamos a predecir su clase con el árbol podado):
?, 8737.2, 1839, 2462, 95.0 ?, 2134.4, 3156, 4203, 34.0
?, 1023.2, 3192, 4121, 70.0 ?, 3842.11, 3890, 4022, 30.0
?, 2490.8, 2104, 3913, 50.0
Y nos ha predicho lo siguiente: Curiosamente, nos ha clasificado todos como la

inst#	actual	predicted	error	prediction
1	1:?	1:Run		0.947
2	1:?	1:Run		0.947
3	1:?	1:Run		0.947
4	1:?	1:Run		0.947
5	1:?	1:Run		0.947

clase "Run". Si echamos un vistazo a nuestro árbol, vemos cómo cada distinta instancia acaba de alguna forma u otra en una hoja de la misma clase. Esto requeriría una investigación para saber por qué es que todos han sido clasificados de tal forma, y si es que el árbol no está funcionando como debería, o es porque los valores que hemos ofrecido no han sido lo suficientemente buenos.

Sexto Laboratorio (25-26 de octubre)

En este laboratorio se nos pide escoger una competición y responder unas preguntas.

1. **¿Quién es el patrocinador de la competición? Tamaño del training set y del test set.**

Bueno pues vamos a escoger la competición llamada "Earthquake Damage Prediction". Al ser una competición comunitaria, no tiene ningún patrocinador. No podemos ver el test set porque es solo para invitados, pero consta de 18.75MB,

mientras que el test set consta de 4.69MB. Tras hacer un poco más de investigación, podemos ver como en el siguiente enlace: https://vidhur2k.github.io/Earthquake-Prediction/
Se aportan más datos sobre el dataset.

2. **¿Cuál es la clase a predecir? ¿Está el problema desbalanceado? ¿Cuántas variables predictoras hay en total?**

Lo que se pretende es predecir el grado de daño causado a un edificio debido a un terremoto. Hay tan solo tres grados de daño: 1, 2 y 3. Está desbalanceado, ya que la mayoría de casos caen en la clase 2 (podemos verlo en el análisis que se lleva a cabo en el enlace). Tiene 39 variables predictoras.

1. **Métrica Score.**

La métrica score se puede encontrar en el enlace que kaggle cuelga:
`https://www.drivendata.org/competitions/57/nepal-earthquake/page/136/`:

$$F_{\text{micro}} = \frac{2 \cdot P_{\text{micro}} \cdot R_{\text{micro}}}{P_{\text{micro}} + R_{\text{micro}}},$$
$$P_{\text{micro}} = \frac{\sum_{k=1}^{3} TP_k}{\sum_{k=1}^{3} (TP_k + FP_k)},$$
$$R_{\text{micro}} = \frac{\sum_{k=1}^{3} TP_k}{\sum_{k=1}^{3} (TP_k + FN_k)}$$

El, modelo que mediante esa fórmula obtengan el valor más alto será el ganador. Es algo distinto al usual F1 Score que nosotros conocemos porque estamos tratando de predecir tres variables.

3. **Trata de entender y luego de explicar en la respuesta, al menos 3-4 variables predictoras que pudieran ser relevantes en la clasificación.**

- **Age:** La edad de un edificio, esta variable es de las más importantes. Si un edificio es antiguo, tendrá más probabilidades de sufrir daños más graves que un edificio más nuevo.
- **Land surface condition:** El estado en el que el territorio donde dicho edifcio se encuentra. Ya puede ser un edificio nuevo, que si se encuentra en un territorio inestable, el terremoto bien podría causar que los cimientos del edificio den de sí.
- **Foundation type:** El tipo de cimiento usado como base para el edificio también es crucial, por lo que si se ha usado un material no tan fiable, un terremoto podría ser causante de una desgracia.

