Tarea-Examen 3: Aprendizaje Supervisado

Curso Avanzado de Estadística. Profa. Guillermina Eslava Gómez.

Aldo Sayeg Pasos Trejo. César Cossio Guerrero.

Posgrado en Ciencias Matemáticas. Universidad Nacional Autónoma de México.

16 de abril de 2020

1. Problema 1

Buscamos realizar un estudio de clasificación sobre la base de datos `Pima`, que consiste en 532 observaciones de 7 variables usadas para describir el estado de salud de las mujeres de la tribu Pima, cuya tasa de enfermedad de Diabetes Mellitus es inusualmente alda. La descripción [1] de las variables se muestra en la tabla 13.

Variable	Descripción
"npreg"	número de embarazos
"glu"	Concentración de glucosa en el plasma para una prueba oral
"bp"	Presión arterial diastólica (mmHg)
"skin"	grueso de la piel del triceps (mm)
"bmi"	Índice de masa corporal (kg/m^2)
"ped"	Función pedigree para diabetes
"age"	Edad (años)
"type"	Indicador de diabetes ("yes" o "no")

Tabla 1: Descripción de las variables

Antes de realizar cualquier estudio sobre el sistema, analizamos la relación entre sus variables analizando su correlación en la figura 1, su separación por clase en una gráfica a pares como en la figura 2, por componentes principales como en 3 y sus distribuciones individuales en las figuras 4,5,6,7,8,9 y 10.

De ese análisis, podemos concluir rápidamente que "bmi" presenta una alta correlación con "skin". Esto se puede entender claramente ya que el índice de masa corporal debe de tener un efecto claro

sobre la piel en el tríceps. De igual manera, la correlación entre "age" y "npreg" es totalmente obvia debido a que el número de embarazos se relaciona directamente con la edad.

En cuanto a las diferencias entre clases, nuevamente no se pueden observar muchas diferencias sustanciales y las que se muestran como la de la figura 10 se entiende nuevamente de manera intuitiva ya que la incidencia de Diabetes debe de ser poca en mujeres jóvenes. La diferencia entre la distribución de "glu" en la figura 5 también se puede explicar de manera trivial al igual que la de la figura 8 al ser la obesidad un factor de riesgo para la diabetes.

Esta análisis nos permite especular que las variables con mayor significancia estadística pueden ser "bmi","glu","age" y que podemos usar una sola variable pare representar variables altamente correlacionadas.En principio, nos interesa usar 4 modelos para clasificar

a) Análisis de discriminante lineal

b) Naive Bayes

c) Regresión logística

d) Suppor Vector Machines (SVM)

Yendo un poco más lejos, consideramos realizar un análsis incluyento también el Análisis de discriminante cuadrático y K-Neighbors. La figura 11 muestra las tasas de clasificación global y local para esos modelos para distintos tamaños del conjunto de entrenamiento de los datos. La separación de los datos se realizó sin tomar en cuenta el mantener la proporcionalidad entre las clases. Notemos que la tasa aparente corresponde a la fracción $s = 1.0$. Por limitaciones del poder de cómputo disponible, se realizaron $n = 10$ iteraciones para las tasas no aparentes.

Podemos observar que, en cuanto a la tasa de error global aparente, el modelo K-Neighbors presenta el menor valor.

Sin embargo, pareciera intutivo pensar que las tasas no aparentes deben de mejorar si ahora forzamos a que la separación entre conjunto de entrenamiento y de prueba mantenga la proporcionalidad existen en las clases. La tabla 26 y la gráfica 12 muestran dichos resultados.

En efecto, se muestra que las tasas no aparentes mejoran aunque no significativamente. Si pensamos en la capacidad predictiva del modelo y en el contexto del diagnóstico médico, es claro que la tasa de aceptación que más nos interesa son los falsos negativos pues nos interesa diagnosticar con mayor precisión. En ese sentido, los modelos entrenados manteniendo la proporcionalidad presentan tasas muy elevadas de error para fracciones pequeñas.

Antes de proceder modelo por modelo, buscamos para cada variable numérica y positiva del modelo, cual es el valor λ que maximiza la verosimilitud de una transformación Box-Cox para esa variable. Dichos valores, junto con sus intervalos de confianza, se pueden consultar en la tabla 27

Habiendo realizado primero este análisis, ahora nos disponemos a presentar los 4 modelos relevantes.

1.1. Análisis de discriminante lineal

Para discriminante lineal, no hay muchso parámetros que podamos ajustar para posibles modelos. Nos enfocamos más bien en buscar transformaciones o interacciones entre las variables.

Despues de mostrar que las transformaciones Box-Cox no presentan una mejoría sustancial para ninguno de los modelos, y que esta es considerada la mejor clase de transformaciones dado que las figuras 1, 2 no nos muestra que otra tranfromación podría mejorar la separación con una variable, nos limitamos a no realizar transformaciones sobre las variables.

Así, solo añadimos posibles interacciones. Entre interacciones a pares, encontramos que la interacción que presentaba menor tasa global aparente de error era la interacción entre "ped" y "age".

Añadiendo esta interacción al modelo, despues consideramos interacciones terciarias que lo hagan óptimo. Sin embargo, todas las tasas de interacciones muestran tasas similares (mayores 1 0.20), por lo que no consideramos que representen una mejoría sobre nuestro modelo y, así, nos quedamos con este modelo en interacción a pares como el mejor clasificador.

1.2. Naive Bayes

Nuevamente, no existen muchos parámetros modificables en el método de Naive Bayes y, por los argumentos anteriormente mostrados, recurrimos a otra interacción binaria, la interacción entre "ped" e "bp".

1.3. Regresión Logística

Existen mejores estimadores para determinar los valores de la regresión logística. La tabla de los p-values para una regresión con todas las variables la podemos presentar en la tabla 28.

Despues de revisar la sigfinicancia estadística de la regresión, nos podemos quedar solamente con las variables "glu","bmi","ped" y "age". Ademas, considerando que la edad presenta una gran influencia, podemos incluir un término cuadrático para la edad "age^2". Así, este modelo es el que consideramos óptimo y representativo de este método.

1.4. Support Vector Machines

Por ser el método que presenta más parámetros para el clasificador, primero realizamos una corrida para kernels lineales, de funciones de base radial de grado 1,3 y 5 y polinomiales de los mismos grados. Los resultados se pueden consultar en la figura 13. Aunque existen también los Kernels sigmoides [2], al probar con estos encontramos tasas de clasificación muy bajas, por lo que los excluimos del análisis.

El kernel lineal se desempeña muy bien también al ser comparado por los otros. Tanto clasificadores de base radial como los polinomiales muestran muy buen desempeño aun para grados bajos. Sin embargo, el clasificador que consideramos como el óptimo es el que minimiza la tasa global aparante, es decir, **el de kernel polinomial y grado 5**.

Por las limitaciones del poder computacional, resultó imposible analizar distintos valores del parámetro γ. Aunque ahora podríamos analizar interacciones, por la estructura de los parámetros del sis-

Método	Variables en el modelo	Notas extra
Análisis de Discriminante Lineal	"npreg", "glu", "bp", "skin", "bmi", "ped", "age", "type", "ped*age"	-
Naive Bayes	"npreg", "glu", "bp", "skin", "bmi", "ped", "age", "type", "ped*bp"	-
Regresión logística	"'glu", "bmi", "ped", "age", "age^2"	-
Support Vector Machines	"npreg", "glu", "bp", "skin", "bmi", "ped", "age", "type"	Kernel Polynomial de grado 5

Tabla 2: Variables de los modelos

tema, sabemos que estas ya se están contemplando al aumentar el grado del kernel. Así, dejamos este modelo como el definitivo para este clasificador y presentamos sus tasas de error de manera gráfica individual como tabular en la figura 14.

1.5. Conclusiones

Para resumir, en la tabla 2 mostramos las variables utilizadas por el modelo correspondiente a cada método: Comparamos los cuatro modelos seleccionados para cada método. La tabla 30 y la figura 15 muestran dicha comparación. De Inmediatamente podemos concluir que ningún método alcanza una tasa de error global menor al 20 % . Todos los modelos logran mantener a la tasa de error global, tanto aparente como no aparente, en valores menores a 25 %. Sintetizamos dichos resultados en las tablas 4 y 6.

	LDA	Naive Bayes	Logistic	SVM, kernel = poly, deg = 5
Mean global error, entrenamiento: 1.0	0.205	0.205	0.207	0.205
Mean Yes error, entrenamiento: 1.0	0.412	0.412	0.390	0.475
Mean No error, entrenamiento: 1.0	0.101	0.101	0.115	0.070

Tabla 4: Tasas de error aparente locales y globales para los cuatro métodos con conjuntos divididos manteniendo proporcionalidad

	LDA	Naive Bayes	Logistic	SVM, kernel = poly, deg = 5
Mean global error, entrenamiento: 0.7	0.216	0.210	0.207	0.223
Mean Yes error, entrenamiento: 0.7	0.412	0.413	0.381	0.496
Mean No error, entrenamiento: 0.7	0.106	0.104	0.117	0.071

Tabla 6: Tasas de error aparente locales y globales para los cuatro métodos con conjuntos divididos manteniendo proporcionalidad

En general, el modelo que menos falsos negativos da es la SVM. Sin embargo, su tasa de falsos positivos siempre es muy alta. En general, consideramos que el mejor método resulta ser el modelo de regresión logística pues sus tasas de error locales son bastante estables independientemente del tamaño de la muestra y, aunque la global no aparente no tiene tanta estabilidad, la global aparente presente un valor bajo y no distinto a los otros valores.

2. Problema 2

Tenemos una base de datos, consistente en 236 observaciones que representa a pacientes de una clínica danesa de cardiología [3], con 14 variables categóricas cuya descripción muestra la tabla 7

Variable	Descripción
"Sex"	Sexo ("male" or "female")
"AngPec"	Indicador de si padece angina de pecho ("Atypical", "None" or "Typical")
"AMI"	Indicador de problemas cardiacos ("yes" or "no")
"QWave"	Indicador de presencia de una onda en el electrocardiograma (ECG) ("yes" or "no")
"QWavecode"	Indicador de is es representativo la presencia de la onda Q ("Usable" or "Nonusable")
"STcode"	Indicador de si es representativa la presencia de una onda T en el ECG ("Usable" or "Nonusable")
"STchange"	Indicador de la presencia de una onda T en el ECG ("yes" or "no")
"SuffHeartF"	Indicador de si ha presentado infartos cardiacos ("yes" o "no")
"Hyptertrophi"	Indicador de si ha habido hipertrofia arterial ("yes" o "no")
"Hyperchol"	Indicador de si hay colesterol alto ("yes" o "no")
"Smoker"	Indicador de si es fumador ("yes" o "no")
"Inherit"	Indicador de si hay predisposiciones hereditarias ("yes" o "no")
"Heartfail"	Indicador de si ha tenido paros cardiacos ("yes" o "no")
"CAD"	Indicador de presencia de la enfermedad de las arterias coronarias ("yes" o "no")

Tabla 7: Descripción de las variables

Nos interesa clasificar los valores de "CAD" correctamente. Para visualizar las variables no podemos graficar ni realizar un cambio a componentes principales de estas variables al ser categóricas, podemos ver un histograma de sus distintos valores entre las clases que las componen, como muestran las figuras 16,18,17,19,20,21,22,23,24,25,26,27 y28.

De la valoración cualitativa, vemos que las variables que muestran una mayor diferencia en sus frecuencias para cada categoría entre sus dos grupos son "AngPec", "AMI", "QWave" y "Hyperchol". Es intuitivo pensar que el alto colesterol, la angina de pecho y la presencia de malestares cardiacos en general puede contribuir a la enfermedad CAD, por lo que las diferencias entre los grupos nos parecen razonables. Sin embargo, sin mayor conocimiento sobre las variables, no podemos realizar más a severaciones.

Procedemos a construir modelos de clasificación sobre con los siguientes métodos

a) Naive Bayes

b) Regresión logística

c) Support Vector machines

Nuevamente, de manera previa, realizamos un barrido con muchos métodos de clasificación para obtener un panorama general de su utilidad. La tabla 32 y la figura 29 muestran los resultados de dichas cosas. La tasa global de error aparente es menor al 20 % para todos, lo que indica alta precisión en todos los métodos. También observamos que las tasas locales para cada clase tienen valores cercanos a la local, lo que indica la estabilidad entre clases de los métodos. Procedemos a analizar método tras método.

2.1. Naive Bayes

Nuevamente, no nos es posible refinar este método de muchas maneras, por lo que buscamos la interacción binaria que minimiza el error global aparente. Encontramos que es de la variable "Sex" con "AMI". Mantenemos nuestro modelo con ese clasificador.

2.2. Regresión logística

Realizando de manera inmediata una regresión logística sobre los datos, podemos obtener los p-values para estimar la significancia estadística de cada variable. La tabla 33 muestra dichos resultados. Como podemos observar, las únicas que presentan un p-value que podríamos considerar como significativo son las variables "AngPec","AMI", "STcode","STchange" y "Hyperchol", por lo que son las únicas que consideraremos para el clasificador.

2.3. Support Vector machines

Realizamos el mismo análisis que en el ejercicio anterior, primero realizamos una corrida para kernels lineales, de funciones de base radial de grado 1,3 y 5 y polinomiales de los mismos grados. Los resultados se pueden consultar en la figura . Notamos que el clasificador polinomial de grado 5 es el que presenta la menor tasa de error global aparente, por lo que la consideramos la más adecuada para nuestros datos.

Las limitaciones computacionales nos impiden explorar valores de la variable γ distintos.

Método	Variables en el modelo	Notas extra
Naive Bayes	"Sex", "AngPec", "AMI", "QWave", "QWavecode", "STcode", "STchange", "SuffHeartF", "Hyptertrophi", "Hyperchol", "Smoker", "Inherit", "Heartfail", "CAD", "Sex*AMI"	-
Regresión logística	"AngPec","AMI", "STcode","STchange", "Hyperchol"'	-
Support Vector Machines	"Sex", "AngPec", "AMI", "QWave", "QWavecode", "STcode", "STchange", "SuffHeartF", "Hyptertrophi", "Hyperchol", "Smoker", "Inherit", "Heartfail", "CAD"	Kernel Polynomial de grado 5

Tabla 8: Variables de los modelos

2.4. Conclusiones

En la tabla 8 mostramos las variables utilizadas por el modelo correspondiente a cada método: La gráfica 31 y la tabla 35 muestran los resultados para los tres modelos presentados. Sin ninguna duda, el modelo de la SVM presenta la menor tasa global de error aparente, así como estabilidad directa sobre la diferencia entre las tasas de errores locales no aparentes. En general, sus tasas locales y globales no aparentes también son las menores entre todos los valores presentes. Presentamos un resumen de dichas tablas en las tablas 10 y 12

	Naive Bayes	Logistic	SVM, kernel = poly, deg = 5
Mean global error, entrenamiento: 1.0	0.144	0.169	0.038
Mean Yes error, entrenamiento: 1.0	0.150	0.243	0.037
Mean No error, entrenamiento: 1.0	0.140	0.109	0.039

Tabla 10: Tasas de error aparente locales y globales para los cuatro métodos con conjuntos divididos mantieniendo proporcionalidad

	Naive Bayes	Logistic	SVM, kernel = poly, deg = 5
Mean global error, entrenamiento: 0.7	0.170	0.201	0.200
Mean Yes error, entrenamiento: 0.7	0.171	0.196	0.104
Mean No error, entrenamiento: 0.7	0.154	0.153	0.060

Tabla 12: Tasas de error aparente locales y globales para los cuatro métodos con conjuntos divididos mantieniendo proporcionalidad

Cabe destacar que dicho modelo, el de SVM, no utiliza ninguna interacción ni ninguna transformación de variables, por lo que lo consideramos óptimo en el sentido de que maneja los datos de manera directa

3. Problema 3

Se nos presenta una base de datos de 145 observaciones de 5 variables numéricas y una variable categórica, las cuales describimos a continuación

Variable	Descripción
"Weight"	Peso Kg()
"Fglucose"	Concentración de glucosa en el plasma en ayunoas
"Glucoseint"	Area bajo una curva de glucosa para una medición constante en 3 horas
"InsulineResist"	Area bajo una curva de insulina para una medición constante en 3 horas
"InsulinResp"	Respuesta de insulina a la glucosa
"Class"	Clase de paciente ("Overt","Chemical" y "Normal")

Tabla 13: Descripción de las variables

Nuevamente, primero analizamos la relación entre sus variables analizando su correlación en la figura 32, su separación por clase en una gráfica a pares como en la figura 33 y sobre las componentes principales en la figura 34.

El análisis exploratorio nos muestra que existe una alta correlación entre "Glucoseint" y "Fglucose", lo cual se puede explicar de manera inmediata. Si realizamos un ajuste con un clasificador multinomial tomando como referencia la clase 3 y un clasificador que compara cada clase con las demás (one vs others), se obtienen las tasas mostradas en la figura 39 y en la tabla 37. Usaremos el clasificador multinomial para el análisis.

Para seleccionar modelos predictivos y descriptivos, primero se revisó el artículo en cuestión [**3-3**] que trabajó con los datos. El modelo de referencia es el que el artículo propone y consta de las siguientes tres varaibles "GlucoseInt", "InsulinResp" y "InsulineResist". Cabe mencionar que los nombres utilizados en el artículo original para denotar a las variables son distintos de los presentados en la base de datos.

Por otra parte, seleccionamos la categoría de "Chemical" como referencia, esto debido a que en el artículo parecen estar interesados en diferenciar esta con la categoría Over. Esta elección, desde luego, obedece más a qué pregunta queramos contestar. Sin embargo, en nuestro caso lo utilizamos porque parece que nos ayudaría a identificar con mayor facilidad los grupos.

3.1. Modelo descriptivo

Se realizaron diversas funciones para encontrar la combinación óptima dentro de un conjunto de modelos que consideraba términos simples, interacciones y términos cuadráticos. Para lograr esto se implemento una rutina que utiliza una búsqueda tabú con una función de optimización el AIC, BIC y Deviance de cada modelo. En dichas rutinas se consideró la que todas las variables involucradas fueran significativas respecto de la estadística F. El resultado de dicho análisis arrojó los modelos presentados en la tabla 14 En la tabla se muestran modelos solo de tres términos, y esto se debe

Variables en el modelo	AIC	BIC	Deviance
"Fglucose" , "Fglucose^2" , "GlucoseInt^2"	34.9	58.7	18.9
"Fglucose" , "GlucoseInt" , "Fglucose^2"	30.7	54.5	14.7
"Weight" , "Weight*InsulinResp" , "InsulinResp^2"	230.2	254.0	214.2

Tabla 14: Resumen de los posibles modelos descriptivos con los parámetros de su ajuste

principalmente a dos motivos: por una parte, como indíca el análisis del artículo, el número debería ser próximo a 3 términos; y por el otro lado, casi todos los modelos que optimizaron cada medida tenían ese número de términos (como se puede apreciar en la tabla 14 El mejor modelo, en términos descriptivos es el segundo, que tomaremos como el modelo descriptivo.

3.2. Modelo predictivo

Para calcular un modelo predictivo, para lo cual se utilizó la búsqueda tabú pero ahora la función objetivo que se buscó minimizar fue el mínimo del máximo de las tasas de error por grupos en errores no aparentes para una fracción de entrenamiento de 0.7. La tabla 15 resume tres posibles modelos, mientras que la tabla 17 muestra sus tasas de error no aparentes.

Nombre del modelo	Variables en el modelo
Modelo 1	"Fglucose" , "GlucoseInt*GlucoseInt"
Modelo 2	"InsulinResp", "Fglucose*InsulinResp" , "GlucoseInt*InsulinResp"
Modelo 3	"Fglucose" , "Weight*InsulineResist" , "Fglucose*GlucoseInt" , "GlucoseInt*GlucoseInt"

Tabla 15: Resumen de los posibles modelos predictivos con sus respectivas tasas de error no aparente utilizando K fold cross-validation

	Modelo 1	Modelo 2	Modelo 3
Mean global error, entrenamiento: 0.7	0.140	0.030	0.230
Mean 1 error, entrenamiento: 0.7	0.121	0.009	0.002
Mean 2 error, entrenamiento: 0.7	0.474	0.062	0.894
Mean 3 error, entrenamiento: 0.7	0.005	0.010	0.009

Tabla 17: Tasas de error no aparente locales y globales para los tres modelos con conjuntos divididos manteniendo proporcionalidad

El modelo que minimiza dichos valores, muy por encima de todos los otros contrincantes, es el modelo 2, por lo que lo seleccionamos como el modelo predictivo.

