Tecnológico de Monterrey

Tecnológico y de Estudios Superiores de Monterrey

Inteligencia artificial avanzada para la ciencia de datos I

Profesor: Jorge Adolfo Ramírez Uresti

Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo

Liam Garay Monroy A01750632

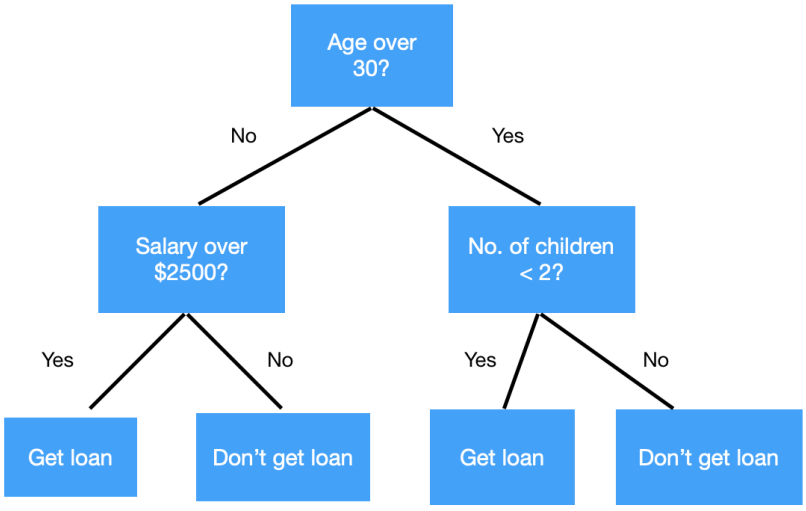
13 de Septiembre de 2022

Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo

Para este trabajo utilizaremos el algoritmo de random forest con el objetivo de poder analizar como es que cambiando los parámetros del mismo, podemos realizar distintos ajustes para llegar a un modelo más preciso, así como poder observar los cambios que se presentan entre un data set de pruebas, así como el de test, Tomando en cuenta distintos medidores y gráficas que nos permitan identificar dichos cambios.

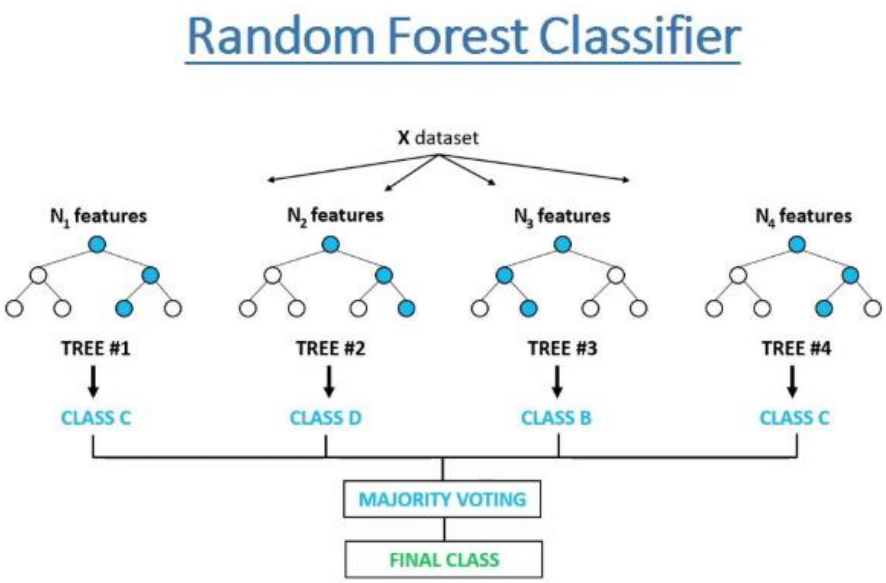
Primero, comenzaremos dando una introducción breve a nuestro algoritmo, y para ello, primero tenemos que entender como es que funciona un árbol de decisiones y la relación dentro de un random forest.

En esta imagen podemos apreciar un breve ejemplo de un árbol de decisiones.



