Instituto Tecnológico de Estudios Superiores de Monterrey
Campus Monterrey

“Apegándome a la Integridad Académica de los Estudiantes del Tecnológico de Monterrey, me comprometo a que mi actuación en esta actividad esté regida por la integridad académica. En congruencia con el compromiso adquirido, realizaré este trabajo de forma honesta y personal, para reflejar, a través de él, mi conocimiento y aceptar, posteriormente, la evaluación obtenida”

Inteligencia artificial avanzada para la ciencia de datos I
Gpo 102

Módulo 2: Aprendizaje máquina.
Análisis y reporte sobre el desempeño del modelo.

Alumno:
A01750164 | Paul Martín García Morfín

Profesor:
César Javier Guerra Páramo

Fecha: 18/09/2022

Análisis y reporte sobre el desempeño del modelo

El modelo seleccionado y que se analiza en el presente documento es la implementación de un Random Forest Regressor incluido en la biblioteca de Sklearn.
El conjunto de datos utilizado es el de “Boston Housing” el cual contiene información relativa a las viviendas en la zona de Boston, Massachusetts recopilada por el Servicio de Censos de EE.UU.

Dentro del conjunto de datos existen 14 atributos que se describen a continuación:

1. CRIM - tasa de criminalidad per cápita por ciudad
2. ZN - proporción de terrenos residenciales con zonificación para lotes de más de 25.000 pies cuadrados
3. INDUS - proporción de acres comerciales no minoristas por ciudad.
4. CHAS - Variable ficticia del río Charles (1 si el tramo limita con el río; 0 en caso contrario)
5. NOX - concentración de óxidos nítricos (partes por 10 millones)
6. RM - número medio de habitaciones por vivienda
7. EDAD - proporción de unidades ocupadas por sus propietarios construidas antes de 1940
8. DIS - distancias ponderadas a cinco centros de empleo de Boston
9. RAD - índice de accesibilidad a las autopistas radiales
10. TAX - tasa de impuesto sobre la propiedad de valor total por 10.000 dólares
11. PTRATIO - ratio alumno-profesor por ciudad
12. B - 1000(Bk - 0,63)^2 donde Bk es la proporción de negros por ciudad
13. LSTAT - % de estatus inferior de la población
14. MEDV - Valor medio de las viviendas ocupadas por sus propietarios en 1000 dólares.

La variable objetivo, es decir, de la que se busca obtener predicciones en función de las otras variables disponibles, es el valor medio de las viviendas (**MEDV**).

Para iniciar, se realiza una pequeña exploración de esta base de datos para entender la información disponible.

```
Dataset shape:
(506, 14)

Summary:
         count    mean     std     min     25%     50%     75%     max
CRIM     506.0    3.61    8.60    0.01    0.08    0.26    3.68   88.98
ZN       506.0   11.36   23.32    0.00    0.00    0.00   12.50  100.00
INDUS    506.0   11.14    6.86    0.46    5.19    9.69   18.10   27.74
CHAS     506.0    0.07    0.25    0.00    0.00    0.00    0.00    1.00
NOX      506.0    0.55    0.12    0.38    0.45    0.54    0.62    0.87
RM       506.0    6.28    0.70    3.56    5.89    6.21    6.62    8.78
AGE      506.0   68.57   28.15    2.90   45.02   77.50   94.07  100.00
DIS      506.0    3.80    2.11    1.13    2.10    3.21    5.19   12.13
RAD      506.0    9.55    8.71    1.00    4.00    5.00   24.00   24.00
TAX      506.0  408.24  168.54  187.00  279.00  330.00  666.00  711.00
PTRATIO  506.0   18.46    2.16   12.60   17.40   19.05   20.20   22.00
B        506.0  356.67   91.29    0.32  375.38  391.44  396.22  396.90
LSTAT    506.0   12.65    7.14    1.73    6.95   11.36   16.96   37.97
MEDV     506.0   22.53    9.20    5.00   17.02   21.20   25.00   50.00
```

```
Numerical variables:
        CRIM
        ZN
        INDUS
        CHAS
        NOX
        RM
        AGE
        DIS
        RAD
        TAX
        PTRATIO
        B
        LSTAT
        MEDV

Categorical variables:

Missing data count:
Series([], dtype: int64)

Duplicated data count:
0
```

Se realiza una separación y evaluación del modelo con un conjunto de prueba y un conjunto de validación (Train/Test/Validation). Para ello se utiliza la función `train_test_split()` con una división del 80% para el de entrenamiento y 20% para el de prueba.

```python
# Split the data into training (80%) and test (20%)
X_train, X_test, y_train, y_test = train_test_split(X_trans, y, test_size=0.2, random_state=42)
```