3.3. Conclusiones

La tabla 18 muestra las variables de los dos modelos trabajados

Modelo	Variables	Notas
Descriptivo	"Fglucose" , "GlucoseInt" , "Fglucose^2"	-
Predictivo	"InsulinResp", "Fglucose*InsulinResp" , "GlucoseInt*InsulinResp"	-

Tabla 18: Modelos descriptivo y predictivo

A continuación, mostramos las tasas de error aparente y no aparente para el modelo predictivo y descriptivo en la tablas

	Descriptivo	Predictivo
Mean global error, entrenamiento: 1.0	0.021	0.021
Mean 1 error, entrenamiento: 1.0	0.000	0.000
Mean 2 error, entrenamiento: 1.0	0.056	0.056
Mean 3 error, entrenamiento: 1.0	0.013	0.013

Tabla 20: Tasas de error aparente locales y globales para el modelo descriptivo y predictivo

	Descriptivo	Predictivo
Mean global error, entrenamiento: 0.7	0.023	0.023
Mean 1 error, entrenamiento: 0.7	0.000	0.002
Mean 2 error, entrenamiento: 0.7	0.056	0.057
Mean 3 error, entrenamiento: 0.7	0.022	0.011

Tabla 22: Tasas de error no aparente locales y globales para el modelo descriptivo y predictivo con conjuntos divididos mantieniendo proporcionalidad

Es claro que el modelo descriptivo y el modelo predictivo difieren. También, en el último no aparecen los términos individuales de las interacciones; no obstante, al agregar dichos términos individuales al modelo no eran significativos según la estadística F. Ambos modelos tienen un alto poder predictivo, mientras que la interpretación del descriptivo es mucho más sencilla.

Anexo 1: Tablas relevantes

Problema 1

	LDA	QDA	Naive Bayes	Logistic	KNC, k=5	SVM
Mean global error, entrenamiento: 0.1	0.241	0.278	0.250	0.241	0.258	0.243
std, entrenamiento: 0.1	0.017	0.027	0.020	0.017	0.027	0.036
Mean Yes error, entrenamiento: 0.1	0.406	0.475	0.352	0.426	0.515	0.537
std, entrenamiento: 0.1	0.089	0.089	0.088	0.098	0.127	0.180
Mean No error, entrenamiento: 0.1	0.144	0.157	0.190	0.139	0.118	0.094
std, entrenamiento: 0.1	0.042	0.054	0.057	0.038	0.046	0.049
Mean global error, entrenamiento: 0.2	0.229	0.266	0.247	0.234	0.250	0.240
std, entrenamiento: 0.2	0.014	0.017	0.021	0.014	0.015	0.034
Mean Yes error, entrenamiento: 0.2	0.390	0.423	0.342	0.405	0.429	0.584
std, entrenamiento: 0.2	0.044	0.099	0.038	0.049	0.048	0.151
Mean No error, entrenamiento: 0.2	0.136	0.158	0.187	0.135	0.143	0.070
std, entrenamiento: 0.2	0.021	0.044	0.038	0.024	0.028	0.038
Mean global error, entrenamiento: 0.3	0.219	0.251	0.234	0.228	0.251	0.232
std, entrenamiento: 0.3	0.014	0.014	0.014	0.013	0.014	0.019
Mean Yes error, entrenamiento: 0.3	0.429	0.394	0.357	0.412	0.408	0.523
std, entrenamiento: 0.3	0.041	0.034	0.034	0.049	0.040	0.074
Mean No error, entrenamiento: 0.3	0.110	0.145	0.164	0.123	0.143	0.077
std, entrenamiento: 0.3	0.019	0.014	0.025	0.019	0.022	0.033
Mean global error, entrenamiento: 0.4	0.233	0.242	0.241	0.227	0.265	0.224
std, entrenamiento: 0.4	0.020	0.014	0.013	0.018	0.014	0.019
Mean Yes error, entrenamiento: 0.4	0.412	0.394	0.347	0.406	0.459	0.521
std, entrenamiento: 0.4	0.032	0.047	0.026	0.022	0.054	0.046
Mean No error, entrenamiento: 0.4	0.125	0.140	0.175	0.122	0.119	0.075
std, entrenamiento: 0.4	0.015	0.024	0.017	0.014	0.019	0.018
Mean global error, entrenamiento: 0.5	0.215	0.240	0.247	0.227	0.250	0.220
std, entrenamiento: 0.5	0.016	0.020	0.018	0.015	0.018	0.017

Continued on next page

	LDA	QDA	Naive Bayes	Logistic	KNC, k=5	SVM
Mean Yes error, entrenamiento: 0.5	0.407	0.406	0.351	0.420	0.396	0.493
std, entrenamiento: 0.5	0.028	0.029	0.025	0.024	0.034	0.017
Mean No error, entrenamiento: 0.5	0.117	0.133	0.174	0.112	0.128	0.084
std, entrenamiento: 0.5	0.013	0.015	0.015	0.013	0.016	0.010
Mean global error, entrenamiento: 0.6	0.217	0.248	0.234	0.228	0.263	0.236
std, entrenamiento: 0.6	0.020	0.031	0.016	0.019	0.020	0.036
Mean Yes error, entrenamiento: 0.6	0.425	0.408	0.346	0.414	0.398	0.482
std, entrenamiento: 0.6	0.021	0.016	0.020	0.023	0.037	0.033
Mean No error, entrenamiento: 0.6	0.108	0.134	0.174	0.119	0.123	0.093
std, entrenamiento: 0.6	0.012	0.008	0.015	0.013	0.017	0.021
Mean global error, entrenamiento: 0.7	0.204	0.252	0.228	0.230	0.250	0.221
std, entrenamiento: 0.7	0.021	0.029	0.030	0.034	0.032	0.028
Mean Yes error, entrenamiento: 0.7	0.414	0.384	0.339	0.422	0.337	0.477
std, entrenamiento: 0.7	0.016	0.019	0.009	0.030	0.032	0.023
Mean No error, entrenamiento: 0.7	0.116	0.139	0.181	0.111	0.140	0.094
std, entrenamiento: 0.7	0.011	0.011	0.007	0.011	0.008	0.015
Mean global error, entrenamiento: 0.8	0.213	0.246	0.238	0.254	0.266	0.221
std, entrenamiento: 0.8	0.036	0.033	0.053	0.035	0.044	0.038
Mean Yes error, entrenamiento: 0.8	0.423	0.390	0.337	0.421	0.329	0.496
std, entrenamiento: 0.8	0.018	0.012	0.016	0.021	0.022	0.021
Mean No error, entrenamiento: 0.8	0.109	0.139	0.180	0.109	0.121	0.082
std, entrenamiento: 0.8	0.009	0.005	0.006	0.007	0.012	0.014
Mean global error, entrenamiento: 0.9	0.209	0.263	0.259	0.217	0.252	0.231
std, entrenamiento: 0.9	0.046	0.062	0.048	0.031	0.054	0.048
Mean Yes error, entrenamiento: 0.9	0.417	0.397	0.348	0.426	0.317	0.482
std, entrenamiento: 0.9	0.011	0.009	0.006	0.013	0.011	0.009
Mean No error, entrenamiento: 0.9	0.114	0.141	0.179	0.111	0.114	0.090
std, entrenamiento: 0.9	0.006	0.005	0.004	0.006	0.015	0.009

Continued on next page

	LDA	QDA	Naive Bayes	Logistic	KNC, k=5	SVM
Mean global error, entrenamiento: 1.0	0.212	0.224	0.239	0.212	0.171	0.216
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000	0.000	0.000
Mean Yes error, entrenamiento: 1.0	0.424	0.390	0.345	0.424	0.294	0.480
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000	0.000	0.000
Mean No error, entrenamiento: 1.0	0.107	0.141	0.186	0.107	0.110	0.085
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000	0.000	0.000

Tabla 24: Tasas de error locales y globales para varios clasificadores con conjuntos de entrenamiento divididos sin mantener proporcionalidad

	LDA	QDA	Naive Bayes	Logistic	KNC, k=5	SVM
Mean global error, entrenamiento: 0.1	0.247	0.287	0.252	0.235	0.258	0.247
std, entrenamiento: 0.1	0.025	0.024	0.014	0.015	0.020	0.032
Mean Yes error, entrenamiento: 0.1	0.455	0.461	0.377	0.429	0.449	0.600
std, entrenamiento: 0.1	0.055	0.104	0.046	0.082	0.058	0.164
Mean No error, entrenamiento: 0.1	0.130	0.175	0.183	0.130	0.149	0.067
std, entrenamiento: 0.1	0.033	0.051	0.032	0.041	0.029	0.037
Mean global error, entrenamiento: 0.2	0.229	0.266	0.251	0.233	0.252	0.230
std, entrenamiento: 0.2	0.011	0.021	0.017	0.013	0.018	0.007
Mean Yes error, entrenamiento: 0.2	0.428	0.437	0.363	0.426	0.425	0.521
std, entrenamiento: 0.2	0.028	0.073	0.045	0.047	0.044	0.065
Mean No error, entrenamiento: 0.2	0.120	0.147	0.175	0.126	0.148	0.081
std, entrenamiento: 0.2	0.015	0.031	0.032	0.025	0.030	0.026
Mean global error, entrenamiento: 0.3	0.227	0.242	0.241	0.231	0.265	0.229
std, entrenamiento: 0.3	0.020	0.010	0.013	0.012	0.014	0.011
Mean Yes error, entrenamiento: 0.3	0.382	0.407	0.350	0.410	0.420	0.528
std, entrenamiento: 0.3	0.025	0.042	0.026	0.038	0.033	0.036
Mean No error, entrenamiento: 0.3	0.135	0.138	0.175	0.123	0.154	0.073

Continued on next page

	LDA	QDA	Naive Bayes	Logistic	KNC, k=5	SVM
std, entrenamiento: 0.3	0.020	0.014	0.019	0.015	0.022	0.013
Mean global error, entrenamiento: 0.4	0.226	0.251	0.237	0.221	0.258	0.227
std, entrenamiento: 0.4	0.016	0.011	0.013	0.014	0.019	0.012
Mean Yes error, entrenamiento: 0.4	0.413	0.408	0.346	0.408	0.402	0.511
std, entrenamiento: 0.4	0.030	0.035	0.024	0.028	0.037	0.052
Mean No error, entrenamiento: 0.4	0.115	0.140	0.175	0.120	0.145	0.078
std, entrenamiento: 0.4	0.011	0.014	0.022	0.021	0.021	0.022
Mean global error, entrenamiento: 0.5	0.212	0.238	0.241	0.225	0.253	0.223
std, entrenamiento: 0.5	0.013	0.017	0.020	0.012	0.015	0.012
Mean Yes error, entrenamiento: 0.5	0.410	0.402	0.338	0.405	0.383	0.510
std, entrenamiento: 0.5	0.022	0.025	0.010	0.016	0.036	0.026
Mean No error, entrenamiento: 0.5	0.114	0.136	0.171	0.121	0.134	0.078
std, entrenamiento: 0.5	0.013	0.012	0.011	0.009	0.018	0.015
Mean global error, entrenamiento: 0.6	0.225	0.246	0.230	0.220	0.251	0.225
std, entrenamiento: 0.6	0.025	0.023	0.022	0.024	0.024	0.024
Mean Yes error, entrenamiento: 0.6	0.412	0.382	0.354	0.412	0.379	0.481
std, entrenamiento: 0.6	0.021	0.024	0.019	0.017	0.026	0.033
Mean No error, entrenamiento: 0.6	0.116	0.142	0.167	0.118	0.122	0.093
std, entrenamiento: 0.6	0.009	0.010	0.012	0.010	0.011	0.019
Mean global error, entrenamiento: 0.7	0.222	0.244	0.236	0.237	0.238	0.225
std, entrenamiento: 0.7	0.034	0.028	0.023	0.020	0.020	0.026
Mean Yes error, entrenamiento: 0.7	0.420	0.385	0.341	0.406	0.351	0.477
std, entrenamiento: 0.7	0.016	0.014	0.015	0.015	0.022	0.018
Mean No error, entrenamiento: 0.7	0.113	0.143	0.182	0.120	0.121	0.095
std, entrenamiento: 0.7	0.009	0.006	0.009	0.009	0.017	0.013
Mean global error, entrenamiento: 0.8	0.232	0.234	0.245	0.219	0.250	0.233
std, entrenamiento: 0.8	0.031	0.034	0.034	0.022	0.032	0.031
Mean Yes error, entrenamiento: 0.8	0.415	0.393	0.339	0.431	0.356	0.485
std, entrenamiento: 0.8	0.008	0.022	0.007	0.010	0.033	0.005

Continued on next page

	LDA	QDA	Naive Bayes	Logistic	KNC, k=5	SVM
Mean No error, entrenamiento: 0.8	0.116	0.136	0.179	0.110	0.116	0.089
std, entrenamiento: 0.8	0.008	0.005	0.009	0.008	0.014	0.006
Mean global error, entrenamiento: 0.9	0.215	0.239	0.228	0.237	0.259	0.198
std, entrenamiento: 0.9	0.051	0.046	0.054	0.053	0.057	0.039
Mean Yes error, entrenamiento: 0.9	0.419	0.390	0.339	0.422	0.315	0.488
std, entrenamiento: 0.9	0.008	0.009	0.007	0.011	0.013	0.008
Mean No error, entrenamiento: 0.9	0.110	0.139	0.181	0.114	0.110	0.083
std, entrenamiento: 0.9	0.006	0.003	0.004	0.006	0.006	0.005
Mean global error, entrenamiento: 1.0	0.212	0.224	0.239	0.212	0.171	0.216
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000	0.000	0.000
Mean Yes error, entrenamiento: 1.0	0.424	0.390	0.345	0.424	0.294	0.480
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000	0.000	0.000
Mean No error, entrenamiento: 1.0	0.107	0.141	0.186	0.107	0.110	0.085
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000	0.000	0.000

Tabla 26: Tasas de error locales y globales para varios clasificadores con conjuntos de entrenamiento divididos sin mantener proporcionalidad

	lambda	Lower confidence interval, alpha = 0.05	Upper confidence interval, alpha = 0.05
glu	-0.1487	-0.4522	0.1580
bp	1.0763	0.7520	1.4125
skin	0.5490	0.3709	0.7263
bmi	0.1249	-0.2044	0.4510
ped	-0.0090	-0.1269	0.1084
age	-1.4435	-1.7747	-1.1202

Tabla 27: Valores de λ que maximizan la verosimilitud de una transformación Box-Cox

Coefficient	Estimate	Std. Error	z value	Pr($p > \|z\|$)
(Intercept)	-0.97191	0.12103	-8.030	9.72e-16
glu	1.06690	0.13040	8.182	2.80e-16
bp	-0.10413	0.12553	-0.830	0.406802
skin	0.09456	0.15746	0.601	0.548172
bmi	0.53147	0.15900	3.342	0.000830
ped	0.42914	0.12360	3.472	0.000516
age	0.54557	0.12267	4.448	8.68e-06

Tabla 28: Coeficientes y p-values para la regresión logística del modelo completo

	LDA	Naive Bayes	Logistic	SVM, kernel = poly, deg = 5
Mean global error, entrenamiento: 0.1	0.231	0.249	0.219	0.253
std, entrenamiento: 0.1	0.018	0.031	0.015	0.023
Mean Yes error, entrenamiento: 0.1	0.389	0.434	0.392	0.529
std, entrenamiento: 0.1	0.071	0.047	0.075	0.123
Mean No error, entrenamiento: 0.1	0.143	0.145	0.128	0.098
std, entrenamiento: 0.1	0.029	0.052	0.035	0.039
Mean global error, entrenamiento: 0.2	0.224	0.236	0.216	0.245
std, entrenamiento: 0.2	0.010	0.015	0.011	0.012
Mean Yes error, entrenamiento: 0.2	0.392	0.423	0.392	0.533
std, entrenamiento: 0.2	0.028	0.042	0.044	0.083
Mean No error, entrenamiento: 0.2	0.128	0.128	0.121	0.085
std, entrenamiento: 0.2	0.027	0.021	0.021	0.029
Mean global error, entrenamiento: 0.3	0.213	0.232	0.207	0.237
std, entrenamiento: 0.3	0.008	0.010	0.010	0.018
Mean Yes error, entrenamiento: 0.3	0.410	0.407	0.395	0.519
std, entrenamiento: 0.3	0.028	0.023	0.027	0.042
Mean No error, entrenamiento: 0.3	0.110	0.121	0.111	0.077
std, entrenamiento: 0.3	0.014	0.018	0.019	0.018
Mean global error, entrenamiento: 0.4	0.216	0.219	0.211	0.243
std, entrenamiento: 0.4	0.009	0.012	0.021	0.014

Continued on next page

	LDA	Naive Bayes	Logistic	SVM, kernel = poly, deg = 5
Mean Yes error, entrenamiento: 0.4	0.404	0.401	0.387	0.472
std, entrenamiento: 0.4	0.025	0.018	0.021	0.031
Mean No error, entrenamiento: 0.4	0.115	0.119	0.113	0.087
std, entrenamiento: 0.4	0.016	0.011	0.012	0.013
Mean global error, entrenamiento: 0.5	0.215	0.202	0.213	0.229
std, entrenamiento: 0.5	0.020	0.020	0.014	0.010
Mean Yes error, entrenamiento: 0.5	0.413	0.406	0.393	0.490
std, entrenamiento: 0.5	0.023	0.019	0.022	0.024
Mean No error, entrenamiento: 0.5	0.103	0.110	0.113	0.075
std, entrenamiento: 0.5	0.012	0.012	0.011	0.009
Mean global error, entrenamiento: 0.6	0.214	0.223	0.196	0.224
std, entrenamiento: 0.6	0.023	0.032	0.024	0.019
Mean Yes error, entrenamiento: 0.6	0.407	0.415	0.385	0.484
std, entrenamiento: 0.6	0.015	0.016	0.015	0.020
Mean No error, entrenamiento: 0.6	0.111	0.109	0.111	0.077
std, entrenamiento: 0.6	0.008	0.016	0.006	0.010
Mean global error, entrenamiento: 0.7	0.216	0.210	0.207	0.223
std, entrenamiento: 0.7	0.018	0.021	0.023	0.032
Mean Yes error, entrenamiento: 0.7	0.412	0.413	0.381	0.496
std, entrenamiento: 0.7	0.018	0.015	0.009	0.024
Mean No error, entrenamiento: 0.7	0.106	0.104	0.117	0.071
std, entrenamiento: 0.7	0.014	0.010	0.008	0.009
Mean global error, entrenamiento: 0.8	0.190	0.218	0.207	0.228
std, entrenamiento: 0.8	0.034	0.025	0.035	0.043
Mean Yes error, entrenamiento: 0.8	0.410	0.415	0.388	0.484
std, entrenamiento: 0.8	0.013	0.014	0.010	0.016
Mean No error, entrenamiento: 0.8	0.100	0.107	0.114	0.072

Continued on next page

	LDA	Naive Bayes	Logistic	SVM, kernel = poly, deg = 5
std, entrenamiento: 0.8	0.008	0.006	0.008	0.004
Mean global error, entrenamiento: 0.9	0.224	0.226	0.209	0.222
std, entrenamiento: 0.9	0.044	0.055	0.036	0.043
Mean Yes error, entrenamiento: 0.9	0.418	0.406	0.383	0.481
std, entrenamiento: 0.9	0.007	0.011	0.009	0.009
Mean No error, entrenamiento: 0.9	0.104	0.106	0.114	0.071
std, entrenamiento: 0.9	0.008	0.006	0.005	0.004
Mean global error, entrenamiento: 1.0	0.205	0.205	0.207	0.205
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000
Mean Yes error, entrenamiento: 1.0	0.412	0.412	0.390	0.475
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000
Mean No error, entrenamiento: 1.0	0.101	0.101	0.115	0.070
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000

Tabla 30: Tasas de error locales y globales para varios clasificadores con conjuntos de entrenamiento divididos sin mantener proporcionalidad

Problema 2

	LDA	QDA	Naive Bayes	Logistic	KNC, k=5	SVM
Mean global error, entrenamiento: 0.1	0.258	0.378	0.295	0.214	0.187	0.215
std, entrenamiento: 0.1	0.054	0.079	0.081	0.033	0.032	0.040
Mean Yes error, entrenamiento: 0.1	0.293	0.307	0.207	0.262	0.214	0.246
std, entrenamiento: 0.1	0.045	0.273	0.080	0.062	0.049	0.092
Mean No error, entrenamiento: 0.1	0.193	0.381	0.360	0.149	0.150	0.155
std, entrenamiento: 0.1	0.081	0.232	0.193	0.053	0.050	0.068
Mean global error, entrenamiento: 0.2	0.207	0.320	0.252	0.176	0.171	0.180
std, entrenamiento: 0.2	0.028	0.111	0.103	0.033	0.016	0.029

