## ASSESSMENT 01: ENSEMBLE METHODS AND GENETIC ALGORITHMS

|  |  |  |  |
| --- | --- | --- | --- |
| **Author/s** | | | |
| **Group:** |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name:** | Ignacio |  | **Surname:** | Pérez-Cea Espantaleón |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name:** | Pablo |  | **Surname:** | Sanz Caperote |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name:** | Álvaro |  | **Surname:** | Rodríguez González |

|  |
| --- |
| **General guidelines** |
| * **This practice must be done using R.** * **Use English (o español, como queráis)** * **Do not change the format of this report.** * **Add your R code at the end of this report.** * **Submit your work via Moodle:**   + **This report (pdf format)**   + **Your scripts (.m)** |

|  |
| --- |
| **Comments** |
| *Add here relevant comments related to the course of the assessment, if any* |

|  |  |
| --- | --- |
| **TASK 1** | UNDERSTANDING BOOTSTRAP Use the script AssML2\_T1.A1.R |
| *Use bootstrap in order to estimate the variance of the complexity (“number of leaves”) and accuracy of a classification tree fitted using rpart. It is obvious that these variances are very sensitive to the sample size, the larger the dataset, the lower the variance should be. There are other sources of variability such as the number of variables used as inputs, the selected complexity of the model and the inherent complexity of the problem at hand.*  *In the script AssML2\_T1.A1.R you can find the basis for completing this task. Note that using the “boot” function it is also possible to get an empirical idea of the distribution of “accuracy” or “number of leaves” by using the histogram.*  *In particular, you should analyze the influence of the size of the training set, the number the variables and the cp parameter in the variability of the number of leaves and the accuracy of the classification trees. Try with two different problems:*   * *SimDataRECTANGLEwithNoise.dat* * *SimDataRECTANGLEwithoutNoise.dat*   *Both problems have 5000 observations with 6 input variables and one output Y, with values I (inside the rectangle) or O (outside the rectangle)* | |
| **EXPLORATORY ANALYSIS**  **TASK 1** | **Include here a basic exploratory analysis of the problem.**  **Justify your answers, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

En primer lugar, cargamos los datos y aplicamos la función str y summary para conseguir un primer vistazo de las variables. Empezando por el dataset SimDataRECTANGLEwithoutNoise.dat, transformamos la variable Y a factor y usamos la función ggpairs para una mejor visualización de la distribución y correlación de los datos, obteniendo el siguiente gráfico.

Aplicación

Descripción generada automáticamente con confianza baja

Ilustración 1 ggpairs del dataset sin ruido

En este gráfico observamos que las variables X1 con X4 y X2 con X5 presentan una correlación muy alta (en torno al 0.98). Además, vemos que la variable de salida está bastante descompensada, habiendo casi el doble de ‘O’ que de ‘I’. Para evitar que esto genere problemas a la hora de entrenar los modelos que vayamos a usar, se puede obligar a cada sample del set de entrenamiento a estar completamente balanceado.

Además, se podría anticipar que las variables más importantes cuando realicemos árboles serán X1 y X2 ya que muestran cortes más limpios al representar los datos.

Siguiendo con el dataset de SimDataRECTANGLEwithNoise.dat, transformamos también la variable de salida a factor y representamos los datos con ggpairs obteniendo la siguiente gráfica.

Imagen que contiene Aplicación

Descripción generada automáticamente

Ilustración 2 ggpairs para el dataset con ruido.

Gracias a esta gráfica, vemos que este dataset se comporta igual que el anterior por lo que las conclusiones obtenidas para el dataset sin ruido serían las misma que para este. Tanto X1 con X4 como X2 con X5 presentan un alto grado de correlación, la variable de output está desbalanceada y las variables más importantes, seguramente seán X1 y X2.

|  |  |
| --- | --- |
| **QUESTION 1.1** | ***Consider the cases with and without noise, using the same small cp (e.g. 0.001) and a subset of 1000 observations with all the candidate inputs.***  **In which case the variability of the trees is larger? Why?**  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

Para realizar este ejercicio hacemos uso del ultimo script de AssML2\_t1A1 que entrenará un árbol con un cp de 0.001 y usará 1000 muestras bootstrap, devolviéndonos el número de hojas del árbol y su accuracy.

Primero aplicamos esta función al dataset con ruido obteniendo los siguientes resultados:

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | original | bias | Std. error |
| Nº hojas | 75 | -20.59 | 4.647 |
| Accuracy | 1 | -0.045 | 0.0060 |

En esta tabla observamos el sesgo y la varianza de los resultados obtenidos con las muestras bootstrap comparado con el árbol generado con todo el dataset completo. Podemos observar que es alto, cogiendo un número de hojas medias bastante menor que el original y generando una varianza alta. En cuanto al accuracy, vemos que ni el sesgo ni la varianza son altos.

Ahora mostramos los histogramas que representan la distribución de estos valores obtenidos a partir de las muestras bootstrap. En el caso del número de hojas, vemos que la distribución de estos resultados es normal, habiendo una mayoría de árboles con un número de hojas que varía entre 50 y 60, tomando un valor medio de 54.41 hojas y 0.9544 de accuracy. Esta bajada del número de hojas con respecto al original resulta normal ya que, al tener este ruido, siempre que se coja una muestra que no contemple valores de este tipo, disminuirán los cortes, como seguramente pase con el dataset sin ruido.

Gráfico, Histograma

Descripción generada automáticamente

Ilustración Distribución del número de hojas en el dataset con ruido

Gráfico, Histograma

Descripción generada automáticamente

Ilustración 4 Distribución del accuracy en el dataset con ruido

Seguidamente, ejecutamos la función con el dataset sin ruido obteniendo la siguiente tabla.

|  |  |  |  |
| --- | --- | --- | --- |
| metrics | original | bias | Std. error |
| Nº hojas | 5 | 0.5180 | 1.005 |
| Accuracy | 1 | -0.001 | 0.001 |

En esta tabla observamos el sesgo y la varianza de los resultados obtenidos con las muestras bootstrap comparado con el árbol generado con todo el dataset completo. Podemos observar que el sesgo es bajo, cogiendo un número de hojas medias prácticamente igual y estimando una varianza bastante baja. En cuanto al accuracy, vemos que tanto el sesgo como la varianza son prácticamente nulos.

Ahora mostramos los histogramas que representan la distribución de estos valores obtenidos a partir de las muestras bootstrap. En el caso del número de hojas, vemos que la distribución de estos resultados deja de ser normal, tomando prácticamente todos los árboles 5 hojas y 0.9988 de accuracy.

Gráfico, Histograma

Descripción generada automáticamente

Ilustración Distribución del accuracy en el dataset sin ruido

Gráfico, Histograma

Descripción generada automáticamente

Ilustración 5 Distribución del accuracy en el dataset sin ruido

|  |  |
| --- | --- |
| **QUESTION 1.2** | ***Consider using a subset of 1000 observations of the case with noise (all candidate inputs). Compute the variability of trees when cp is small (0.001) and large (e.g. 0.3).***  **In which case the variability of the trees is larger? Why?**  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

En el caso de usar un cp de 0.001 obtendríamos los mismos resultados que en el apartado anterior por lo que nos centramos en analizar el modelo con un cp de 0.3.

Previo a analizar los histogramas, podemos afirmar que con un cp tan alto lo que ocurrirá será que todas las muestras generarán el mismo árbol ya que en el momento en el que estén presentes las variables más importantes X1 o X2 cortará por ahí.

Al obtener los resultados de la función vemos un sesgo y varianza completamente nulo reafirmando la hipótesis inicial de que los árboles serían iguales.

Tras esto, analizamos la distribución de las hojas donde se ve que, con un grado de complejidad tan alto, todos los árboles tienen solo 1 hoja.

Gráfico

Descripción generada automáticamente

Ilustración Distribución de las hojas par un cp de 0.3

Al representar la distribución de la accuracy, obtenemos un gráfico similar al anterior y un accuracy de 0.656 para todos los árboles, siendo un valor mucho menor al de los árboles con un cp pequeño.

Gráfico

Descripción generada automáticamente

Ilustración Distribución del accuracy para un cp de 0.3

|  |  |
| --- | --- |
| **QUESTION 1.3** | ***Using the dataset SimDataRECTANGLEwithNoise and setting a small cp. Compare the results with 500, 1000 and 5000 observations.***  **In which case the variability of the trees is larger? Is the result you expected?**  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

En el caso de 1000 observaciones obtenemos los mismos resultados que en el apartado 1, por lo que nos centraremos en 500 y 5000 observaciones.

Para un N de 500, obtenemos los siguientes resultados.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | original | bias | Std. error |
| Nº hojas | 43 | -12.369 | 3.504 |
| Accuracy | 1 | -0.050 | 0.008 |

En esta tabla observamos el sesgo y la varianza de los resultados obtenidos con las muestras bootstrap comparado con el árbol generado con todo el dataset completo. Podemos observar que el sesgo es alto, cogiendo un número de hojas medias menor al original y estimando una varianza alta también. En cuanto al accuracy, vemos que tanto el sesgo como la varianza son prácticamente nulos.

