



Instituto Politécnico Nacional
Escuela Superior de Cómputo

Proyecto Regresión Lineal

Unidad de Aprendizaje: Data Mining
Grupo: 3CV6

Alumno(s):

Lara Cázares Jaime Arturo
Ramos Diaz Enrique

Profesor(a):

Ocampo Botello Fabiola

23 de noviembre de 2019

Índice

Enunciado del problema	3
Diccionario de datos	3
Desarrollo: Proceso KDD	4
Carga de la base de datos	4
Limpieza de datos	4
Integración y selección de datos	4
Correlaciones entre atributos	4
Transformación de los datos	5
Minería de datos	6
Partición de los datos	6
Modelo de Regresión Lineal: Aprendizaje	8
Modelo de Regresión Lineal: Predicción	10
Evaluación del modelo	12
Uso manual del modelo	13
Conclusiones	13
Anexos	14

Enunciado del problema

IRIS es quizás la base de datos más conocida que se encuentra en la literatura de reconocimiento de patrones. El conjunto de datos contiene 3 clases de 50 instancias cada una, donde cada clase se refiere a un tipo de planta de iris. Una clase es linealmente separable de las otras 2; estas últimas NO son linealmente separables entre sí.

El objetivo es predecir la anchura del pétalo en centímetros de una flor iris con base en la longitud del mismo, utilizando modelos predictivos de regresión lineal.

Diccionario de datos

Dataset obtenido de la siguiente página: <https://www.kaggle.com/saurabh00007/iriscsv>

Atributo	Significado	Tipo de datos y dominio
SepalLengthCm	Longitud del sépalo de la flor en centímetros. Los sépalos son cada una de las hojas, generalmente de color verde que forman el cáliz de la flor.	Tipo de dato numérico con un rango de [4.3, 7.9]
SepalWidthCm	Ancho del sépalo de la flor en centímetros.	Tipo de dato numérico con un rango de [2.0, 4.4]
PetalLengthCm	Longitud del pétalo de la flor en centímetros. Los pétalos son cada una de las partes que componen la corola de la flor.	Tipo de dato numérico con un rango de [1.0, 6.9]
PetalWidthCm	Ancho del pétalo de la flor en centímetros.	Tipo de dato numérico con un rango de [0.1, 2.5]
Species	Clasificación del tipo de iris. también es tomado como el target.	Tipo de dato nominal (cadena de texto) con el dominio [iris-virginica, iris-setosa, iris-versicolor]

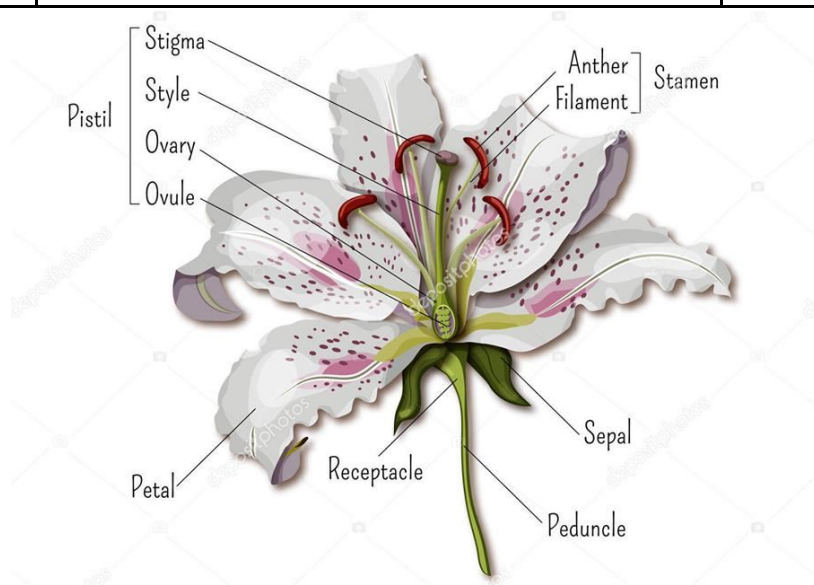


Figura 1. Partes de una flor

Desarrollo: Proceso KDD

Carga de la base de datos

La base de datos a utilizar tiene como nombre **iris.csv**

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
Id					
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
...
146	6.7	3.0	5.2	2.3	Iris-virginica
147	6.3	2.5	5.0	1.9	Iris-virginica
148	6.5	3.0	5.2	2.0	Iris-virginica
149	6.2	3.4	5.4	2.3	Iris-virginica
150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

Figura 2. Registros de la base de datos **iris.csv**

Limpieza de datos

Si revisamos la existencia de registros nulos, notaremos que no los hay, por lo que no es necesario hacer una limpieza en la base de datos.