Continued on next page

	LDA	QDA	Naive Bayes	Logistic	KNC, k=5	SVM
Mean Yes error, entrenamiento: 0.2	0.229	0.252	0.172	0.193	0.174	0.177
std, entrenamiento: 0.2	0.068	0.250	0.047	0.041	0.037	0.051
Mean No error, entrenamiento: 0.2	0.150	0.335	0.294	0.126	0.154	0.136
std, entrenamiento: 0.2	0.035	0.276	0.187	0.038	0.026	0.062
Mean global error, entrenamiento: 0.3	0.165	0.245	0.260	0.169	0.181	0.149
std, entrenamiento: 0.3	0.030	0.090	0.122	0.020	0.022	0.011
Mean Yes error, entrenamiento: 0.3	0.182	0.179	0.199	0.187	0.192	0.134
std, entrenamiento: 0.3	0.027	0.087	0.120	0.029	0.047	0.028
Mean No error, entrenamiento: 0.3	0.113	0.211	0.284	0.117	0.130	0.116
std, entrenamiento: 0.3	0.024	0.224	0.245	0.036	0.023	0.028
Mean global error, entrenamiento: 0.4	0.168	0.224	0.263	0.162	0.159	0.150
std, entrenamiento: 0.4	0.033	0.086	0.117	0.025	0.019	0.015
Mean Yes error, entrenamiento: 0.4	0.189	0.139	0.179	0.177	0.167	0.121
std, entrenamiento: 0.4	0.034	0.028	0.070	0.033	0.029	0.023
Mean No error, entrenamiento: 0.4	0.108	0.218	0.291	0.105	0.120	0.116
std, entrenamiento: 0.4	0.028	0.179	0.260	0.037	0.037	0.022
Mean global error, entrenamiento: 0.5	0.160	0.214	0.196	0.167	0.152	0.138
std, entrenamiento: 0.5	0.017	0.053	0.036	0.028	0.026	0.035
Mean Yes error, entrenamiento: 0.5	0.153	0.166	0.195	0.178	0.158	0.134
std, entrenamiento: 0.5	0.030	0.028	0.037	0.034	0.032	0.035
Mean No error, entrenamiento: 0.5	0.116	0.146	0.167	0.109	0.110	0.090
std, entrenamiento: 0.5	0.031	0.111	0.017	0.026	0.029	0.022
Mean global error, entrenamiento: 0.6	0.158	0.173	0.182	0.162	0.155	0.156
std, entrenamiento: 0.6	0.032	0.034	0.029	0.039	0.019	0.033
Mean Yes error, entrenamiento: 0.6	0.175	0.118	0.164	0.164	0.166	0.130
std, entrenamiento: 0.6	0.016	0.032	0.017	0.024	0.027	0.032
Mean No error, entrenamiento: 0.6	0.101	0.125	0.169	0.109	0.091	0.093
std, entrenamiento: 0.6	0.027	0.021	0.020	0.023	0.025	0.025

Continued on next page

	LDA	QDA	Naive Bayes	Logistic	KNC, k=5	SVM
Mean global error, entrenamiento: 0.7	0.155	0.158	0.168	0.168	0.149	0.151
std, entrenamiento: 0.7	0.027	0.040	0.040	0.033	0.042	0.028
Mean Yes error, entrenamiento: 0.7	0.159	0.094	0.166	0.154	0.166	0.106
std, entrenamiento: 0.7	0.017	0.025	0.017	0.018	0.039	0.020
Mean No error, entrenamiento: 0.7	0.095	0.121	0.164	0.116	0.086	0.100
std, entrenamiento: 0.7	0.021	0.017	0.016	0.022	0.020	0.021
Mean global error, entrenamiento: 0.8	0.127	0.196	0.204	0.144	0.173	0.156
std, entrenamiento: 0.8	0.040	0.045	0.053	0.064	0.032	0.038
Mean Yes error, entrenamiento: 0.8	0.152	0.114	0.165	0.160	0.174	0.129
std, entrenamiento: 0.8	0.009	0.016	0.015	0.018	0.019	0.025
Mean No error, entrenamiento: 0.8	0.096	0.120	0.163	0.091	0.089	0.074
std, entrenamiento: 0.8	0.013	0.014	0.021	0.016	0.007	0.016
Mean global error, entrenamiento: 0.9	0.158	0.204	0.196	0.171	0.113	0.125
std, entrenamiento: 0.9	0.069	0.057	0.046	0.088	0.079	0.059
Mean Yes error, entrenamiento: 0.9	0.150	0.114	0.167	0.150	0.161	0.111
std, entrenamiento: 0.9	0.012	0.013	0.011	0.014	0.020	0.023
Mean No error, entrenamiento: 0.9	0.098	0.119	0.155	0.119	0.078	0.078
std, entrenamiento: 0.9	0.014	0.016	0.011	0.013	0.014	0.020
Mean global error, entrenamiento: 1.0	0.114	0.097	0.153	0.127	0.123	0.089
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000	0.000	0.000
Mean Yes error, entrenamiento: 1.0	0.140	0.093	0.150	0.150	0.178	0.112
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000	0.000	0.000
Mean No error, entrenamiento: 1.0	0.093	0.101	0.155	0.109	0.078	0.070
std, entrenamiento: 1.0	0.000	0.000	0.000	0.000	0.000	0.000

Tabla 32: Tasas de error locales y globales para varios clasificadores con conjuntos de entrenamiento divididos sin mantener proporcionalidad

Coefficient	Estimate	Std. Error	z value	Pr(¿—z—)
(Intercept)	-1.9907	1.3717	-1.451	0.146690
SexMale	0.4269	0.5242	0.814	0.415394
AngPecNone	-0.9167	0.7861	-1.166	0.243542
AngPecTypical	2.0258	0.6704	3.022	0.002514
AMINotCertain	-1.8638	0.5475	-3.404	0.000663
QWaveYes	1.7011	0.5516	3.084	0.002043
QWavecodeUsable	0.8020	1.0821	0.741	0.458595
STcodeUsable	-1.9136	0.7652	-2.501	0.012394
STchangeYes	2.4963	0.6071	4.112	3.92e-05
SuffHeartFYes	0.3750	0.5581	0.672	0.501609
HypertrophiYes	-0.7961	0.5992	-1.329	0.183919
HypercholYes	1.2769	0.4521	2.825	0.004733
SmokerYes	0.3492	0.5504	0.634	0.525844
InheritYes	0.5909	0.4846	1.219	0.222732
HeartfailYes	-0.8047	0.6662	-1.208	0.227045

Tabla 33: Coeficientes y p-values para la regresión logística del modelo completo

	Naive Bayes	Logistic	SVM, kernel = poly, deg = 5
Mean global error, entrenamiento: 0.1	0.281	0.204	0.200
std, entrenamiento: 0.1	0.095	0.041	0.028
Mean Yes error, entrenamiento: 0.1	0.285	0.277	0.272
std, entrenamiento: 0.1	0.173	0.125	0.062
Mean No error, entrenamiento: 0.1	0.253	0.135	0.105
std, entrenamiento: 0.1	0.217	0.047	0.037
Mean global error, entrenamiento: 0.2	0.332	0.192	0.201
std, entrenamiento: 0.2	0.104	0.032	0.031
Mean Yes error, entrenamiento: 0.2	0.179	0.201	0.254
std, entrenamiento: 0.2	0.123	0.050	0.052
Mean No error, entrenamiento: 0.2	0.429	0.160	0.086
std, entrenamiento: 0.2	0.255	0.048	0.026
Mean global error, entrenamiento: 0.3	0.211	0.192	0.195
std, entrenamiento: 0.3	0.078	0.033	0.027

Continued on next page

	Naive Bayes	Logistic	SVM, kernel = poly, deg = 5
Mean Yes error, entrenamiento: 0.3	0.246	0.229	0.198
std, entrenamiento: 0.3	0.199	0.061	0.061
Mean No error, entrenamiento: 0.3	0.153	0.159	0.093
std, entrenamiento: 0.3	0.035	0.042	0.037
Mean global error, entrenamiento: 0.4	0.292	0.182	0.189
std, entrenamiento: 0.4	0.131	0.039	0.025
Mean Yes error, entrenamiento: 0.4	0.137	0.164	0.173
std, entrenamiento: 0.4	0.078	0.044	0.040
Mean No error, entrenamiento: 0.4	0.378	0.179	0.083
std, entrenamiento: 0.4	0.291	0.019	0.021
Mean global error, entrenamiento: 0.5	0.217	0.190	0.171
std, entrenamiento: 0.5	0.078	0.033	0.039
Mean Yes error, entrenamiento: 0.5	0.174	0.176	0.121
std, entrenamiento: 0.5	0.051	0.038	0.030
Mean No error, entrenamiento: 0.5	0.210	0.158	0.078
std, entrenamiento: 0.5	0.185	0.030	0.024
Mean global error, entrenamiento: 0.6	0.165	0.169	0.183
std, entrenamiento: 0.6	0.035	0.039	0.029
Mean Yes error, entrenamiento: 0.6	0.153	0.199	0.116
std, entrenamiento: 0.6	0.017	0.045	0.031
Mean No error, entrenamiento: 0.6	0.156	0.157	0.070
std, entrenamiento: 0.6	0.021	0.025	0.013
Mean global error, entrenamiento: 0.7	0.170	0.201	0.200
std, entrenamiento: 0.7	0.042	0.037	0.039
Mean Yes error, entrenamiento: 0.7	0.171	0.196	0.104
std, entrenamiento: 0.7	0.016	0.030	0.031
Mean No error, entrenamiento: 0.7	0.154	0.153	0.060

Continued on next page

	Naive Bayes	Logistic	SVM, kernel = poly, deg = 5
std, entrenamiento: 0.7	0.022	0.031	0.027
Mean global error, entrenamiento: 0.8	0.169	0.188	0.204
std, entrenamiento: 0.8	0.065	0.055	0.057
Mean Yes error, entrenamiento: 0.8	0.169	0.168	0.082
std, entrenamiento: 0.8	0.015	0.044	0.026
Mean No error, entrenamiento: 0.8	0.145	0.155	0.054
std, entrenamiento: 0.8	0.018	0.024	0.010
Mean global error, entrenamiento: 0.9	0.142	0.175	0.162
std, entrenamiento: 0.9	0.068	0.064	0.044
Mean Yes error, entrenamiento: 0.9	0.160	0.197	0.058
std, entrenamiento: 0.9	0.012	0.045	0.017
Mean No error, entrenamiento: 0.9	0.142	0.136	0.039
std, entrenamiento: 0.9	0.005	0.029	0.010
Mean global error, entrenamiento: 1.0	0.144	0.169	0.038
std, entrenamiento: 1.0	0.000	0.000	0.000
Mean Yes error, entrenamiento: 1.0	0.150	0.243	0.037
std, entrenamiento: 1.0	0.000	0.000	0.000
Mean No error, entrenamiento: 1.0	0.140	0.109	0.039
std, entrenamiento: 1.0	0.000	0.000	0.000

Tabla 35: Tasas de error locales y globales para varios clasificadores con conjuntos de entrenamiento divididos sin mantener proporcionalidad

Problema 3

	One-versus-others	multinomial
Mean global error, entrenamiento: 0.1	0.160	0.112

Continued on next page

	One-versus-others	multinomial
std, entrenamiento: 0.1	0.019	0.029
Mean 1 error, entrenamiento: 0.1	0.141	0.212
std, entrenamiento: 0.1	0.052	0.065
Mean 2 error, entrenamiento: 0.1	0.370	0.093
std, entrenamiento: 0.1	0.092	0.112
Mean 3 error, entrenamiento: 0.1	0.039	0.057
std, entrenamiento: 0.1	0.032	0.022
Mean global error, entrenamiento: 0.2	0.135	0.129
std, entrenamiento: 0.2	0.011	0.012
Mean 1 error, entrenamiento: 0.2	0.071	0.101
std, entrenamiento: 0.2	0.038	0.062
Mean 2 error, entrenamiento: 0.2	0.287	0.278
std, entrenamiento: 0.2	0.047	0.079
Mean 3 error, entrenamiento: 0.2	0.039	0.022
std, entrenamiento: 0.2	0.019	0.006
Mean global error, entrenamiento: 0.3	0.046	0.056
std, entrenamiento: 0.3	0.009	0.024
Mean 1 error, entrenamiento: 0.3	0.071	0.071
std, entrenamiento: 0.3	0.038	0.014
Mean 2 error, entrenamiento: 0.3	0.037	0.056
std, entrenamiento: 0.3	0.026	0.039
Mean 3 error, entrenamiento: 0.3	0.013	0.018
std, entrenamiento: 0.3	0.011	0.016
Mean global error, entrenamiento: 0.4	0.065	0.042
std, entrenamiento: 0.4	0.005	0.024
Mean 1 error, entrenamiento: 0.4	0.071	0.030
std, entrenamiento: 0.4	0.029	0.043
Mean 2 error, entrenamiento: 0.4	0.028	0.028

Continued on next page

	One-versus-others	multinomial
std, entrenamiento: 0.4	0.023	0.023
Mean 3 error, entrenamiento: 0.4	0.031	0.022
std, entrenamiento: 0.4	0.012	0.022
Mean global error, entrenamiento: 0.5	0.055	0.050
std, entrenamiento: 0.5	0.019	0.036
Mean 1 error, entrenamiento: 0.5	0.081	0.061
std, entrenamiento: 0.5	0.029	0.049
Mean 2 error, entrenamiento: 0.5	0.019	0.019
std, entrenamiento: 0.5	0.013	0.026
Mean 3 error, entrenamiento: 0.5	0.009	0.013
std, entrenamiento: 0.5	0.012	0.011
Mean global error, entrenamiento: 0.6	0.092	0.057
std, entrenamiento: 0.6	0.043	0.035
Mean 1 error, entrenamiento: 0.6	0.061	0.040
std, entrenamiento: 0.6	0.049	0.014
Mean 2 error, entrenamiento: 0.6	0.056	0.009
std, entrenamiento: 0.6	0.060	0.013
Mean 3 error, entrenamiento: 0.6	0.018	0.022
std, entrenamiento: 0.6	0.016	0.016
Mean global error, entrenamiento: 0.7	0.076	0.008
std, entrenamiento: 0.7	0.060	0.011
Mean 1 error, entrenamiento: 0.7	0.020	0.000
std, entrenamiento: 0.7	0.014	0.000
Mean 2 error, entrenamiento: 0.7	0.065	0.000
std, entrenamiento: 0.7	0.035	0.000
Mean 3 error, entrenamiento: 0.7	0.022	0.004
std, entrenamiento: 0.7	0.022	0.006
Mean global error, entrenamiento: 0.8	0.057	0.046

Continued on next page

	One-versus-others	multinomial
std, entrenamiento: 0.8	0.016	0.016
Mean 1 error, entrenamiento: 0.8	0.020	0.000
std, entrenamiento: 0.8	0.029	0.000
Mean 2 error, entrenamiento: 0.8	0.046	0.000
std, entrenamiento: 0.8	0.035	0.000
Mean 3 error, entrenamiento: 0.8	0.018	0.018
std, entrenamiento: 0.8	0.012	0.006
Mean global error, entrenamiento: 0.9	0.089	0.022
std, entrenamiento: 0.9	0.031	0.031
Mean 1 error, entrenamiento: 0.9	0.010	0.000
std, entrenamiento: 0.9	0.014	0.000
Mean 2 error, entrenamiento: 0.9	0.102	0.000
std, entrenamiento: 0.9	0.073	0.000
Mean 3 error, entrenamiento: 0.9	0.018	0.004
std, entrenamiento: 0.9	0.012	0.006
Mean global error, entrenamiento: 1.0	0.034	0.000
std, entrenamiento: 1.0	0.000	0.000
Mean 1 error, entrenamiento: 1.0	0.000	0.000
std, entrenamiento: 1.0	0.000	0.000
Mean 2 error, entrenamiento: 1.0	0.083	0.000
std, entrenamiento: 1.0	0.000	0.000
Mean 3 error, entrenamiento: 1.0	0.026	0.000
std, entrenamiento: 1.0	0.000	0.000

Tabla 37: Tasas de error locales y globales para varios clasificadores con conjuntos de entrenamiento divididos sin mantener proporcionalidad

Anexo 2: Figuras relevantes

Problema 1

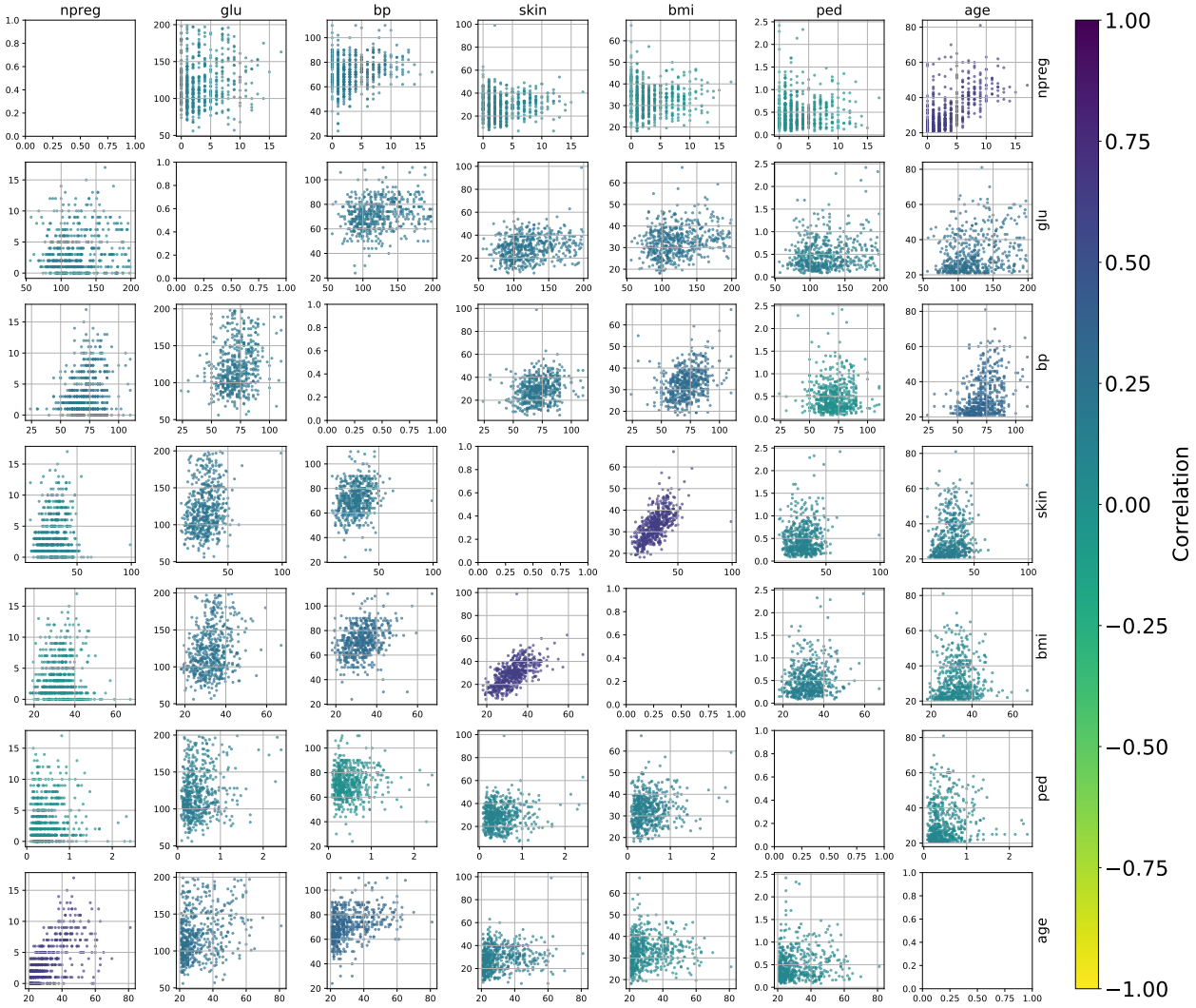

Figura 1: Gráfica de correlación para las variables numericas del problema 1

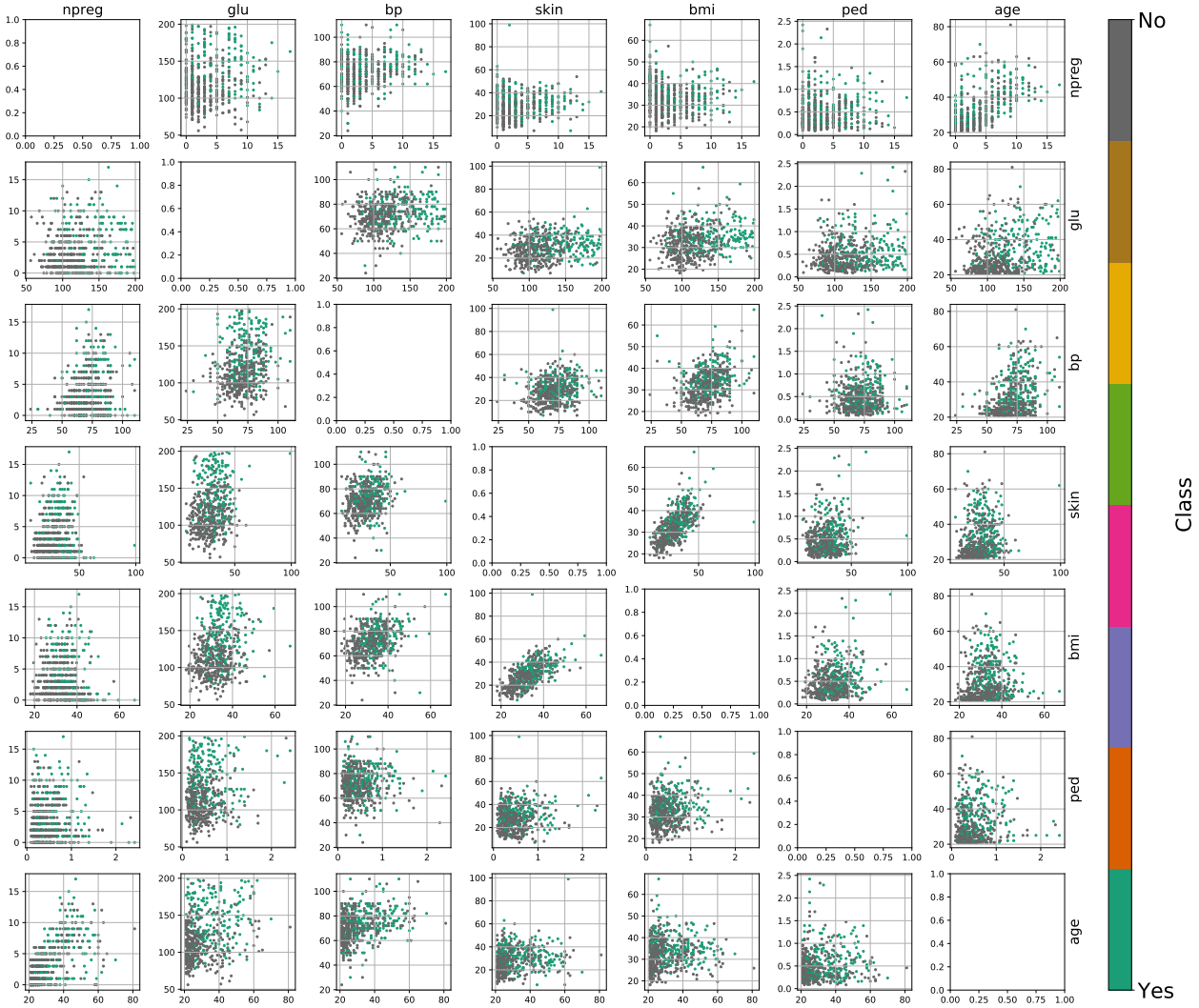

Figura 2: Gráficas de una variable contra otra divididas en clases para las variables numéricas del problema 1

