Tecnológico de Monterrey

Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo. (Portafolio Análisis)

Materia:

Inteligencia artificial avanzada para la ciencia de datos I (Gpo 101)

Profesor:

Docente: Ivan Mauricio Amaya Contreras

Alumno

Gilberto Ramos Salinas A01734128

Fecha

08 de Septiembre del 2022

Introducción

A lo largo de este módulo tuve la oportunidad de realizar diferentes implementaciones sobre el uso de Machine Learning en distintas bases de datos con el objetivo de mejorar dicha implementación en futuras entregas. Por ello, el análisis de este documento será sobre un proyecto realizado con frameworks como pandas, numpy y sklearn. Este proyecto fue ejecutado con una base de datos proveniente de Kaggle denominada Iris, la cual contiene datos provenientes como la longitud cepal, ancho cepal, longitud del pétalo y ancho del pétalo. El propósito principal es generar una predicción del subtipo de la planta iris para facilitar el trabajo de clasificación.

Exploración de datos

Antes de realizar el análisis es necesario revisar el dataset para ver la calidad de la información. Esto se hace con el objetivo de observar si el dataset cuenta con espacios vacíos o con información incompleta. También se puede ver la relación que existe entre las variables para ver si existe una correlación grande entre las especies y sus atributos.
Esta visualización puede ser obtenida a través de los siguientes comandos:

El análisis debe de contener los siguientes elementos:

```
##Checar valores no nulos
df.notnull().sum()
```

```
Id              150
SepalLengthCm   150
SepalWidthCm    150
PetalLengthCm   150
PetalWidthCm    150
Species         150
No_Species      150
dtype: int64
```

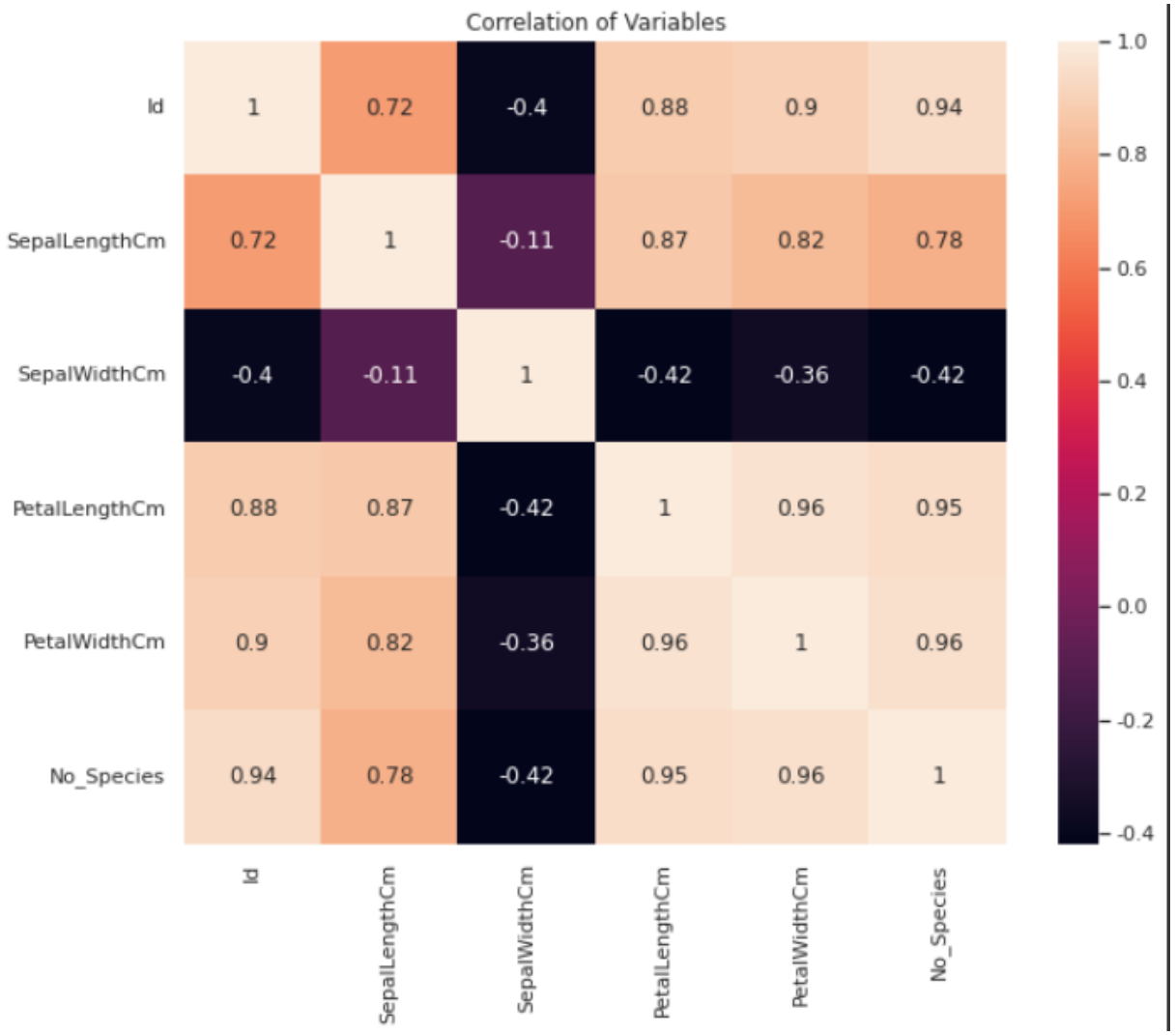
```
[71] ##Checar valores nulos
     df.isnull().sum()
```

```
Id              0
SepalLengthCm   0
SepalWidthCm    0
PetalLengthCm   0
PetalWidthCm    0
Species         0
No_Species      0
dtype: int64
```

```
##Checar blancos
df.isna().sum()
```

```
Id              0
SepalLengthCm   0
SepalWidthCm    0
PetalLengthCm   0
PetalWidthCm    0
Species         0
No_Species      0
dtype: int64
```