4. **¿Por qué ha llamado tu atención, por qué te "gusta"? ¿Por qué has escogido éstas y no otras? ¿Qué otros datasets has considerado y ojeado y finalmente no has incluido, cuáles?**

Me ha llamado la atención porque no me esperaba ver una competición sobre algo tan serio como lo es el daño que puede causar un terremoto. Simplemente me esperaba otro tipo de competiciones, en las que se predijera algo menos chocante, por lo que al ver una base de datos sobre ésto, me decanté por él. He escogido esta porque las demás eran competiciones que no cumplían con el criterio que tenían que cumplir para poder escogerlas. Entre ellas se encuentran sobre todo modelos basados en reconocimiento de imágenes, por ejemplo, "Judging a book by its cover", donde dada una foto de una portada de un libro, se predecía a qué genero pertence. Otra interesante era: "Cats vs Dogs", donde un modelo debía clasificar una imagen si se trataba de un perro o un gato. La primera era una clasificación categórica, pero al ser el dataset de imágenes, no podía hablar mucho al respecto, lo mismo ocurre con la última —salvo que era una clasificación binaria. Otra que me llamó la atención fue "Microsoft Malware Classification", donde dados unos datos de una aplicación, se debía determinar la probablidad de que era un malware. Obviamente, al ser el output una variable numérica, se descartó al instante.

Séptimo Laboratorio (2-3 de noviembre)

Este laboratorio está dedicado a tratar la selección de variables, y distintos métodos.

1. **Explica la diferencia principal entre los métodos de selección de variables "filter" y "wrapper".**

En filter lo que hacemos es escoger un número concreto, llamémoslo K, de variables, y dejamos el resto de lado. Éstas K variables que escogemos no son cualesquiera, sino que son las k primeras variables que tienen una mayor información mutua. Mientras tanto, con wrapper, hacemos una aproximación más bruta, puesto que comprobamos cada combinación posible entre las variables del dataset y nos quedamos el mejor modelo clasificatorio de todos. Sobra decir que éste último será bastante costoso, puesto que comprobar todas y cada una de las combinaciones lleva su tiempo. Aun así, en problemas con dimensionalidad baja ésta opción puede que no resulte tan loca.

2. **Escoge la base de datos adult.arff. Describe el problema y describe al menos cuatro variables predictoras.**

El adult datataset es una famosa colección de datos de UCI - machine learning repository. La idea es predecir si los ingresos excenden los 50 mil dólares por año dado unos datos obtenidos del censo de EEUU de 1994. Para poder predecir si un sujeto pasa el umbral seleccionado, se usan 14 variables. De las cuales vamos a explicar cuatro:

- **Age:** La edad del sujeto actual, es una variable numérica.

- **Workclass:** Es una variable cuantitativa con un total de 8 categorías: Privado, Autónomo con Empresa, Autónomo sin Empresa, Gobierno federal, Gobierno Local, Gobierno Estatal, Sin paga, Jamás ha trabajado.
- **Relationship:** El estado civil del individuo, consiste en: Casado civilmente, Divorciado, Jamás Casado, Separado, Viudo, Casado con esposo ausente o casado con pareja en la armada.
- **Horas por semana:** Atributo numérico, representa las horas que el sujeto trabaja por semana.

3. **Realiza dicho ranking mediante una de las métricas filter citadas, y muestra la expresión fórmula que utiliza para realizar dicho rankeo.**

Vamos a usar el GainRatio. La fórmula que usa es la siguiente:

$$\mathrm{I}(\mathrm{Class}, \mathrm{Attribute}) = \frac{H(\mathrm{Class}) - H(\mathrm{Class}|\mathrm{Attribute})}{H(\mathrm{Attribute})}$$

Es la misma fórmula que usamos en clase, salvo que esta vez lo dividimos por H(Attribute). El ranking resultante se puede ver en la figura 7 (al fondo de la página).

4. **¿Hay algunas variables con valor nulo? ¿Qué quiere decir?**

Las hay, éstas se denotan por medio del valor "?". Simplemente quiere decir que hay valores que no han sido registrados en el censo, los podríamos poner nosotros mediante los filtros aprendidos en laboratorios previos.

5. **Comprueba si las variables que aparecen en la parte alta del ranking tienen un papel relevante en el árbol de clasificación J48.**

Como podemos ver en la figura 8, sí que tiene que ver con el árbol obtenido. Esto es porque el árbol de decisión se hace con el Gain Ratio. Curiosamente, hours-per-week, pese a no estar alto en el ranking ni mucho menos, se encuentra en el segundo nivel. Podemos intuir que esto se debe a que los ingresos están correlacionados con las horas que trabajamos por semana, pero a su misma vez no guarda una gran correlación con el hecho de que gane más de 50 mil dólares al año o no. Lo mismo ocurre entre los ingresos y el hecho de que el sujeto esté casado o no, si no pasa un umbral, iremos directamente a dicho nodo. Es más, si resulta que el sujeto está casado, iremos al nodo de capital-loss, también curioso.