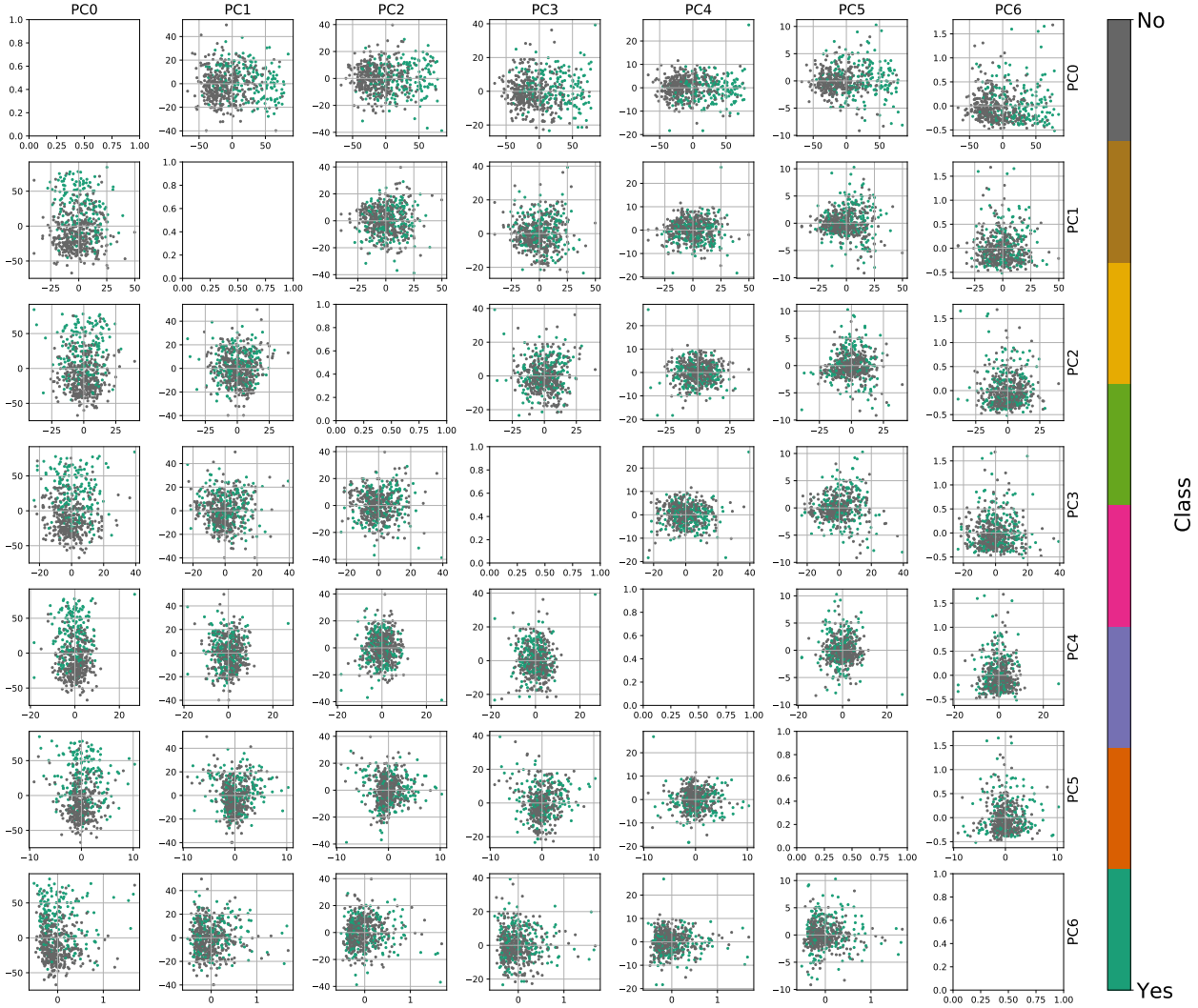

Figura 3: Gráficas de una variable contra otra divididas en clases para las componentes principales del problema 1

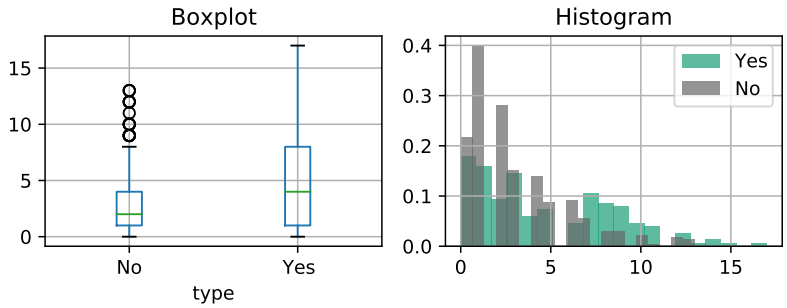

Figura 4: Distribución de la variable “npreg”

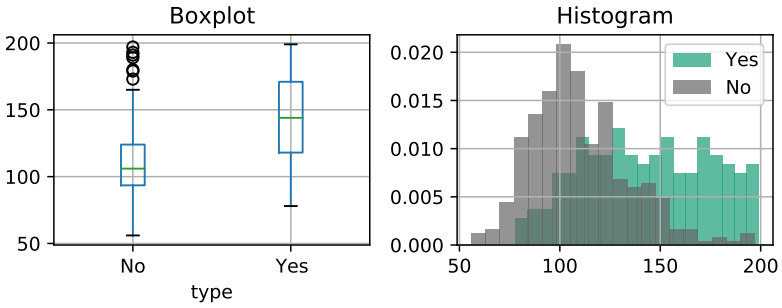

Figura 5: Distribución de la variable “glu”

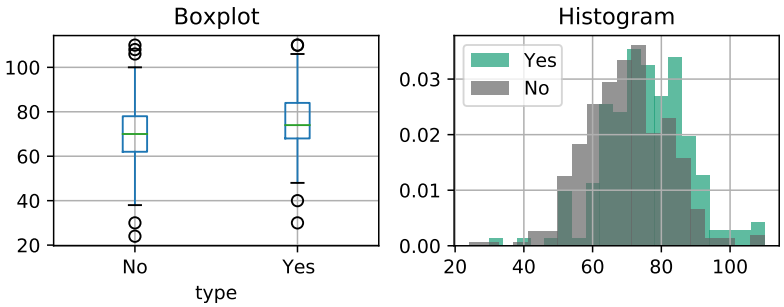

Figura 6: Distribución de la variable “bp”

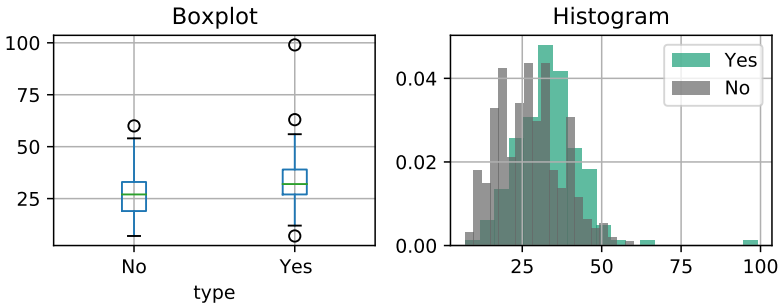

Figura 7: Distribución de la variable “skin”

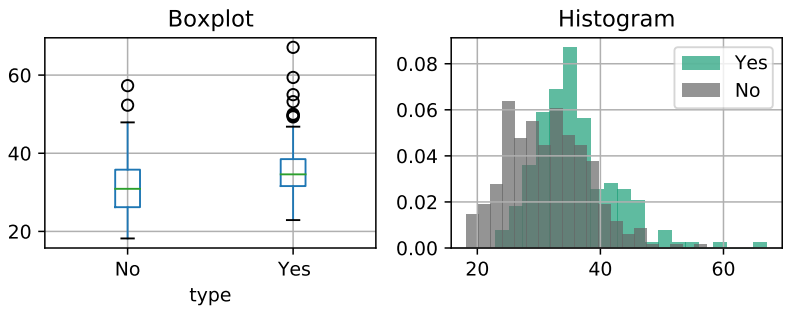

Figura 8: Distribución de la variable “bmi”

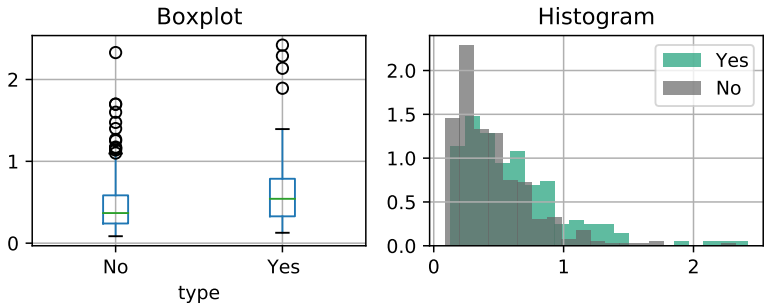

Figura 9: Distribución de la variable “ped”

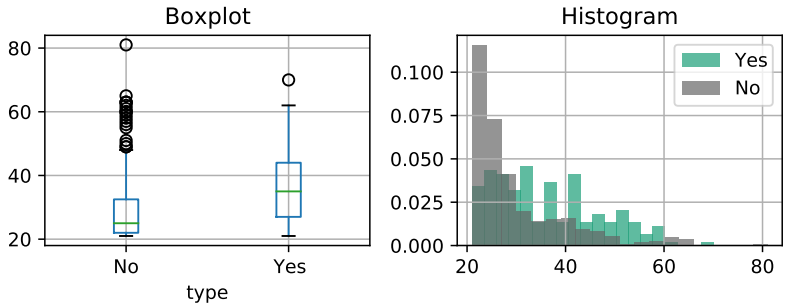

Figura 10: Distribución de la variable “age”

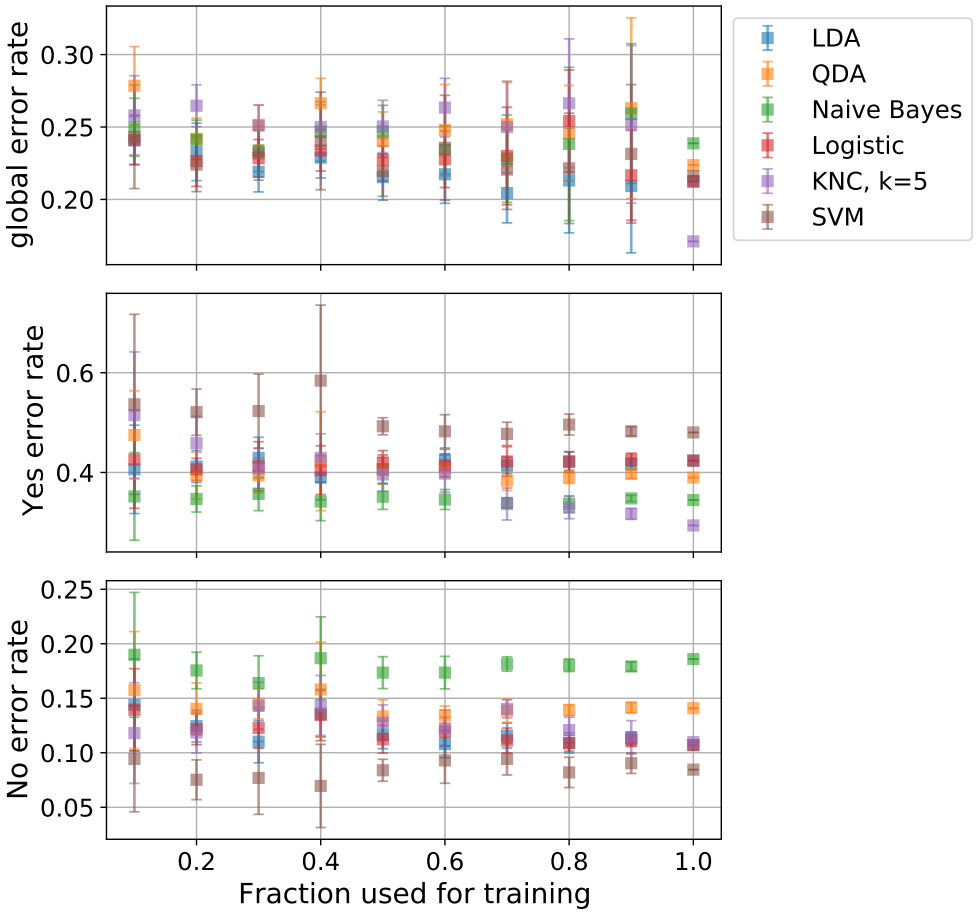

Figura 11: Tasas globales y locales para varios modelos, con conjuntos de entrenamiento divididos sin mantener proporcionalidad

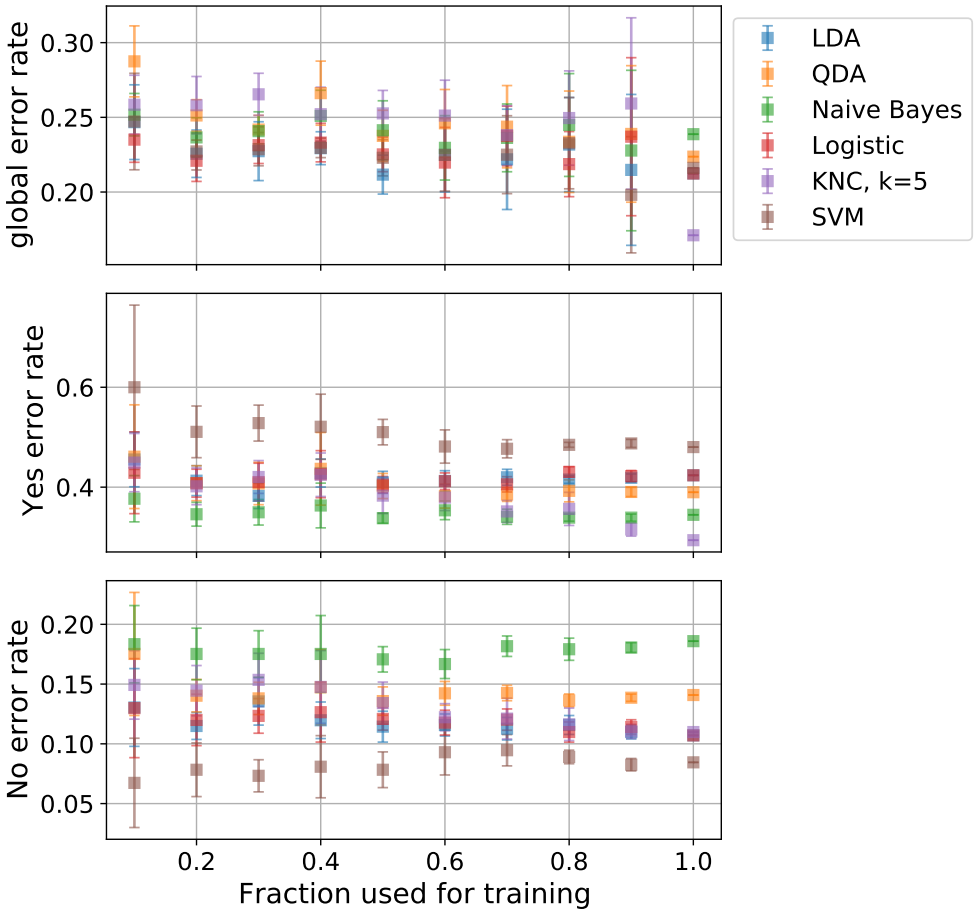

Figura 12: Tasas globales y locales para varios modelos, con conjuntos de entrenamiento divididos manteniendo proporcionalidad

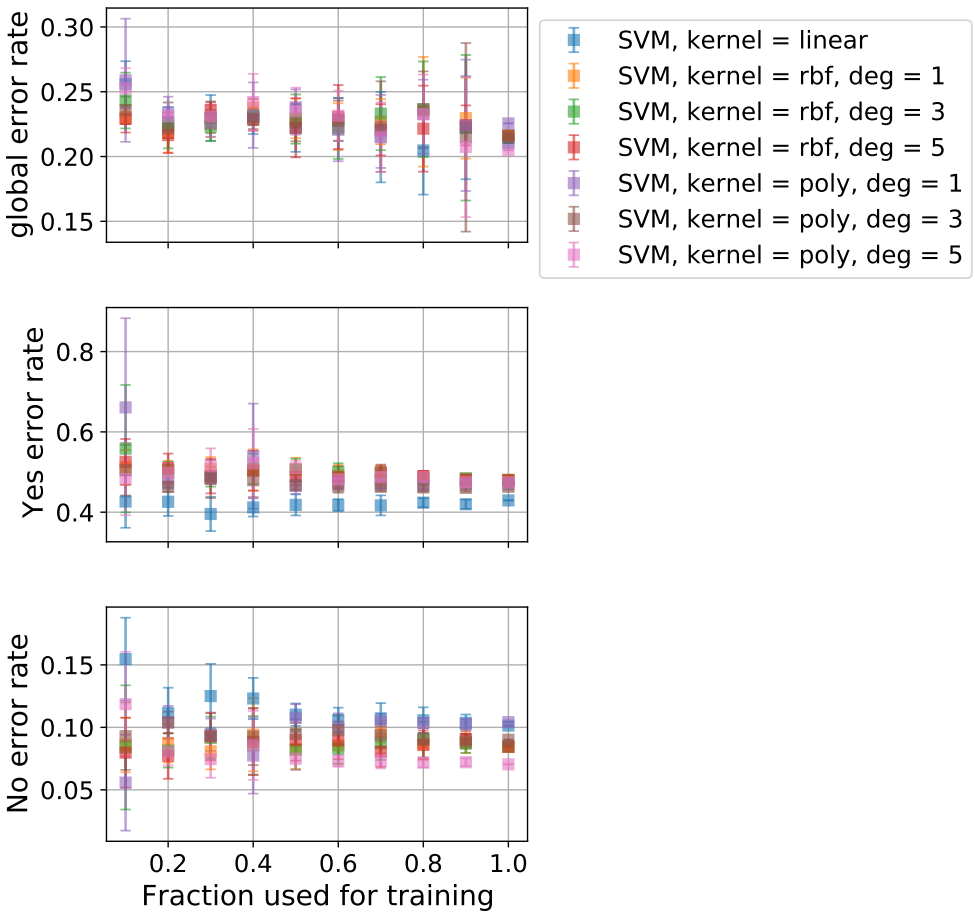

Figura 13: Tasas globales y locales para modelos SVM, con conjuntos de entrenamiento divididos manteniendo proporcionalidad