Ahora mostramos los histogramas que representan la distribución de estos valores obtenidos a partir de las muestras bootstrap. En el caso del número de hojas, vemos que la distribución de estos resultados es normal, tomando una media de hojas de 30.631 y 0.9494 de accuracy.

Gráfico, Histograma

Descripción generada automáticamente

Ilustración Distribución del número de hojas para un R de 500

Gráfico, Histograma

Descripción generada automáticamente

Ilustración Distribución del accuracy para un R de 500

En el caso de usar un N de 5000, obtenemos los siguientes resultados.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | original | bias | Std. error |
| Nº hojas | 48 | 29.369 | 10.076 |
| Accuracy | 0.9388 | -0.005 | 0.002 |

En esta tabla observamos el sesgo y la varianza de los resultados obtenidos con las muestras bootstrap comparado con el árbol generado con todo el dataset completo. Podemos observar que el sesgo es muy alto, cogiendo un número de hojas medias bastante mayor que el original y generando una varianza alta. En cuanto al accuracy, vemos que, tanto sesgo como varianza, son nulos.

Ahora mostramos los histogramas que representan la distribución de estos valores obtenidos a partir de las muestras bootstrap. En el caso del número de hojas, vemos que la distribución de estos resultados es normal, habiendo una mayoría de árboles con un número de hojas que varía entre 50 y 60, tomando un valor medio de 54.41 hojas y 0.9544 de accuracy. Esta bajada del número de hojas con respecto al original resulta normal ya que, al tener este ruido, siempre que se coja una muestra que no contemple valores de este tipo, disminuirán los cortes, como seguramente pase con el dataset sin ruido.

Gráfico, Histograma

Descripción generada automáticamente

Ilustración Distribución del número de hojas para un R de 5000

Gráfico, Histograma

Descripción generada automáticamente

Ilustración Distribución del accuracy para un R de 5000

Tras ver los resultados individuales al variar el tamaño de la muestra, vemos que, claramente, este aumento es directamente proporcional a la varianza y al sesgo, siendo el dataset con 500 muestras el que tiene estos valores más pequeños, seguidos del de 100 muestras y, finalmente el de 5000.

|  |  |
| --- | --- |
| **QUESTION 1.4** | ***Using the dataset SimDataRECTANGLEwithoutNoise, with small cp and a subset of 1000 observations, consider two cases:***   * ***Case A: Build the trees only with candidate variables X1 and X2*** * ***Case B: Build the trees only with candidate variables X4 and X5***   **In which case the variability of the trees is larger? Why?**  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

En primer lugar, trataremos el caso A obteniendo los siguientes resultados.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | original | bias | Std. error |
| Nº hojas | 5 | 0.015 | 0.211 |
| Accuracy | 1 | -0.0004 | 0.0004 |

En esta tabla observamos el sesgo y la varianza de los resultados obtenidos con las muestras bootstrap comparado con el árbol generado con todo el dataset completo. Podemos observar que el sesgo es prácticamente nulo, cogiendo un número de hojas medias igual y estimando una varianza muy baja. En cuanto al accuracy, vemos que tanto el sesgo como la varianza son prácticamente nulos.

Ahora mostramos los histogramas que representan la distribución de estos valores obtenidos a partir de las muestras bootstrap. En el caso del número de hojas, vemos que la distribución de estos resultados deja de ser normal, tomando todos los árboles 5 hojas y 0.9995 de accuracy.

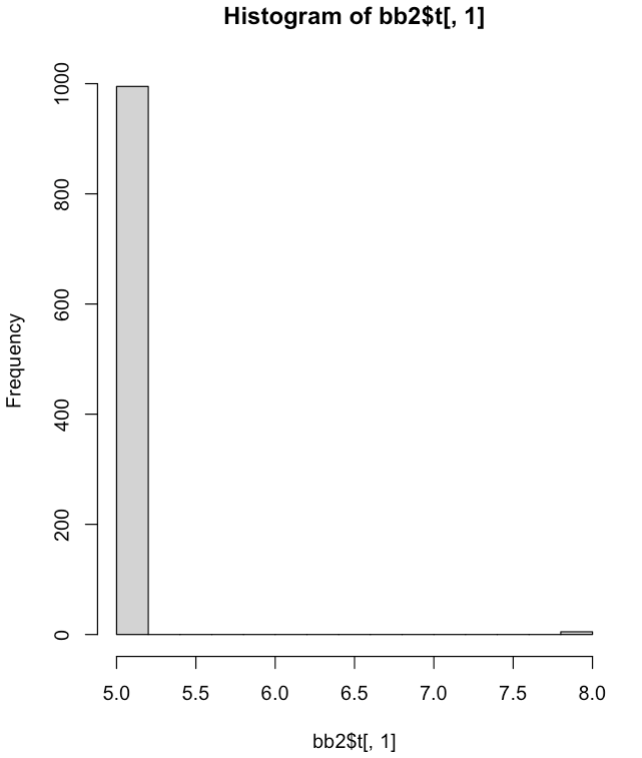


Ilustración Distribución del número de hojas para el dataset sin ruido usando X1 y X2

Gráfico, Histograma

Descripción generada automáticamente

Ilustración Distribución del accuracy para el dataset sin ruido usando X1 y X2

En lo referente al caso B, obtenemos los siguientes resultados.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | original | bias | Std. error |
| Nº hojas | 28 | 41.084 | 11.796 |
| Accuracy | 0.9134 | 0.001 | 0.002 |

En esta tabla observamos el sesgo y la varianza de los resultados obtenidos con las muestras bootstrap comparado con el árbol generado con todo el dataset completo. Podemos observar que el sesgo es altísimo, estimando una varianza muy alta. En cuanto al accuracy, vemos que tanto el sesgo como la varianza son prácticamente nulos.

Ahora mostramos los histogramas que representan la distribución de estos valores obtenidos a partir de las muestras bootstrap. En el caso del número de hojas, vemos que la distribución de estos resultados deja de ser normal, tomando un número de hojas medio de 69.084 y 0.9144 de accuracy.

Gráfico, Histograma

Descripción generada automáticamente

Ilustración Distribución del número de hojas para el dataset sin ruido usando X4 y X5

Gráfico, Histograma

Descripción generada automáticamente

Ilustración 13 Distribución del accuracy para el dataset sin ruido usando X3 y X4

Tras este análisis individual, vemos que, aún presentado una correlación altísima con X1 y X2, tanto el sesgo como la varianza, al usar X4 y X5 empeora muchísimo.

|  |  |
| --- | --- |
| **CONCLUSIONS**  **TASK 1** | **Summarize here your conclusions resulting from the previous experiments. In particular, discuss the impact of the sample size, the *cp* parameter and the number of input variables in the accuracy and complexity of the model.** |

Las conclusiones obtenidas tras el desarrollo de los apartados anteriores serían las siguientes:

* El tamaño de la muestra influye de manera directa en el sesgo y la varianza aumentándolas considerablemente.
* El cp escogido debe ser lo suficientemente pequeño para poder estimar, de forma eficiente, la varianza con bootstrap ya que cuantos menos complejos sean, más similares serán siempre que estén presentes las variables más importantes.
* El número de variables dentro del modelo afecta enormemente si se evitan las variables más importantes ya que estos árboles serían muy distintos entre ellos.

|  |  |
| --- | --- |
| **TASK 2** | USING BASIC TREES, BAGGED TREES, RANDOM FORESTS AND GRADIENT BOOSTING Use the script AssML2\_T2.A1.R |
| *Fit your best set of models for the problem SimDataCIRCLE, using the dataset in SimDataCIRCLETR.dat. After that, check the “generalization error” of all those models using the dataset SimDataCIRCLETS.dat, avoiding refitting models as result of this final comparison.*  *This problem has 6 input variables and one output Y, with values I (inside the circle), or O (outside the circle). The TR dataset has 500 observations whereas the TS has 10000.*  *Note that Y has been obtained from the inputs in a pure deterministic way, i.e. given the inputs, the output is perfectly determined.* | |
| **EXPLORATORY ANALYSIS**  **TASK 2** | **Include here a basic exploratory analysis of the problem.**  **Justify your answers, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

En primer lugar, cargamos los datos de train y aplicamos la función str y summary para conseguir un primer vistazo de las variables. Gracias a ello, observamos que la variable de salida es de tipo chr, por lo que transformamos la variable Y a factor y usamos la función ggpairs para una mejor visualización de la distribución y correlación de los datos, obteniendo el siguiente gráfico.

Imagen que contiene Diagrama

Descripción generada automáticamente

Ilustración 14 ggpairs del dataset de círculo

En este gráfico observamos que las variables X1 con X4 y X2 con X5 presentan una correlación muy alta (en torno al 0.98). Además, vemos que la variable de salida no está demasiado descompensada, a diferencia de lo que ocurría en el primer dataset. Para corregir esto, que puede hacer que genere problemas a la hora de entrenar los modelos que vayamos a usar, se puede obligar a cada sample del set de entrenamiento a estar completamente balanceado.

Además, se podría anticipar que las variables más importantes cuando realicemos árboles serán X1 y X2 ya que muestran cortes más limpios al representar los datos.