Integración y selección de datos

Solo nos basta con la base de datos principal **iris.csv**.

Para seleccionar qué atributos vamos a utilizar, primero revisamos las correlaciones entre ellos.

Correlaciones entre atributos

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

Figura 3. Valores de correlaciones entre atributos

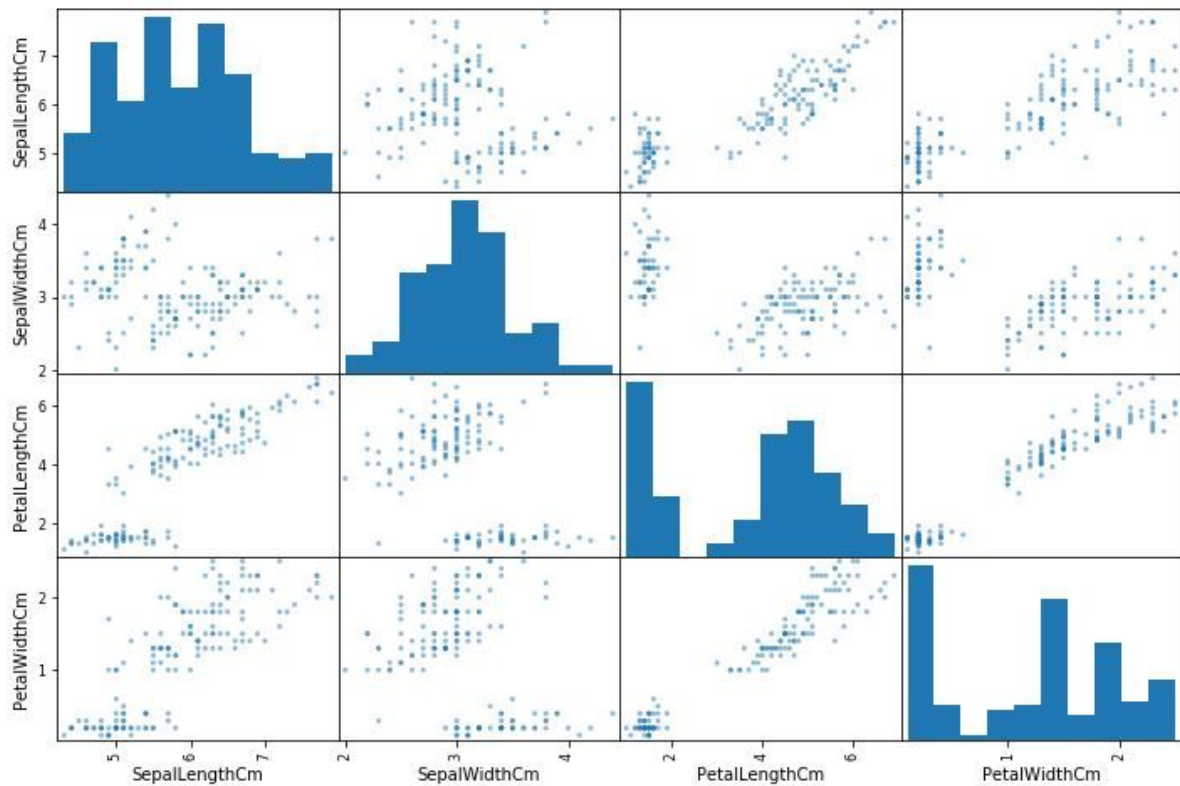


Figura 4. Gráficas de correlaciones entre atributos

Al analizar la tabla y las gráficas de correlación, nos damos cuenta que los atributos que poseen la más alta entre ellos son **PetalLengthCm** y **PetalWidthCm**, seleccionando al primero la variable independiente X, y el segundo la dependiente Y para **la ecuación de la línea de estimación**.