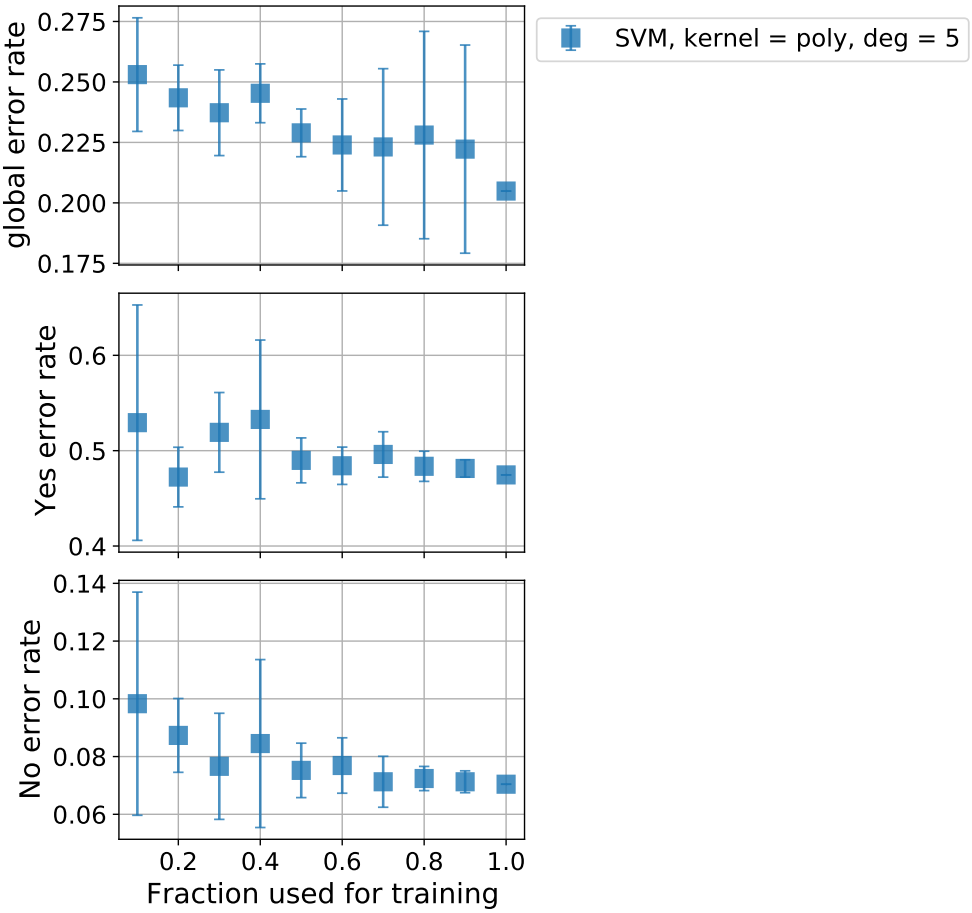

Figura 14: Tasas globales y locales para modelos SVM, con conjuntos de entrenamiento divididos manteniendo proporcionalidad

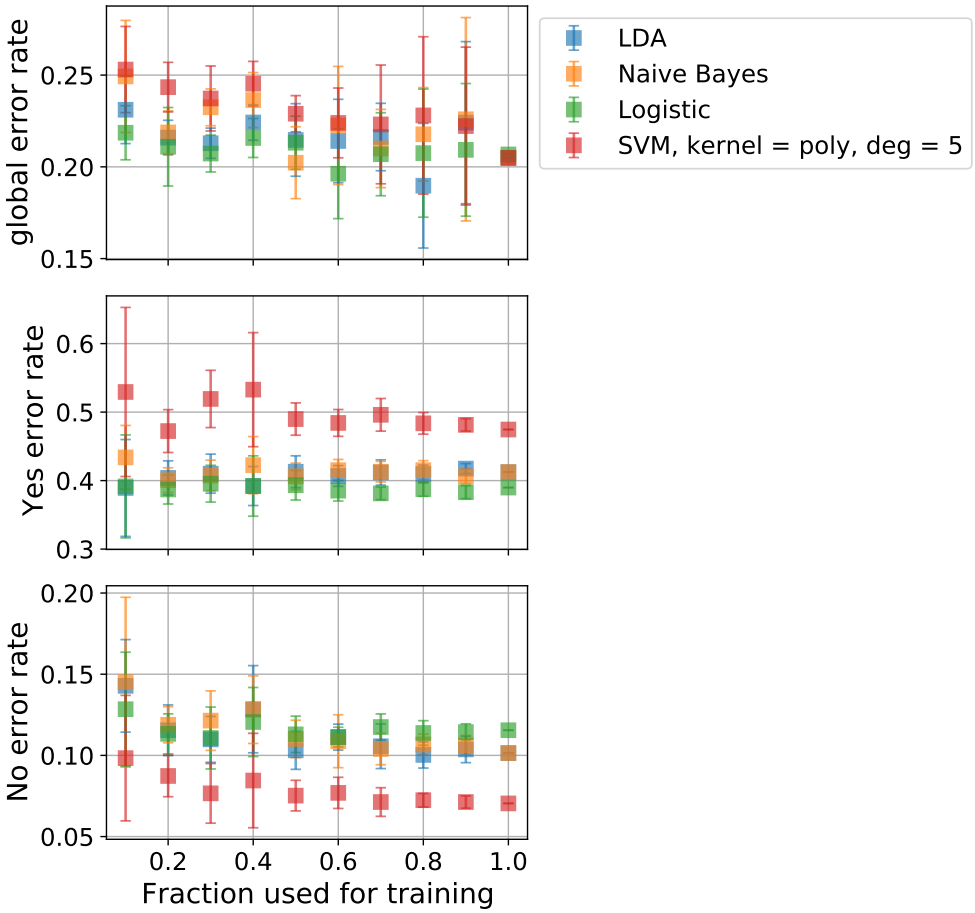

Figura 15: Tasas globales y locales para modelos SVM, con conjuntos de entrenamiento divididos manteniendo proporcionalidad

Problema 2

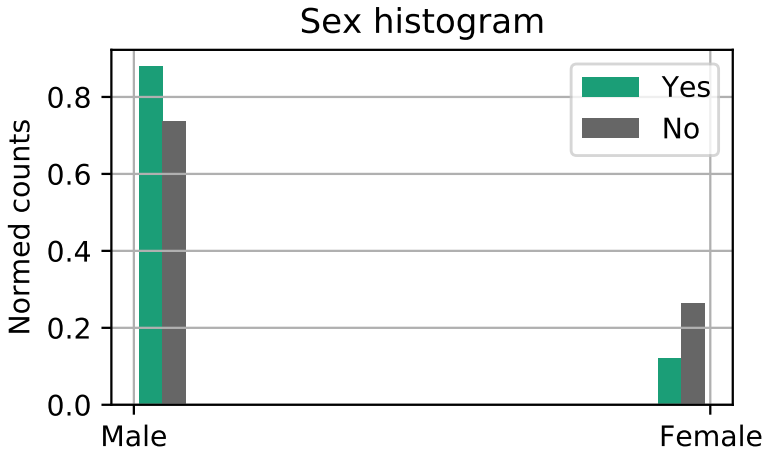

Figura 16: Distribución de la variable "Sex"

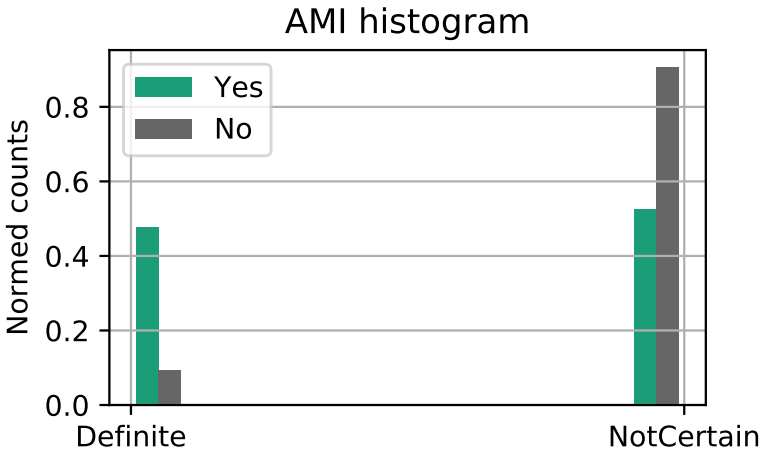

Figura 17: Distribución de la variable "AngPec"

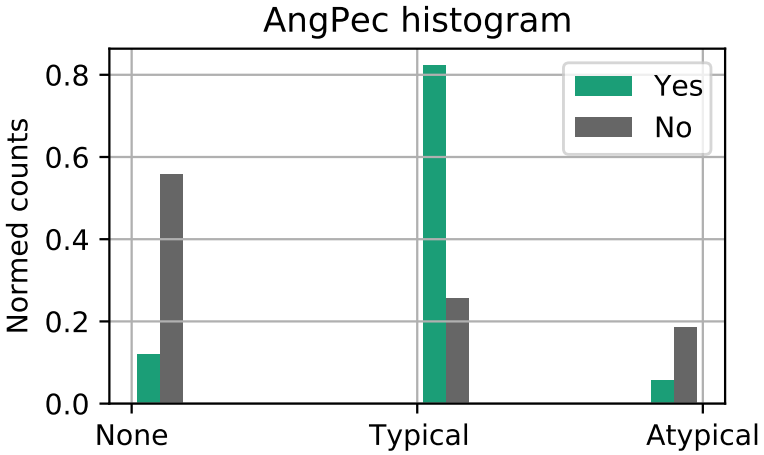

Figura 18: Distribución de la variable "AngPec"

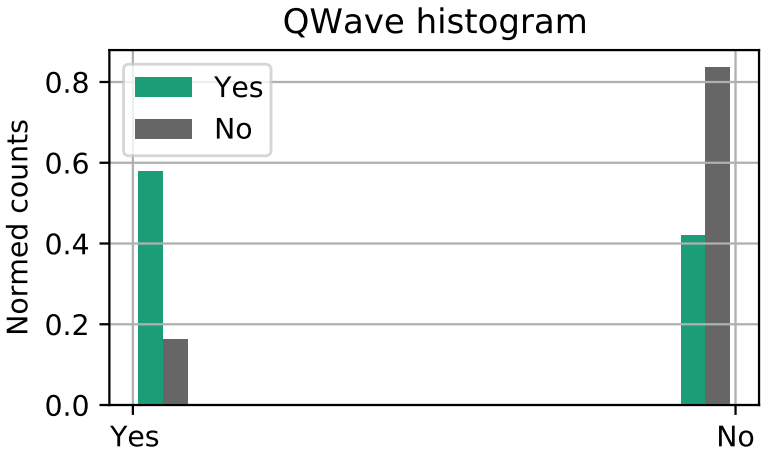

Figura 19: Distribución de la variable "QWave"

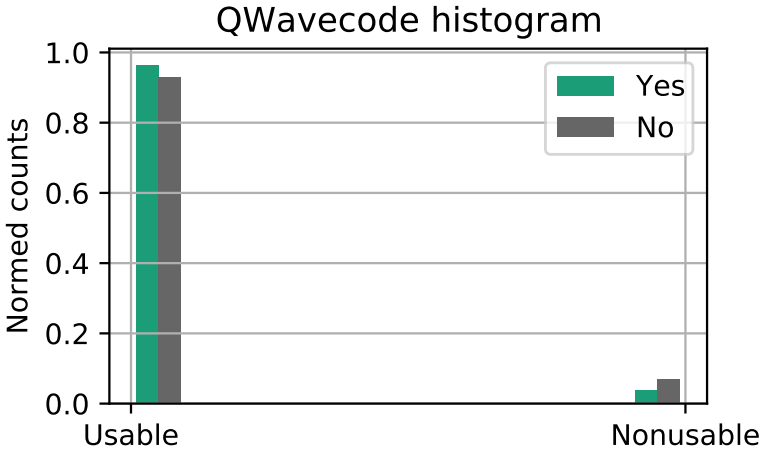

Figura 20: Distribución de la variable "QWavecode"

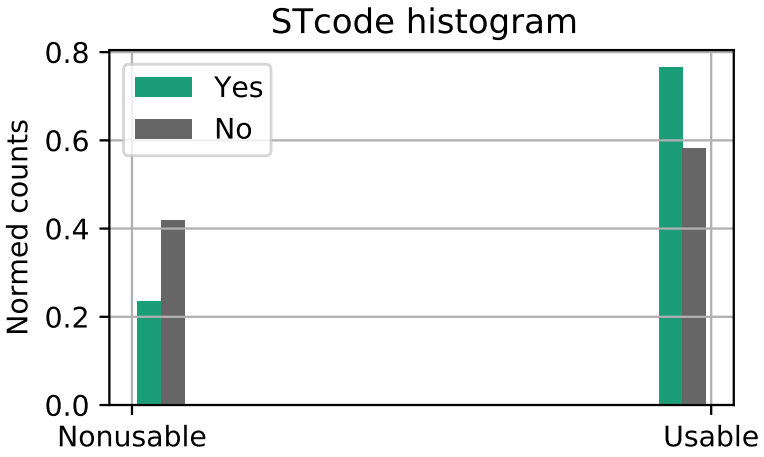

Figura 21: Distribución de la variable "STcode"

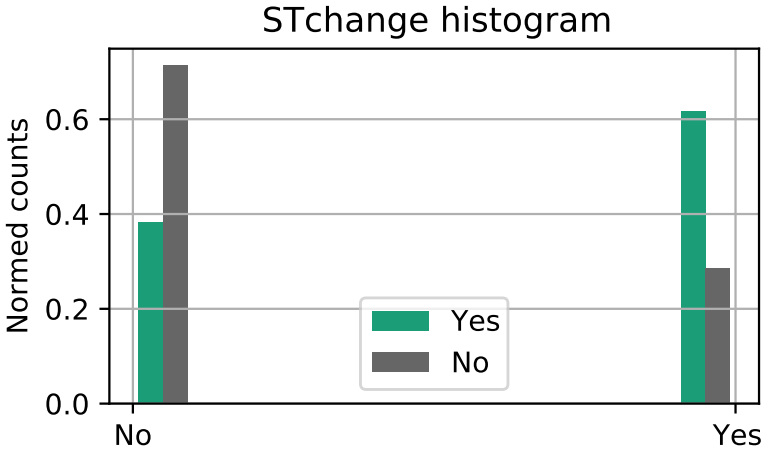

Figura 22: Distribución de la variable "STchange"

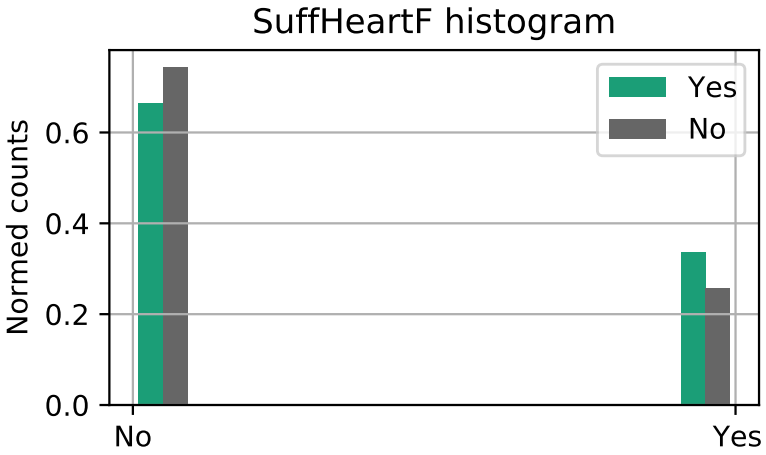

Figura 23: Distribución de la variable "SuffHeartF"

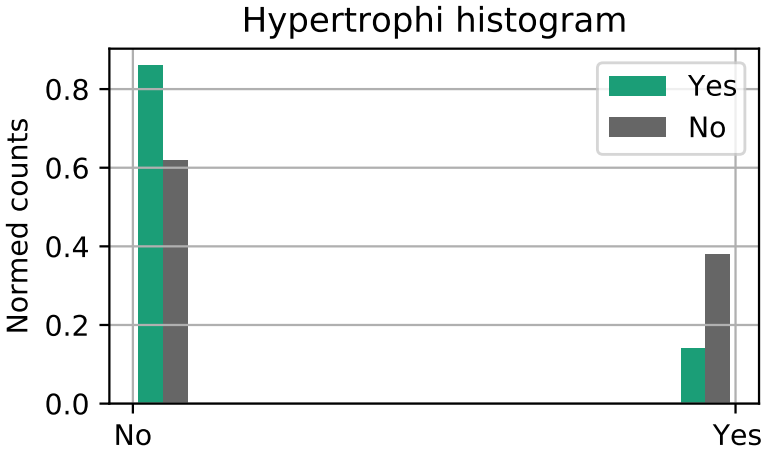

Figura 24: Distribución de la variable "Hypertrophi"

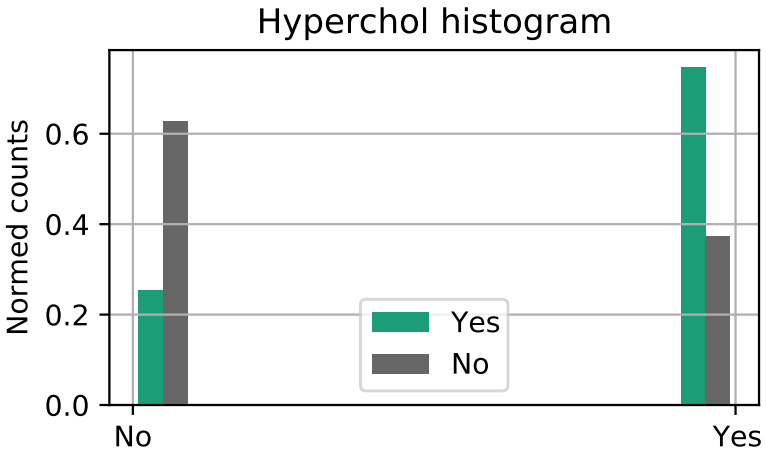

Figura 25: Distribución de la variable "Hyperchol"

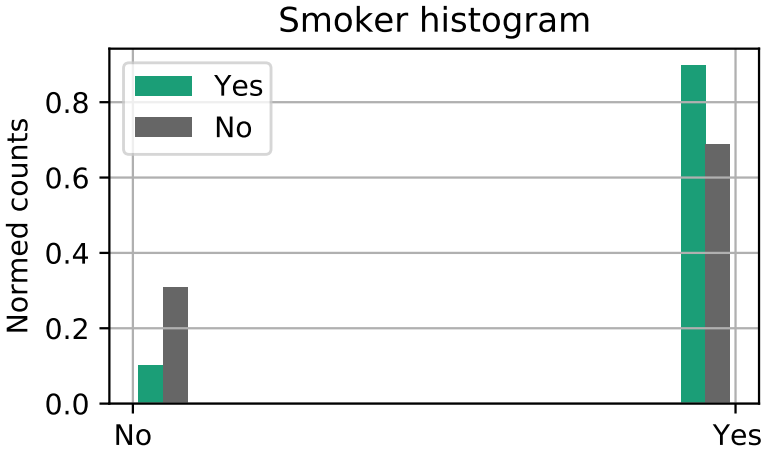

Figura 26: Distribución de la variable "Smoker"

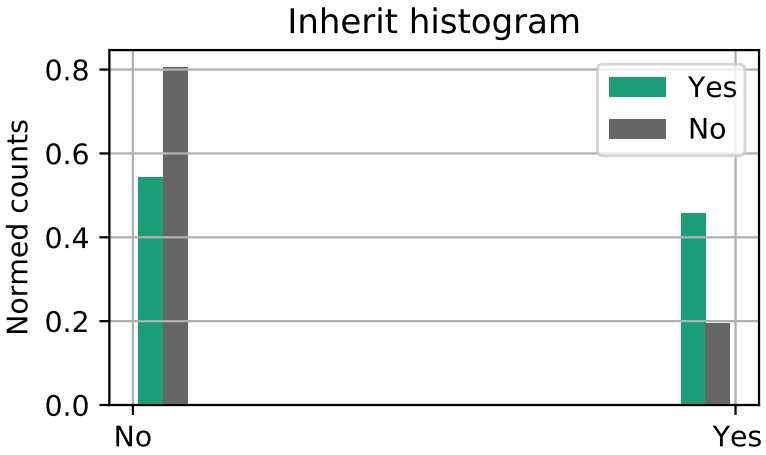

Figura 27: Distribución de la variable "Inherit"

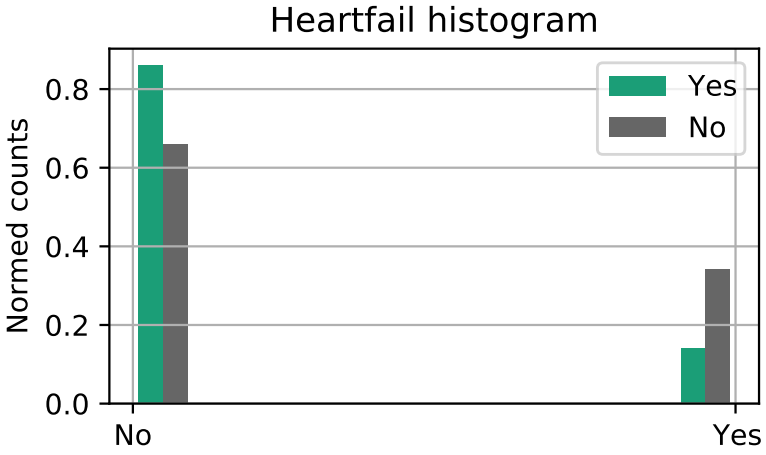

Figura 28: Distribución de la variable "Heartfail"

