Revista Ingeniería y Ciencia

Volumen 1, número 1, páginas 1-17

Estimación del índice de rotura de oleaje mediante técnicas de aprendizaje automático

Camilo Andres Cabrera Meneses[1]

Recepción: 09 de diciembre de 2022

Se aceptan comentarios y/o discusiones al artículo.

Resumen

La rotura de olas tiene una estrecha relación con varios procesos físicos en las regiones costeras, como el transporte de sedimentos, las corrientes costeras y la presión de las ondas de choque. Por lo tanto, es fundamental predecir con precisión el índice de rompimiento, como la altura y la profundidad de rompimiento de las olas, para múltiples aplicaciones de ingeniería costeras. Los estudios representativos sobre el rompimiento de las olas proporcionan muchas fórmulas empíricas para la predicción del índice de rompimiento, principalmente a través de experimentos con modelos hidráulicos. Sin embargo, las fórmulas empíricas existentes para romper el índice determinan los coeficientes de la ecuación asumida a través del análisis estadístico de datos bajo la suposición de una ecuación específica. En este trabajo, se aplican algoritmos representativos de aprendizaje automático supervisado de base lineal, para la predicción de índices de rotura: Altura de ola en rotura H_b y Profundidad en rotura h_b, a partir de datos experimentales publicados previamente. Los mejores ajustes para los datos de prueba (test) se obtuvieron con los modelos Random Forest Regressor (RFR) y Support Vector Regression (SVR), obteniendo un R^2 de 0.821 para H_b y 0.798 para h_b en el caso de RFR; y un R^2 de 0.826 para H_b y 0.719 para h_b en el caso de SVR.

Palabras claves: Aprendizaje automático, Regresión, Métodos supervisados, Olas, Rotura, Coeficiente de determinación.

[1] Estudiante de maestría en Recursos Hidráulicos, Departamento de Geociencias y Medio Ambiente, Universidad Nacional de Colombia, Medellín, Colombia, cacabreram@unal.edu.co .

1. Introducción

Las olas del océano que se propagan hacia la costa experimentan la disminución de la profundidad del agua, lo que acorta su longitud de onda y aumenta su altura. La ola continua ganando altura a medida que se acerca cada vez más a tierra, hasta llegar al punto en que no puede ganar más altura y rompe sin mantener su forma, lo que se define como el fenómeno de “Rotura de oleaje” El rompimiento de las olas causa varios eventos en el proceso de dispersión de la energía de las olas (transportada desde aguas profundas), como las olas de impacto, las corrientes costeras, las corrientes de retorno y el transporte de sedimentos, y también afecta a las embarcaciones y a la estabilidad de las estructuras costeras. La comprensión del proceso de rotura de las olas es fundamental para el diseño y mantenimiento de las estructuras costeras. Además, se considera un factor necesario para predecir y responder al transporte de sedimentos y cambios morfológicos en el área cercana a la costa. Particularmente entre las magnitudes físicas relacionadas con el rompimiento de las olas, las propiedades más importantes son la altura de la ola y la profundidad del agua en la posición del rompimiento de la ola, que se definen como "Altura de rompimiento de la ola" H_b y "Profundidad de rompimiento del agua" h_b, respectivamente (ver Figura 1). Estos valores cuantitativos, llamados índices de rompimiento de olas, indican el punto de inicio de la ola rompiente y la altura máxima de la ola cerca de la costa, y se han realizado numerosos estudios para estimar estos valores. Sin embargo, debido a las fuertes características turbulentas y la no linealidad, la observación y predicción de las olas rompientes costeras es compleja, y muchos estudios aún se ven cuestionados por este tema.

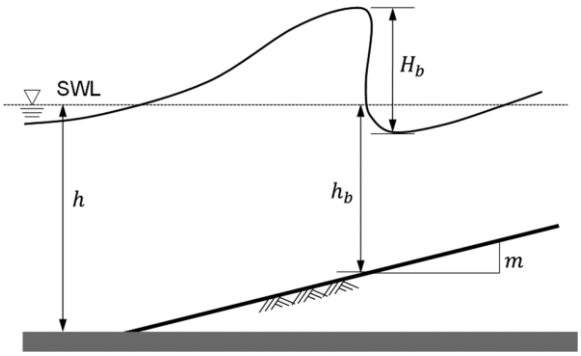

Figura 1. Esquema de definición de ola en rompimiento incipiente. Tomado de: (Choi et al., 2020).

Numerosos estudios han intentado estimar fórmulas empíricas de rompimiento de olas para la altura del rompimiento de olas y la profundidad del agua mediante la reproducción de varias condiciones de olas en experimentos de laboratorio de canal

de olas y la obtención de transformaciones de olas consecutivas dependiendo de la profundidad del agua. Sin embargo, el rendimiento constante se vio limitado en condiciones generalizadas, ya que estas fórmulas empíricas para romper olas dependían de ciertos datos de experimentos de laboratorio. Además, los métodos y formas básicas del índice de rompimiento utilizados para desarrollar las fórmulas empíricas de rompimiento de olas eran diferentes entre sí. En particular, las fórmulas empíricas de rompimiento de olas sugeridas por Goda (2007), Robertson et al. (2018) y otros tenían un alto grado de reproducibilidad porque la altura estimada de la ola rompiente y la profundidad del agua se incluyeron como valores de entrada para las ecuaciones, pero eso los hace insuficientes para las aplicaciones de ingeniería costera.