6. **¿Y en las reglas inducidas por el clasificador Jrip? ¿Era necesario en "Test Options" tener escogida una k-fold cross-validation?**

No era necesario tener escogido ningún k. Si estamos únicamente interesados en el modelo generado por JRip y no su evaluación, podemos escoger no seleccionar un k-fold cross-validation.

```
Ranked attributes:
 0.17351  11 capital-gain
 0.11623  12 capital-loss
 0.08553   6 marital-status
 0.07676   8 relationship
 0.04003  10 sex
 0.03705   5 education-num
 0.03142   4 education
 0.02767   1 age
 0.0264   13 hours-per-week
 0.02423   7 occupation
 0.01098   2 workclass
 0.0103    9 race
 0.00973  14 native-country
 0         3 fnlwgt
```

Figure 7: Ranking

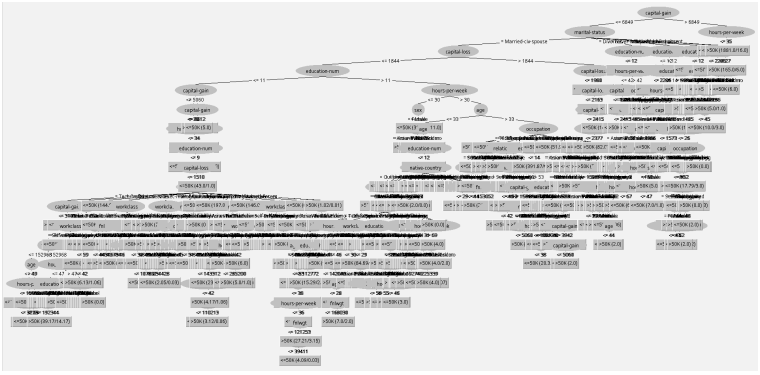


Figure 8: Árbol de clasificación J48.

7. **¿Qué opción de "test options" es la computacionalmente la más económica, y con la que sigas obteniendo el modelo final?**

La más económica si solo queremos el modelo final se encuentra en test options - More Options... Y deseleccionar todas las opciones salvo output model. De esta forma WEKA no se molesta en dedicar tiempo para construir una matriz de confusión, ahorrándonos una parte considerablemente grande de la computación.

8. **Construye un clasificador k-nn del "vecino más próximo" que contenga únicamente las top 5 predictoras del ranking aprendido. ¿Tiene mejor porcentaje de acierto que el que usa todas las variables predictoras?**

Como podemos ver en las figuras 9 y 10, con el top 5 del ranking hemos obtenido un mejor TPR que sin el top 5. Parece ser que filtering con k=5 estamos quitando variables que más que ayudar podrían estar obstaculizándonos la predicción. También tiene su sentido, ya que como hemos visto previamente en la figura 5, a partir de la posición 5 del top la información mutua empieza a caer.

9. **Construye el mismo clasificador pero que contenga únicamente el top 10, ¿tienen mejor porcentaje de acierto que el clasificador aprendido con todas las predictoras?**

Vamos a usar el clasificador k-nn con k = 3, el resultado se encuentra en la figura 11. Una vez más, es mejor que si decidieramos escoger todas las variables, pero ligeramente peor que si escogiesemos las primeras 5. Por lo tanto, podría ser que nuestro número de atributos ideal se encuentre entre 5 y 10 atributos (o menos, pero no más).

10. **¿Esta manera de seleccionar variables te garantiza que entre las "top-K" seleccionadas para aprender el clasificador no haya dos variables redundantes? ¿Qué opinas?**

No lo garantiza realmente, aunque podría darnos alguna pista. Es decir, si vieramos dos variables que tienen la misma información mutua con respecto a la clase o una muy similar podría ser que fueran practicamente iguales, pero no nos garantiza nada. Si realmente quisieramos saber si son redundantes, o bien tendríamos que calcular la correlación entre ambos, o a simple vista podríamos llegar a dicha conclusión.

```
Correctly Classified Instances       40518               82.9573 %
Incorrectly Classified Instances      8324               17.0427 %
Kappa statistic                          0.3881
Mean absolute error                      0.223
Root mean squared error                  0.3354
Relative absolute error                 61.2404 %
Root relative squared error             78.6179 %
Total Number of Instances            48842

=== Confusion Matrix ===

     a     b   <-- classified as
  3519  8168 |     a = >50K
   156 36999 |     b = <=50K
```

