Módulo 2: Análisis y Reporte sobre el desempeño del modelo

Random Forest Classifier

Ximena Montserrat Sánchez Rubio

A01378326

Septiembre 2023

- Código del modelo disponible en: https://github.com/XimenaMRubio/Module_implementation.git

Random Forest Classifier

Para este proyecto, se eligió el data set de Iris. En este podemos encontrar información de diversos marcadores de plantas. Los datos que incluye son, el largo y ancho del sépalo, así como el ancho y el largo del pétalo. Por su parte, el target cuenta con las tres clases en las que se dividen las flores de iris (Setosa, Versicolor y Virginica). Considero que, a pesar de contar con pocos datos en comparación con otros datasets, estos son suficientes para el entrenamiento, test y validación. Otro aspecto importante, es que las clases están balanceadas. El método de Random Forest Classifier funciona adecuadamente cuando se busca poder separar en categorías.

División de los datos

Lo que quiero explicar con la figura 1 es que, en general, los datos aún después de separarlos tienen una distribución balanceada. Si bien se puede observar que hay una pequeña discrepancia entre la parte de entrenamiento, prueba y validación, las clases se mantienen. Esto indica que de principio es menos probable que exista algún tipo de bias relacionado a la distribución desigual de los datos. En este caso, tomar en cuenta este aspecto es importante, puesto que ayuda a que el modelo no tenga "preferencias" por cierta categoría.

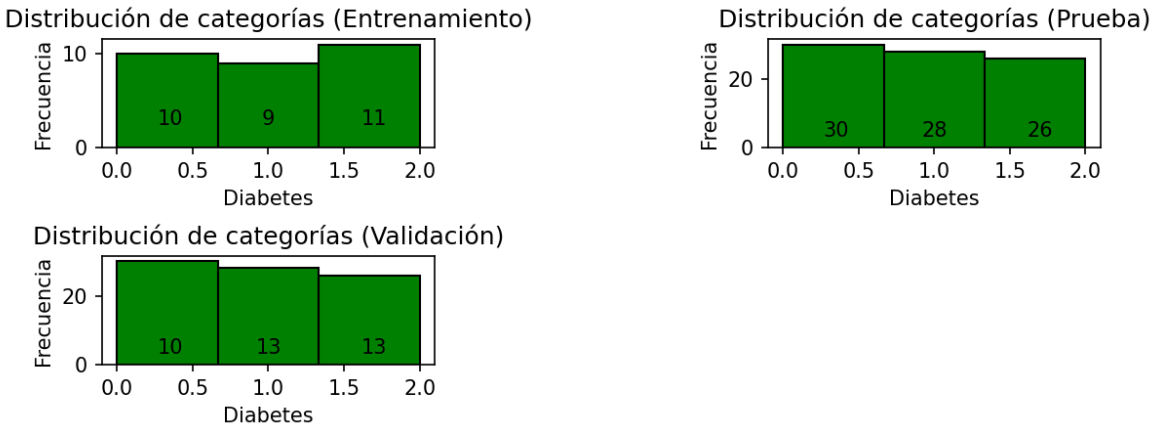

Figura 1. Comparando la distribución de categorías

Inicialmente se dividió el data set en dos partes: training y testing como se muestra en la figura 2. Para obtener el set de validación, lo que se hizo fue separar el testing en dos partes. En la figura 3, podemos observar de manera más detallada la separación final.