El aprendizaje automático de máquinas surge como una alternativa para estimar los índices de rotura a través de datos de experimentales recolectados en trabajos anteriores. A partir de variables como la pendiente del fondo en el que se propaga la ola, su altura en aguas profundas y la longitud o periodo de la onda es posible entrenar modelos de aprendizaje automático para la predicción de la altura de ola y la profundidad de agua en rotura.

2. Metodología

2.1. Datos

En este trabajo, se utilizaron datos experimentales de laboratorio de canales de olas publicados abiertamente para entrenar diferentes modelos de aprendizaje automático predicción del índice de olas rompientes. Una parte sustancial de los datos se obtuvo de Gaughan (1973) y Smith & Kraus (1990), dado que la fecha de publicación de estos trabajos es antigua, se requirió hacer una digitalización de estos datos para conformar la base de datos inicial (ver Figura 2) . Los datos restantes se recopilaron de Lara et al. (2006), Kouvaras & Dhanak (2018) y Xie et al. (2019). Cada conjunto de datos consta de valores de entrada, como la altura de las olas en aguas profundas, la pendiente del fondo, el período o la longitud de onda, y las alturas de las olas rompientes y las profundidades del agua obtenidas de experimentos de laboratorio de canal de olas con pendientes distribuidas entre 0,009 y 0,225.

m	T sec	h_i cm	H_i cm	H_b cm	H_b swl cm	h_b cm	H_∞/L_∞	H_b/H_∞	H_b/h_b
				2. Iversen, continued					
0.033	2.10	43.9	3.50	6.56		8.37	0.0052	1.83	0.782
0.033	2.29	43.5	3.54	7.01		8.55	0.0042	2.04	0.822
0.033	2.52	43.5	3.57	6.10		8.07	0.0035	1.76	0.755
0.033	2.52	43.2	2.84	5.79		7.00	0.0027	2.16	0.826
0.033	2.65	43.0	2.96	5.49		7.44	0.0025	2.00	0.737
0.050	1.40	54.9	10.07	12.8		16.14	0.0360	1.16	0.792
0.050	1.50	48.8	9.08	12.2		14.00	0.0280	1.24	0.870
0.050	1.59	48.8	7.80	12.2		14.63	0.0210	1.48	0.834
0.050	1.89	47.8	6.85	11.6		13.40	0.0130	1.60	0.864
0.050	2.24	47.8	5.88	11.0		11.90	0.0076	1.85	0.925
0.050	1.04	53.3	11.68	10.7		16.50	0.0730	0.87	0.649
0.050	1.15	48.8	9.30	9.45		11.90	0.0480	1.05	0.795
0.050	1.26	47.8	7.92	10.1		10.40	0.0350	1.16	0.971
0.050	1.33	48.8	7.25	9.14		10.40	0.0290	1.14	0.884
0.050	1.41	47.5	6.15	8.24		10.05	0.0220	1.20	0.819
0.050	1.67	46.0	5.43	8.24		8.84	0.0130	1.45	0.931
0.050	1.93	45.4	4.39	7.62		7.62	0.0079	1.66	1.000
0.050	0.74	47.2	6.52	5.79		8.84	0.0767	0.88	0.660
0.050	0.93	45.7	6.29	6.40		8.25	0.0480	0.99	0.780
0.050	1.03	45.7	5.65	5.49		7.62	0.0360	--	0.720
0.050	1.12	45.7	5.03	5.79		7.02	0.0270	1.10	0.826
0.050	1.17	45.7	4.42	6.10		6.41	0.0220	1.30	0.953
0.050	1.34	45.7	3.35	4.27		4.87	0.0130	1.17	0.875
0.050	1.55	44.8	2.86	4.57		5.48	0.0083	1.47	0.834

Figura 2. Visualización de registros de laboratorio consignados en Smith & Kraus (1990).

Una vez digitalizados los datos de los reportes antiguos y consolidados con los datos de los artículos más actuales se procede a cargar la base de datos a Python para su posterior tratamiento y análisis, así como el entrenamiento y validación de los modelos. La base de datos esta compuesta por 277 registros; cuatro variables independientes: pendiente del fondo (m), periodo de ola (T), Longitud de onda (Lo) y Altura de ola en aguas profundas (Ho); dos variables independientes: Altura de ola en el punto de rotura (Hb) y Profundidad del agua en el punto de rotura (hb). La Figura 3 muestra los primeros registros de la base de datos en formato DataFrame y una descripción con los principales parámetros estadísticos de cada variable.

	m (slope)	T (s)	Lo (cm)	Ho (cm)	Hb (cm)	hb (cm)
0	0.0125	1.2	224.6	8.8	8.6	12.5
1	0.0125	1.2	224.6	9.1	9.1	12.5
2	0.0125	1.2	224.6	12.0	12.2	13.8
3	0.0125	1.2	224.6	14.8	14.2	16.3
4	0.0125	1.2	224.6	11.7	11.6	16.3

	m (slope)	T (s)	Lo (cm)	Ho (cm)	Hb (cm)	hb (cm)
count	277.000	277.000	277.000	277.000	277.000	277.000
mean	0.059	1.661	497.681	10.581	12.570	14.367
std	0.054	0.793	677.935	10.729	12.504	16.125
min	0.003	0.700	76.400	1.036	2.438	3.048
25%	0.012	1.200	224.600	7.200	9.144	9.900
50%	0.050	1.500	351.300	9.800	11.440	12.500
75%	0.100	2.000	613.000	12.200	13.700	16.300
max	0.200	6.000	5623.560	137.000	150.000	200.000

Figura 3. Visualización y descripción de la base de datos.