En segundo lugar, cargamos los datos de train y aplicamos la función str y summary para conseguir un primer vistazo de las variables. Gracias a ello, observamos que la variable de salida es de tipo chr, por lo que transformamos la variable Y a factor y usamos la función ggpairs para una mejor visualización de la distribución y correlación de los datos, obteniendo el siguiente gráfico.

Forma

Descripción generada automáticamente con confianza baja

Ilustración 15 ggpairs del dataset de círculo

En este gráfico observamos que las variables X1 con X4 y X2 con X5 presentan una correlación muy alta (en torno al 0.98). Además, vemos que la variable de salida no está demasiado descompensada,a diferencia de lo que ocurría en el primer dataset. Para corregir esto, que puede hacer que genere problemas a la hora de entrenar los modelos que vayamos a usar, se puede obligar a cada sample del set de entrenamiento a estar completamente balanceado.

Además, se podría anticipar que las variables más importantes cuando realicemos árboles serán X1 y X2 ya que muestran cortes más limpios al representar los datos.

|  |  |
| --- | --- |
| **QUESTION 2.1** | ***Summarize the accuracy of the different models in the TR and TS datasets. Include some measure of complexity of the resultant model.***  ***What is the best model? Consider both complexity and accuracy***  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

En primer lugar, ajustamos un árbol de decisión básico usando un tunegrid variable entre 0, 0.01 y 0.001 y como control usamos un cv con un k\_fold de 10 y se obtienen un accuracy de 0.858 y un kappa de 0.714, viendo que el cp óptimo es 0.1. Además, en la siguiente tabla vemos los resultados del train y el test.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Set | Accuracy | 95% CI | Kappa | Sensitivity | Specificity |
| Train | 0.908 | (0.879,0.931) | 0.815 | 0.893 | 0.925 |
| Test | 0.870 | (0.864,0.877) | 0.742 | 0.847 | 0.894 |

Además, vemos en la siguiente gráfica la importancia de cada variable.

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

Ilustración VarImp del árbol básico

Tras esto, entrenamos un árbol con bagging con el mismo control, obteniendo un accuracy de 0.894 y un kappa de 0.786. Además, en esta tabla se observa una gran diferencia entre los resultados del train y test por lo que estamos ante un caso de overfitting.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Set | Accuracy | 95% CI | Kappa | Sensitivity | Specificity |
| Train | 0.994 | (0.982,0.998) | 0.987 | 0.996 | 0.991 |
| Test | 0.906 | (0.900,0.912) | 0.813 | 0.903 | 0.910 |

Además, vemos en la siguiente gráfica la importancia de cada variable.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Ilustración 15 VarImp del árbol con bagging

Ahora aplicamos un algoritmo de random forest con 100 árboles y un m óptimo de 3 obteniendo un accuracy de 0.902 y un Kappa de 0.802. En cuanto al train y test, en este caso ocurre igual que en el anterior, hay indicios claros de overfitting debido a que el accuracy del train dista mucho de la del test.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Set | Accuracy | 95% CI | Kappa | Sensitivity | Specificity |
| Train | 1 | (0.9926, 1) | 1 | 1.000 | 1.000 |
| Test | 0.914 | (0.908,0.919) | 0.828 | 0.895 | 0.933 |

Además, vemos en la siguiente gráfica la importancia de cada variable.

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

Ilustración 16 VarImp del random forest

Por último, entrenamos un modelo de Gradient Boosting utilizando para el modelo 1000 árboles, una profundidad de árbol de 1 (tocones), una tasa de aprendizaje por árbol de 0.1. Usando estos valores, se obtiene un accuracy de 0.914 y un Kappa de 0.8263556. Vemos que para el train y test vuelve a ocurrir lo que, en los dos casos anteriores, teniendo el train un accuracy de 1 y el test un accuracy lo suficientemente menor como para admitir overfitting.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Set | Accuracy | 95% CI | Kappa | Sensitivity | Specificity |
| Train | 1 | (0.9926, 1) | 1 | 1.000 | 1.000 |
| Test | 0.954 | (0.950,0.958) | 0.909 | 0.956 | 0.953 |

Tras este análisis individual, mostramos en la siguiente tabla el accuracy y kappa obtenidos en cada modelo con el TS, sus curvas ROC y sus curvas de calibración. En ellas vemos que los modelos van mejorando progresivamente, siendo el peor el árbol básico, seguido del árbol con Bagging, el Random forest y el Gradient Boosting. Esto resulta normal ya que, tal y como vimos en clase, Bagging mejoraba el árbol normal mediante bootstrap, RF mejoraba Bagging eliminando la correlación entre los árboles y Gradient Boosting era mejor que RF debido a que en cada iteración se da mayor prioridad a los valores mal clasificados.

|  |  |  |
| --- | --- | --- |
| Modelo | Accuracy | Kappa |
| Basic Tree | 0.870 | 0.742 |
| Bagged Tree | 0.904 | 0.813 |
| Random forest | 0.914 | 0.828 |
| Gradient Boosting | 0.954 | 0.909 |

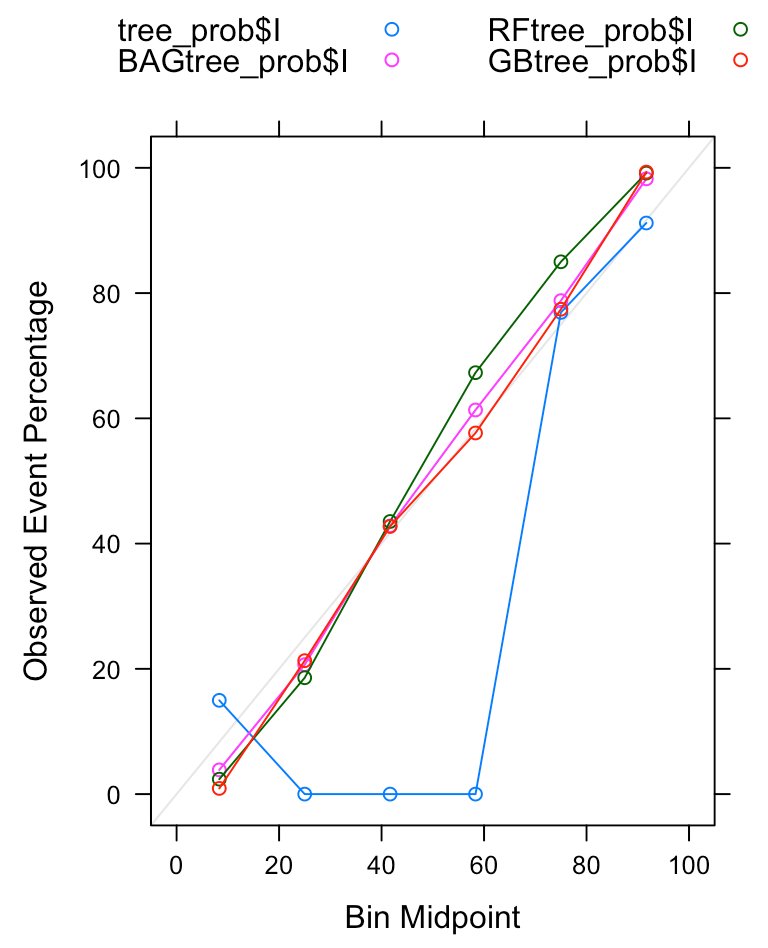
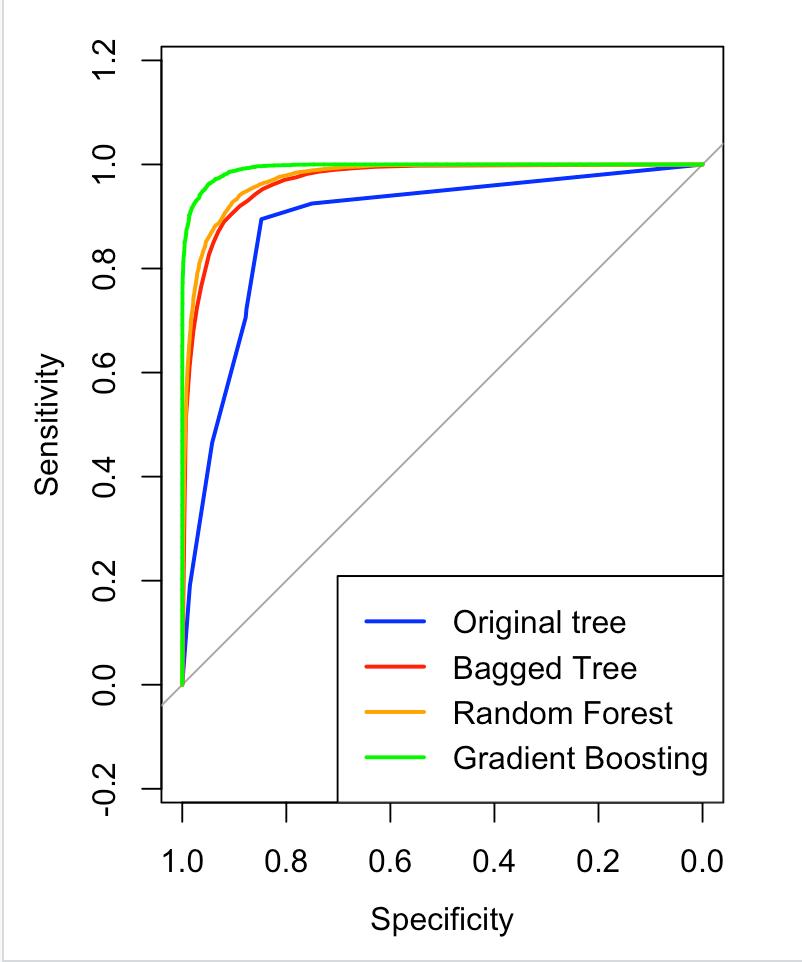