Por medio del siguiente diagrama de calor podemos observar que unas variables presentan una mayor correlación que otras. Sin embargo, esto no fue considerado determinante para la realización del proyecto.



Definición del modelo Regresión Logística

Al tratarse de una clasificación de subtipos se puede implementar el concepto de variable dicotómica tomando en cuenta los únicos 3 valores que puede dar de resultado (Iris-setosa, Iris-versicolor, Iris-virginica). Por ello, se decidió utilizar el modelo de regresión logística para facilitar el proceso de clasificación de plantas, al predecir el resultado por un conjunto de predictores.

La variable dependiente sería la clasificación de de las plantas que en este caso se clasificaron en notación numérica. (No_Species)

Las variables independientes son: 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm'.

```
[22] name ='Regresion Logistica'
     logistic = LogisticRegression()
```

```
x = df[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']].values
classes = {'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2}
y = df['No_Species'].values  # variable dependiente
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
 6   No_Species     150 non-null    int64
dtypes: float64(4), int64(2), object(1)
memory usage: 8.3+ KB
```

Separación y evaluación del modelo con un conjunto de prueba y un conjunto de validación (Train/Test/Validation).

La separación es obtenida a través de la librería sklearns con X_train, x_test, Y_train, y_test. La validación es obtenida por el método K Fold, el cual revisa la veracidad de un modelo machine learning. Para este caso al tratarse de la base de datos Iris se decidió implementar un modelo de regresión logística.

```
[ ]  from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(x, y, random_state=42)
```

```
from sklearn.model_selection import KFold
from sklearn.metrics import make_scorer
from sklearn.model_selection import cross_validate

kfold = KFold(n_splits=30, shuffle=True, random_state=42)
scorer = make_scorer(accuracy_score)

veracity = cross_validate(logistic, x, y, cv=kfold, scoring=scorer)

acur = veracity['test_score']
var = veracity['test_score'].var()
m = veracity['test_score'].mean()
sd = veracity['test_score'].std()
```

```
Accuracy of the Logistic Regretion model with k-fold cross validation
K-fold accuracies:  [1.  1.  1.  1.  1.  1.  1.  1.  1.  0.8 1.  1.  1.  1.  1.  1.  0.8 0.8
 0.8 0.8 1.  1.  1.  1.  1.  1.  1.  1.  1.  0.8]
Mean : 0.9600000000000001
Variance:  0.006399999999999997
Standard deviation:  0.07999999999999997
Bias:  0.039999999999999925
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (
```

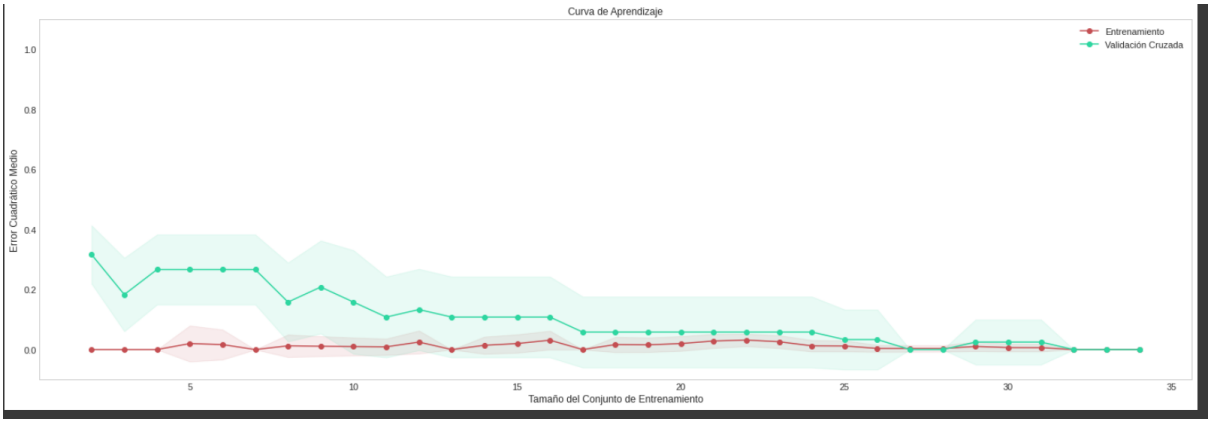
Diagnóstico y explicación el grado de bias o sesgo: bajo medio alto

Anteriormente se puede apreciar como los resultados del K Fold demuestran como la diferencia entre el promedio del valor predicho contra el real es 0.96 estando cerca del 1. En otras palabras, demuestra un modelo capaz de comprender la información y retornar resultados de clasificación gratificantes.

A través de estos resultados podemos obtener que el sesgo es de 0.039 demostrando que el modelo de regresión logística tiene un sesgo bajo dado que el resultado está bastante cerca del 0.

Diagnóstico y explicación el grado de varianza: bajo medio alto

La varianza como tal nos explica la distribución de los datos respecto a la media. Por ello, la varianza en este caso nos da .0063 Esto demuestra que la distribución de datos respecto a la media es baja.



Diagnóstico y explicación el nivel de ajuste del modelo: underfitt fitt overfitt

Basándonos en los resultados previos se pudo apreciar que la veracidad del modelo es de 0.96 indicando que el modelo presenta un overfitting. Esto se puede extrapolar gracias a que los errores en el entrenamiento fueron bastante bajos como se aprecia en la imagen anterior y los de validación altos.

Basándote en lo encontrado en tu análisis utiliza técnicas de regularización o ajuste de parámetros para mejorar el desempeño de tu modelo y documenta en tu reporte cómo mejoró este.