2.2. EDA – Análisis Exploratorio de Datos

Con el dataset cargado en Python se procede realizar el análisis exploratorio de datos para determinar relaciones entre variables independientes y correlaciones con las

variables dependientes. En primer lugar, se efectúan los grafico de histogramas y boxplot para cada una de las 4 variables independientes. La Figura 4 muestra los gráficos univariados para cada variable, podemos observar los valores para la pendiente se concentran en la franja entre 0.03 y 0.1, sin presencia de datos atípicos en el registro, para las demás variables se observa concentración en los valores bajos de la distribución con presencia de pocos datos extremos. Los valores de altura de ola se concentran entre 1 y 20 cm, con un par de datos marcados fuertemente como extremos.

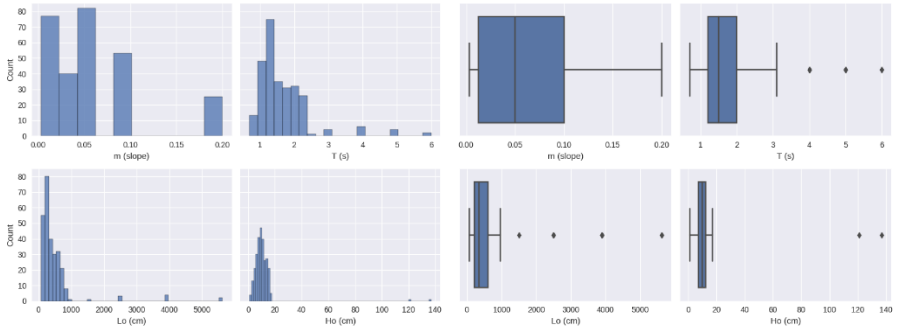

Figura 4. Gráficos de análisis univariado de variables independientes (Histograma - BoxPlot)

Posteriormente procedemos a efectuar el análisis multivariado entre las variables del conjunto de datos para determinar correlaciones entre ellas. La Figura 5 muestra la matriz de correlación generada con la librería *seaborn* de Python. En la matriz se puede observar una correlación alta entre las variables independientes T y Lo, a su vez se muestra una alta correlación entre la variable independiente Ho, y las variables dependientes que se busca predecir, lo cual puede indicar que esta sería una variable calve para la predicción futura.

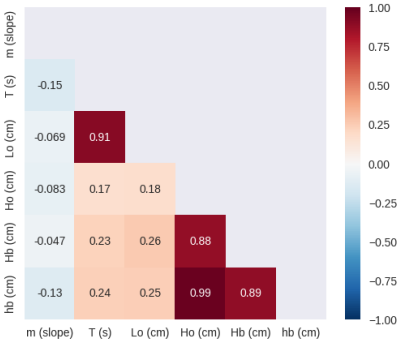

Figura 5. Matriz de correlación.

2.3. Selección de variables

Para la selección de variables empleamos el método **Recursive Feature Elimination** (RFE), en el cual se utiliza un modelo de *Machine Learning* para seleccionar las variables, eliminando las de menor importancia en un proceso iterativo. El argumento *step* corresponde al número de variables eliminadas en cada iteración. El código para la ejecución del método es el siguiente:

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression

rfe=RFE(estimator=LinearRegression(),n_features_to_select = 3, step = 1)
fit=rfe.fit(X,Y)
print(fit.n_features_)
print(fit.support_)
print(X.columns[fit.support_])
print(fit.ranking_)
```

El método da como resultado la eliminación de la variable *Lo (cm)*, la cual tenía una alta correlación con la variable Periodo (T). En lo relacionado a la física del problema sabemos que existe una relación entre la Longitud y Periodo de una ola, en el marco de la teoría lineal de oleaje, por esta razón la eliminación de la variable Longitud puede resultar adecuada ya que su información del problema está contenida en *T (s)*.

2.4. Validación cruzada

Para garantizar una evaluación acertada de los modelos de aprendizaje automático y un buen desempeño de estos se debe, como buena práctica, dividir los datos en subgrupos de entrenamiento (train) y validación (test), de forma que se entrene el modelo con el conjunto de entrenamiento y se valide con los datos de prueba. Para dividir los datos se hace uso de la función *train_test_split* de la librería **scikit-learn** con un 80% del total de los datos para entrenamiento y 20% para prueba**.** Las dimensiones de los vectores resultantes se muestran a continuación, donde Y1 corresponde a la variable dependiente Hb (cm) y Y2 a hb (cm).

```
Dimensiones de la matriz para entrenar: (221, 4)
Dimensiones del vector Y1 para entrenar: (221,)
Dimensiones del vector Y2 para entrenar: (221,)
Dimensiones de la matriz para validar: (56, 4)
Dimensiones del vector Y1 para validar: (56,)
Dimensiones del vector Y2 para validar: (56,)
```

Adicionalmente se hace uso del algoritmo *K-fold Cross Validation* de la librería **scikit-learn,** el cual divide los datos en un número de K de subconjuntos (k = 5 ó k = 10). Cada partición es denominada un *fold*. El algoritmo es entonces entrenado con K-1 subconjuntos y un subconjunto es utilizado para validar. Esto es k veces repetido por lo que se obtienen k valores de *score*. El propósito es evaluar que tan bien generaliza un modelo determinado ante el suministro de otros datos diferentes al entrenamiento. A continuación, se muestra los resultados obtenidos al aplicar *k-fold* para un modelo de Random Forest Regressor con los datos de entrenamiento para las variables Y1 y Y2.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold

kfold = KFold(n_splits=5, shuffle= True,random_state=1)
model = RandomForestRegressor()
results_Y1 = cross_val_score(model, X_train, Y1_train, cv=kfold,
scoring='r2')
results_Y2 = cross_val_score(model, X_train, Y2_train, cv=kfold,
scoring='r2')
```

```
Results for RF Regressor Y1:
[ 0.78760142 -3.35498724  0.96737731 -0.03360597 -1.66531173]
Mean: -0.6597852418932965
Std_D: 1.6376742016992487

Results for RF Regressor Y2:
[0.89105573 0.59792919 0.83473299 0.82292038 0.82165194]
Mean: 0.7936580459133242
Std_D: 0.10111694885816751
```