X = PetalLengthCm

Y = PetalWidthCm

Transformación de los datos

No es necesario realizar ninguna transformación o darle algún formato a la base de datos.

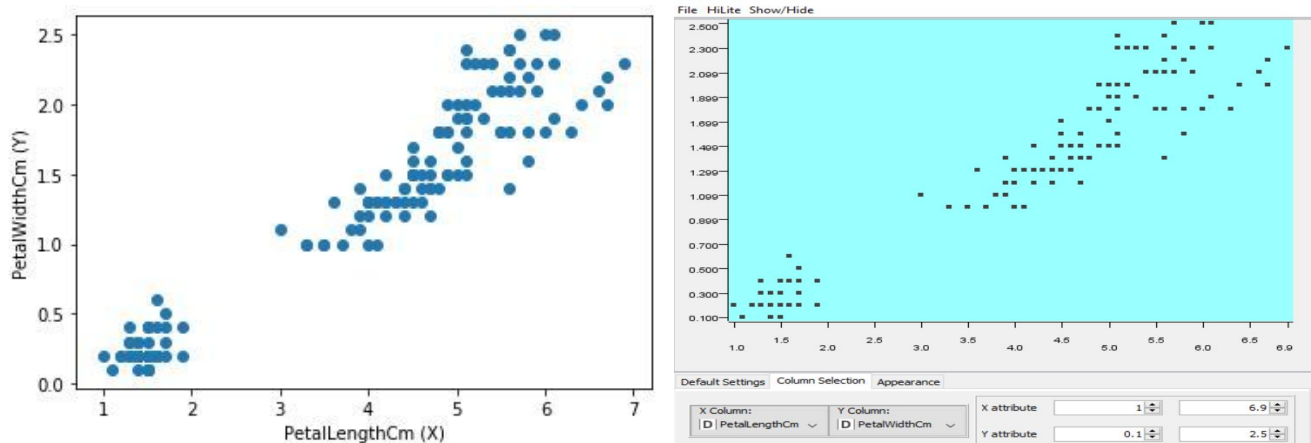


Figura 5. Gráfica de **PetalLengthCm** y **PetalWidthCm** en Python y KNIME

Minería de datos

Partición de los datos

Al ser la regresión lineal un algoritmo de aprendizaje supervisado, de forma similar a los árboles de decisión, es necesario dividir el conjunto de datos para generar el modelo. Una parte será para el entrenamiento o aprendizaje, que sería el 80 % de todos los datos. La otra parte servirá para ser evaluada por **la ecuación de la línea de estimación** del modelo generado, que corresponde al 20 % de los datos restantes.

Herramienta	Python
Parámetros de configuración	<p>Se utiliza el método <i>train_test_split</i>, que recibe como parámetros los valores x, los valores y, el tamaño del conjunto de entrenamiento (0.8 = 80%), y una semilla aleatoria.</p> <p>Esto nos da como resultado 4 arreglos, dos para entrenamiento para la variable independiente X (PetalLengthCm) y dos para la predicción de la variable dependiente Y (PetalWidthCm)</p> <pre>X_values = np.array(data[['PetalLengthCm']]) Y_values = np.array(data[['PetalWidthCm']]) x_train, x_test, y_train, y_test = train_test_split(X_values, Y_values, train_size=0.8, random_state=0) df = pd.DataFrame(x_train, columns=["PetalLengthCm"]) df = df.join(pd.DataFrame(y_train, columns=["PetalWidthCm"])) df.info() df.head(10)</pre>

Figura 6. Código para particionar datos en Python

<pre><class 'pandas.core.frame.DataFrame'> RangeIndex: 120 entries, 0 to 119 Data columns (total 2 columns): PetalLengthCm 120 non-null float64 PetalWidthCm 120 non-null float64 dtypes: float64(2) memory usage: 2.0 KB</pre>	<pre><class 'pandas.core.frame.DataFrame'> RangeIndex: 30 entries, 0 to 29 Data columns (total 2 columns): PetalLengthCm 30 non-null float64 PetalWidthCm 30 non-null float64 dtypes: float64(2) memory usage: 608.0 bytes</pre>				
PetalLengthCm	PetalWidthCm	PetalLengthCm	PetalWidthCm		
0	5.5	1.8	0	5.1	2.4
1	4.5	1.5	1	4.0	1.0
2	1.5	0.2	2	1.4	0.2
3	4.9	1.8	3	6.3	1.8
4	5.6	2.2	4	1.5	0.2
5	3.9	1.4	5	6.0	2.5
6	1.7	0.3	6	1.3	0.3
7	5.1	1.6	7	4.7	1.5
8	4.2	1.5	8	4.8	1.4
9	4.0	1.2	9	4.0	1.3