Ilustración Curvas de calibración de los modelos.



|  |  |
| --- | --- |
| **QUESTION 2.2** | ***What are the two more important variables according to the different models? Is X3 more important than X6?***  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

Como se veía en los gráficos anteriores de las variables más importantes, las más determinantes a la hora de realizar el árbol son X1 y X2. Además, en todo momento X3 es mucho más importante que X6, siendo esta última variable la menos determinante.

|  |  |
| --- | --- |
| **QUESTION 2.3** | ***Create some new variables derived from the raw ones. The idea is to build some variable/s that simplify the classification process using trees.***  ***Create a simple tree and compare with the previous one. Is it better? Why?***  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

*(Add here your analysis)*

|  |  |
| --- | --- |
| **TASK 3** | USING GENETIC ALGORITHMS Use the script AssML2\_T3.A1.R |
| *Use a genetic algorithm to estimate the two parameters defining a straight line (i.e. a simple linear regression model).*  *In the script AssML2\_T3.A1.R you can find the basis for completing this task. Note that you can use the R function “lm” in order to fit a regression line to the data by ordinary least squared (OLS), see: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/lm*  *In particular, you should analyze the obtained results using GAs, comparing with the theoretical target line and the line obtained by OLS. Impact of main hyperparameters of the GA in the results are also very interesting.* | |
| **EXPLORATORY ANALYSIS**  **TASK 3** | **Include here the parameters defining the true underlying straight line of the problem (slope and intercept), as well as the estimated parameters using OLS regression with the R function lm.**  **Justify your answers, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

El dataset usado en este apartado es creado a partir de la generación de puntos a lo largo de una recta de pendiente 1 y un intercept de 5. Además, a estos puntos se les añade un ruido obteniendo lo siguiente:

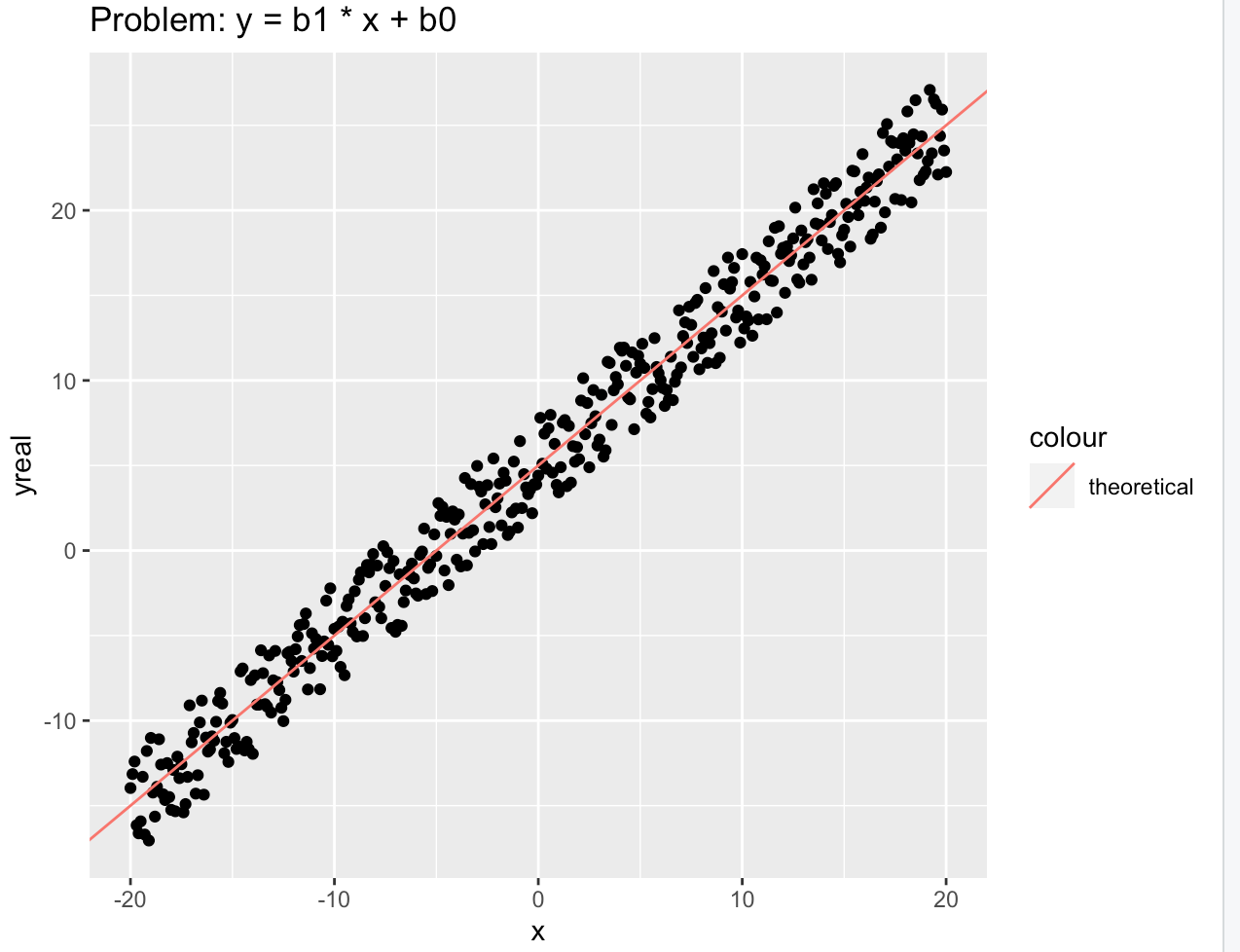


Ilustración Dataset apartado 3

Tras esto, se aplica una regresión lineal para estimar, tras el ruido, estos parámetros de la recta, obteniendo una pendiente de 0.998 y un intercept de 4.949, prácticamente similar a la inicial. De hecho, en la siguiente gráfica vemos que la estimación teórica y la recta con la regresión lineal se superponen.