Cada decisión va en forma de cascada, una detrás de otra, por ejemplo en el primer apartado podemos observar que se pregunta si la edad supera los 30 años, dependiendo de la respuesta, nuestro árbol tomara un camino u otro, y de esta forma llegará a clasificar nuestro conjunto de opciones dentro de una clase, por ejemplo en este caso se contemplan dos clases: recibe préstamo o no recibe préstamo.

Si ampliamos este panorama tenemos una serie de varios árboles de decisiones, los cuales tratan con muestras del data set principal (pudiendo ser este el de train o en dado caso de una predicción, el de test) teniendo esos fragmentos se realizan distintas predicciones por cada árbol, teniendo así una variedad de resultados con clases que pueden ser diferentes las unas de las otras.



Por lo el output de un random forest, es como lo indica la imagen, la clase que la mayoría de los árboles tuvieron como output se toma como la clase final, este tipo de modelo se considera mucho mejor a un árbol de decisiones normal, ya que ayuda a evitar el overfitting de los árboles hacia los datos de entrenamiento.

Ahora que sabemos un poco más sobre los árboles de decisión y los random forest, podemos entrar en materia directamente y realizar las pruebas necesarias para poder analizar nuestro modelo, en conjunto con los datos disponibles.

Lo primero es dividir nuestro dataset en dos partes, una de entrenamiento y otra para test

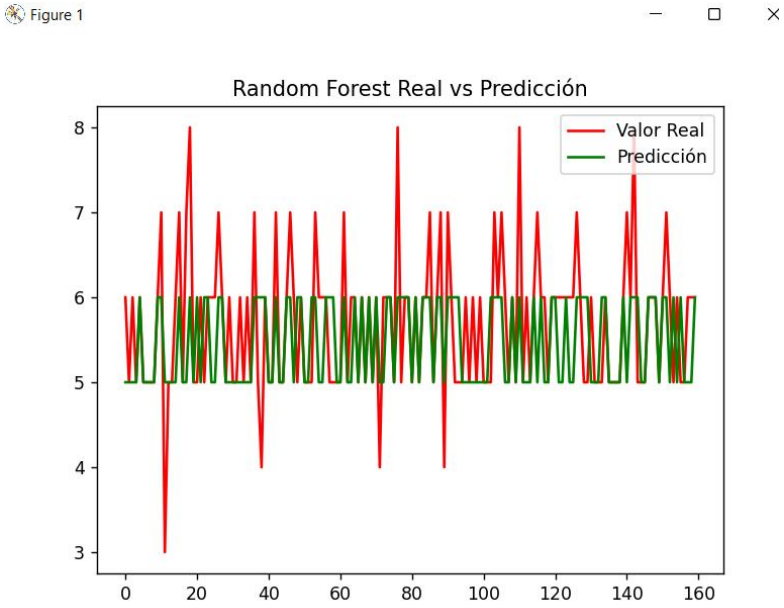
```
PEntrenamiento = 90
X_train, X_test, y_train, y_test =  train_test_split(X, y, test_size=(100-PEntrenamiento)/100,random_state=42)
```

Utilizaremos un random state, para poder replicar los datos y de esta forma ver como es que nuestro híper parámetros realmente afectan a nuestro modelo, así como un data set de entrenamiento del 90%.

Probaremos con el data set del vino, en donde intentaremos predecir su calidad únicamente con las columnas de “pH” y “alcohol”

Se utilizaron los siguientes parámetros:

```
NEstimadores = 3
NHojas = 4
rnd_clf = RandomForestClassifier(n_estimators=NEstimadores, max_leaf_nodes=NHojas, n_jobs=-1)
```