Figure 9: Escogiendo el top 5 del ranking.

```
Correctly Classified Instances       40001               81.8988 %
Incorrectly Classified Instances      8841               18.1012 %
Kappa statistic                          0.4867
Mean absolute error                      0.2083
Root mean squared error                  0.3718
Relative absolute error                 57.2275 %
Root relative squared error             87.1474 %
Total Number of Instances            48842

=== Confusion Matrix ===

     a     b   <-- classified as
  6732  4955 |     a = >50K
  3886 33269 |     b = <=50K
```

Figure 10: Sin escoger el top 5 del ranking.

```
Correctly Classified Instances       40188               82.2816 %
Incorrectly Classified Instances      8654               17.7184 %
Kappa statistic                          0.5018
Mean absolute error                      0.2052
Root mean squared error                  0.3621
Relative absolute error                 56.3633 %
Root relative squared error             84.8807 %
Total Number of Instances            48842

=== Confusion Matrix ===

    a     b   <-- classified as
 6967  4720 |     a = >50K
 3934 33221 |     b = <=50K
```

Figure 11: Escogiendo el top 10 del ranking.

11. **¿Qué hace WEKA para calcular la correlación entre variables numéricas y la variable de la clase nominal?**

Como podemos ver en el siguiente enlace:
`https://weka.sourceforge.io/doc.dev/weka/attributeSelection/InfoGainAttributeEval.html#InfoGainAttributeEval--`
El método que se encarga de calcular la información mutua hace una llamada a discretize(). Este método se encarga de particionar el conjunto de números, de forma que se obtendrán distintos subconjuntos con los que ya se podrá calcular la información mutua como si se tratara de una variable nominal cualquiera. Mirando el árbol de decisión J48 en la figura 6, se puede observar que, por ejemplo, con capital gain, se ha decidio hacer una partición en 6849. Con capital loss también se ha hecho una división a partir del número 1844. Para más información, el siguiente enlace es de gran utilidad:
`https://weka.sourceforge.io/doc.dev/weka/filters/supervised/attribute/Discretize.html`
Nótese que también tenemos la opción de algo así como "binarizar" los atributos numéricos. Si marcásemos este apartado a True, entonces haría lo siguiente:

- **Si el valor numérico es 0:** Entonces el valor de dicho atributo al binarizarlo será de 0.
- **Si el valor número es desconocido:** Entonces el valor de dicho atributo también será desconocido.
- **Si es otro valor:** Entonces el valor de dicho atributo al binarizarlo será de 1.

Para más información:
`https://weka.sourceforge.io/doc.dev/weka/filters/supervised/attribute/Discretize.html`

12. **Wrapper feature selection.**

El número distintos de subconjuntos que podemos generar son de órden de 2^d, donde d es el número de variables que tiene el dataset. Un coste exponencial, cabe esperar que este método de encontrar el subconjunto óptimo será de un coste alto. En nuestro caso, adult.arff consta de 15 atributos, luego: $2^{15} = 32768$. Por éste mismo motivo es por el que vamos a realizar una busqueda local — una búsqueda global es demasiado costosa. Entonces, escogemos "WrapperSubsetEval", escogemos 5 folds, y el árbol de decisión J48. En cuanto al método de búsqueda, usaremos GreedyStepWise. Este método de busqueda es, como su nombre dice, avaricioso: dado un subconjunto inicial, irá añadiendo nuevas variables mientras evalua el resultado del modelo resultante, en el momento que un modelo sea peor que el previo, se detendrá. Nótese que también nos dan la opción de searchBackwards, que si lo pusieramos a True, en vez de ir añadiendo variables, las iría quitando. El resultado se encuentra en la figura 12.