Podemos observar que la métrica para la variable Y1 fluctúa entre valores bastante diferentes, lo cual se refelaja en el R2 promedio bajo y una desviación estándar alta. Esto muestra indicios de que la variable dependiente Y1 puede presentar problemas de varianza dada su alta fluctuación en las métricas ante el suministro de nuevos conjuntos de datos. En cambio, la variable Y2 presenta una métrica en promedio alta (0.794) y una desviación estándar baja, lo cual señala que esta variable tendrá un ajuste mejor que su variable auxiliar.

2.5. Modelos supervisados

2.5.1. Regresión Linear

Para aplicación del método de Regresión Lineal se hace uso de la función *OLS* de la librería **statsmodels**. La instanciación del método se efectúa de manera sencilla a través de las siguientes líneas de código.

```
lm_Y1=sm.OLS(Y1_train_org, sm.add_constant(X_train_org)).fit()
print("****************** FOR Y1 - Train Data ****************")
print(lm_Y1.summary())
```

Se agrega el intercepto a la función para dejar que este parámetro absorba una fracción del error producto del ajuste. Los resultados del ajuste para los datos de entrenamiento se muestran en la Figura 6. Podemos observar un muy buen ajuste para los datos de entrenamiento en el caso de la variable Y2 con un R^2 de 0.986, lo cual concuerda con lo hallado en validación cruzada de la subsección anterior, además todos los coeficientes son estadísticamente significativos (valor p <0.05), dando certeza del buen ajuste de este modelo. Para el modelo ajustado a los datos de entrenamiento de la variable Y1 encontramos un buen R^2 de 0.786, sin embargo, los coeficientes del intercepto y la pendiente (m) no son significativos.

```
********** FOR Y1 - Train Data **********
                            OLS Regression Results
==============================================================================
Dep. Variable:                Hb (cm)   R-squared:                       0.786
Model:                            OLS   Adj. R-squared:                  0.783
Method:                 Least Squares   F-statistic:                     265.4
Date:                Thu, 08 Dec 2022   Prob (F-statistic):           2.58e-72
Time:                        01:50:23   Log-Likelihood:                -724.28
No. Observations:                 221   AIC:                             1457.
Df Residuals:                     217   BIC:                             1470.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.9271      1.236     -0.750      0.454      -3.364       1.510
m (slope)      7.9268      8.428      0.941      0.348      -8.684      24.537
T (s)          1.4251      0.591      2.413      0.017       0.261       2.589
Ho (cm)        1.0152      0.038     26.986      0.000       0.941       1.089
==============================================================================
Omnibus:                      451.620   Durbin-Watson:                   1.944
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           259305.000
Skew:                          12.227   Prob(JB):                         0.00
Kurtosis:                     169.018   Cond. No.                         312.
==============================================================================
```

```
********** FOR Y2 - Train Data **********
                            OLS Regression Results
==============================================================================
Dep. Variable:                hb (cm)   R-squared:                       0.986
Model:                            OLS   Adj. R-squared:                  0.986
Method:                 Least Squares   F-statistic:                     5023.
Date:                Thu, 08 Dec 2022   Prob (F-statistic):          3.77e-200
Time:                        01:58:19   Log-Likelihood:                -480.56
No. Observations:                 221   AIC:                             969.1
Df Residuals:                     217   BIC:                             982.7
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -3.1716      0.410     -7.728      0.000      -3.980      -2.363
m (slope)    -10.9767      2.798     -3.924      0.000     -16.491      -5.463
T (s)          1.6244      0.196      8.285      0.000       1.238       2.011
Ho (cm)        1.4659      0.012    117.388      0.000       1.441       1.491
==============================================================================
Omnibus:                       18.783   Durbin-Watson:                   2.198
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               34.372
Skew:                          -0.449   Prob(JB):                     3.44e-08
Kurtosis:                       4.710   Cond. No.                         312.
==============================================================================
```

Figura 6. Resultados de la aplicación de OLS para predicción de variable Hb (cm) (izquierda) y hb (cm) (derecha)

Con los modelos ajustados se procede a realizar las predicciones de los valores de Y1 y Y2, a partir de los datos de X reservados para test. La Figura 7 muestra los gráficos de dispersión para las dos variables para los subconjuntos de prueba (test). Los R^2 con

los datos de prueba obtenidos para la variable Y1 y Y2 son 0.800 y 0.741 respectivamente.