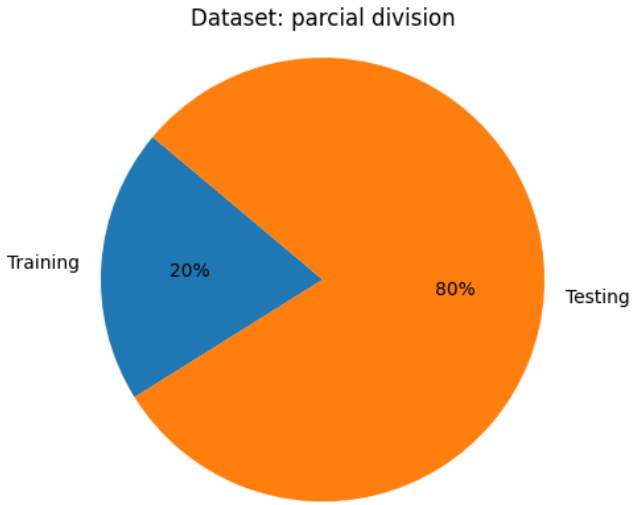

Figura 2. Gráfica de pastel que indica la división parcial del data set.

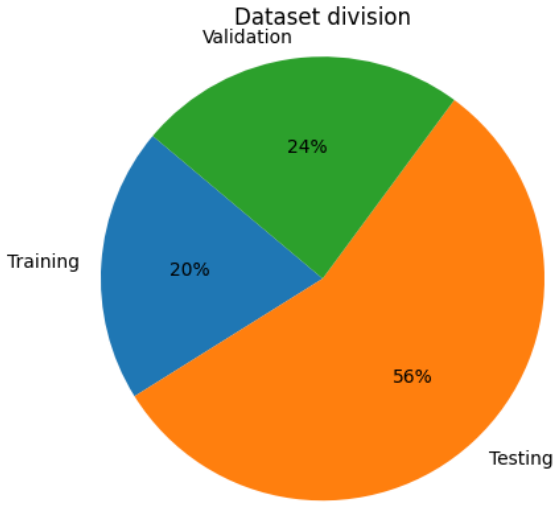

Figura 3. Gráfica de pastel que indica la división final del data set.

```
El dataset tiene  150  datos en total.
El set de training tiene  30  datos.
El set de testing tiene  84  datos.
El set de validación tiene  36  datos.
```

Figura 4. Número de datos presentes en el entrenamiento, test y validación.

En la figura 4 podemos ver el total de datos dentro del data set y como quedaron los demás después de dividirlos. A deferencia de las gráficas anteriores, en este recuadro podemos ver la cantidad que representa el porcentaje. Por supuesto esto se puede cambiar, sin embargo, considero que esta configuración funcionó adecuadamente con el modelo. Es importante mencionar que busqué darle poco porcentaje al training para evitar overfiting.

Modelo

Para el modelo inicial busqué ponerle parámetros no ideales con el fin de obtener resultados poco favorables y poder observar como el modelo va mejorando cuando estos van cambiando. Es importante observar que este generaliza.

```
MODELO 1
La configuración de este modelo es:  RandomForestClassifier(max_depth=1, n_estimators=3)

              precision    recall  f1-score   support

           0       0.81      1.00      0.90        30
           1       1.00      0.46      0.63        28
           2       0.74      0.96      0.83        26

    accuracy                           0.81        84
   macro avg       0.85      0.81      0.79        84
weighted avg       0.85      0.81      0.79        84

MAE: 0.20238095238095238
MSE: 0.2261904761904762
Overall specificity score:  0.9056741470534574
```