Figura 7. Datos para entrenamiento y datos para predicción en Python

Figura 7. Datos para entrenamiento y datos para predicción en Python

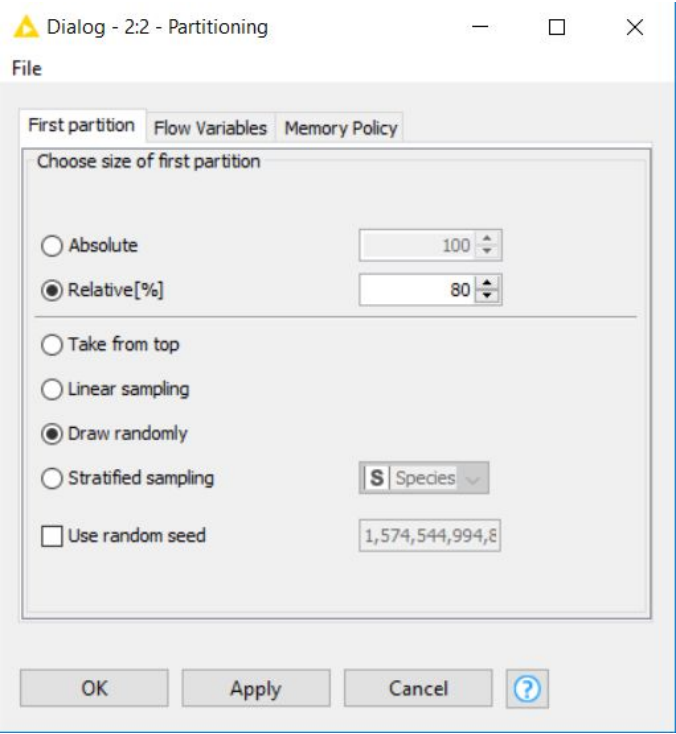
Herramienta	KNIME
Parámetros de configuración	<p>Se opta por tomar el 80% de datos para entrenamiento y el 20% de datos para pruebas, esto basado en la regla de 80 - 20 y en el tamaño del dataset utilizado</p> 

Figura 8. Configuración del nodo **Partitioning** en KNIME

D	PetalLe...	D	PetalWi...	D	PetalLe...	D	PetalWi...
1.4	0.2	1.5	0.2	1.5	0.2		
1.5	0.2	1.5	0.2	1.3	0.4		
1.4	0.2	1.6	0.4	1.6	0.4		
4.7	1.4	1.5	0.4	1.4	0.2		
4.5	1.5	1.2	0.2	1.3	0.3		
4.9	1.5	4.6	1.5	3.9	1.4		
4	1.3	4	1	4.5	1.5		
4.5	1.3	4.8	1.4	5.1	1.6		
4.7	1.6	4.5	1.5	4.4	1.2		
3.3	1	4.6	1.4	4	1.2		
4.6	1.3	4.2	1.3	4.2	1.2		
3.5	1	4.2	1.3	4.2	1.3		
4.2	1.5	4.5	1.7	5.8	1.8		
4.7	1.4	6.1	2.5	5	2		
3.6	1.3	5	2	5.3	2.3		
4.4	1.4	5.5	1.8	5	1.5		
4.5	1.5	6.7	2	5.4	2.1		
4.1	1						
3.9	1.1						
4.8	1.8						
4	1.3						
4.9	1.5						
4.7	1.2						
4.3	1.3						
4.4	1.4						
5	1.7						
4.5	1.5						
3.5	1						
3.8	1.1						
3.7	1						
3.9	1.2						
4.5	1.6						
4.7	1.5						
4.4	1.3						
4.1	1.3						
4	1.3						
3.3	1						