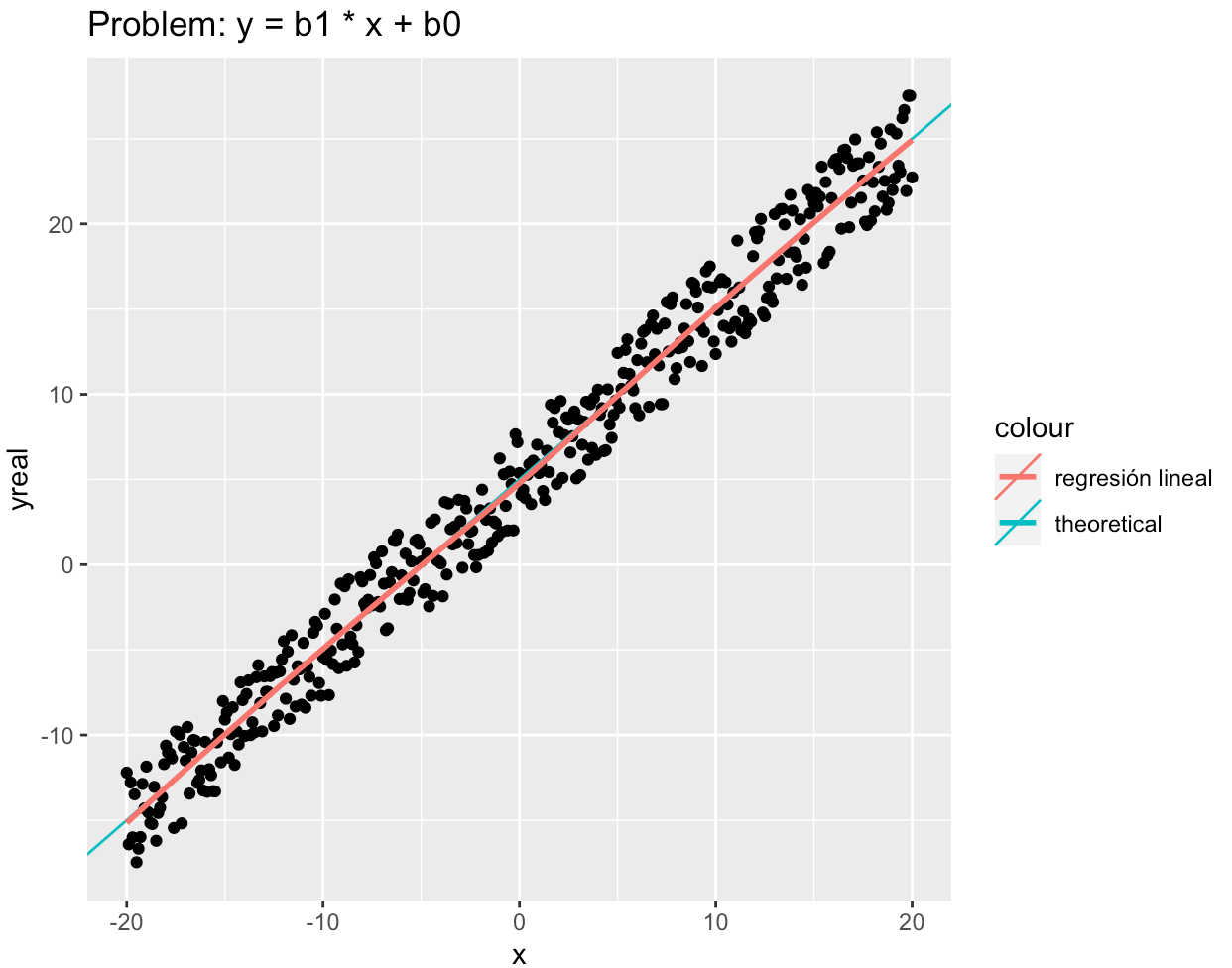


Ilustración 19 Estimación teórica y regresión lineal

|  |  |
| --- | --- |
| **QUESTION 3.1** | ***Fit the regression line using a GA. Set the hyperparameters to obtain a good result in a reasonable number of iterations. Include in the answer your set of GA’s parameters and the best linear model obtained using GA (summary(GA)).***  ***Show a plot with the individuals in a generation where the population is very different and in another generation where it is more stable.***  ***Show the fitness evolution. Discuss why your set of GA params is reasonable.***  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

Para el entrenamiento de esta regresión lineal mediante GA usamos 100 iteraciones, con una población de 100 individuos, en la cual cada individuo son una pareja de x1 y x2. Los valores escogidos para la probabilidad de cruce y la mutación, son 0.8 y 0.1 respectivamente. Esto se debe a que, en el caso de la mutación, un número mayor provocaría que los individuos de generaciones posteriores distasen mucho de los padres y al haber mayor probabilidad de cruce entre padres con mejor fitness, alteraría completamente el valor de este fitness, pudiendo empeorarlo notablemente.

En el caso de la probabilidad de cruce, 0.8 es un valor estándar ya que al disminuirlo obtendríamos generaciones demasiado iguales a las anteriores y habría muy poca mejora.

Tras entrenar la regresión lineal obtenemos lo siguiente.

|  |  |  |
| --- | --- | --- |
| Fitness Value | X1 value | X2 value |
| -3.207 | 1.001 | 4.969 |

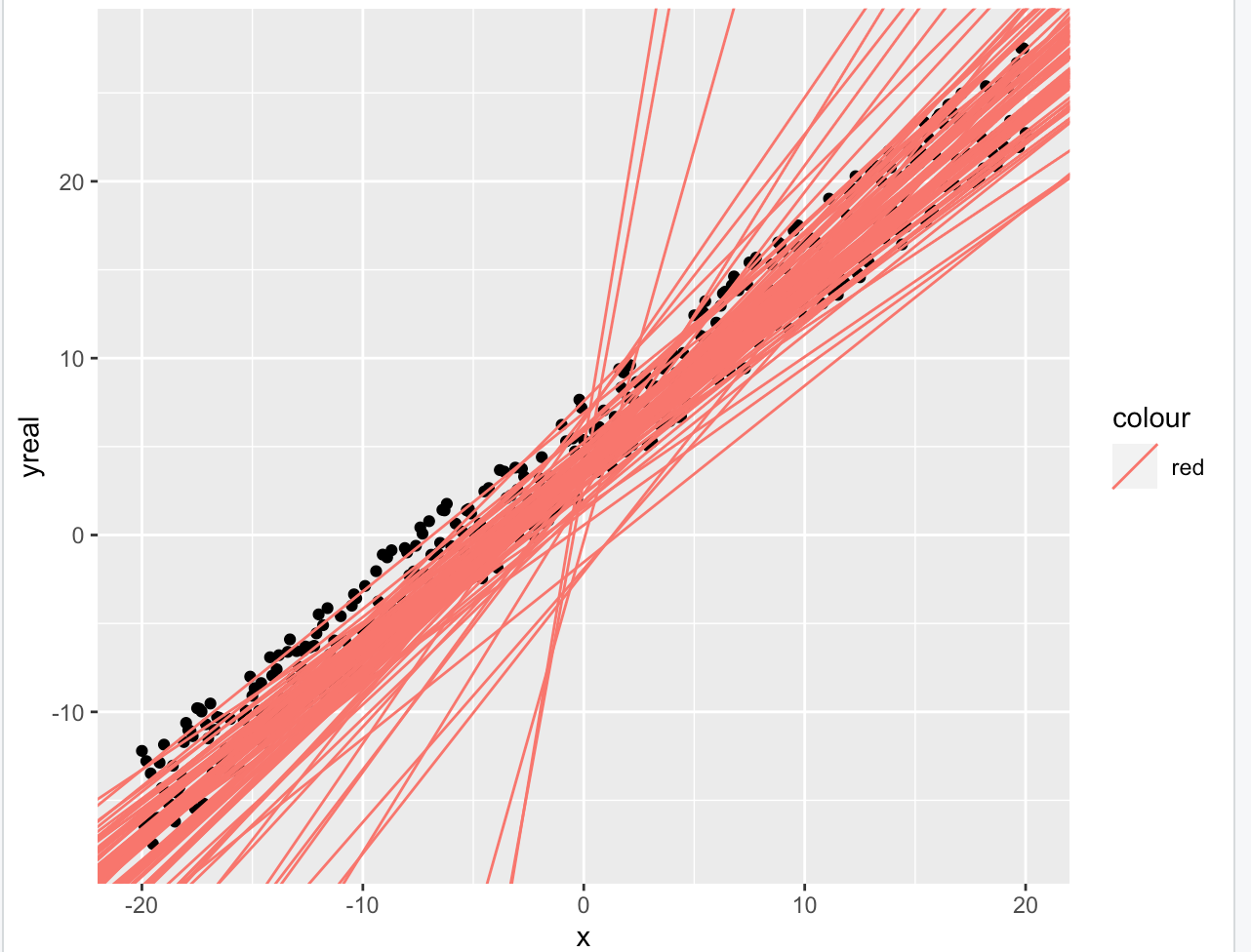


Ilustración 20 Regresión lineal con GA

Tras ver el resultado, analizamos la evolución del fitness mediante la siguiente gráfica.

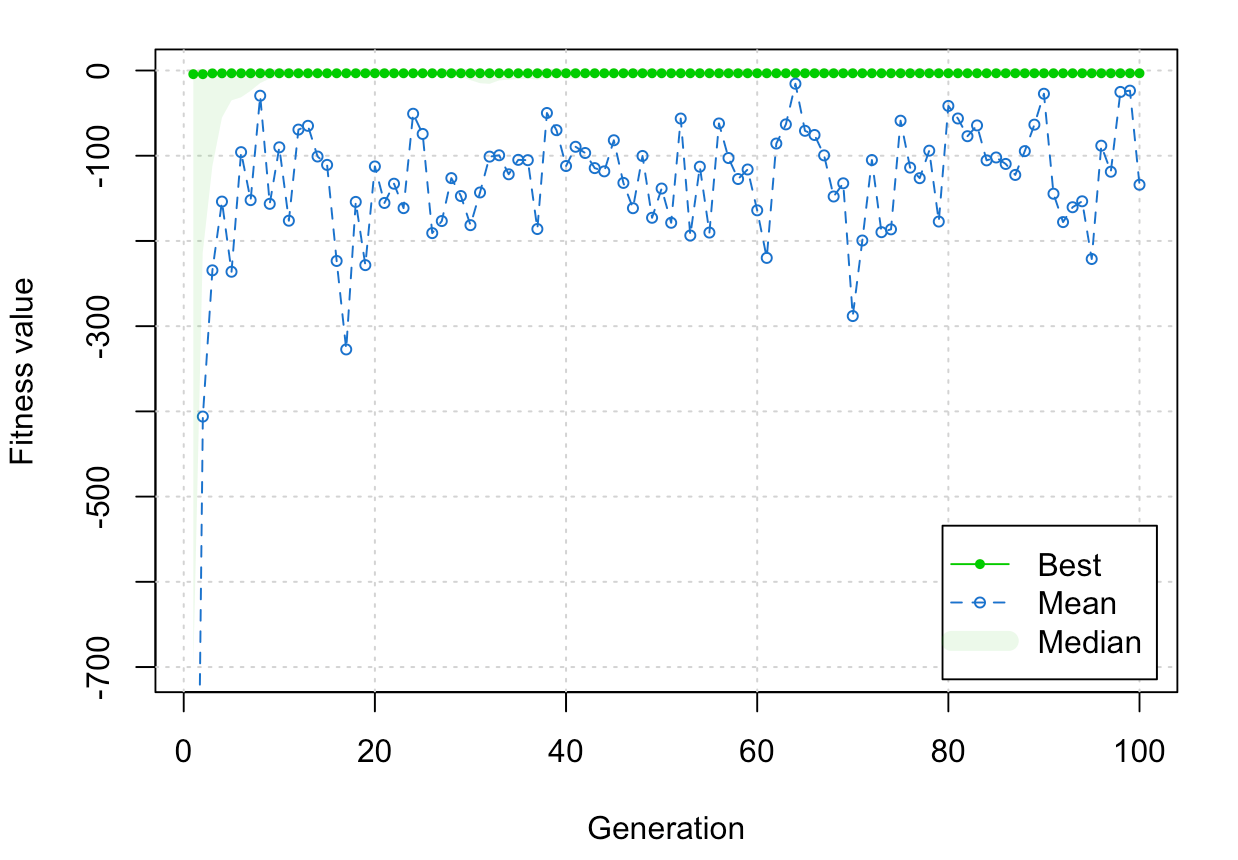


Ilustración 21 Evolución del fitness

En la gráfica anterior observamos que en las primeras generaciones el valor del fitness es muy bajo, estabilizándose, la mediana, a partir de la decimoquinta generación.

|  |  |
| --- | --- |
| **QUESTION 3.2** | ***Compare the OLS model and the GA model.***  ***What is the best model? Why?***  ***Quantify in only one number the quality of each solution and, according to such as quantity, explain why one model is better than the other is.***  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

Tras haber entrenado los modelos anteriores, realizamos la comparación observando el RMSE. Para el caso de la regresión lineal con GA, tomamos el valor del fitness (3.201), ya que hemos estipulado que la función fitness sea el menor RMSE.