Figura 5. Métricas del primer modelo

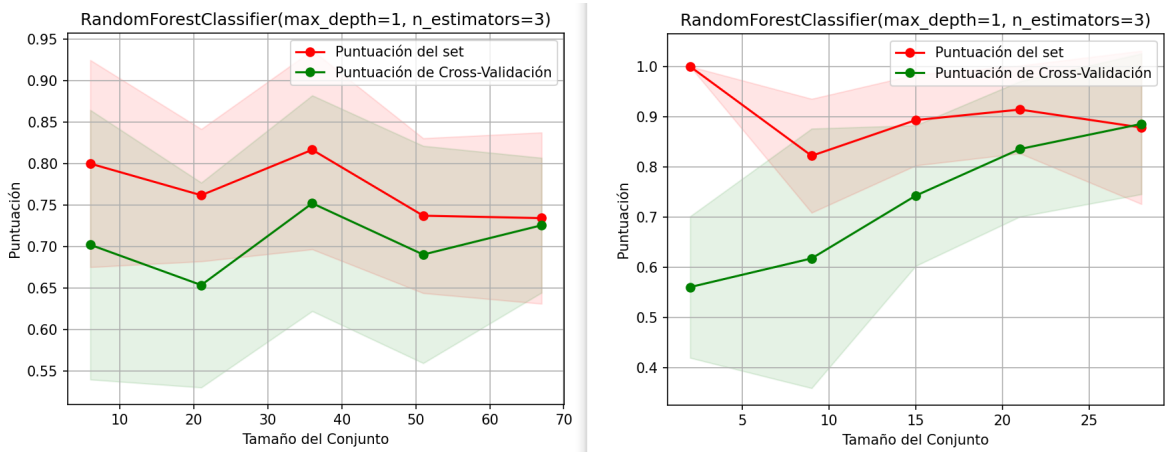

Figura 6. Curva de aprendizaje del primer modelo (test & validation sets)

De este primer modelo podemos observar que se tiene un accuracy general de 81. Para cada clase se obtuvieron las métricas de precision, recall y f1. La clase que clasificó de una forma más certera fue la clase 0. Hay un buen puntaje de f1, lo que indica un buen balance entre las predicciones correctas y las que identificó como correctas. De ahí, le sigue la clase 2 con un f1 de 0.83 y finalmente la clase 1. En esta última podemos observar que el modelo es muy preciso en sus predicciones positivas, pero no es tan sensible a la detección de todos los casos positivos presentes en el conjunto de datos.

Con un MAE y un MSE relativamente bajos, indica que la diferencia entre las predicciones y los valores reales no tienen una varianza tan grande. El specificity muestra que el 90.56% de los casos negativos reales se clasifican correctamentte como negativos.

En la figura 6 podemos observar dos gráficas. La del lado izquierdo corresponde a la curva de aprendizaj del test y la del derecho a la curva de aprendizaje del validation. En la primera podemos observar que en la puntuación del set, la puntuación disminuye y en el cross-validation aumenta un

poco. De la segunda gráfica (lado derecho), podemos ver que mejora en la prueba de validación y el puntaje general. Esto indica que tiene un buen rendimiento en ambos pero la varianza aumentó. Esto se puede apreciar más debido a la separación de las lineas.

Conclusiones del modelo:

- Bias: Bajo
- Varianza: Media
- Nivel de ajuste: Fit

Una posible solución es realizar un modelo un poco más complejo.

```
MODELO 2
La configuración de este modelo es:  RandomForestClassifier(max_depth=1, n_estimators=5)

              precision    recall  f1-score   support

           0       0.80      0.93      0.86        30
           1       0.89      0.57      0.70        28
           2       0.81      0.96      0.88        26

    accuracy                           0.82        84
   macro avg       0.83      0.82      0.81        84
weighted avg       0.83      0.82      0.81        84

MAE: 0.19047619047619047
MSE: 0.21428571428571427
Overall specificity score:  0.9097792373654443
```

Figura 7. Métricas del segundo modelo (técnica de regularización 1).