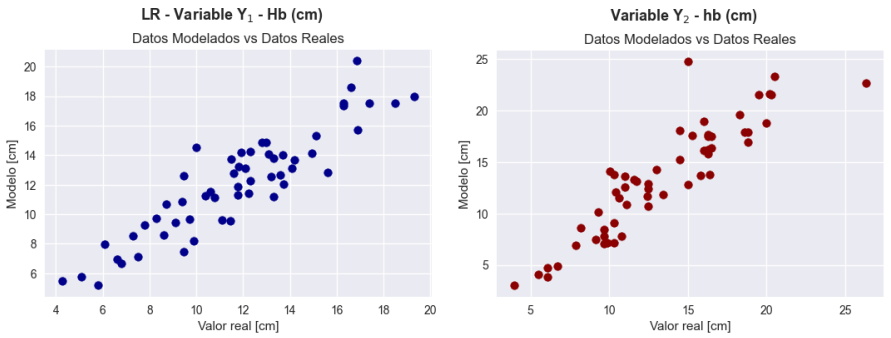

Figura 7. Scatter plot entre los datos de prueba y los datos predichos por el modelo de Regresión Lineal.

Con la finalidad de comprobar que las consideraciones de un modelo de regresión lineal se cumplan, se procede a graficar los valores ajustados vs los residuales para cada uno de los dos modelos ajustados, además se grafica el histograma y Q-Q plot de los residuos para cada variable (ver Figura 8 y Figura 9).

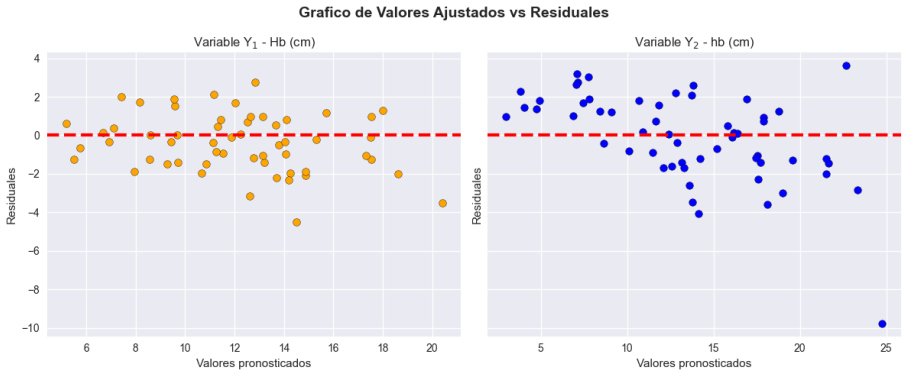

Figura 8. Scatter plot de valores pronosticados vs residuales de la Regresión Lineal

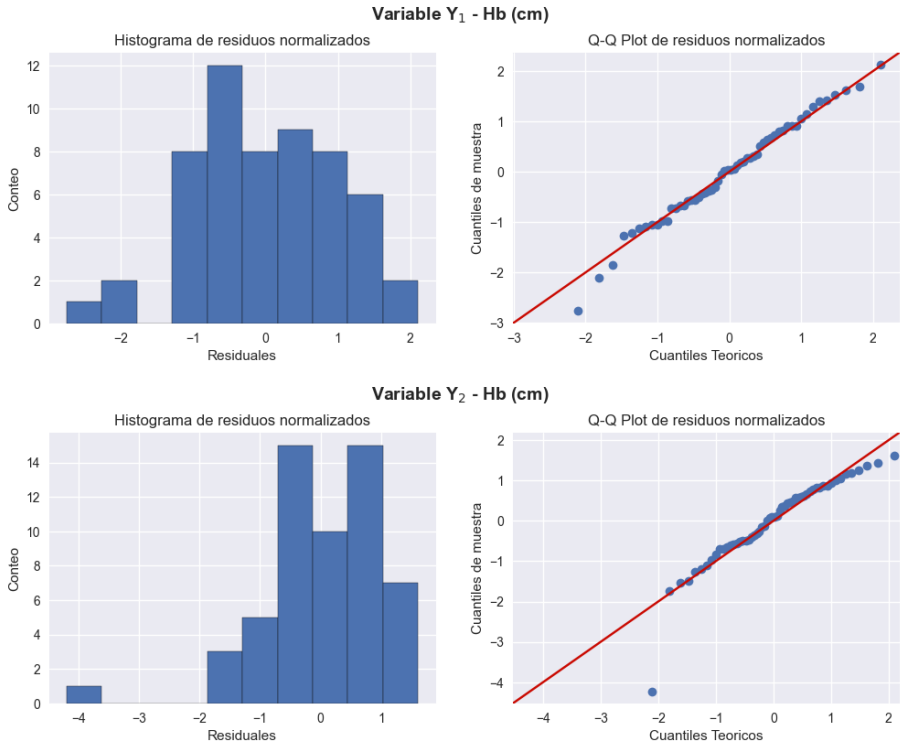

Figura 9. Gráficos de diagnóstico de normalidad para residuales del modelo de Regresión Lineal.

Podemos observar de la Figura 8 que los resídelas para ambos modelos se concentran alrededor de cero, además su varianza se mantiene constante con concentraciones de los residuales entre -4 y 4, esto demuestra la homocedasticidad del conjunto de residuales. Además los gráficos de diagnostico de normalidad mostrados en la Figura 9 muestran que para ambos modelos los residuales siguen aproximadamente una distribución normal, concentrándose alrededor de la línea de 45° en el Q-Q plot y un histograma aproximadamente simétricos centrado en cero. Lo anterior indica que los modelos ajustados obtienen buenas métricas de ajuste, tanto en entrenamiento como en validación y cumplen las consideraciones teóricas de un modelo de Regresión Lineal.