En el caso de la primera regresión lineal, obtuvimos que el RMSE es 3.065, por lo que afirmamos que el mejor modelo es la regresión lineal sin GA. Esta situación es común en conjuntos de datos muy sencillos en los que el ajuste no requiere de un modelo demasiado complejo.

|  |  |
| --- | --- |
| **QUESTION 3.3** | ***Analyze the importance of the mutation probability (pmutation) in this problem. Compare three cases: no mutation, pmutation = 0.1 and pmutation = 0.8***  ***Describe the impact of this probability in the result. Discuss the effect of using pmutation = 0 and pmutation = 0.8***  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

Para un valor de mutación 0.1 se han realizado los apartados anteriores por lo que nos centraremos en un valor de mutación de 0.8.

Como vemos en la siguiente imagen, la mayoría de rectas de regresión no convergen, esto es debido a lo explicado en el primer apartado; un mayor valor de mutación generará, en cada población, parejas muy distintas a las generaciones anteriores, perdiendo el buen fitness.

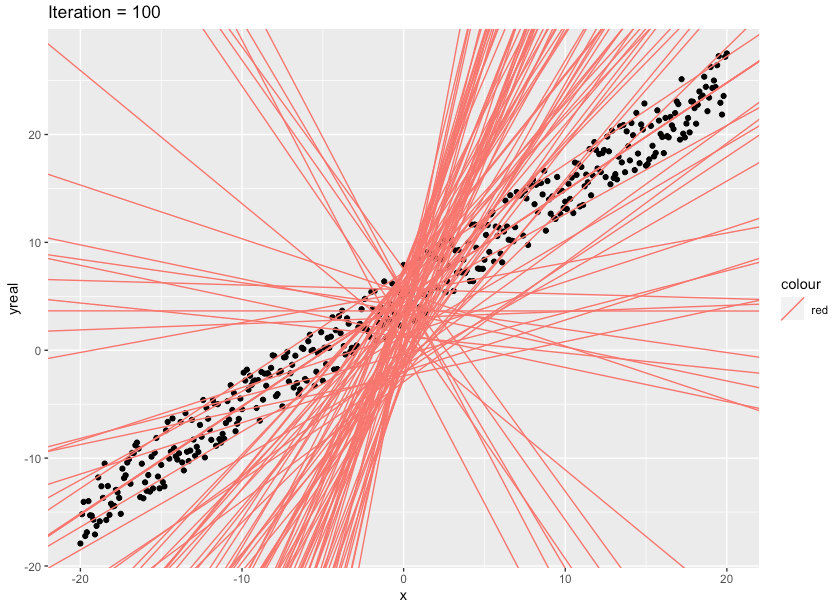


Ilustración Regresión lineal con GA y pmutation = 0.8

Además, en esta segunda imagen, observamos que tanto la media, como la mediana en cada generación tiene un valor de fitness mucho menor al caso de pmutation = 0.1 y mucho más variable, por lo que el error será mucho mayor. De hecho, vemos como en ningún momento se estabiliza la mediana.



Ilustración Análisi del fitness con un pmutation = 0.8

Para el caso de un pmutation = 0, cabría esperar que en el momento en el que encuentre un fitness mínimo, se mantenga ahí y no cambien las parejas en las generaciones posteriores. En casos en los que el problema es muy sencillo, es posible que encuentre la solución muy rápido, aunque esto no quiere decir que sea mejor que el caso de pmutation = 0.1.

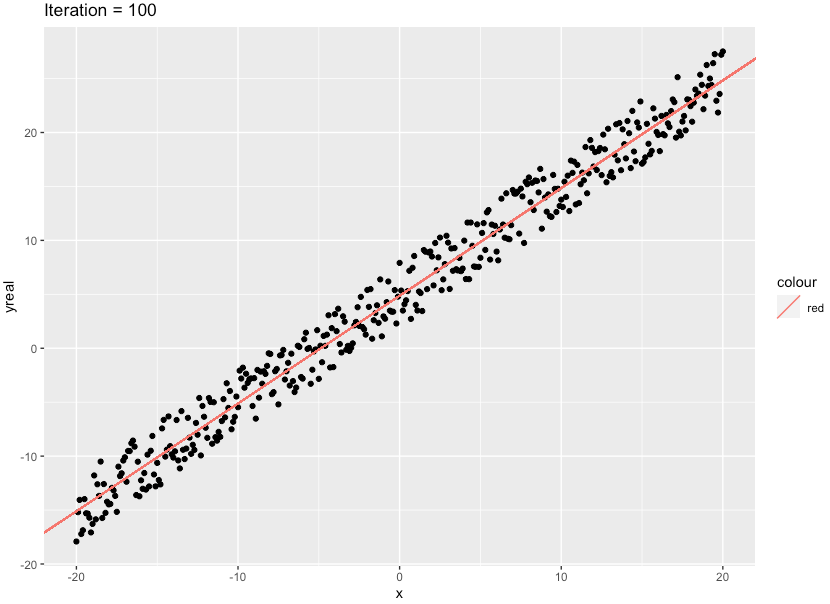


Ilustración Regresión lineal con GA y un pmutation nulo

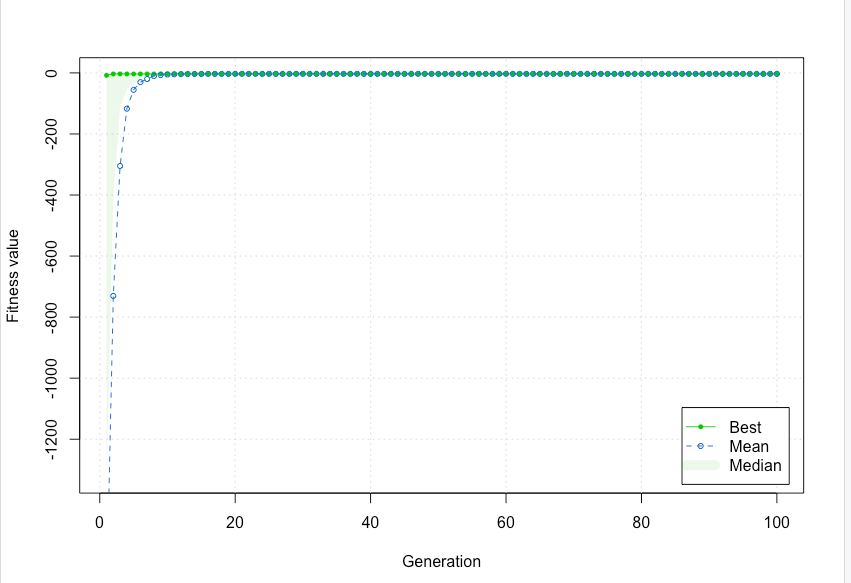


Ilustración 25 Análisi del fitness con un pmutation nulo

Como vemos en estas dos imágenes, se confirma la teoría inicial de que encuentra el valor con mejor fitness muy rápido y se mantiene ahí sin variar el resto de generaciones.

|  |  |
| --- | --- |
| **QUESTION 3.4** | ***Change your GA implementation to fit the linear model using, instead the MSE, the MAE.***  ***Is this new GA model better than the previous GA model? Why?***  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

Tras cambiar la función de fitness de MSE a MAE entrenamos la regresión lineal con GA obteniendo los siguientes resultados.