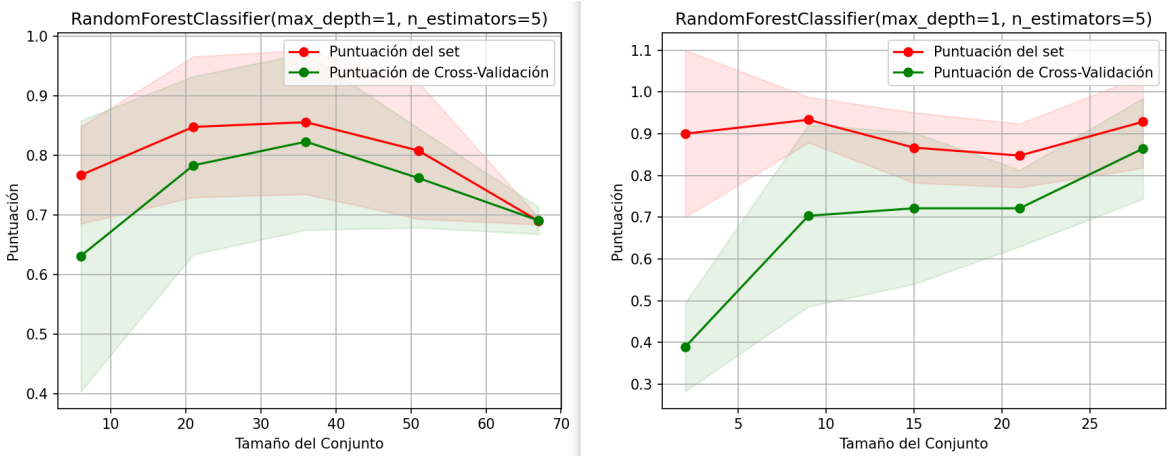

Figura 8. Curva de aprendizaje del segundo modelo (test & validation sets)

De la figura 7 podemos observar que tenemos una mejora en las métricas, lo que indica que los cambios realizados al modelo ayudaron a mejorarlo.

Algo curioso es que en la primera curva de aprendizaje del test (izquierda) tiene un rendimiento bajo, lo cual puede indicar que hay sesgo en los datos. Sin embargo, en el set de validación podemos ver que tiene un buen desempeño y la linea de cross-validation indica que el modelo mejora conforme se le agregan más datos.

Conclusiones del modelo:

- Bias: Bajo

- Varianza: Media
- Nivel de ajuste: Underfit
  Nuevamente considero que una posible solución es realizar un modelo un poco más complejo.

```
MODELO 3
La configuración de este modelo es:  RandomForestClassifier(max_depth=3, max_leaf_nodes=3, n_estimators=50)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        30
           1       0.92      0.82      0.87        28
           2       0.83      0.92      0.87        26

    accuracy                           0.92        84
   macro avg       0.92      0.91      0.91        84
weighted avg       0.92      0.92      0.92        84

MAE: 0.08333333333333333
MSE: 0.08333333333333333
Overall specificity score:  0.9614121510673235
```

Figura 9. Métricas del tercer modelo (técnica de regularización 2).

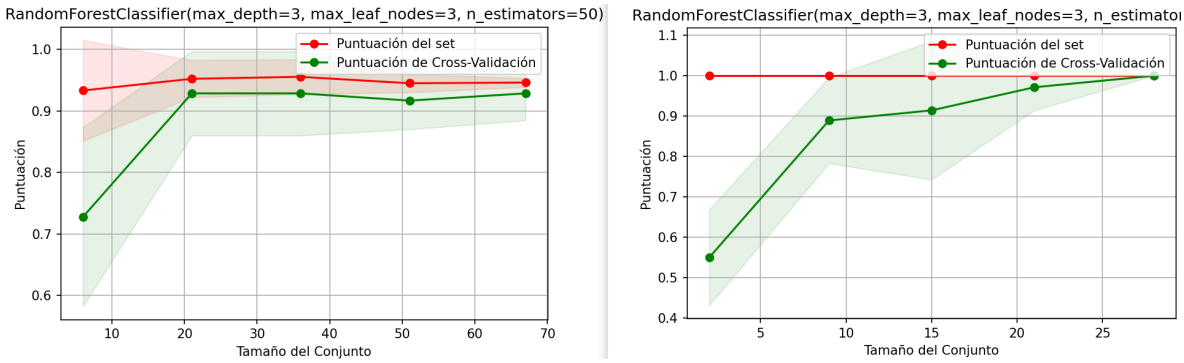

Figura 10. Curva de aprendizaje del tercer modelo (test & validation sets)

El modelo es capaz de identificar la complejidad de los datos y en general podemos decir que tiene un buen desempeño. Podemos decir que tiene un nivel de bias medio porque auque el MAE y MSE tienen un puntaje bajo y el f1-score indica que para las clases 1 y 2 hay un balance aceptable pero no tan bueno. La linea del cross-validation indica que mientras hay más datos, la varianza aumenta un poco.