2.5.2. KNN - Regressor

Como primera medida para una correcta aplicación del modelo de Vecinos Más Cercanos en su variante de regresión debemos hallar los hiperparámetros óptimos que me permitan obtener las mejores métricas para train y test. El hiperparámetro más importante en este modelo es el número de vecino (n_neighbors), por lo tanto, se

procede a graficar la curva de validación en función de este parámetro para encontrar su valor optimo. Los puntos de la curva se determinan mediante la función *validation_curve* de la librería **scikit-learn**. La Figura 10 muestra las curvas de validación para las variables Y1 y Y2. Podemos observar que para el caso de la variable Y1, el numero de vecinos que genera los valores más altos del score de train y test en conjunto es 2, mientras que para la variable Y2 es 3.

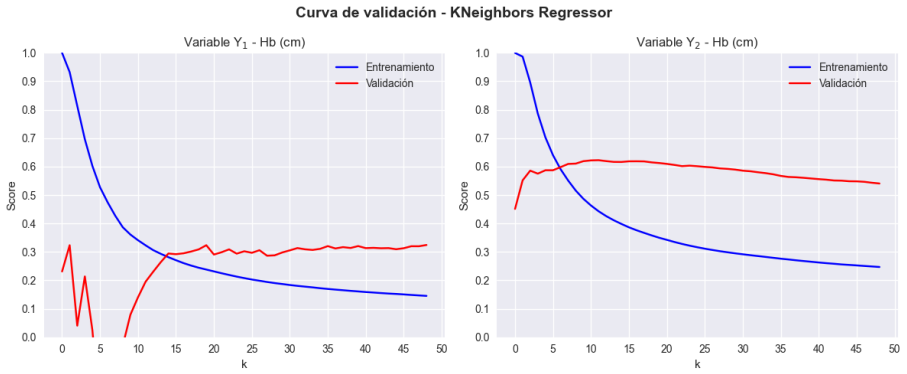

Figura 10. Curva de validación en función del hiperparámetro n_neighbors.

Para aplicación del método de KNN-Regressor se hace uso de la función *KNeighborsRegressor* de la librería **scikit-learn**. La instanciación del método se efectúa de manera sencilla a través de las siguientes líneas de código.

```
KNN_Y1 = KNeighborsRegressor(n_neighbors=2)
KNN_Y1.fit(X_train_org,Y1_train_org)
print('R2 Y1 - KNN:',KNN_Y1.score(X_train_org,Y1_train_org))
```

Los resultados del coeficiente de determinación R^2 para los datos de entrenamiento para el modelo KNN de la variable Y1 y Y2 son 0.944 y 0.895 respectivamente. De nuevo el juste para la variable Y2 muestra un ajuste muy superior en comparación con la variable Y1. Con los modelos ajustados se procede a realizar las predicciones de los valores de Y1 y Y2, a partir de los datos de X reservados para test. La Figura 11Figura 7 muestra los gráficos de dispersión para las dos variables para los subconjuntos de prueba (test). Los R^2 con los datos de prueba obtenidos para la variable Y1 y Y2 son 0.743 y 0.780 respectivamente.

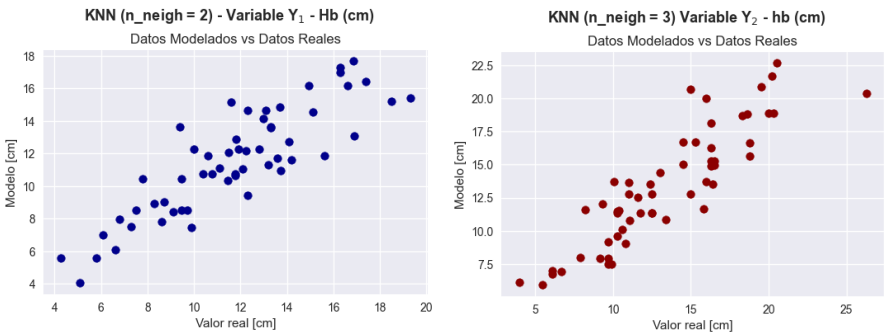

Figura 11. Scatter plot entre los datos de prueba y los datos predichos por el modelo de KNN-Regressor

2.5.3. Support Vector Regression

Como primera medida para una correcta aplicación del modelo de Support Vector Machine en su variante de regresión debemos hallar los hiperparámetros óptimos que me permitan obtener las mejores métricas para train y test. Entre los hiperparámetros más importante en este modelo se encuentran el *kernel*, *C* y *degree*, por lo tanto, se procede a ajustar diferentes modelos con diferentes hiperparámetros dentro de un proceso iterativo mediante la función *RandomizedSearchCV* de la librería **scikit-learn**. La selección de hiperparámetros para Support Vector Regression se efectúa mediante el siguiente código.

```
param_distribs = {'kernel' : ('linear', 'poly', 'rbf', 'sigmoid'), 'C' :
np.arange(0,100,1),'degree' : [2,3,4,5,6,7]}

rnd_search = RandomizedSearchCV(SVR(), param_distributions=param_distribs,
n_iter=100, cv=5, scoring='r2', verbose=2, random_state=42)

rnd_search.fit(df_X, Y1)
print(rnd_search.best_score_)
print(rnd_search.best_estimator_)
```

Obteniendo como resultado las siguientes modelos para ajustar con los datos de train:

```
svm_reg_Y1 = SVR(kernel="linear",C=49)
svm_reg_Y1.fit(X_train_org,Y1_train_org)

svm_reg_Y2 = SVR(kernel="linear",C=15)
svm_reg_Y2.fit(X_train_org,Y2_train_org)
```

Los resultados del coeficiente de determinación R^2 para los datos de entrenamiento para el modelo SVR de la variable Y1 y Y2 son 0.776 y 0.985 respectivamente. De nuevo el juste para la variable Y2 muestra un ajuste muy superior en comparación con

la variable Y1. Con los modelos SVR ajustados se procede a realizar las predicciones de los valores de Y1 y Y2, a partir de los datos de X reservados para test. La Figura 12Figura 7 muestra los gráficos de dispersión para las dos variables para los subconjuntos de prueba (test). Los R^2 con los datos de prueba obtenidos para la variable Y1 y Y2 son 0.826 y 0.719 respectivamente.