13. **Feature selection versus feature extraction. (Con iris.arff).**

La combinación lineal que nos ha salido después de haber aplicado PCA se encuentra en la figura 13. La variable que forma el primer eje no es más que una combinación lineal entre pettallength, petalwidth, sepalwidth y sepallength. Por otra parte, el segundo eje está formado por otra combinación lineal entre sepalwidth, sepallength, petalwidth y petallength.
En feature selection nosotros estamos descartando algunas variables, en wrapper selection se ve claro: empezando con un subconjunto de variables, añadimos más variables hasta que hayamos encontrado el máximo local. Por ende, estamos dejando de lado otras variables que o bien hemos pasado por alto o no aportan lo suficiente. La pregunta es, ¿con feature extraction estamos descartando variables?
La respuesta es que realmente, no. Refiriéndonos de nuevo a la figura 13, el caso es que aunque aunque hayamos creado dos nuevas variables, no son más que una combinación lineal entre las variables del dataset. Es decir, estamos *transformando* los componentes en una forma más compacta, y perdiendo dimensionalidad de esta forma.

```
Search Method:
        Greedy Stepwise (forwards).
        Start set: no attributes
        Merit of best subset found:    0.861

Attribute Subset Evaluator (supervised, Class (nominal): 15 class):
        Wrapper Subset Evaluator
        Learning scheme: weka.classifiers.trees.J48
        Scheme options: -C 0.25 -M 2
        Subset evaluation: classification accuracy
        Number of folds for accuracy estimation: 5

Selected attributes: 1,5,6,8,11,12,13 : 7
                     age
                     education-num
                     marital-status
                     relationship
                     capital-gain
                     capital-loss
                     hours-per-week
```

Figure 12: Resultado Wrapper.

```
Ranked attributes:
 0.2723  1 -0.581petallength-0.566petalwidth-0.522sepallength+0.263sepalwidth
 0.042   2 0.926sepalwidth+0.372sepallength+0.065petalwidth+0.021petallength
```

Figure 13: Ranked attributes.

14. **Ejecuta las siguientes líneas de código en R y explica su funcionamiento.**

Comenta cada línea de código, aportando tu toque personal y sin traducir el manual sin entenderlo, interpreta lo que indica la ayuda, y responde a las preguntas que se te hacen en el script.

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  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
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Como podemos ver, este es una representación del dataset de iris.arff. De esto se encarga la función summary, que en este caso, dado un objeto de data frame nos ha ofrecido un análisis de los datos. Vamos a listarlos:

- **Min.** El valor mínimo de una variable en todo el data set.
- **1st Quartile.** El primer cuartil es el número que se encuentra en la mitad entre el último número y el mediano. Debajo de él se encuentran el % 25 de los datos.
- **Median.** Median, o el mediano, es el equivalente al segundo cuartil. Es el número que se encuentra justo en la mitad de una variable dada del dataset. Corta los datos por la mitad — el 50% de los datos se encuentra por encima, y viceversa.
- **Mean.** El valor medio de una variable.
- **3rd Quartile.** El tercer cuartil, el número que se encuentra en la mitad entre el número de la mitad y el número más grande.

```
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
```

Primero de todo expliquemos la primera línea de código. Se está haciendo una llamada a la función prcomp, no sin antes pasar únicamente las variables predictoras, evitando así mandar la clase a predecir. Finalmente, con scale = TRUE lo que se hace es estandarizar las variables, lo que viene a ser establecer la media a cero y establecer la desviación estandar a uno. Con summary podemos visualizar el objeto, que es de la clase prcomp, pero, **¿qué hace prcomp exactamente?**