Figura 9. Datos para entrenamiento y datos para predicción en KNIME

Modelo de Regresión Lineal: Aprendizaje

Herramienta	Python
Parámetros de configuración	<p>Se crea el objeto de Regresión Lineal utilizando el método <i>linear_model.LinearRegression</i> y luego se entrena con los arreglos de entrenamiento de PetalLengthCm y PetalWidthCm.</p> <pre># Creamos el objeto de Regresión Lineal regr = linear_model.LinearRegression() # Entrenamos nuestro modelo regr.fit(x_train, y_train) # Construimos la ecuación de la línea de # estimación (en este caso, al ser 2D) y_pred1 = regr.predict(x_train) coef = float(regr.coef_[0]) termInd = float(regr.intercept_)</pre> <p>Figura 10. Código para generar el modelo de Regresión Lineal en Python</p>

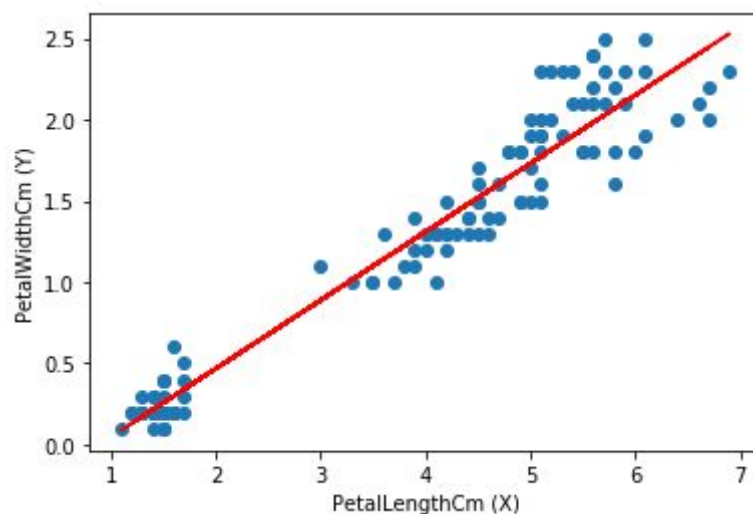


Figura 11. Modelo de Regresión Lineal en Python

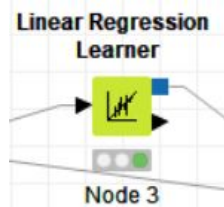
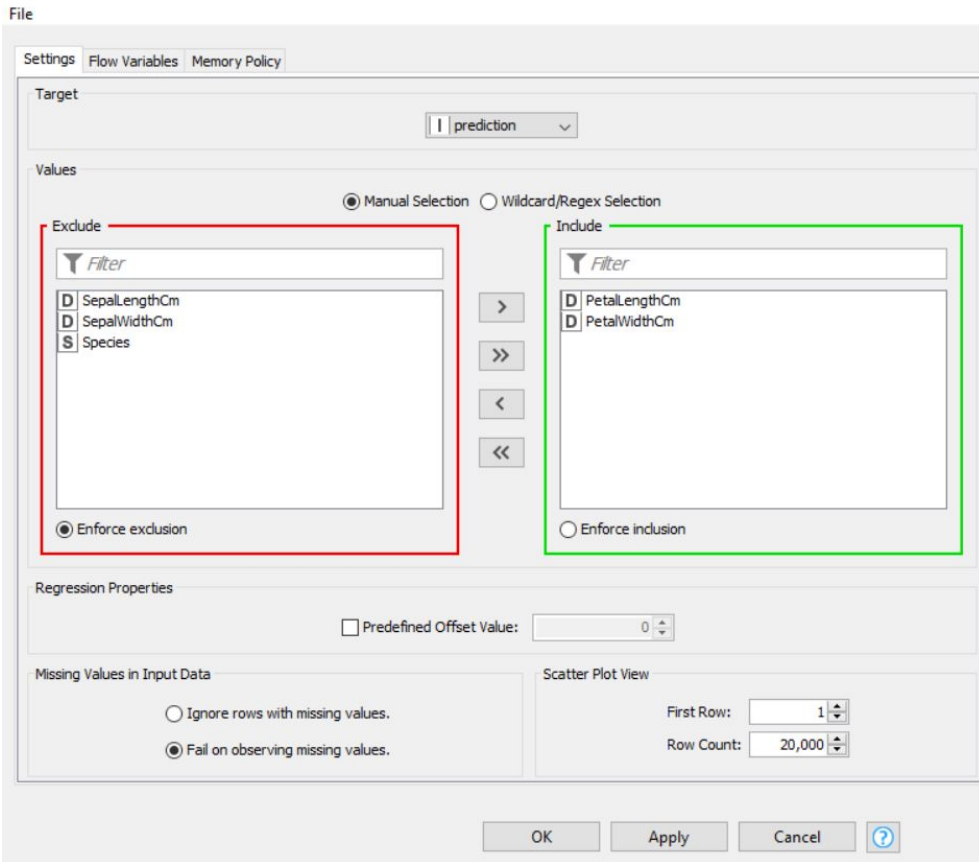
$$Y = 0.42X - 0.37$$