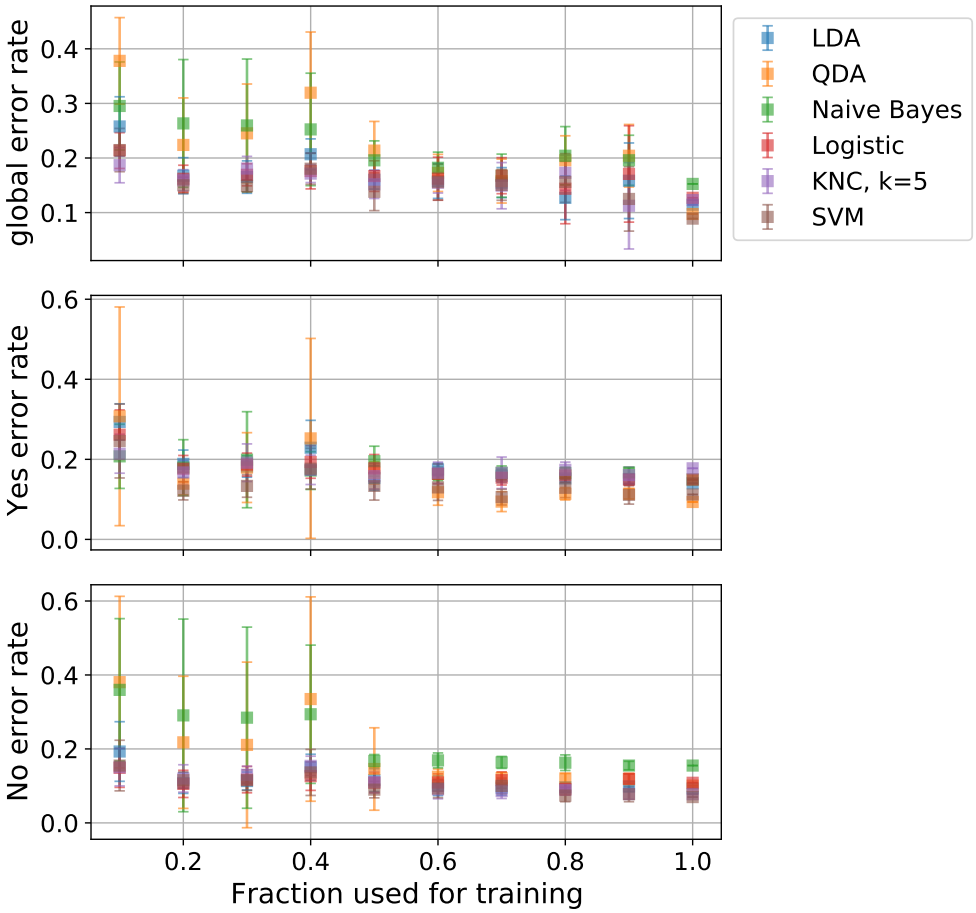

Figura 29: Tasas globales y locales para varios modelos, con conjuntos de entrenamiento divididos manteniendo proporcionalidad

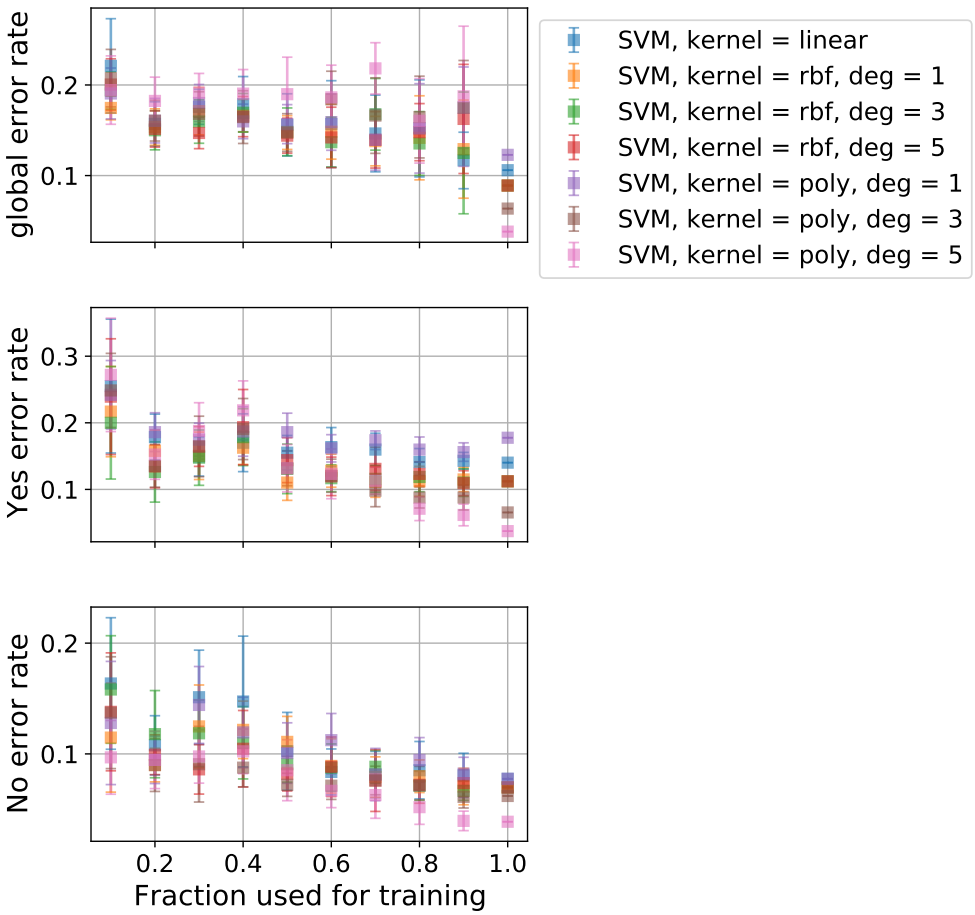

Figura 30: Tasas globales y locales para modelos SVM, con conjuntos de entrenamiento divididos manteniendo proporcionalidad

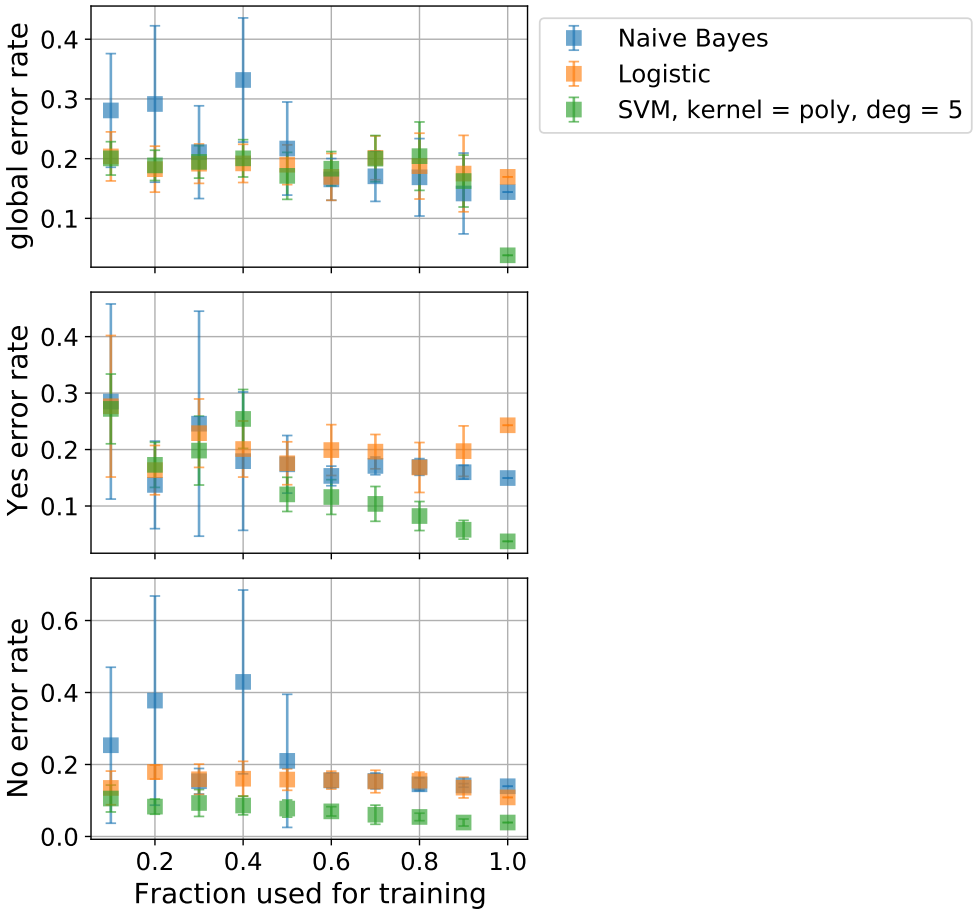

Figura 31: Tasas globales y locales para modelos SVM, con conjuntos de entrenamiento divididos manteniendo proporcionalidad

Problema 3

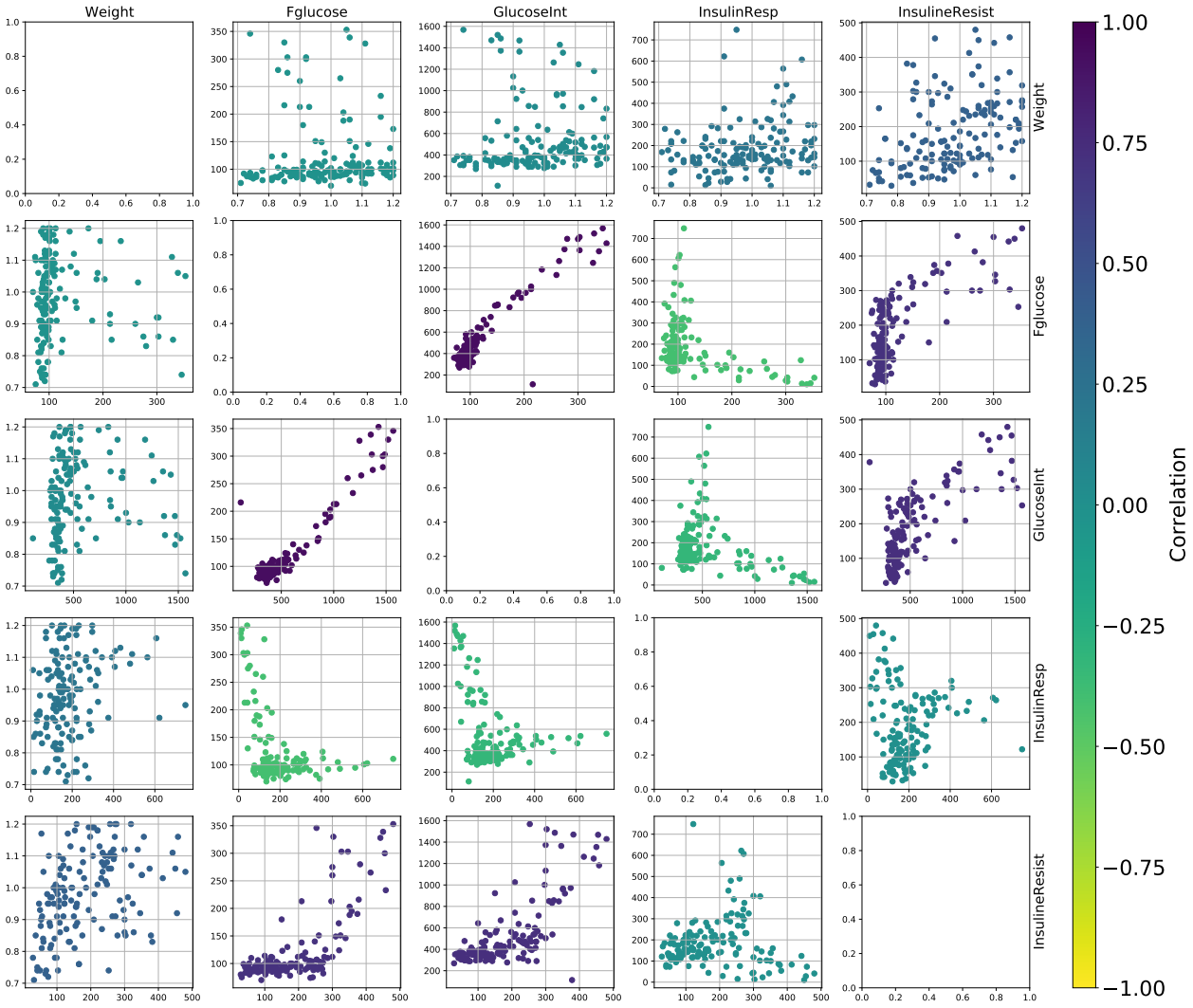

Figura 32: Gráfica de correlación para las variables numericas del problema 3

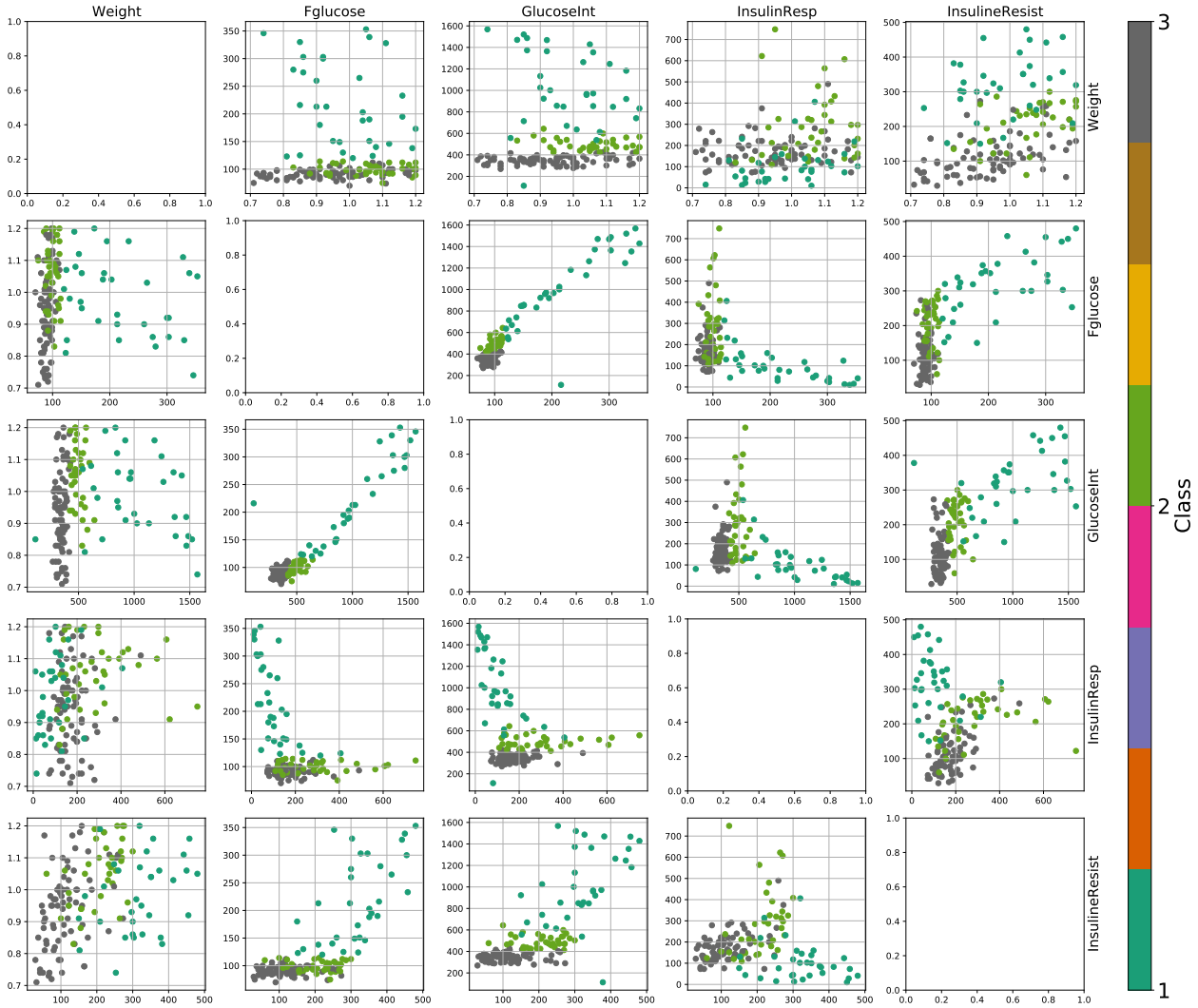

Figura 33: Gráficas de una variable contra otra divididas en clases para las variables numéricas del problema 3

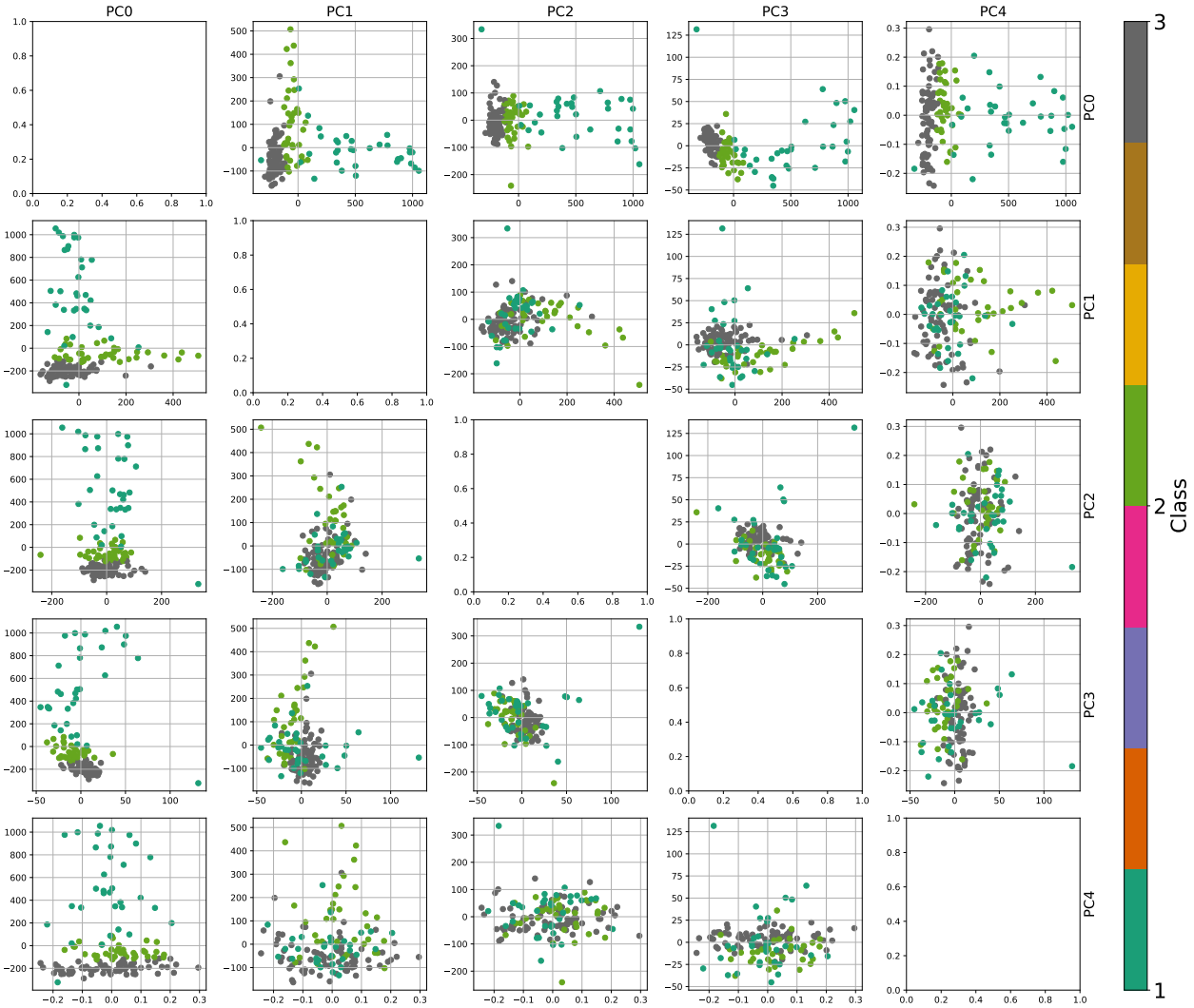

Figura 34: Gráficas de una variable contra otra divididas en clases para las variables numéricas del problema 3