Conclusiones del modelo:

- Bias: Medio
- Varianza: Media
- Nivel de ajuste: Fit
  El siguiente paso es buscar una configuración que no incremente a overfit y mejore las métricas.

```
MODELO 4
La configuración de este modelo es:  RandomForestClassifier(max_depth=5, max_leaf_nodes=6, n_estimators=50)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        30
           1       0.90      0.93      0.91        28
           2       0.92      0.88      0.90        26

    accuracy                           0.94        84
   macro avg       0.94      0.94      0.94        84
weighted avg       0.94      0.94      0.94        84

MAE: 0.05952380952380952
MSE: 0.05952380952380952
Overall specificity score:  0.9714696223316912
```

Figura 11. Métricas del cuarto modelo (técnica de regularización 3).

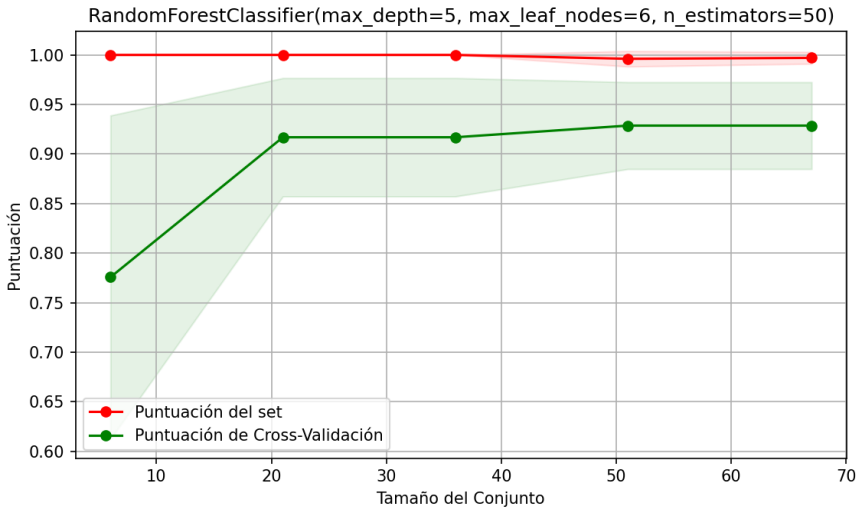

Figura 12. Curva de aprendizaje del cuarto modelo (test set)

Finalmente, en el modelo 4, podemos observar que el accuracy tuvo una mejora, así como el f1-score. Esto indica que el balance entre precision y recall mejoró pero aún podemos identificar un poco de bias. El MSE y MAE también indican que el sesgo en mucho más bajo en comparación con los otros modelos.
Podemos observar que la curva de aprendizaje del test set tuvo un buen puntaje que posiblemente puede significar un poco de overfit. En la prueba de cross-validation también se observa un buen desempeño.

El specificity muestra que el 97.14% de los casos negativos reales se clasifican correctamentte como negativos.

Conclusiones:

- Bias: Bajo-Medio
- Varianza: Bajo
- Nivel de ajuste: Fit-Overfit

Como comentario, considero que para mejorar el modelo, pueden agregarse más datos al dataset y de esta manera crear un modelo más complejo. Considero que el tamaño del dataset fue un reto para el desarrollo del modelo porque al ser muy pequeño y estar bien balanceado, funcionaba muy

bien con los parámetros iniciales. Justamente mencioné que por cuestiones experimentales, busqué iniciar con parámetros que no favorecían el desempeño del modelo.