Error máximo: 0.52

Error medio absoluto: 0.15

Error medio cuadrado: 0.037

Puntuación de varianza: 0.93

Herramienta	KNIME
Parámetros de configuración	<p>Ocupamos el nodo Linear Regression Learner para crear el modelo de decisiones que entrenamos con el 80% de los datos.</p>  <p>Figura 12. Nodo Linear Regression Learner en KNIME</p>  <p>Figura 13. Configuración del nodo Linear Regression Learner en KNIME</p>

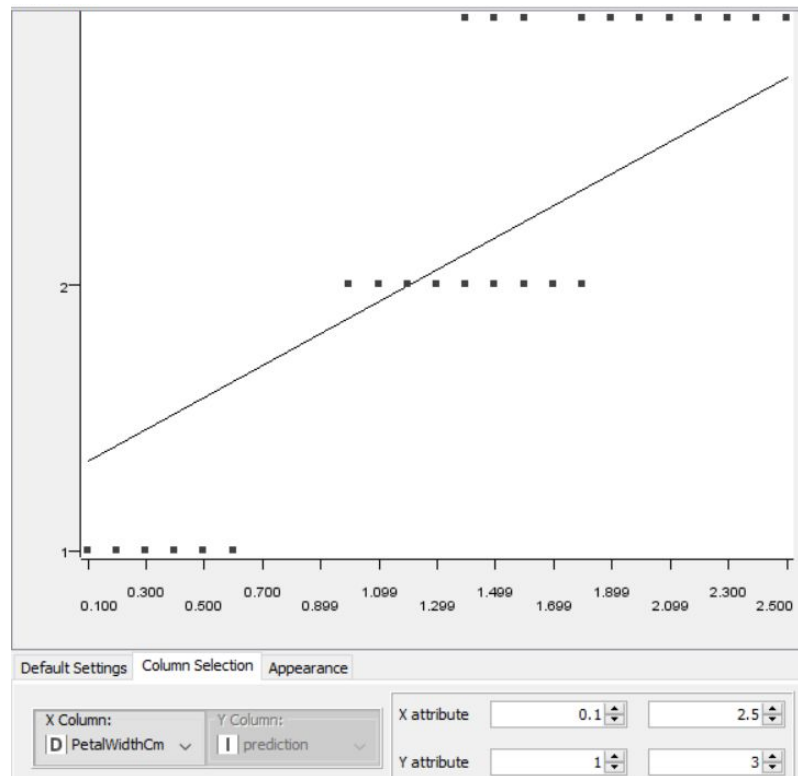


Figura 14. Gráfica del modelo de Regresión Lineal en KNIME

Row ID	S Variable	D Coeff.	D Std. Err.	D t-value	D P> t
Row1	PetalLengthCm	0.191	0.042	4.507	0
Row2	PetalWidthCm	0.6	0.097	6.197	0
Row3	Intercept	0.57	0.059	9.634	0