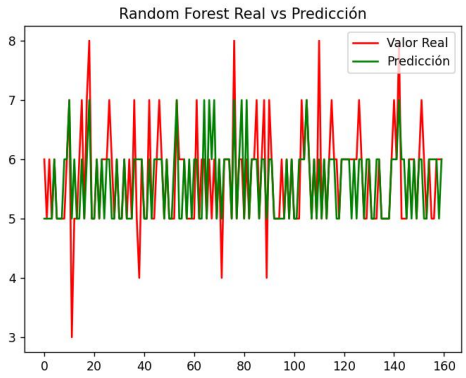
Podemos observar que nuestro modelo en este data set, tienen un high bias, ya que realmente no esta siendo muy preciso en los puntos de las predicciones, pero todos se encuentran dentro de un cierto rango establecido.

```
=============Entrenamiento del modelo=============
avg expected loss: 0.613
Bias: 0.487
Variance: 0.126
```

A pesar de ello tenemos un considerable promedio de perdida y podemos apreciar el high bias.

Ahora procederemos a cambiar estos parámetros para revisar la afectación que estos tienen con el modelo.

```
NEstimadores = 400
NHojas = 100
rnd_clf = RandomForestClassifier(n_estimators=NEstimadores, max_leaf_nodes=NHojas, n_jobs=12)
```

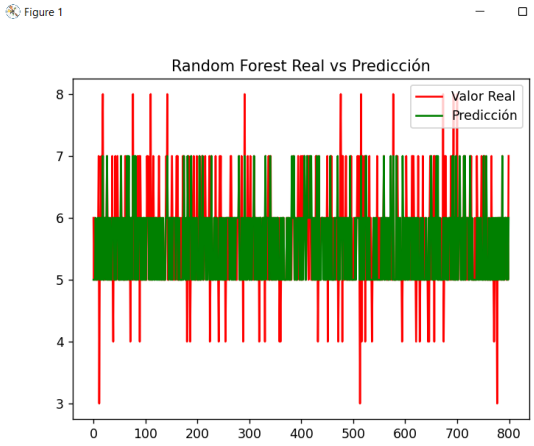


Podemos observar que si hay un cambio significativo en el Bias de nuestro modelo, ya que se acerca más a los valores reales, además de que ahora no se limita a ciertas clases, sino que explora distintas posibilidades

```
=============Entrenamiento del modelo=============
avg expected loss: 0.434
Bias: 0.345
Variance: 0.088
```

Observando los datos numéricos también nos podemos dar cuenta de que ambos datos disminuyeron en su valor, hay menos Bias y también menor varianza.

Probando con parámetros diferentes, así como número diferente de data set de train y test, esta vez utilizaremos un 50% de set de datos para train y test




```
=============Entrenamiento del modelo=============
avg expected loss: 0.513
Bias: 0.419
Variance: 0.094
```

Podemos apreciar que el bias, la varianza y la perdida aumentaron a los híper parámetros anteriores de 400 estimadores y 100 hojas, teniendo como parámetros en este 50% test y train los siguientes:

```
NEstimadores = 100
NHojas = 50
```

En este particular caso, con el data frame trabajado:

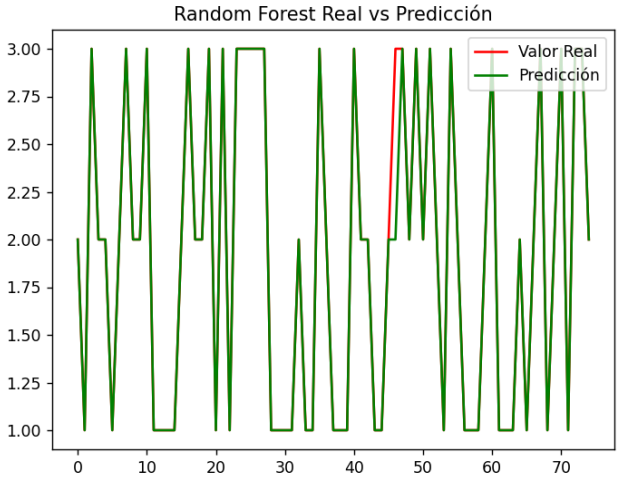
Podríamos argumentar que se esta haciendo un pequeño underfitting y muchas de las predicciones son generalizaciones, esto se puede apreciar por el alto bias, pero baja varianza, ya que no está realmente buscando predecir de forma concreta nuestros valores, sino que más bien esta siendo muy general, con la finalidad de tener un poco más datos promedio dentro del rango de nuestras clases, sesgando de cierta manera a nuestro modelo.