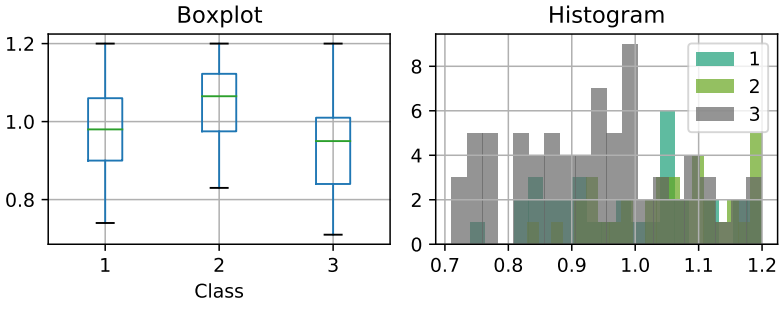

Figura 35: Distribución de la variable “Weight”

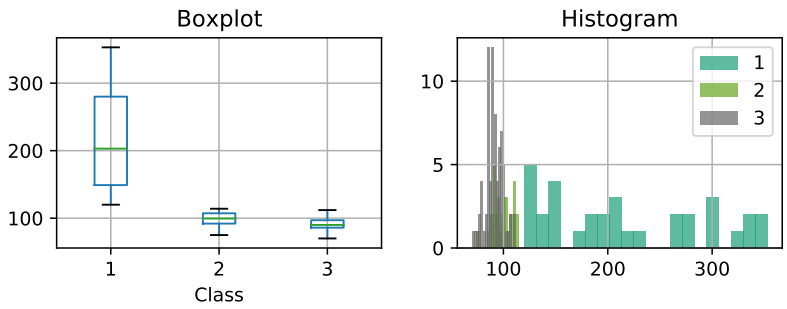

Figura 36: Distribución de la variable “Fglucose”

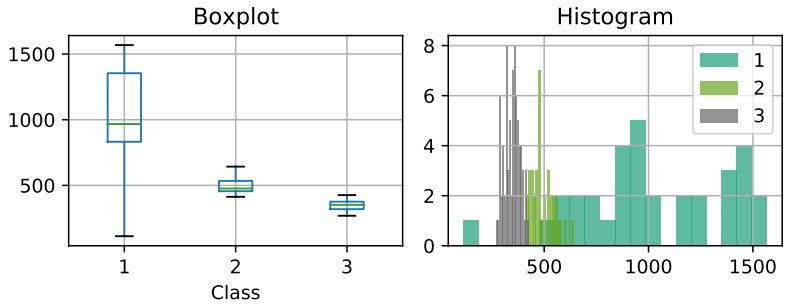

Figura 37: Distribución de la variable “Glucoseint”

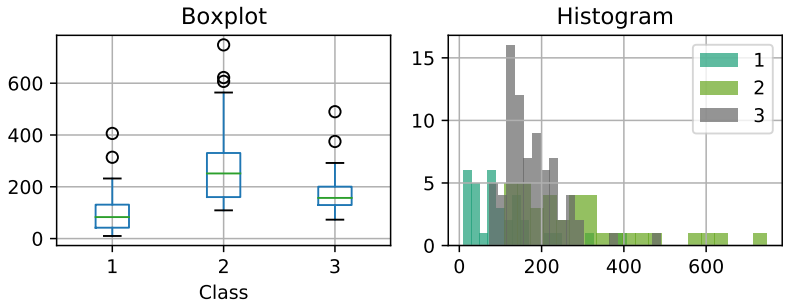

Figura 38: Distribución de la variable “InsulinResp”

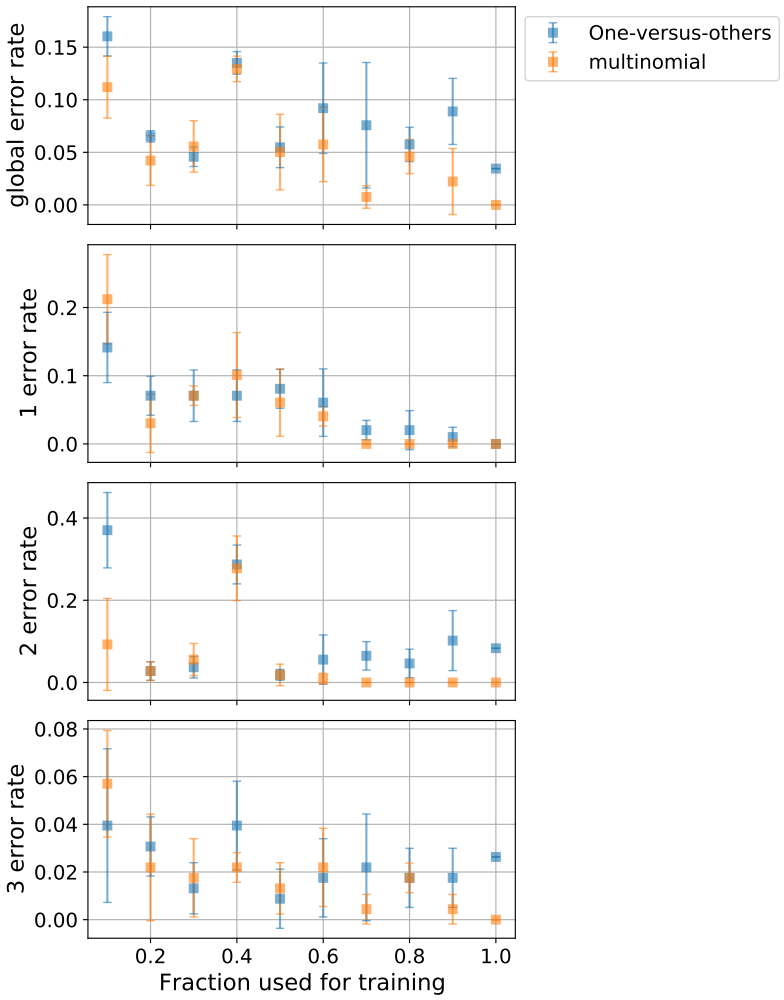

Figura 39: Distribución de la variable “InsulineResist”

Anexo 3: Código en Python de los problemas

```
#!/usr/bin/env python
# coding: utf-8
# Funciones auxiliares
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import matplotlib.cm as cm
from scipy.stats import boxcox
import sklearn.linear_model as lm
import sklearn.discriminant_analysis as da
import sklearn.svm as svm
import sklearn.neighbors as ng
import sklearn.naive_bayes as nb
import sklearn.model_selection as ms
import sklearn.preprocessing as prep
import sklearn.decomposition as de
import statsmodels.api as sm
if not(os.path.isdir("tarea")):
    mkdir("tarea")
def dataScaled(df):
    # Creating dictionary to store the different data frames
    data = {"original":df}
    # Standarizing data to have mean 0 and variance 1
    scaler = pr.StandardScaler()
    scaler.fit(df)
    data["standarized"] = pd.DataFrame(scaler.transform(df),index=df.index
    ,columns=df.columns)
    # Centering data to have variance 1
    scaler = pr.StandardScaler(with_mean=False)
    scaler.fit(df)
    data["withmean"] = pd.DataFrame(scaler.transform(df),index=df.index,
    columns=df.columns)
    return data
def correlationPlots(df,saveName="",cmap=cm.viridis_r,f=16,width=50,**
    kwargs):
    variables = [var for var in df.columns if np.issubdtype(df[var].dtype,
    np.number)]
    nvar = len(variables)
    values = [df[var].to_numpy() for var in variables]
    correlations = np.corrcoef(values)
    norm = mcolors.Normalize(vmin=-1,vmax=1)
    fig , ax = plt.subplots(ncols=nvar,nrows=nvar,figsize=(19,16),
    constrained_layout=True)
    for i in range(nvar):
        for j in range(nvar):
            if i!=j:
                ax[i][j].scatter(values[i],values[j],color=cmap(norm(
    correlations[i][j])),**kwargs)
                ax[i][j].grid()
        ax[0][i].set_title(str(variables[i]),fontsize=f)
```

```
47          ax[i][nvar-1].yaxis.set_label_position("right")
48          ax[i][nvar-1].set_ylabel(str(variables[i]),fontsize=f)
49      cbar = fig.colorbar(cm.ScalarMappable(norm=norm,cmap=cmap),ax=ax.ravel
    ().tolist(),aspect=width)
50      cbar.ax.tick_params(labelsize=1.5*f)
51      cbar.ax.yaxis.set_label_position("right")
52      cbar.ax.set_ylabel("Correlation",fontsize=1.5*f)
53      if bool(saveName):
54          plt.savefig(os.path.join("tarea","{0}-correlation.pdf".format(
    saveName)))
55      plt.show()
56
57  def classPlots(X,y,saveName="",cmap=cm.Dark2,f=16,width=50,**kwargs):
58      variables = [var for var in X.columns if np.issubdtype(X[var].dtype,np
    .number)]
59      nvar = len(variables)
60      values = [X[var].to_numpy() for var in variables]
61      classes = list(set(y))
62      nclass = len(classes)
63      norm = mcolors.Normalize(vmin=0,vmax=nclass-1)
64      colors = [cmap(norm(classes.index(clas))) for clas in y]
65      fig , ax = plt.subplots(ncols=nvar,nrows=nvar,figsize=(19,16),
    constrained_layout=True)
66      for i in range(nvar):
67          for j in range(nvar):
68              if i!=j:
69                  ax[i][j].scatter(values[i],values[j],color=colors,**kwargs
    )
70                  ax[i][j].grid()
71          ax[0][i].set_title(str(variables[i]),fontsize=f)
72          ax[i][nvar-1].yaxis.set_label_position("right")
73          ax[i][nvar-1].set_ylabel(str(variables[i]),fontsize=f)
74      cbar = fig.colorbar(cm.ScalarMappable(norm=norm,cmap=cmap),ax=ax.ravel
    ().tolist(),aspect=width,ticks=range(nclass))
75      cbar.ax.tick_params(labelsize=1.5*f)
76      cbar.ax.yaxis.set_label_position("right")
77      cbar.ax.set_ylabel("Class",fontsize=1.5*f)
78      cbar.ax.set_yticklabels(classes)
79      if bool(saveName):
80          plt.savefig(os.path.join("tarea","{0}-classes.pdf".format(saveName
    )))
81      plt.show()
82
83  def pcaPlots(X,y,saveName="",cmap=cm.Dark2,f=16,width=50,ncomps=0,**kwargs
    ):
84      variables = [var for var in X.columns if np.issubdtype(X[var].dtype,np
    .number)]
85      nvar = len(variables)
86      values = [X[var].to_numpy() for var in variables]
87      aux = np.transpose(values)
88      pca = de.PCA()
89      pca.fit(aux)
90      Z = pca.transform(aux)
91      variables = ["PC{0}".format(i) for i in range(len(variables))]
```

```
92      values = np.transpose(Z)
93      classes = list(set(y))
94      nclass = len(classes)
95      norm = mcolors.Normalize(vmin=0,vmax=nclass-1)
96      colors = [cmap(norm(classes.index(clas))) for clas in y]
97      if ncomps==0:
98          ncomps=nvar
99      fig , ax = plt.subplots(ncols=ncomps,nrows=ncomps,figsize=(19,16),
    constrained_layout=True)
100     for i in range(ncomps):
101         for j in range(ncomps):
102             if i!=j:
103                 ax[i][j].scatter(values[i],values[j],color=colors,**kwargs
    )
104                 ax[i][j].grid()
105         ax[0][i].set_title(str(variables[i]),fontsize=f)
106         ax[i][nvar-1].yaxis.set_label_position("right")
107         ax[i][nvar-1].set_ylabel(str(variables[i]),fontsize=f)
108     cbar = fig.colorbar(cm.ScalarMappable(norm=norm,cmap=cmap),ax=ax.ravel
    ().tolist(),aspect=width,ticks=range(nclass))
109     cbar.ax.tick_params(labelsize=1.5*f)
110     cbar.ax.yaxis.set_label_position("right")
111     cbar.ax.set_ylabel("Class",fontsize=1.5*f)
112     cbar.ax.set_yticklabels(classes)
113     if bool(saveName):
114         plt.savefig(os.path.join("tarea","{0}-pca.pdf".format(saveName)))
115     plt.show()
116
117
118 def indVarDistPlots(df,resCol,saveName="",cmap=cm.Dark2,**kwargs):
119     if not(resCol in df.columns):
120         raise ValueError("Name {0} is not in columns".format(resCol))
121     classes = list(set(df[resCol]))
122     nclass = len(classes)
123     norm = mcolors.Normalize(vmin=0,vmax=nclass-1)
124     colors = [cmap(norm(i)) for i in range(nclass)]
125     for col in df.columns:
126         if col != resCol and np.issubdtype(df[col].dtype,np.number):
127             fig, ax = plt.subplots(ncols=2,nrows=1,figsize=(6,2.5))
128             df.boxplot(column=col,by=resCol,ax=ax[0])
129             for i,val in enumerate(classes):
130                 df[df[resCol]==val][col].hist(ax=ax[1],label=str(val),
    color=colors[i],**kwargs)
131             ax[0].set_title("Boxplot")
132             ax[1].legend()
133             ax[1].set_title("Histogram")
134             fig.suptitle("{0} distribution".format(col),y=1.1)
135         elif col != resCol and (df[col].dtype == np.dtype('O')):
136             fig,ax = plt.subplots(nrows=1,ncols=1,figsize=(4,2.5))
137             labels = [str(val) for val in set(df[resCol])]
138             weights = [[1/df[df[resCol]==val][col].shape[0] for i in range
    (df[df[resCol]==val][col].shape[0])] for val in set(df[resCol])]
139             weights = np.transpose(weights)
140             ax.hist(np.transpose([df[df[resCol]==val][col].to_numpy() for
```

```
    val in set(df[resCol])]),
141                     weights=weights,
142                     label=labels,
143                     color=colors,
144                     **kwargs)
145             ax.legend()
146             ax.grid()
147             ax.set_ylabel("Normed counts")
148             ax.set_title("{0} histogram".format(col))
149         elif col != resCol:
150             print("Column {0} cannot be interpreted".format(col))
151         if bool(saveName):
152             plt.tight_layout()
153             plt.savefig(os.path.join("tarea","{0}-{1}-dist.pdf".format(
    saveName,col)))
154         plt.show()
155
156 def boxcoxLambdaTable(df,resCol,alpha=0.05):
157     names = []
158     lambdas = []
159     intervalsBot = []
160     intervalsTop = []
161     for col in df.columns:
162         if col != resCol and np.issubdtype(df[col].dtype,np.number):
163             if (df[col]>0).prod():
164                 names.append(col)
165                 bx = boxcox(df[col],alpha=alpha)
166                 lambdas.append(bx[1])
167                 intervalsBot.append(bx[2][0])
168                 intervalsTop.append(bx[2][1])
169             else:
170                 print("Can't convert column {0}: not entirely positive".
    format(col) )
171     fin = pd.DataFrame.from_dict({"lambda":lambdas,"Lower confidence
    interval, alpha = {0}".format(alpha):intervalsBot,"Upper confidence
    interval, alpha = {0}".format(alpha):intervalsTop})
172     fin.index = names
173     return fin.transpose()
174
175 def errorRates(mod,X,y,n=100,size=0.5,equalRatios=False):
176     fin = np.zeros(n)
177     classes = list(set(y))
178     perClass = [np.zeros(n) for val in classes]
179     for i in range(n):
180         if size<1.0:
181             if equalRatios:
182                 X_train, X_test, y_train, y_test = ms.train_test_split(X,y
    ,train_size=size,stratify=y)
183             else:
184                 X_train, X_test, y_train, y_test = ms.train_test_split(X,y
    ,train_size=size)
185             fit = mod.fit(X_train,y_train)
186             res = fit.predict(X_test)
187             fin[i] = np.mean(y_test != res)
```

```
            for j,val in enumerate(classes):
                curr = X[y==val]
                res = fit.predict(curr)
                perClass[j][i] = np.mean(y[y==val] != res)
        elif size==1:
            fit = mod.fit(X,y)
            res = fit.predict(X)
            fin[i] = np.mean(y != res)
            for j,val in enumerate(classes):
                curr = X[y==val]
                res = fit.predict(curr)
                perClass[j][i] = np.mean(y[y==val] != res)
    final = [np.mean(fin),np.std(fin)]
    for cla in perClass:
        final.append(np.mean(cla))
        final.append(np.std(cla))
    return final

def errorRatesTable(models,X,y,n=100,size=0.5,equalRatios=False,names=[]):
    if not bool(names):
        names = [str(mod).split("(")[0] for mod in models]
    elif len(names) != len(models):
        raise ValueError("length of names must match length of models")
    errors = [errorRates(mod,X,y,n=n,size=size,equalRatios=equalRatios)
    for mod in models]
    fin = pd.DataFrame(errors)
    fin.index = names
    cols = ["Mean global error","std"]
    classes = list(set(y))
    for i in range(len(classes)):
        cols.append("Mean {0} error".format(classes[i]))
        cols.append("std")
    fin.columns = cols
    return fin.transpose()

def errorRatesSizeTable(models,X,y,sizes,n=100,equalRatios=False,names=[])
    :
    tables = []
    indexes = []
    for i,size in enumerate(sizes):
        print("Simulating size: {0}".format(size))
        tab = errorRatesTable(models,X,y,n=n,size=size,equalRatios=
    equalRatios,names=names)
        tables.append(tab)
        newIndex = ["{0}, entrenamiento: {1}".format(name,np.round(sizes[i
    ],decimals=2)) for name in tab.index]
        indexes.extend(newIndex)
    df = pd.concat(tables,ignore_index=True)
    df.index = indexes
    return df

def errorRatesSizesPlot(errorTable,saveName="",f1=16,f2=14,**kwargs):
    sizes = [float(name.split(":")[-1]) for name in errorTable.index]
    sizes = list(set(sizes))
```

```
    nclass = int(errorTable.shape[0]/(2*len(sizes)))
    names = [name.split(" ")[1] for name in errorTable.index]
    names = [names[i] for i in range(0,2*nclass,2)]
    fig,ax = plt.subplots(nrows=nclass,ncols=1,sharex=True,figsize=(8,2.5*nclass))
    for col in errorTable.columns:
        for j in range(nclass):
            vals = errorTable.iloc[range(2*j,errorTable.shape[0],2*nclass)][col].to_numpy()
            stds = errorTable.iloc[range(2*j+1,errorTable.shape[0],2*nclass)][col].to_numpy()
            ax[j].errorbar(sizes,vals,yerr=stds,label=str(col),fmt="o",**kwargs)
            ax[j].grid(True)
            ax[j].set_ylabel("{0} error rate".format(names[j]),fontsize=f1)
            ax[j].tick_params(axis="both",labelsize=f2)
    ax[nclass-1].set_xlabel("Fraction used for training",fontsize=f1)
    ax[0].legend(bbox_to_anchor=(1,1),fontsize=f2)
    plt.subplots_adjust(hspace=0.05)
    if bool(saveName):
        plt.tight_layout()
        plt.savefig(os.path.join("tarea","{0}-sizeDependence.pdf".format(saveName)))
    plt.show()

def pairInteractionMin(df,resCol,model):
    transformed = df.copy()
    vals = []
    inters = []
    for col1 in df.columns:
        for col2 in df.columns:
            if col1 != resCol and col2!= resCol and col1!=col2:
                transformed["inter"] = transformed[col1]*transformed[col2]
                models = [model]
                names = ["LDA"]
                sizes=[1.0]
                res = errorRatesSizeTable(models,transformed.drop(resCol,axis=1),transformed[resCol],sizes=sizes,n=10,equalRatios=True)
                vals.append(res.iloc[0,0])
                inters.append(col1 + "*" + col2)
    m = np.argmin(vals)
    print("error minimizing interacion: {0}".format(inters[m]))
    print("Value: {0}".format(vals[m]))
    return inters[m].split("*")

def multinomialLogisticRegressionClassifier(X,y,reference,**kwargs):
    classes = list(set(y))
    nclasses = len(classes)
    classifiers = []
    newY = y.astype("category").cat.codes
    refCode = newY[y==reference].iloc[0]
    newY = (newY - refCode) % nclasses
    newY.name = y.name
```

```
285     conversion = {}
286     for c in classes:
287         code = newY[y==c].iloc[0]
288         conversion.update({c:code})
289     mod = sm.MNLogit(exog=X,endog=newY)
290     fit = mod.fit(maxiters=10**5)
291     return conversion,fit
292 
293 # Problema 1
294 df = pd.read_csv("pimate.csv")
295 df = df.append(pd.read_csv("pimatr.csv"),ignore_index=True)
296 print(df.head())
297 
298 correlationPlots(df,saveName="1",alpha=0.7,s=5)
299 
300 classPlots(df.drop("type",axis=1),df["type"],saveName="1",s=5)
301 
302 pcaPlots(df.drop("type",axis=1),df["type"],saveName="1",s=5)
303 
304 indVarDistPlots(df,"type",saveName="1",alpha=0.7,density=True,bins=20)
305 
306 models = [
307     da.LinearDiscriminantAnalysis(),
308     da.QuadraticDiscriminantAnalysis(),
309     nb.GaussianNB(),
310     lm.LogisticRegression(dual=False,max_iter=10**6),
311     ng.KNeighborsClassifier(),
312     svm.SVC()
313 ]
314 names = [
315     "LDA",
316     "QDA",
317     "Naive Bayes",
318     "Logistic",
319     "KNC, k=5",
320     "SVM"
321 ]
322 sizes=np.linspace(0.1,1.0,10)
323 
324 res = errorRatesSizeTable(models,df.drop("type",axis=1),df["type"],sizes=
        sizes,n=10,names=names)
325 tab = "p{4cm}"
326 for col in res.columns:
327     tab +="|p{1.5cm}"
328 res.to_latex(buf=os.path.join("tarea","1-gen-tab.tex"),float_format="{:0.3
        f}".format,longtable=True,column_format=tab)
329 errorRatesSizesPlot(res,saveName="1-gen",alpha=0.6,capsize=3,marker="s")
330 
331 sizes=np.linspace(0.1,1.0,10)
332 res = errorRatesSizeTable(models,df.drop("type",axis=1),df["type"],sizes=
        sizes,n=10,equalRatios=True,names=names)
333 tab = "p{4cm}"
334 for col in res.columns:
335     tab +="|p{1.5cm}"
```

```
336 res.to_latex(buf=os.path.join("tarea","1-gen-eq-tab.tex"),float_format="
    {:0.3f}".format,longtable=True,column_format=tab)
337 errorRatesSizesPlot(res,saveName="1-gen-eq",alpha=0.6,capsize=3,marker="s"
    )
338
339 transTable = boxcoxLambdaTable(df,"type",alpha=0.05)
340 tab = "p{4cm}"
341 for col in transTable.transpose().columns:
342     tab += "|p{3cm}"
343 transTable.transpose().to_latex(buf=os.path.join("tarea","1-bxparam-tab.
    tex"),float_format="{:0.4f}".format,column_format=tab)
344
345 transformed = df.copy()
346 for col in transTable.columns:
347     transformed[col] = boxcox(transformed[col],lmbda = transTable[col]["
    lambda"])
348
349 sizes=np.linspace(0.1,1.0,10)
350 res = errorRatesSizeTable(models,transformed.drop("type",axis=1),
    transformed["type"],sizes=sizes,n=10,equalRatios=True)
351 errorRatesSizesPlot(res,saveName="1-boxcox-eq",alpha=0.6,capsize=3,marker=
    "s")
352
353 # LDA analysis
354 mod = da.LinearDiscriminantAnalysis()
355 col1,col2 = pairInteractionMin(df,"type",mod)
356
357 transformed = df.copy()
358 transformed["pairIter"] = transformed[col1]*transformed[col2]
359 models = [mod]
360 names = ["LDA"]
361 sizes=np.linspace(0.1,1.0,10)
362 res1 = errorRatesSizeTable(models,transformed.drop("type",axis=1),
    transformed["type"],sizes=sizes,n=10,equalRatios=True,names = names)
363 tab = "p{4cm}"
364 for col in res1.columns:
365     tab +="|p{1.5cm}"
366 res1.to_latex(buf=os.path.join("tarea","1-lda-eq-tab.tex"),float_format="
    {:0.3f}".format,longtable=True,column_format=tab)
367 errorRatesSizesPlot(res1,saveName="1-lda-eq",alpha=0.6,capsize=3,marker="s
    ")
368
369 # Naive Bayes
370 mod = nb.GaussianNB()
371 col1,col2 = pairInteractionMin(df,"type",mod)
372
373 transformed = df.copy()
374 transformed["bmi*ped"] = transformed[col1]*transformed[col2]
375 models = [mod]
376 names = ["Naive Bayes"]
377 sizes=np.linspace(0.1,1.0,10)
378 res2 = errorRatesSizeTable(models,transformed.drop("type",axis=1),
    transformed["type"],sizes=sizes,n=10,equalRatios=True,names=names)
379 tab = "p{4cm}"
```

```
for col in res2.columns:
    tab +="|p{1.5cm}"
res2.to_latex(buf=os.path.join("tarea","1-nb-eq-tab.tex"),float_format="{:0.3f}".format,longtable=True,column_format=tab)
errorRatesSizesPlot(res2,saveName="1-nb-eq",alpha=0.6,capsize=3,marker="s")

# Logistic regression
transformed = df.copy()
transformed = transformed[["glu","bmi","ped",'age',"type"]]
transformed["age2"] = transformed["age"]*transformed["age"]
models = [lm.LogisticRegression(max_iter=10**6)]
names = ["Logistic"]
sizes=np.linspace(0.1,1.0,10)
res3 = errorRatesSizeTable(models,transformed.drop("type",axis=1),transformed["type"],sizes=sizes,n=10,equalRatios=True,names=names)
tab = "p{4cm}"
for col in res3.columns:
    tab +="|p{1.5cm}"
res3.to_latex(buf=os.path.join("tarea","1-log-eq-tab.tex"),float_format="{:0.3f}".format,longtable=True,column_format=tab)
errorRatesSizesPlot(res3,saveName="1-log-eq",alpha=0.6,capsize=3,marker="s")

# SVM analysis
models = [
    svm.SVC(kernel="linear"),
    svm.SVC(kernel="rbf",degree=1),
    svm.SVC(kernel="rbf",degree=3),
    svm.SVC(kernel="rbf",degree=5),
    svm.SVC(kernel="poly",degree=1),
    svm.SVC(kernel="poly",degree=3),
    svm.SVC(kernel="poly",degree=5),
]
names = [
    "SVM, kernel = linear",
    "SVM, kernel = rbf, deg = 1",
    "SVM, kernel = rbf, deg = 3",
    "SVM, kernel = rbf, deg = 5",
    "SVM, kernel = poly, deg = 1",
    "SVM, kernel = poly, deg = 3",
    "SVM, kernel = poly, deg = 5"
]

res = errorRatesSizeTable(models,df.drop("type",axis=1),df["type"],sizes=sizes,n=10,names=names)
errorRatesSizesPlot(res,saveName="1-svm",alpha=0.6,capsize=3,marker="s")

res = errorRatesSizeTable(models,df.drop("type",axis=1),df["type"],sizes=sizes,n=10,names=names,equalRatios=True)
errorRatesSizesPlot(res,saveName="1-svm-eq",alpha=0.6,capsize=3,marker="s")

models = [
```

```
426     svm.SVC(kernel="poly",degree=5)
427 ]
428 names = [ "SVM, kernel = poly, deg = 5"
429 ]
430 sizes=np.linspace(0.1,1,10)
431 res4 = errorRatesSizeTable(models,df.drop("type",axis=1),df["type"],sizes=
    sizes,n=10,names=names,equalRatios=True)
432 tab = "p{4cm}"
433 for col in res4.columns:
434     tab +="|p{1.5cm}"
435 res4.to_latex(buf=os.path.join("tarea","1-svm-fin-eq-tab.tex"),
    float_format="{:0.3f}".format,longtable=True,column_format=tab)
436
437 errorRatesSizesPlot(res4,saveName="1-svm-fin-eq",alpha=0.8,capsize=3,
    marker="s",markersize=10)
438
439 final = pd.concat([res1,res2,res3,res4],axis=1)
440 tab = "p{4cm}"
441 for col in final.columns:
442     tab +="|p{1.5cm}"
443 final.to_latex(buf=os.path.join("tarea","1-final-tab.tex"),float_format="
    {:0.3f}".format,longtable=True,column_format=tab)
444 errorRatesSizesPlot(final,saveName="1-final",alpha=0.65,capsize=3,marker="
    s",markersize=8)
445
446 neds = [res.iloc[range(0,res.shape[0],2)[-3:],:] for res in [res1,res2,
    res3,res4]]
447 final = pd.concat(neds,axis=1)
448 tab = "p{4cm}"
449 for col in final.columns:
450     tab +="|p{1.5cm}"
451 final.to_latex(buf=os.path.join("tarea","1-final-correg-aparent-tab.tex"),
    float_format="{:0.3f}".format,longtable=True,column_format=tab)
452
453 neds = [res.iloc[range(0,res.shape[0],2)[-12:-9],:] for res in [res1,res2,
    res3,res4]]
454 final = pd.concat(neds,axis=1)
455 tab = "p{4cm}"
456 for col in final.columns:
457     tab +="|p{1.5cm}"
458 final.to_latex(buf=os.path.join("tarea","1-final-correg-nonaparent-tab.tex
    "),float_format="{:0.3f}".format,longtable=True,column_format=tab)
459
460 # Problema 2
461 df = pd.read_csv("cad1.csv",index_col=0)
462 print(df.head())
463
464 indVarDistPlots(df,"CAD",saveName="2")
465
466 resCol="CAD"
467 dfCoded = df.copy()
468 for col in df.columns:
469     if col!=resCol and df[col].dtype==np.dtype("O"):
470         dfCoded[col] = df[col].astype("category").cat.codes
```

```
471
472 dfCoded
473
474 models = [
475     da.LinearDiscriminantAnalysis(),
476     da.QuadraticDiscriminantAnalysis(),
477     nb.GaussianNB(),
478     lm.LogisticRegression(dual=False,max_iter=10**6),
479     ng.KNeighborsClassifier(),
480     svm.SVC()
481 ]
482 names = [
483     "LDA",
484     "QDA",
485     "Naive Bayes",
486     "Logistic",
487     "KNC, k=5",
488     "SVM"
489 ]
490
491 sizes=np.linspace(0.1,1.0,10)
492 res = errorRatesSizeTable(models,dfCoded.drop("CAD",axis=1),dfCoded["CAD"
    ],sizes=sizes,n=10,equalRatios=True,names=names)
493 tab = "p{4cm}"
494 for col in res.columns:
495     tab +="|p{1.5cm}"
496 res.to_latex(buf=os.path.join("tarea","2-gen-eq-tab.tex"),float_format="
    {:0.3f}".format,longtable=True,column_format=tab)
497 errorRatesSizesPlot(res,saveName="2-gen-eq",alpha=0.6,capsize=3,marker="s"
    )
498
499 # Naive Bayes
500 mod = nb.GaussianNB()
501 col1,col2 = pairInteractionMin(dfCoded,"CAD",mod)
502
503 transformed = dfCoded.copy()
504 transformed["inter"] = transformed[col2]*transformed[col1]
505 models = [mod]
506 names = ["Naive Bayes"]
507 sizes=np.linspace(0.1,1.0,10)
508 res2 = errorRatesSizeTable(models,transformed.drop("CAD",axis=1),
    transformed["CAD"],sizes=sizes,n=10,equalRatios=True,names=names)
509 tab = "p{4cm}"
510 for col in res2.columns:
511     tab +="|p{1.5cm}"
512 res2.to_latex(buf=os.path.join("tarea","2-nb-eq-tab.tex"),float_format="
    {:0.3f}".format,longtable=True,column_format=tab)
513 errorRatesSizesPlot(res2,saveName="2-nb-eq",alpha=0.6,capsize=3,marker="s"
    )
514
515 # logistic Regresion
516 transformed = dfCoded.copy()
517 transformed = transformed[["AngPec","AMI", "STcode","STchange", "Hyperchol
    ","CAD"]]
```

```
models = [lm.LogisticRegression(max_iter=10**6)]
names = ["Logistic"]
sizes=np.linspace(0.1,1.0,10)
res3 = errorRatesSizeTable(models,transformed.drop("CAD",axis=1),
    transformed["CAD"],sizes=sizes,n=10,equalRatios=True,names=names)
tab = "p{4cm}"
for col in res3.columns:
    tab +="|p{1.5cm}"
res3.to_latex(buf=os.path.join("tarea","2-log-eq-tab.tex"),float_format="
    {:0.3f}".format,longtable=True,column_format=tab)
errorRatesSizesPlot(res3,saveName="2-log-eq",alpha=0.6,capsize=3,marker="s
    ")

# SVM analysis
models = [
    svm.SVC(kernel="linear"),
    svm.SVC(kernel="rbf",degree=1),
    svm.SVC(kernel="rbf",degree=3),
    svm.SVC(kernel="rbf",degree=5),
    svm.SVC(kernel="poly",degree=1),
    svm.SVC(kernel="poly",degree=3),
    svm.SVC(kernel="poly",degree=5),
]
names = [
    "SVM, kernel = linear",
    "SVM, kernel = rbf, deg = 1",
    "SVM, kernel = rbf, deg = 3",
    "SVM, kernel = rbf, deg = 5",
    "SVM, kernel = poly, deg = 1",
    "SVM, kernel = poly, deg = 3",
    "SVM, kernel = poly, deg = 5"
]

res = errorRatesSizeTable(models,dfCoded.drop("CAD",axis=1),dfCoded["CAD"
    ],sizes=sizes,n=10,names=names,equalRatios=True)
errorRatesSizesPlot(res,saveName="2-svm-eq",alpha=0.6,capsize=3,marker="s"
    )

models = [
    svm.SVC(kernel="poly",degree=5)
]
names = [ "SVM, kernel = poly, deg = 5"
]
sizes=np.linspace(0.1,1,10)
res4 = errorRatesSizeTable(models,dfCoded.drop("CAD",axis=1),dfCoded["CAD"
    ],sizes=sizes,n=10,equalRatios=True,names=names)
tab = "p{4cm}"
for col in res4.columns:
    tab +="|p{1.5cm}"
res4.to_latex(buf=os.path.join("tarea","2-svm-fin-eq-tab.tex"),
    float_format="{:0.3f}".format,longtable=True,column_format=tab)
errorRatesSizesPlot(res4,saveName="2-svm-fin-eq",alpha=0.8,capsize=3,
    marker="s",markersize=10)