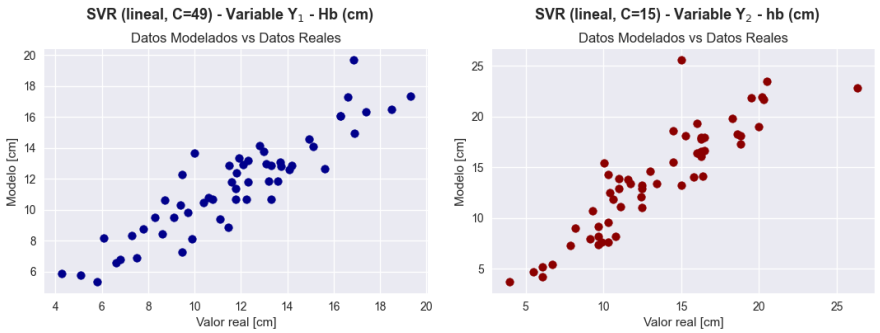

Figura 12. Scatter plot entre los datos de prueba y los datos predichos por el modelo de SV Regression.

2.5.4. Random Forest Regressor

Como primera medida para una correcta aplicación del modelo de Random Forest Regressor debemos hallar los hiperparámetros óptimos que me permitan obtener las mejores métricas para train y test. Entre los hiperparámetros más importante en este modelo se encuentran el *n_estimators*, *max_depth*, *min_samples_split*, *min_samples_leaf* y *bootstrap*, por lo tanto, se procede a ajustar diferentes modelos con diferentes hiperparámetros dentro de un proceso iterativo mediante la función *RandomizedSearchCV* de la librería **scikit-learn**. La selección de hiperparámetros para Random Forest Regressor se efectúa mediante el siguiente código.

```
# Define params and ranges
n_estimators = [int(x) for x in np.linspace(start = 200, stop=2000,num = 10)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 100, num = 10)]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]

# Create the random grid
random_grid = {'n_estimators': n_estimators, 'max_features':
max_features,'max_depth': max_depth,'min_samples_split':
min_samples_split,'min_samples_leaf': min_samples_leaf, 'bootstrap':
bootstrap}
```

```
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator=RandomForestRegressor(),
param_distributions=random_grid, n_iter = 100, scoring =
'neg_mean_absolute_error', cv = 5, verbose=2, random_state=42, n_jobs=-1)

# Fit the random search model
print(rf_random.fit(df_X, Y1))
rf_random.best_params_
```

Obteniendo como resultado las siguientes modelos para ajustar con los datos de train:

```
RFR_Y1 = RandomForestRegressor(n_estimators=1400,min_samples_split=2,
                               min_samples_leaf=2,max_features='auto',
                               max_depth=10,bootstrap=False)
RFR_Y1.fit(X_train_org,Y1_train_org)


RFR_Y2 = RandomForestRegressor(n_estimators=2000,min_samples_split=10,
                               min_samples_leaf=2,max_features='auto',
                               max_depth=90,bootstrap=False)
RFR_Y2.fit(X_train_org,Y2_train_org)
```

Los resultados del coeficiente de determinación R^2 para los datos de entrenamiento para el modelo RFR de la variable Y1 y Y2 son 0.906 y 0.994 respectivamente. De nuevo el juste para la variable Y2 muestra un ajuste muy superior en comparación con la variable Y1. Con los modelos RFR ajustados se procede a realizar las predicciones de los valores de Y1 y Y2, a partir de los datos de X reservados para test. La Figura 13Figura 7 muestra los gráficos de dispersión para las dos variables para los subconjuntos de prueba (test). Los R^2 con los datos de prueba obtenidos para la variable Y1 y Y2 son 0.821 y 0.798 respectivamente.

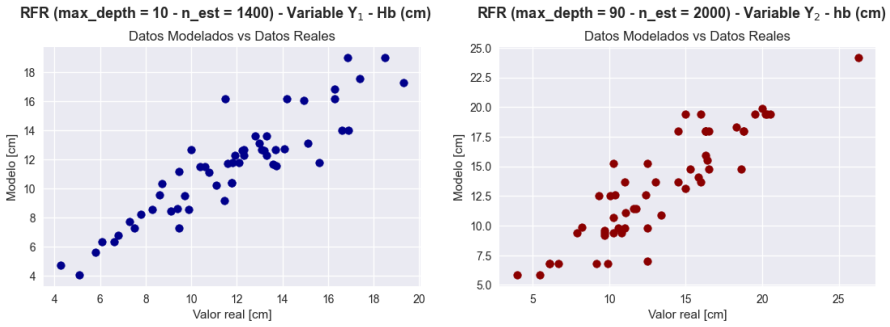

Figura 13. Scatter plot entre los datos de prueba y los datos predichos por el modelo de RF Regressor

3. Resultados y conclusiones

A partir de los resultados obtenidos para cada modelo se construye la Tabla 1 donde se muestran las métricas de ajuste para los datos de entrenamiento y las métricas de evaluación para los datos de testeo. Se puede observar que para el caso de la variable Altura de Ola en el punto de rotura Y1, los modelos que obtienen una mejor métrica con los datos de testeo son Support Vector Regression y Random Forest Regresor respectivamente. Adicionalmente para la variable Y2 se obtienen buenas metricas de evaluación en testeo, por esta razón se eligen estos dos modelos para predecir con la totalidad de las observaciones y establecer la comparativa global del modelo vs las observaciones.