Observemos que pasa con otro data frame (Iris)

Usando el data frame de Iris, podemos observar resultados más balanceados

Parámetros:

```
PEntrenamiento = 50
X_train, X_test, y_train, y_test =  train_test_split(X, y, test_size=(100-PEntrenamiento)/100,random_state=42)
NEstimadores = 3
NHojas = 4
```




```
=============Entrenamiento del modelo=============
avg expected loss: 0.037
Bias: 0.011
Variance: 0.027
```

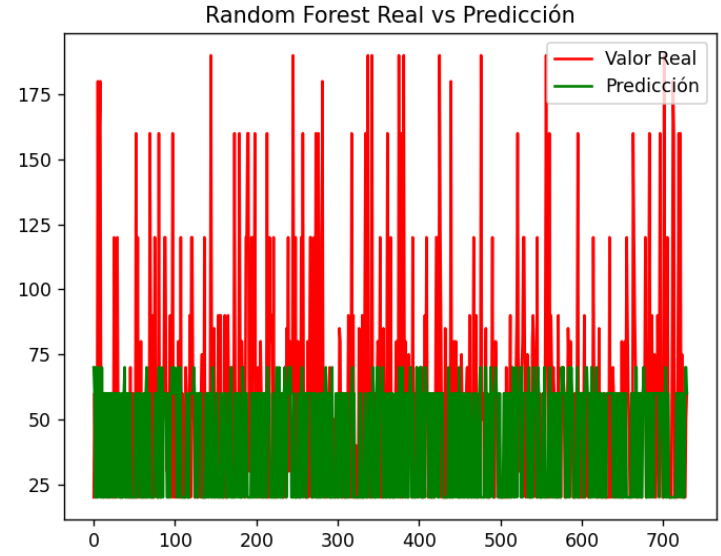
Podemos observar sin duda que se tienen datos mejor balanceados, aunque si se tiene una varianza más grande que el Bias no considero que sea suficiente para considerarse un overfitting ya que no son datos que precisamente estén muy lejos el uno del otro y ambos

es bastante pequeños, además en la gráfica, podemos observar que, a pesar de ser datos separados, todos se encuentran dentro de un mismo rango.

Por último, con otro data frame:

Pero en este cambiaremos un poco la forma de trabajar, elegiremos dos columnas al azar como nuestra X y una como nuestra Y, sin revisar ningún tipo de correlación, para ver la diferencia de un Bias grande y una varianza grande

```
PEntrenamiento = 50
X_train, X_test, y_train, y_test =  train_test_split(X, y, test_size=(100-PEntrenamiento)/100,random_state=42)
NEstimadores = 3
NHojas = 4
```




```
============Entrenamiento del modelo============
avg expected loss: 2269.872
Bias: 2032.408
Variance: 237.465
```

Podemos observar la gran diferencia entre los datos de la predicción y los reales, además de los datos numéricos que comparándolos con los de las anteriores datas frames, son increíblemente abismales, teniendo sin duda predicciones totalmente erróneas en cada iteración.

En conclusión: Con nuestro modelo, podemos observar que en algunos de los data sets se realiza un underfitting y se generalizan en varias de las predicciones esto también lo podemos observar por el alto Bias que se obtiene de las predicciones, en algunas ocasiones también tenemos una varianza alta, con lo cual podríamos estar

presenciando un overfitting del modelo pero al cambiar nuestros híper parámetros podemos observar que nuestro modelo mejora de manera sustancial, teniendo mejores resultado en nuestras predicciones, así como menor Bias y menor Variance, también podemos observar como la perdida promedio va disminuyendo, es necesario recordar que con cosas como Grid search podemos ir midiendo como estos híper parámetros afectan nuestro modelo y de revisar de que forma modificarlos para tener mejor aproximaciones.