Con el objetivo de mejorar el rendimiento del modelo me di la tarea de investigar métodos alternativos para seleccionar parámetros, estimador entre otras configuraciones para el modelo. El resultado de esta búsqueda fue la implementación de hiperparametros por medio de greed search cv el cual retorna la mejor configuración posible para el modelo.

```python
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedKFold

# Create the parameter grid based on the results of random search with iris dataset

param_grid = {
    'penalty': ['l1', 'l2', 'elasticnet', 'none'],
    'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'max_iter': [100, 1000, 2500, 5000]
}

# Instantiate the grid search model
grid_search = GridSearchCV(estimator = logistic, param_grid = param_grid,
                           cv = 3, n_jobs = -1, verbose = 2)

grid_search.fit(X_train, y_train)
```

```
[35] grid_search.best_params_

     {'C': 1, 'max_iter': 100, 'penalty': 'l1', 'solver': 'saga'}

[36] grid_search.best_score_

     0.9729729729729729

[37] grid_search.best_estimator_

     LogisticRegression(C=1, penalty='l1', solver='saga')
```

```
[38] name ='Regresion Logistica'
     logistic = LogisticRegression(C=1, penalty='l1', solver='saga')

[39] logistic.fit(X_train, y_train)
     y_pred = logistic.predict(X_test)

     print("Matriz de Confusión para ", name, confusion_matrix(y_test,y_pred))
     print("Reporte de Clasificación para ", name, classification_report(y_test,y_pred))

     Matriz de Confusión para  Regresion Logistica [[15  0  0]
      [ 0 11  0]
      [ 0  0 12]]
     Reporte de Clasificación para  Regresion Logistica              precision    recall  f1-score   support

               0       1.00      1.00      1.00        15
               1       1.00      1.00      1.00        11
               2       1.00      1.00      1.00        12

        accuracy                           1.00        38
       macro avg       1.00      1.00      1.00        38
    weighted avg       1.00      1.00      1.00        38

     /usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_sag.py:354: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
       ConvergenceWarning,
```

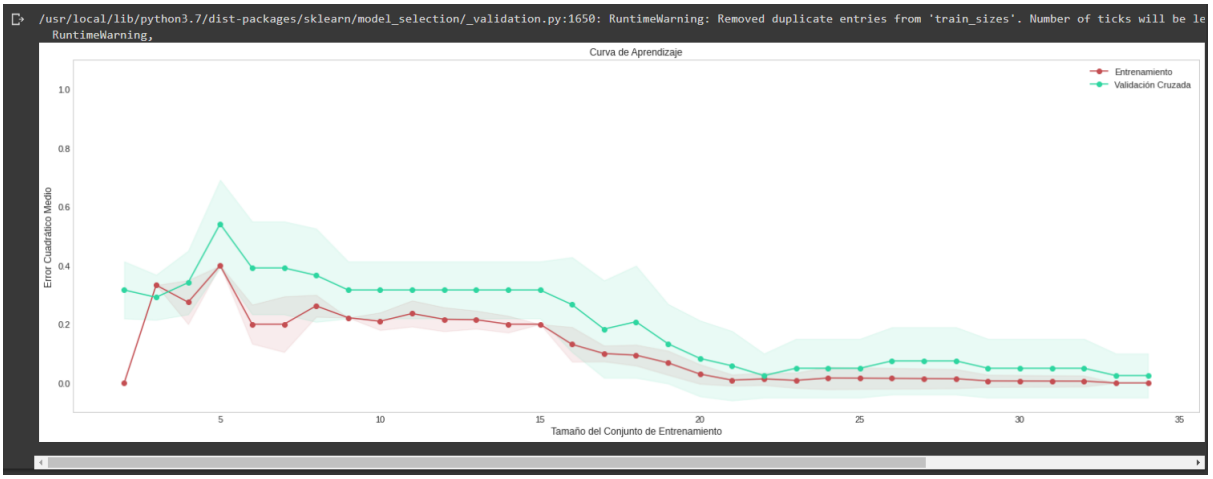
Este nuevo método fue validado por el K-Fold para poder comparar la mejora de los resultados en todos sus ambitos.

```
Accuracy of the Logistic Regretion model with k-fold cross validation
K-fold accuracies:  [1.  1.  1.  1.  1.  1.  1.  1.  1.  0.8 1.  1.  1.  1.  1.  1.  1.  0.8
 0.8 1.  0.8 1.  1.  1.  1.  1.  1.  1.  1.  1. ]
Mean : 0.9733333333333333
Variance:  0.0046222222222222185
Standard deviation:  0.06798692684790376
Bias:  0.026666666666666727
```