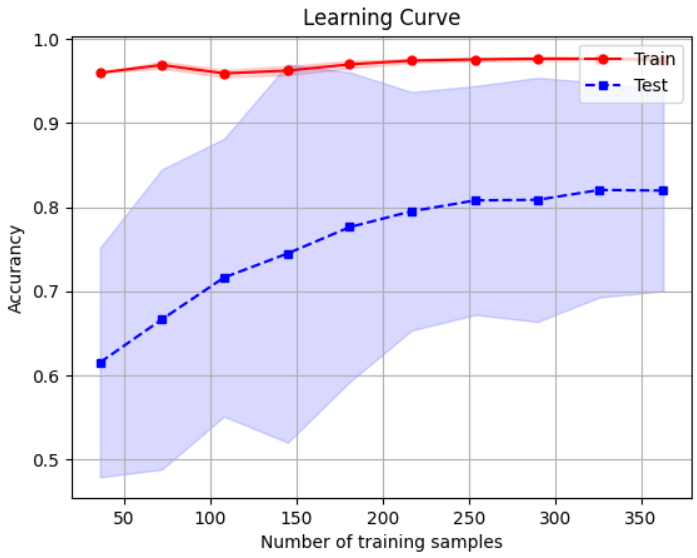
Random Forest Regressor permite hacer predicciones en problemas de regresión, como en este caso. Para esta primer prueba se utilizan los parámetros e hiperparámetos seleccionados de forma aleatoria, como se muestra a continuación:

```
# Random forest model
forest = RandomForestRegressor(n_estimators = 150,
                               criterion    = 'squared_error',
                               max_depth    = 20,
                               max_features = 5,
                               oob_score    = True,
                               n_jobs       = -1,
                               random_state = 42)
forest.fit(X_train, y_train)
```

Con los parámetros configurados se procede a entrenar el modelo y a evaluar su capacidad para realizar predicciones haciendo uso del conjunto de prueba, además se utiliza un gráfico con la curva de aprendizaje para tener una mejor visualización de la precisión entre los datos de entrenamiento y los de evaluación.

```
The model has the following metrics in train:
MSE:  1.638652069393149
RMSE:  1.2800984608197719
MAE:  0.8407289053905372
R2:  0.9786116237316711

The model has the following metrics in test:
MSE:  9.854752345920982
RMSE:  3.139227985655228
MAE:  2.0233580454403977
R2:  0.8153626472406732
```



Gracias a las métricas de evaluación y a la gráfica podemos notar que el modelo creado presenta un sobreajuste, ya que tiene un buen desempeño con los datos de entrenamiento pero su precisión es más baja con los datos de prueba. Esto quiere decir que el modelo memorizó los datos con los que entrenó pero no fue lo suficientemente capaz de generalizar las reglas para aplicarlas a nueva información.
La gráfica también nos da una idea de los niveles de sesgo y de varianza, en este caso se puede observar que hay bajo sesgo pero alta varianza, y el *trade-off* no es bueno.

Es claro que se necesita mejorar el modelo, para ello, se optimizarán los hiper parámetros haciendo uso de *Grid search*.

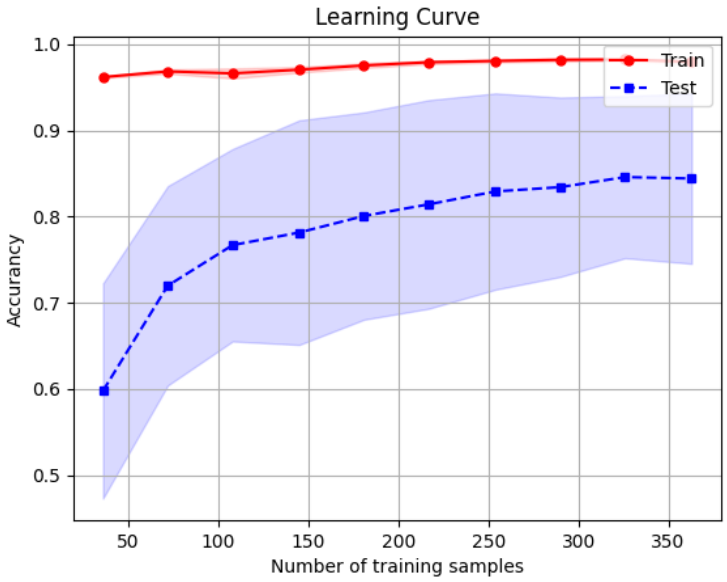
```
# Hyperparameter optimization
# Grid search
param_grid = {'n_estimators': [150],
              'max_features': [5, 7, 9],
              'max_depth'   : [None, 3, 10, 20]
              }

# Cross-validation
grid = GridSearchCV(
        estimator  = RandomForestRegressor(random_state=42),
        param_grid = param_grid,
        scoring    = 'neg_root_mean_squared_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5, n_repeats=3, random_state=42),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )
```

Posteriormente se construyó un nuevo modelo ajustando los parámetros. Los resultados se muestran a continuación.