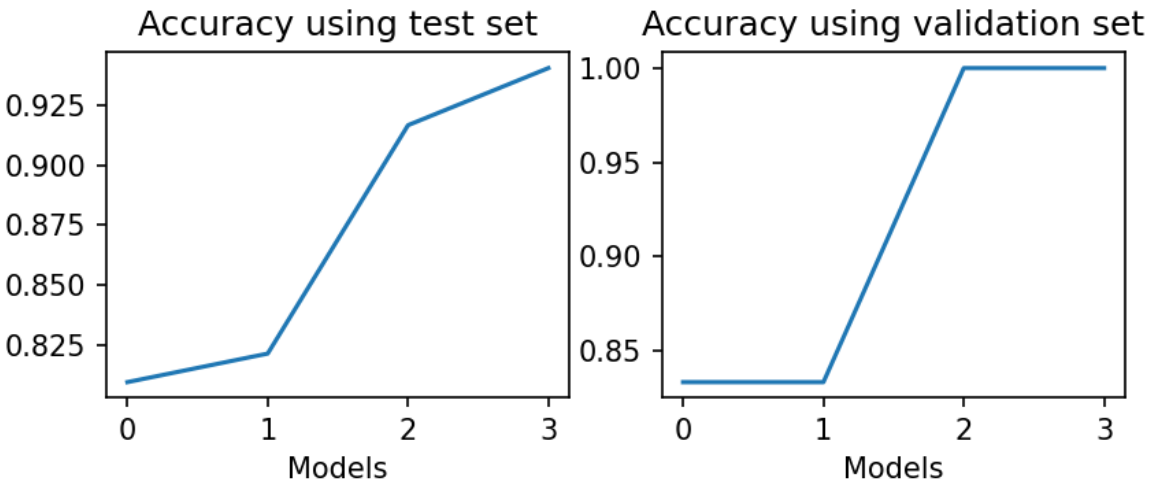

Figura 13. Comparación de los accuracy entre test y validación.

Finalmente, se muestran las gáficas de como aumenta el accuracy en el set de prueba y validación para cada modelo. Podemos observar que en general el accuracy va aumentando. Un problema que observo es que exista overfit en el último modelo.

Algo importante a mencionar es que utilicé las siguientes librerías:

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

El fin de estas librerías es proporcionarte major configuración posible para el modelo en cuestion. En este caso utilicé GridSearch y RandomSearch. Como se puede ver en la figura 18, al final se muestra la mejor combinación de cada método. Las opciones de parámetros fueron previamente establecidas en un diccionario.

```
#diccionario de parámetros
param_grid = {
  'n_estimators': [50, 100, 200, 350],
  'max_features': ['sqrt', 'log2', None],
  'max_depth': [3,5,10],
  'max_leaf_nodes': [3, 6, 9],
}

#Buscando los mejores parámetros utilizando grid search
grid_search = GridSearchCV(RandomForestClassifier(),
                           param_grid=param_grid)
grid_search.fit(Xtrain, ytrain)
print(grid_search.best_estimator_)

#Buscando los mejores parámetros utilizando random search
random_search = RandomizedSearchCV(RandomForestClassifier(),
                                   param_grid)
random_search.fit(Xtrain, ytrain)
print(random_search.best_estimator_)
```

```
RandomForestClassifier(max_depth=3, max_leaf_nodes=3, n_estimators=50)
RandomForestClassifier(max_depth=10, max_leaf_nodes=3)
```

Figura 18. Parte del código donde se aplica la búsqueda de parámetros

Conclusión

Como conclusión puedo decir que el cambiar los parámetros definitivamente puede ayudar a generar un mejor modelo. Si bien, podemos realizar estos cambios mediante nuestros propios experimentos, considero que vale la pena emplear métodos de selección de parámetros. Esto con el fin de darnos una idea de que hacer para poder mejorar el modelo. En este proceso no puede hacerse al azar, es importante tomar en cuenta la varianza de los datos y el bias y tratar de buscar un balance entre ellos. Igualmente, es necesario tomar en cuenta las métricas, ya que pueden ayudarnos a identificar si necesitamos un modelo más complejo o quizá es un problema más relacionado al dataset.