De esta forma podemos apreciar que el modelo mejoró su desempeño, ya que la veracidad incrementó de 0.96 a 0.973. Esto mismo puede ser comparado con los demás rubros:

Modelos	LogisticRegretion()	LogisticRegretion() despues del Greedsearchcv
Accuracy	0.9600	0.9733
Variance	0.0063	0.0046
Standard Deviation	0.0799	0.0679
Bias	0.0399	0.0266

Grado de varianza:



Con este nuevo estudio obtenemos una mejora significativa en los resultados del modelo, retornando clasificaciones gratificantes, ya que el modelo es capaz de comprender la información.

Resultados:

```
[126] def prediction(sepal_length,sepal_width,petal_length,petal_width):
          plants = {0: 'Iris-setosa', 1: 'Iris-versicolor', 2: 'Iris-virginica'}
          pred = logistic.predict([[sepal_length, sepal_width, petal_length, petal_width]])
          print("La especie de la flor es: ", pred," ",plants[pred[0]])
```

```
# Id  SepalLengthCm SepalWidthCm  PetalLengthCm PetalWidthCm  Species     No_Species
# 1   5.1           3.5           1.4           0.2           Iris-setosa 0
prediction(5.1,3.5,1.4,0.2)
# Id  SepalLengthCm SepalWidthCm  PetalLengthCm PetalWidthCm  Species     No_Species
# 51  7.0           3.2           4.7           1.4           Iris-versicolor 1
prediction(7.0,3.2,4.7,1.4)
# Id  SepalLengthCm SepalWidthCm  PetalLengthCm PetalWidthCm  Species     No_Species
# 150 5.9           3.0           5.1           1.8           Iris-virginica  2
prediction(5.9,3.0,5.1,1.8)
# Id  SepalLengthCm SepalWidthCm  PetalLengthCm PetalWidthCm  Species     No_Species
# 23  4.6           3.6           1.0           0.2           Iris-setosa 0
prediction(4.6,3.6,1.0,0.2)
# Id  SepalLengthCm SepalWidthCm  PetalLengthCm PetalWidthCm  Species     No_Species
# 24  5.1           3.3           1.7           0.5           Iris-setosa 0
prediction(5.1,3.3,1.7,0.5)
```

```
La especie de la flor es:  [0]   Iris-setosa
La especie de la flor es:  [1]   Iris-versicolor
La especie de la flor es:  [2]   Iris-virginica
La especie de la flor es:  [0]   Iris-setosa
La especie de la flor es:  [0]   Iris-setosa
```

A través de estos resultados podemos observar como el modelo es capaz de clasificar correctamente las plantas, cumpliendo con éxito los estándares de entrega. Aunque se trate de un proyecto básico, este modelo es capaz de ser utilizado en la industria biológica al clasificar diferentes tipos de especies de plantas para tener un mejor control de un sector o facilitar el estudio de un grupo.

De esta forma se demuestra que la implementación de machine learning en la industria es una herramienta con un potencial muy alto, capaz de acelerar procesos complejos de una forma óptima y rápida.