Figura 15. Modelo de Regresión Lineal en KNIME

Modelo de Regresión Lineal: Predicción

Herramienta	Python
Parámetros de configuración	<p>Ahora simplemente utilizamos el modelo para predecir los valores de PetalWidthCm utilizando el arreglo de entrenamiento de PetalLengthCm</p> <pre># Hacemos las predicciones # utilizando el modelo # (ecuacion de la línea de estimación) y_pred2 = regr.predict(x_test)</pre> <p>Figura 16. Código para realizar predicciones utilizando el modelo de Regresión Lineal en Python</p>

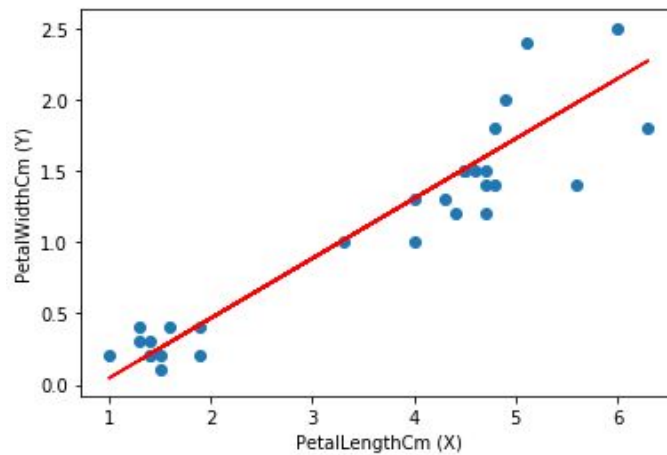


Figura 17. Predicción del modelo en Python

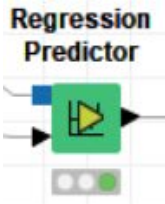
Herramienta	KNIME
Parámetros de configuración	<p>Ocupamos el nodo Regression Predictor para ocupar el modelo previamente creado (entrada 1), en este caso probamos el 20% del conjunto de datos que destinamos para las pruebas (entrada 2).</p> 

Figura 18. Nodo **Regression Predictor** en KNIME

En la siguiente figura se observan los datos predecidos (color rosa) y los datos de target originales (color verde):

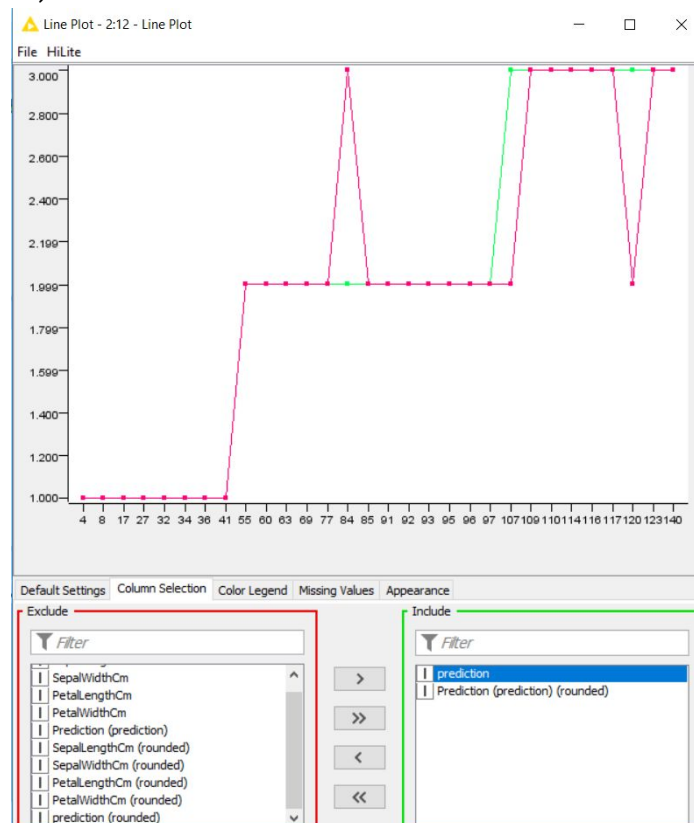
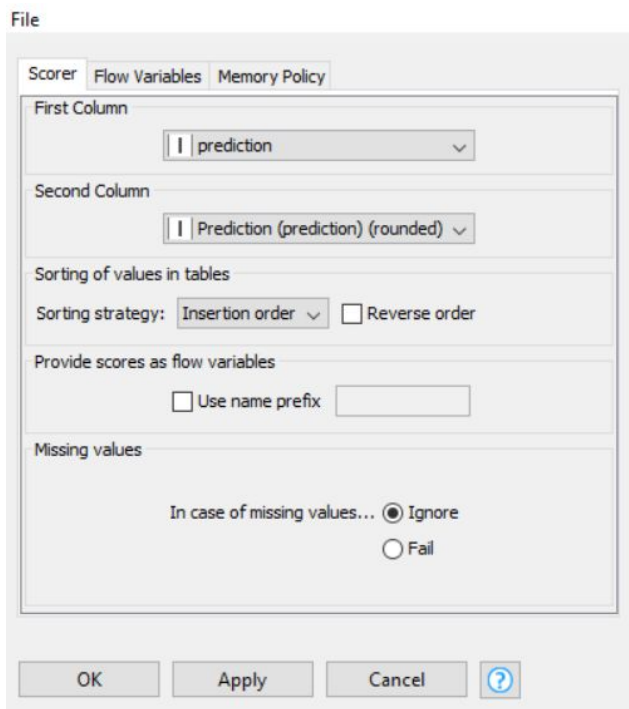