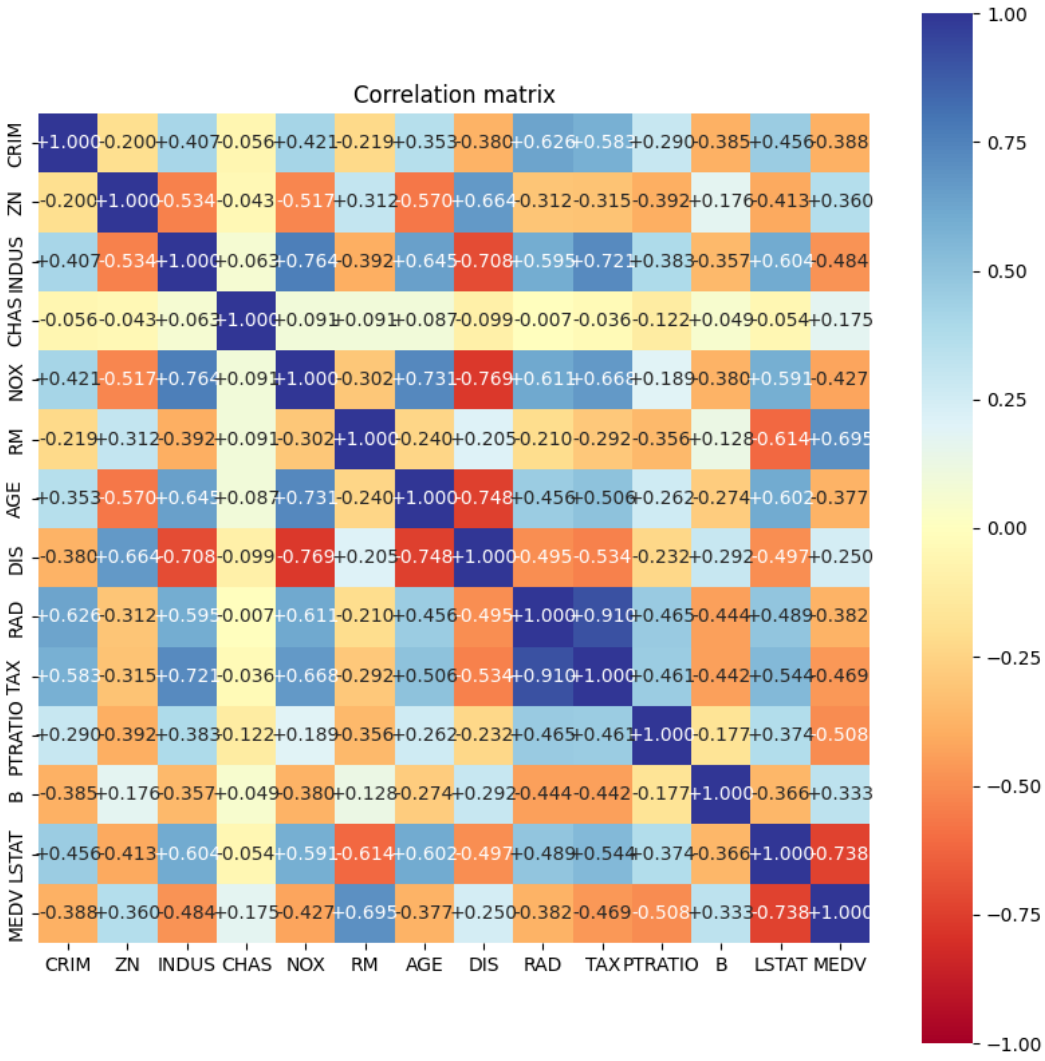
```
The model has the following metrics in train:
MSE:  1.976223970297028
RMSE:  1.4057823338970468
MAE:  0.9026089108910875
R2:  0.9744937811003618

The model has the following metrics in test:
MSE:  7.927145333333336
RMSE:  2.8155186615139556
MAE:  2.047411764705883
R2:  0.8710292448810978
```



Como se puede observar, se consiguió disminuir los errores y aumentar la precisión del modelo con los nuevos parámetros, sin embargo, el modelo aún presenta un sobreajuste, con bajo sesgo y alta varianza, por lo que el *trade-off* sigue sin ser lo suficientemente bueno. Para resolver esto, se necesita abordar el problema de otra forma, una solución puede ser crear más observaciones, es decir, que el conjunto de datos sea más amplio para abarcar más casos y así el modelo pueda generalizar de mejor forma las reglas.

Finalmente, con una matriz de correlación, se puede observar que los atributos que son más influyentes en el modelo son el LSTAT (% de estatus inferior de la población), seguido del RM (número medio de habitaciones por vivienda)


Correlation matrix
CRIM
ZN
INDUS
CHAS
NOX
RM
AGE
DIS
RAD
TAX
PTRATIO
B
LSTAT
MEDV
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00