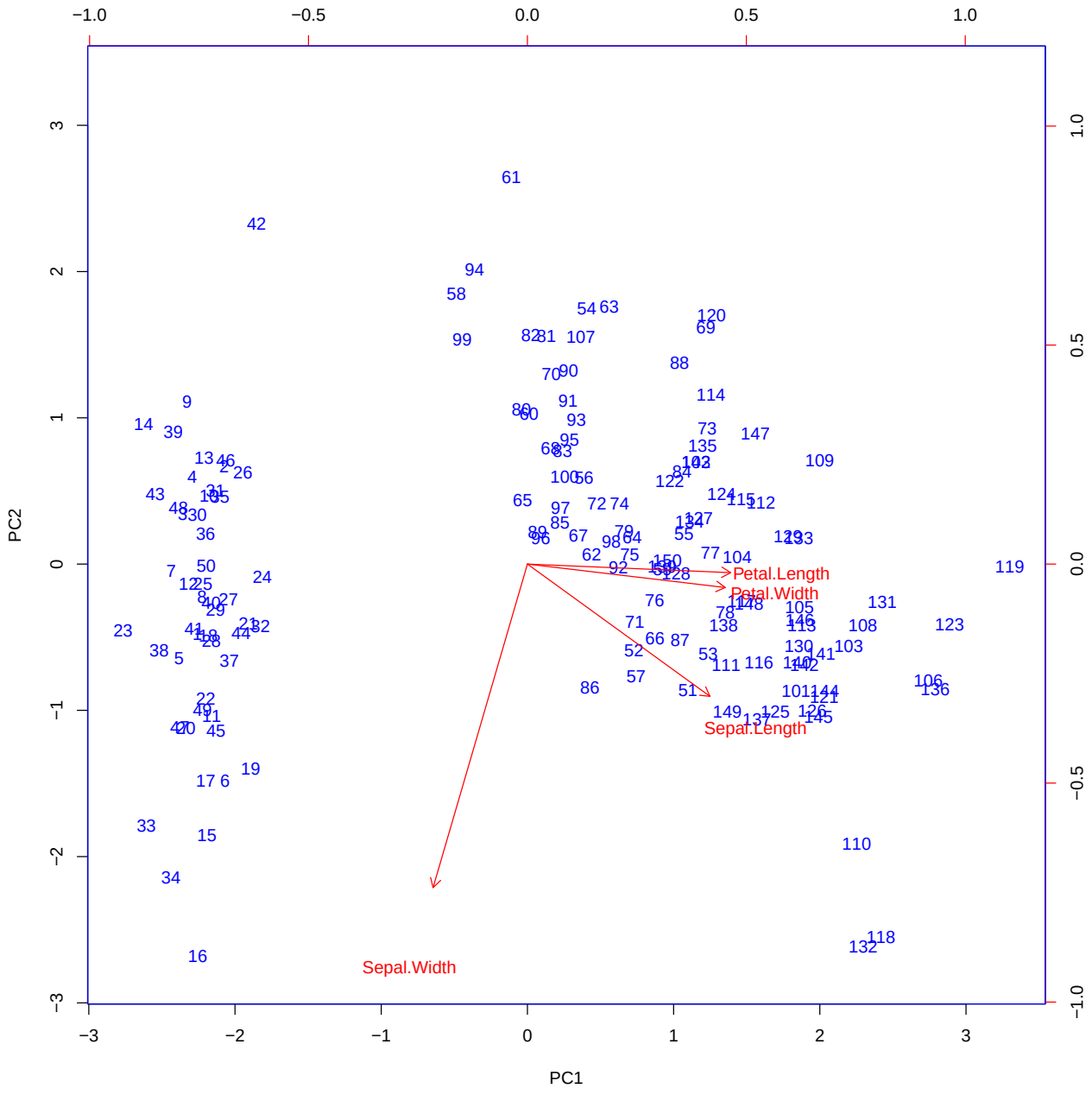
Pues bien, como podemos observar, tenemos cuatro columnas con PC, cuyas iniciales significan "Principal Components". Es decir, son los componentes principales que la función ha calculado dada la base de datos iris.arff, que más tarde podremos usar para usar nuestras predicciones. Además, nos lo pone de una manera sencilla, ya que podemos ver mediante la desviación estándar y la proporción de varianza cuántos datos ha con-

seguido representar cada componente: por lo visto, PC1 consta con una gran varianza, y junto con PC2, según podemos ver con Cumulative Proportion, o la proporción acumulativa, estaríamos cubriendo una gran parte de los datos con tan solo dos dimensiones — una gran diferencia, ya que antes basabamos nuestras predicciones con cuatro dimensiones. Por lo tanto, una opción atractiva sería escoger PC1 y PC2 como "nuevas" variables.

14.1 ¿Qué porcentaje de la varianza recoge la primera y segunda componente?

PC1 consta de 0.7296, y PC2 de 0.2285, es por eso que si sumamos ambos componentes,— y como nos indica el propio objeto—, estaríamos cubriendo un total de 0.9581 de varianza: **Como dicha cifra recoge más del 90 % de la varianza, la visualización de todos los puntos sobre ambas componentes está permitida.**

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



Se trata de una llamada a la función biplot con scale = 0, lo que quiere decir que el usuario no quiere que se realice ningún escalado de los valores originales, (con col(blue,red) especificamos los colores que queremos ver).