```

```
final = pd.concat([res2,res3,res4],axis=1)
tab = "p{4cm}"
for col in final.columns:
    tab +="|p{1.5cm}"
final.to_latex(buf=os.path.join("tarea","2-final-tab.tex"),float_format="{:0.3f}".format,longtable=True,column_format=tab)
errorRatesSizesPlot(final,saveName="2-final",alpha=0.65,capsize=3,marker="s",markersize=8)

neds = [res.iloc[range(0,res.shape[0],2)[-3:],:] for res in [res2,res3,res4]]
final = pd.concat(neds,axis=1)
tab = "p{4cm}"
for col in final.columns:
    tab +="|p{1.5cm}"
final.to_latex(buf=os.path.join("tarea","2-final-correg-aparent-tab.tex"),float_format="{:0.3f}".format,longtable=True,column_format=tab)

neds = [res.iloc[range(0,res.shape[0],2)[-12:-9],:] for res in [res2,res3,res4]]
final = pd.concat(neds,axis=1)
tab = "p{4cm}"
for col in final.columns:
    tab +="|p{1.5cm}"
final.to_latex(buf=os.path.join("tarea","2-final-correg-nonaparent-tab.tex"),float_format="{:0.3f}".format,longtable=True,column_format=tab)

# Problema 3
df = pd.read_csv("Glucose1.txt",index_col="Patient")
df["Class"] = df["Class"].astype("O")
print(df.head())

indVarDistPlots(df,"Class",saveName="3",alpha=0.7,bins = 20)

correlationPlots(df,saveName="3")

pcaPlots(df.drop("Class",axis=1),df["Class"],saveName="3")

classPlots(df.drop("Class",axis=1),df["Class"],saveName="3")

models = [lm.LogisticRegression(max_iter=10**6,multi_class="ovr"),lm.LogisticRegression(max_iter=10**6,multi_class="multinomial")]
names = ["One-versus-others","multinomial"]
sizes=np.linspace(0.1,1.0,10)

res = errorRatesSizeTable(models,df.drop("Class",axis=1),df["Class"].astype("int64"),sizes=sizes,n=3,equalRatios=True,names=names)
tab = "p{4cm}"
for col in res.columns:
    tab +="|p{1.5cm}"
res.to_latex(buf=os.path.join("tarea","3-eq-tab.tex"),float_format="{:0.3f}".format,longtable=True,column_format=tab)
errorRatesSizesPlot(res,saveName="3-eq",alpha=0.6,capsize=3,marker="s")

```

```
609 moddescript = df[["Fglucose","GlucoseInt","Class"]].copy()
610 moddescript["Fglucose^2"] = df["Fglucose"]**2
611
612 models = [lm.LogisticRegression(max_iter=10**6,multi_class="ovr")]
613 names = ["One-versus-others"]
614 sizes=np.linspace(0.1,1.0,10)
615
616 modpredict1 = df[["Fglucose","Class"]].copy()
617 interactions = ["GlucoseInt*GlucoseInt"]
618 for inter in interactions:
619     columns = inter.split("*")
620     modpredict1[inter] = df[columns].product(axis=1)
621
622 res1 = errorRatesSizeTable(models,modpredict1.drop("Class",axis=1),
    modpredict1["Class"].astype("int64"),sizes=sizes,n=20,equalRatios=True,
    names=names)
623
624 modpredict2 = df[["InsulinResp","Class"]].copy()
625 interactions = ["Fglucose*InsulinResp","GlucoseInt*InsulinResp"]
626 for inter in interactions:
627     columns = inter.split("*")
628     modpredict2[inter] = df[columns].product(axis=1)
629
630 res2 = errorRatesSizeTable(models,modpredict2.drop("Class",axis=1),
    modpredict2["Class"].astype("int64"),sizes=sizes,n=20,equalRatios=True,
    names=names)
631
632 modpredict3 = df[["Fglucose","Class"]].copy()
633 interactions = ["Weight*InsulineResist", "Fglucose*GlucoseInt", "
    GlucoseInt*GlucoseInt"]
634 for inter in interactions:
635     columns = inter.split("*")
636     modpredict[inter] = df[columns].product(axis=1)
637
638 res3 = errorRatesSizeTable(models,modpredict3.drop("Class",axis=1),
    modpredict3["Class"].astype("int64"),sizes=sizes,n=20,equalRatios=True,
    names=names)
639
640 neds = [res.iloc[range(0,res.shape[0],2)[-4:],0] for res in [res1,res2,
    res3]]
641 final = pd.concat(neds,axis=1)
642 final.columns = ["Modelo {0}".format(i) for i in range(1,4)]
643 tab = "p{4cm}"
644 for col in final.columns:
645     tab +="|p{2cm}"
646 final.to_latex(buf=os.path.join("tarea","3-descrip-aparent-tab.tex"),
    float_format="{:0.3f}".format,longtable=True,column_format=tab)
647 final
648
649 neds = [res.iloc[range(0,res.shape[0],2)[-16:-12],0] for res in [res1,res2
    ,res3]]
650 final = pd.concat(neds,axis=1)
651 final.columns = ["Modelo {0}".format(i) for i in range(1,4)]
652 tab = "p{4cm}"
```

```
for col in final.columns:
    tab +="|p{2cm}"
final.to_latex(buf=os.path.join("tarea","3-descrip-nonaparent.tex"),
    float_format="{:0.3f}".format,longtable=True,column_format=tab)
final

modpredict = modpredict2

res1 = errorRatesSizeTable(models,moddescript.drop("Class",axis=1),
    moddescript["Class"].astype("int64"),sizes=sizes,n=3,equalRatios=True,
    names=names)
tab = "p{4cm}"
for col in res.columns:
    tab +="|p{1.5cm}"
res1.to_latex(buf=os.path.join("tarea","3-descrip-tab.tex"),float_format="
    {:0.3f}".format,longtable=True,column_format=tab)
errorRatesSizesPlot(res1,saveName="3-descrip",alpha=0.6,capsize=3,marker="
    s")

res2 = errorRatesSizeTable(models,modpredict.drop("Class",axis=1),
    modpredict["Class"].astype("int64"),sizes=sizes,n=20,equalRatios=True,
    names=names)
tab = "p{4cm}"
for col in res.columns:
    tab +="|p{1.5cm}"
res2.to_latex(buf=os.path.join("tarea","3-predict-tab.tex"),float_format="
    {:0.3f}".format,longtable=True,column_format=tab)
errorRatesSizesPlot(res2,saveName="3-predict",alpha=0.6,capsize=3,marker="
    s")

neds = [res.iloc[range(0,res.shape[0],2)[-4:],0] for res in [res1,res2]]
final = pd.concat(neds,axis=1)
final.columns = ["Descriptivo","Predictivo"]
tab = "p{4cm}"
for col in final.columns:
    tab +="|p{1.5cm}"
final.to_latex(buf=os.path.join("tarea","3-final-correg-aparent-tab.tex"),
    float_format="{:0.3f}".format,longtable=True,column_format=tab)

final

neds = [res.iloc[range(0,res.shape[0],2)[-16:-12],0] for res in [res1,res2
    ]]
final = pd.concat(neds,axis=1)
final.columns = ["Descriptivo","Predictivo"]
tab = "p{4cm}"
for col in final.columns:
    tab +="|p{2cm}"
final.to_latex(buf=os.path.join("tarea","3-final-correg-nonaparent-tab.tex
    "),float_format="{:0.3f}".format,longtable=True,column_format=tab)

final
```

Referencias

[1] Jack W Smith y col. "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus". En: *Proceedings of the Annual Symposium on Computer Application in Medical Care*. American Medical Informatics Association. 1988, pág. 261.

[2] *C-Support Vector Classification. sklearn.svm.SVC documentation*. 2020. URL: `https://tex.stackexchange.com/questions/3587/how-can-i-use-bibtex-to-cite-a-web-page` (visitado 05-04-2020).

[3] JF Hansen. "The clinical diagnosis of ischaemic heart disease due to coronary artery disease." En: *Danish medical bulletin* 27.6 (1980), págs. 280-286.