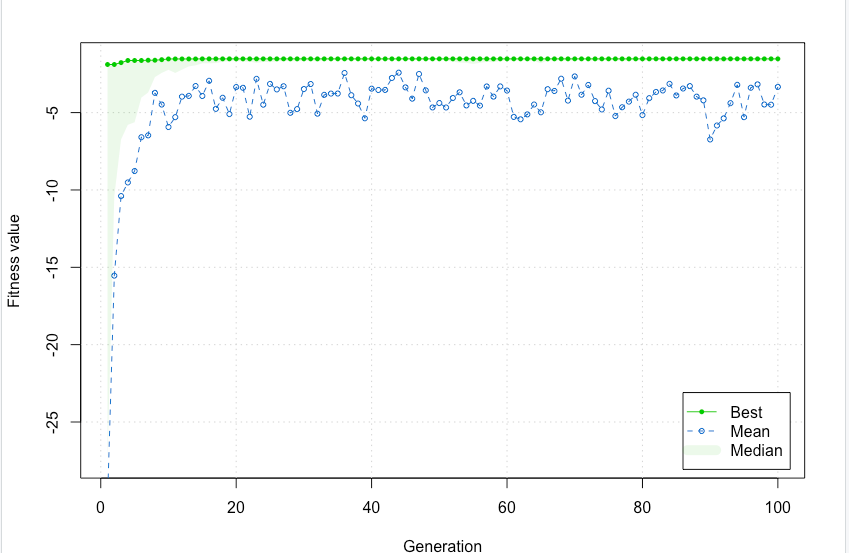


Ilustración 26 Análisis del fitness con MAE

Tanto en este caso como en el que usamos MSE, vemos que la mediana de fitness se estabiliza muy rápido en el mejor valor posible, sin embargo, la media en este caso es mucho menos variantes y además se obtiene un fitness menor (-1.523), por lo que podemos afirmar que el uso de MAE, en este caso, es mejor.

|  |  |
| --- | --- |
| **QUESTION 3.5** | ***Using your GA implementation of question 3.1, modify now the lower and upper limits in the GA from*** “***lower = c(-3, -3), upper = c(8,8)” to “lower = c(-3, -3), upper = c(3,0)”***  ***Is this new GA model better than the previous GA model of question 3.1? Why?***  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

Al alterar ese valor lo que hacemos es establecer el máximo en el origen y una pendiente en 3 lo que conlleva, tal y como se ve en la gráfica, a que exista un desplazamiento hacia la derecha, pasando todas las rectas por el origen. Además, gracias a esta gráfica podemos sar por hecho que este modelo es peor al del apartado anterior.

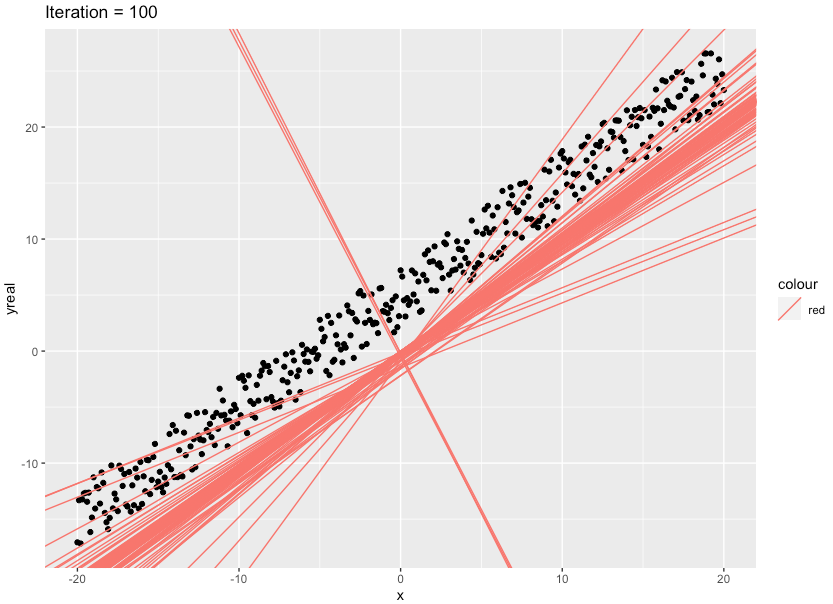


Ilustración 27 Regresión lineal con GA con un maximo de (3,0)

|  |  |
| --- | --- |
| **QUESTION 3.6** | ***Using your GA implementation of question 3.1, modify now the lower and upper limits in the GA from*** “***lower = c(-3, -3), upper = c(8,8)” to “lower = c(-3, -3), upper = c(0,8)”***  ***Is this new GA model better than the previous GA model of question 3.1? Why?***  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

A diferencia del apartado anterior, al obligar a que el límite de la pendiente sea la ordenada, la mayor parte de valores serán una recta horizontal y, el resto, tendrán pendiente negativa. Cabe destacar que ese modelo será mucho peor que el del apartado 3.1.

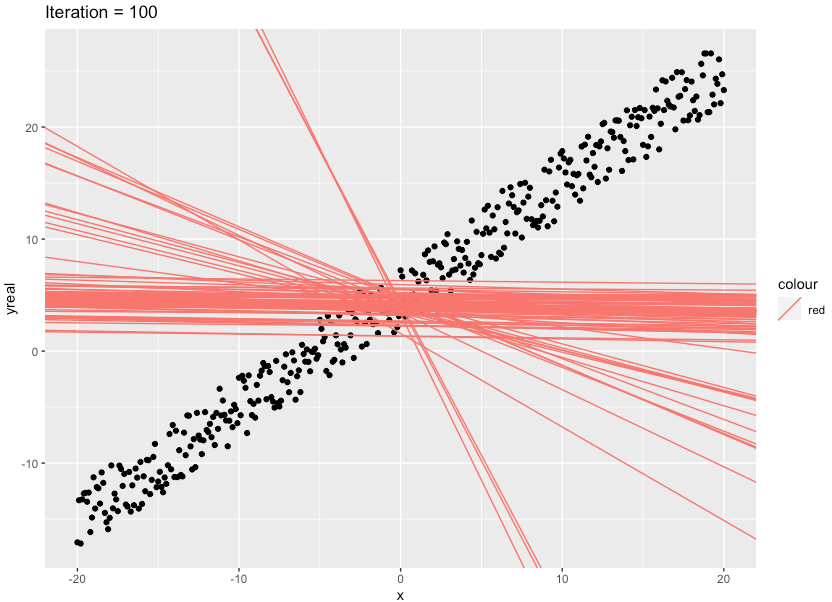


Ilustración Regresión lineal con GA con un maximo de (0,8)

|  |  |
| --- | --- |
| **QUESTION 3.7** | ***Improve (if possible) your solution of question 3.1 using Hybrid GA (set the optional argument optim = TRUE)***  ***What is the best model? Why?***  **Justify your answer, including relevant information that supports your reasoning (e.g. graphs or tables from your data analysis with R)** |

Finalmente, en este apartado aplicamos un GA híbrido, que consiste en ir acotando los valores de búsqueda a medida que vamos encontrando mínimos de la función del fitness En este caso al entrenar el modelo vemos que tras pocas generaciones mantiene la búsqueda en la región con fitness -2.895, siendo finalmente este valor el mejor fitness posible.

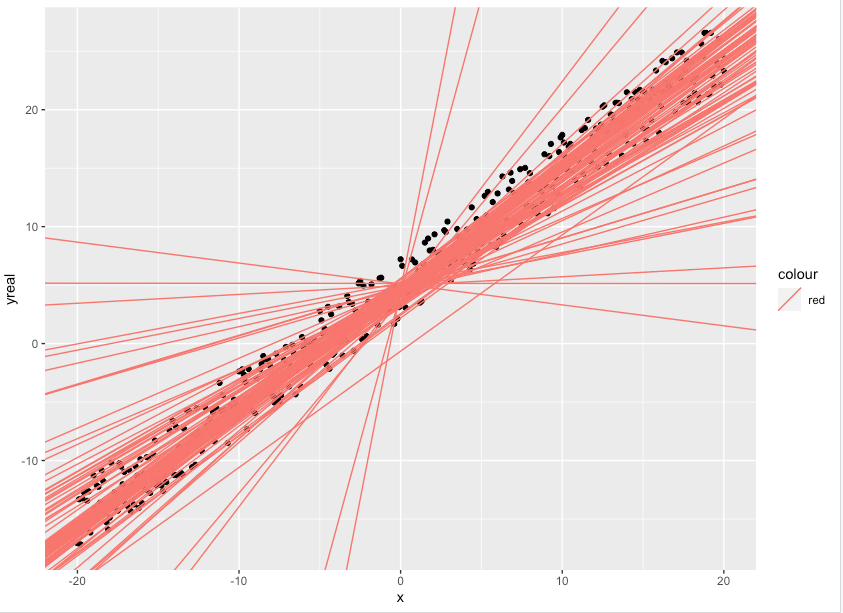


Ilustración Regresión lineal con GA híbrido

Además, en la siguiente gráfica vemos como la solución es muy similar a la del primer apartado, necesitando muy pocas generaciones para tener una mediana estable en el mejor valor del fitness y una media poco variable. En definitiva, ambos modelos son muy similares pero este último presenta un MSE menor por lo que afirmamos que es mejor.