14.2 ¿Qué representan los dos ejes de la visualización?

El nuevo eje "x" es PC1, es la dirección en la que los datos presentaban una mayor varianza en el dataset original, no es más que una combinación lineal de las variables originales. El nuevo eje "y" es PC2, es perpendicular a PC1, y se ha calculado a partir de PC2; aun así, su objetivo sigue siendo capturar la máxima varianza, en este caso, es el segundo "mejor" componente.

14.3 ¿Qué representan las direcciones rojas en la visualización?

Las direcciones rojas representan las variables del dataset original. En concreto, representa la relación entre las distintas variables y su longitud su desviación estándar respectiva.

Si dos direcciones iguales forman un angulo muy pequeño, querrá decir que hay una correlación entre ambas. Si son dos direcciones opuestas, nos querrá decir que guardan también una fuerta relación, pero negativa, en nuestro gráfico no nos ha salido dicho caso.

Bien, **¿y qué quiere decir esto sobre las flechas de nuestro gráfico?**

Podemos sacar varias conclusiones útiles, como la siguiente:

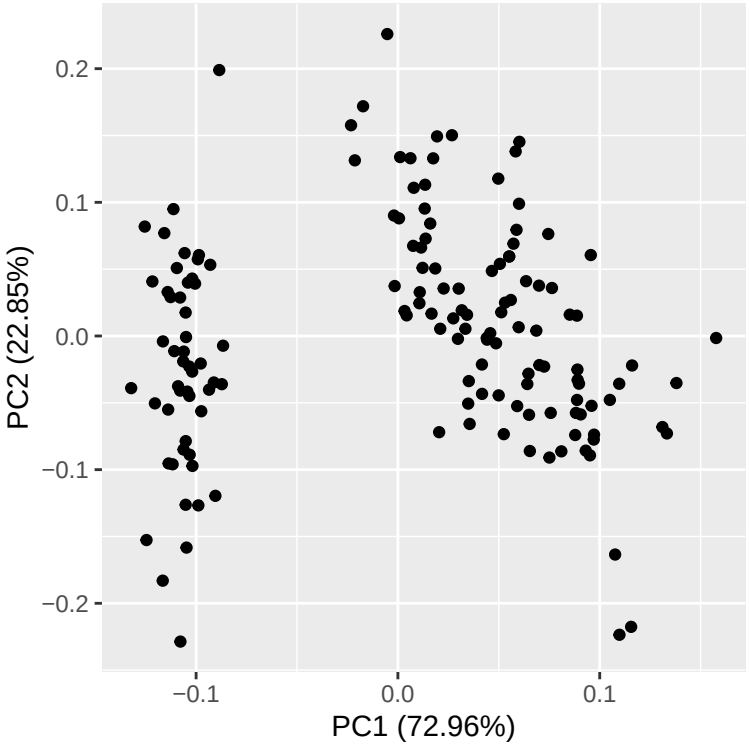
Tanto petal.length como petal.width están muy cerca una de la otra y tienen una dirección parecida. Esto se debe a que guardan una correlación muy fuerte, cosa que tiene sentido, ya que es posible que haya una proporción entre la anchura de sus pétalos y su largura. No ocurre lo mismo entre sepal.length y sepal.width, curioso; de hecho, parece que sepal.length guarda más relación con petal.width que con su respectiva pareja, aunque dejemos el estudio de plantas para otra persona más especializada. El caso es que sepal.width forma un ángulo de casi 90 grados con cada flecha, es decir, ni es opuesto a las otras variables, ni comparte dirección. Es decir, guarda una relación muy débil con ellas, tan débil que uno podría decidir descartar esta variable.