Tabla 1. Resumen de métricas para elección de modelo.

MODELO	Variable Y1 (H_b)		Variable Y2 (h_b)	
	R2 - Train	R2- Test	R2 - Train	R2 - Test
Linear Regression	0.786	0.800	0.986	0.741
KNN - Regresor	0.944	0.743	0.895	0.780
Support Vector Regression	0.775	0.826	0.985	0.719
Random Forest Regresor	0.906	0.821	0.994	0.798

La Figura 14 muestra los scatter plots comparando entre los valores ajustados y valores reales para todo el conjunto de datos. Podemos observar que en ambas variables el modelo SVR genera puntos más alejados de la diagonal central, esto se corresponde con los valores de R^2 calculados para ambos modelos los cuales se detallan en la Tabla 2 .

Tabla 2. Coeficientes de determinación R2 para modelos RFR y SVR.

Modelo / **Variable**	*RFR*	*SVR*
Variable Y1	0.9050	0.7766
Variable Y2	0.9908	0.9808

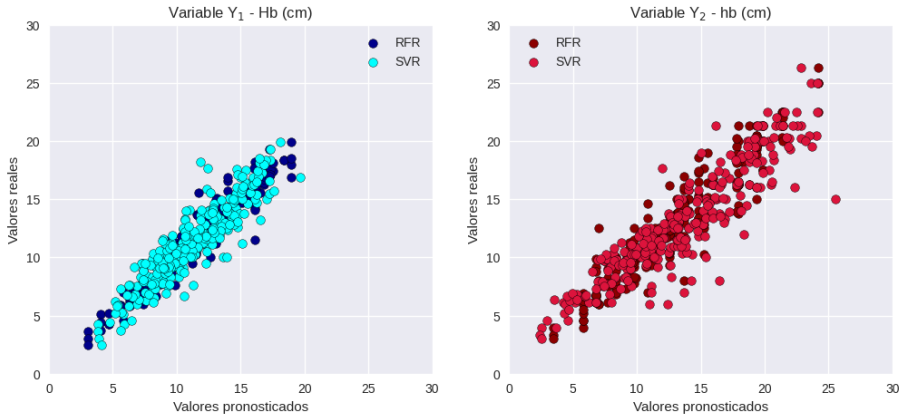

Figura 14. Gráficos de dispersión empleando modelos de SVR y RFR para predicción.

Finalmente se puede concluir que el modelo Random Forest Regresor genera muy buenos pronósticos tanto para Alturas como para Profundidades de rotura, obteniendo coeficientes de determinación superiores a 0.9 al validarse con la totalidad de los registros.

4. Referencias

- Choi, B.-J., Park, C.-W., Cho, Y.-H., Kim, D.-S., & Lee, K.-H. (2020). A Proposal of New Breaker Index Formula Using Supervised Machine Learning. *Journal of Korean Society of Coastal and Ocean Engineers*, *32*(6), 384–395. https://doi.org/10.9765/KSCOE.2020.32.6.384
- Gaughan, M. K. (1973). *Breaking waves : a review of theory and measurements*. https://ir.library.oregonstate.edu/concern/graduate_thesis_or_dissertations/zs25xc86z?locale=en
- Goda, Y. (2007). Reanalysis of Regular and Random Breaking Wave Statistics. *Https://Doi.Org/10.1142/S0578563410002129*, *52*(1), 71–106. https://doi.org/10.1142/S0578563410002129
- Kouvaras, N., & Dhanak, M. R. (2018). Machine learning based prediction of wave breaking over a fringing reef. *Ocean Engineering*, *147*(September 2017), 181–194. https://doi.org/10.1016/j.oceaneng.2017.10.005
- Lara, J. L., Losada, I. J., & Liu, P. L. F. (2006). Breaking waves over a mild gravel slope: Experimental and numerical analysis. *Journal of Geophysical Research: Oceans*, *111*(C11), 11019. https://doi.org/10.1029/2005JC003374
- Robertson, B., Gharabaghi, B., & Hall, K. (2018). Prediction of Incipient

Breaking Wave-Heights Using Artificial Neural Networks and Empirical Relationships. *Https://Doi.Org/10.1142/S0578563415500187*, *57*(4). https://doi.org/10.1142/S0578563415500187

- Smith, E. R., & Kraus, N. C. (1990). *Laboratory Study on Macro-Features of Wave Breaking Over Bars and Artificial Reefs*. https://apps.dtic.mil/sti/citations/ADA225689
- Xie, W., Shibayama, T., & Esteban, M. (2019). A semi-empirical formula for calculating the breaking depth of plunging waves. *Https://Doi.Org/10.1080/21664250.2019.1579459*, *61*(2), 199–209. https://doi.org/10.1080/21664250.2019.1579459