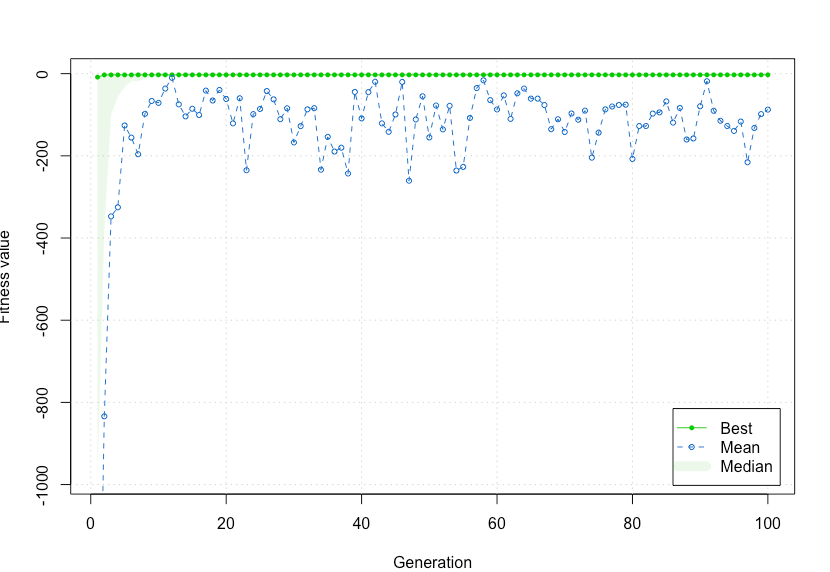


Ilustración Análisis del fitness con GA híbrido

|  |  |
| --- | --- |
| **R CODE** | ***Include here your R code for each task***  ***Justify the text for easy reading*** |

TASK 1

#################################################################################

############## TASK1-ASSESSMENT 1 ENSEMBLE METHODS ############################

############## MACHINE LEARNING II ############################

############## ESU Feb-2022 ############################

#################################################################################

## Set working directory --------------------------------------------------------

getwd()

## Load libraries ---------------------------------------------------------------

library(boot)

library(caret)

library(ggplot2)

library(GGally)

library(MLTools)

library(rpart)

library(randomForest)

## remove all epxisting variables--------------------------------------------------

rm(list=ls())

#################################################################################

# LOAD DATASETS

#################################################################################

#

frectALL = read.table("./Assignment\_1/Data/SimDataRECTANGLEwithoutNoise.dat", sep = "", header = TRUE, stringsAsFactors = FALSE)

frectALL$Y = as.factor(frectALL$Y)

str(frectALL)

summary(frectALL)

ggplot(frectALL)+geom\_point(aes(x=X1,y=X2,color=Y))

frectNoiseALL = read.table("./Assignment\_1/Data/SimDataRECTANGLEwithNoise.dat", sep = "", header = TRUE, stringsAsFactors = FALSE)

frectNoiseALL$Y = as.factor(frectNoiseALL$Y)

str(frectNoiseALL)

summary(frectNoiseALL)

ggplot(frectNoiseALL)+geom\_point(aes(x=X1,y=X2,color=Y))

## Exploratory analysis -------------------------------------------------------------------------------------

#Exploratory analysis for rectangle without noise

ggpairs(frectALL, aes(colour = Y, alpha = 0.4))

#Exploratory analysis for rectangle with noise

ggpairs(frectNoiseALL, aes(colour = Y, alpha = 0.4))

#################################################################################

# SELECT SUBSET OF DATA

#################################################################################

# set number of observations

N=1000

#sampling

indexN=sample(c(1:5000),N,replace = FALSE)

frectTR = frectALL[indexN,]

ggplot(frectTR)+geom\_point(aes(x=X1,y=X2,color=Y))

indexN=sample(c(1:5000),N,replace = FALSE)

frectNoiseTR = frectNoiseALL[indexN,]

ggpairs(frectNoiseTR, aes(colour = Y, alpha = 0.4))

ggplot(frectNoiseTR)+geom\_point(aes(x=X1,y=X2,color=Y))

#################################################################################

# TASK 1: BOOTSTRAP

#################################################################################

# One of the great advantages of the bootstrap approach is that it can be

# applied in almost all situations. No complicated mathematical calculations

# are required. Performing a bootstrap analysis in R entails only two steps.

# First, we must create a function that computes the statistic of interest.

# Second, we use the boot() function, which is part of the boot library, to

# boot() perform the bootstrap by repeatedly sampling observations from the data

# set with replacement.

bootComplexityAccuracyTree.fn=function (data, index){

formula = Y ~ X4+X5

cost = 0.001

# fit with subset of data given by index !!!

tree.fit <- rpart(formula, data=data[index,], method="class", control=rpart.control(minsplit=1, cp=cost) )

#plot(tree.fit)

# Predict with the data (all)

yest = predict(tree.fit, type="class", newdata = data) # predict Y with all dataset

cm <- confusionMatrix(data = yest, # predict, all dataset

reference = data$Y, # Real observations, all dataset

positive = "I") # Class labeled as Positive

accuracy <- cm$overall[1] # get the accuracy from the confusion matrix ( ALL TR SET !!!!! )

nleaves <- sum(tree.fit$frame$var=="<leaf>") # number of leaves (an idea of the complexity)

return (c(nleaves,accuracy))

}

#Check the function

bootComplexityAccuracyTree.fn(frectNoiseTR,1:100)

# Produce R bootstrap estimates for the mean and the variance of y

bb = boot(frectNoiseTR, bootComplexityAccuracyTree.fn,R=1000)

bb

#Histogramas, media de hojas y desviación del dataset con ruido

hist(bb$t[,1])

hist(bb$t[,2])

bb$t[,1]

mean(bb$t[,1])

sd(bb$t[,1])

mean(bb$t[,2])

sd(bb$t[,1])

#Histogramas, media de hojas y desviación del dataset sin ruido

bb2 = boot(frectTR, bootComplexityAccuracyTree.fn,R=1000)

bb2

hist(bb2$t[,1])

hist(bb2$t[,2])

mean(bb2$t[,1])

mean(bb2$t[,2])

var(bb$t[,1])

var(bb2$t[,1])

TASK 2

TASK 3

#################################################################################

############## LabPractice 2.2 GENETIC ALGORITHMS ##########################

############## MACHINE LEARNING II ##########################

############## ESU Jan-2022 ##########################

#################################################################################

## Set working directory --------------------------------------------------------

## Load libraries ---------------------------------------------------------------

library(caret)

library(ggplot2)

library(GA) # install.packages("GA")

# help(package="GA")

## remove all epxisting variables--------------------------------------------------

rm(list=ls())

#-------------------------------------------------------------------------------------------------

#---- SET THE SYNTHETIC LINEAR REGRESSION PROBLEM ---------------------------------------------

#-------------------------------------------------------------------------------------------------

fcreate <- function(coef1, coef2){function(x) coef1\*x+coef2}

f <- fcreate(1,5)

minVal = -20; maxVal = 20;

curve(f, minVal, maxVal)

x <- seq(minVal, maxVal, by = 0.1)

fval <- f(x)

noise <- runif(length(x),-3,3)

fnoise<-noise

freal<-fcreate(1,5)

yreal<- freal(x)+fnoise

data=data.frame(x,yreal)

# show the dataset and the theoretical line

p<- ggplot() + geom\_point(aes(x=x, y=yreal)) + labs(title = "Problem: y = b1 \* x + b0 ") + geom\_abline(aes(intercept=5, slope=1, color="theoretical" ))

plot(p)

y\_est=lm(yreal~x,data=data)

summary(y\_est)

coef=y\_est$coefficients

y\_estim=coef[1]+coef[2]\*x

rmse\_lm=mean((yreal-y\_estim)^2)

rmse\_lm

p<- ggplot() + geom\_point(aes(x=x, y=yreal)) + labs(title = "Problem: y = b1 \* x + b0 ") +

geom\_abline(aes(intercept=5, slope=1, color="theoretical" ))+

geom\_smooth(data=data,aes(x,yreal,color='regresión lineal'),se=FALSE)

plot(p)

#-------------------------------------------------------------------------------------------------

#---- fit LINEAR REGRESSION BY GA ---------------------------------------------

#-------------------------------------------------------------------------------------------------

myfitness <- function(coef) {

festimated<-fcreate(coef[1], coef[2])

yestimated = festimated(x)

-mean((yreal-yestimated)^2) # Negative (GA maximizes fitness)

}

myfitness(c(1,1))

myfitness(c(0,1))

myfitness(c(1,5))

ESUMonitor <- function(obj)

{

lines <- as.data.frame(obj@population)

lines$ind = rownames(lines)

p<- ggplot() + geom\_point(aes(x=x, y=yreal)) + labs(title = paste("Iteration =", obj@iter)) + geom\_abline(aes(intercept=lines[,2], slope=lines[,1], color="red" ))

print(p)

Sys.sleep(0.1)

}

# Run Basic GA using the monitor (see the Plots)

GA <- ga(type = "real-valued", fitness = myfitness,

lower = c(-3, -3), upper = c(8,8),

popSize = 100, # <------- Number of individuals

pcrossover = 0.8,

pmutation = 0.1,

maxiter = 100, # <---------------------- !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

monitor = ESUMonitor, # <---------------------- !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

optim = TRUE

)

abline(v = GA@solution, lty = 3, col = 2)

plot(GA)

summary(GA)

print(GA@solution)

lines <- as.data.frame(GA@population)

lines$ind = rownames(lines)

p<-ggplot() + geom\_point(aes(x=x, y=yreal))+ geom\_abline(aes(intercept=lines[,2], slope=lines[,1], color="red" ))

print(p)