14.4 Indica las diferencias entre las siguientes 3 visualizaciones.

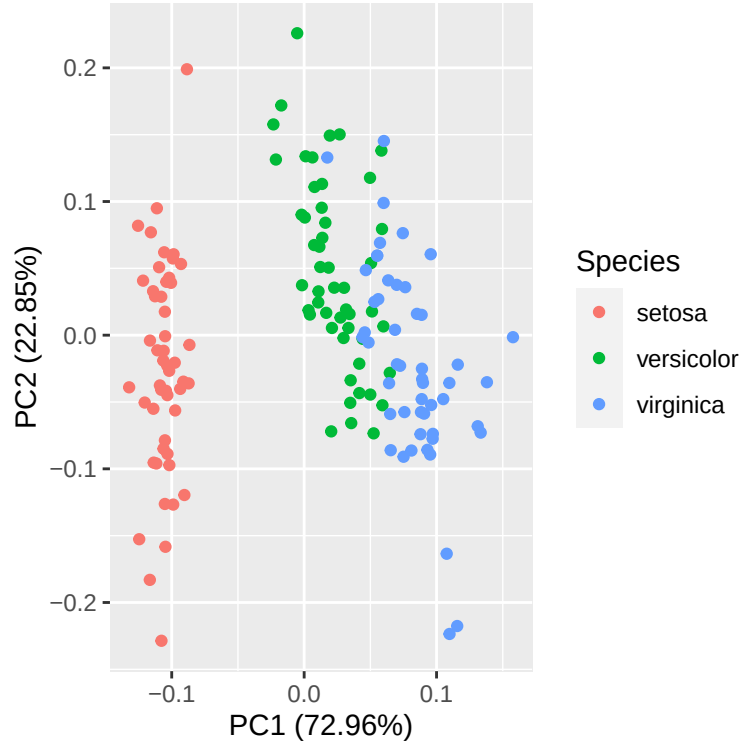
```
library(ggfortify)
```

```
## Loading required package: ggplot2
```

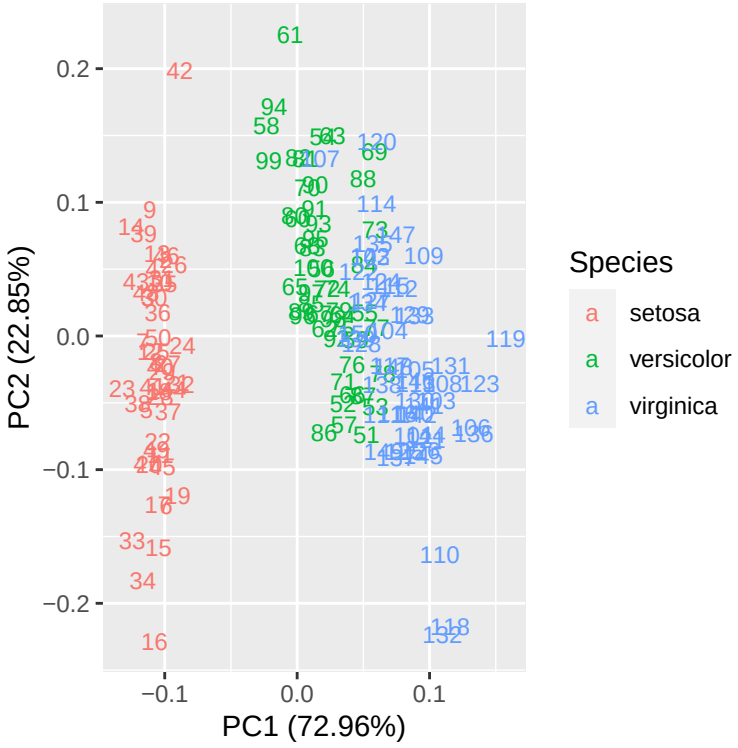
```
library(ggplot2)
autoplot(pca_iris)
```




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




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



El primer gráfico es el más sencillo de todos, puesto que muestra la distribución de las flores en función de PC1 y PC2, los componentes que nosotros hemos escogido (también muestra el porcentaje de varianza que cubre cada uno de ellos). No es más que el fruto de la llamada a la función autoplot, que se encarga de mostrar un gráfico de un objeto (en nuestro caso pca_iris). La segunda llamada da más juego a la función; esta vez asignamos un color distinto a cada clase de flor. Para conseguir esto, lo que se ha hecho es llamar a la función especificando la base de datos con la que estamos trabajando (data = iris), para luego así poder asignar distintos colores (con colour = 'Species'). Vemos cómo el conjunto iris setosa se encuentra más cerca del eje PC2 que ningún otro, esto es porque ésta flor debe ser la que más aporta en el componente PC2. Finalmente, en la última línea de código con shape = FALSE, se hace el gráfico sin los puntos, (y les establece un tamaño con label.size = 3). Con esto se consigue enseñar el número asignado cada flor, y no debería haber ninguno superior a 150: éste es el número total de instancias en iris.arff.