Figura 19. Predicción del modelo en KNIME

Evaluación del modelo

Herramienta	Python
Parámetros de configuración	<pre> # Error Máximo print("Error máximo: ", max_error(y_test, y_pred2)) # Error Medio Absoluto print("Error medio absoluto: ", mean_absolute_error(y_test, y_pred2)) # Error Medio Cuadrado print("Error medio cuadrado: ", mean_squared_error(y_test, y_pred2)) # Puntaje de Varianza. El mejor puntaje es un 1.0 print("Puntuación varianza: ", r2_score(y_test, y_pred2)) </pre> <p>Figura 20. Código para obtener los errores y puntuaciones de la predicción del modelo de Regresión Lineal en Python</p> <p>Error máximo: 0.628 Error medio absoluto: 0.183 Error medio cuadrado: 0.062 Puntuación de varianza: 0.8705</p>

Herramienta	KNIME
Parámetros de configuración	<p>Aplicamos un Scorer para observar cómo funcionó la predicción, con la siguiente configuración:</p>  <p>Figura 21. Configuración del nodo Scorer en KNIME</p>

Confusion Matrix - 2:9 - Scorer

File Hilite

prediction ...	1	2	3
1	8	0	0
2	0	12	1
3	0	2	7

Correct classified: 27	Wrong classified: 3
Accuracy: 90 %	Error: 10 %
Cohen's kappa (κ) 0.845	

Figura 22. Evaluación del modelo en KNIME

Uso manual del modelo

Para obtener un valor de **PetalWidthCm** de una flor proporcionado por la ecuación de la línea de estimación obtenida, simplemente se debe evaluar su ancho del pétalo en ésta.

Por ejemplo, si se desea obtener la **longitud del pétalo** de una flor cuyo ancho es de 6.6 cm:

$$\begin{aligned} X &= \text{PetalWidthCm} = 6.6 \text{ cm} \\ Y &= 0.42(6.6) - 0.37 \\ Y &= \text{PetalLengthCm} = 2.402 \text{ cm} \end{aligned}$$

Conclusiones

Al hacer uso de la regresión lineal podemos encontrar una predicción muy acertada de distintos dataset que tengan una correlación adecuada, con esto se quiere decir que la predicción pueda hacerse estableciendo una variable independiente y otra dependiente. En el caso específico del dataset de iris se pueden establecer con las medidas del pétalo.

Es de suma importancia que se haga un pre análisis de los datos pues hay distintos casos que no permiten la aplicación de la regresión lineal, estos son:

- Falta de independencia
- Falta de homocedasticidad
- Falta de linealidad

Por razones obvias, siempre existirá un margen de error en la predicción de datos, pues al final de cuentas una función lineal es muy pobre cuando se trata de realizar una aproximación sobre un conjunto de datos.

Sin embargo, este método es el más acertado al momento de calcular valores de datos faltantes y/o desconocidos, pues está construido a partir de los valores que ya se conocen dentro del conjunto de datos, evitando así valores atípicos y fuera del contexto de la base de datos.

Anexos

Flujo completo de KNIME:

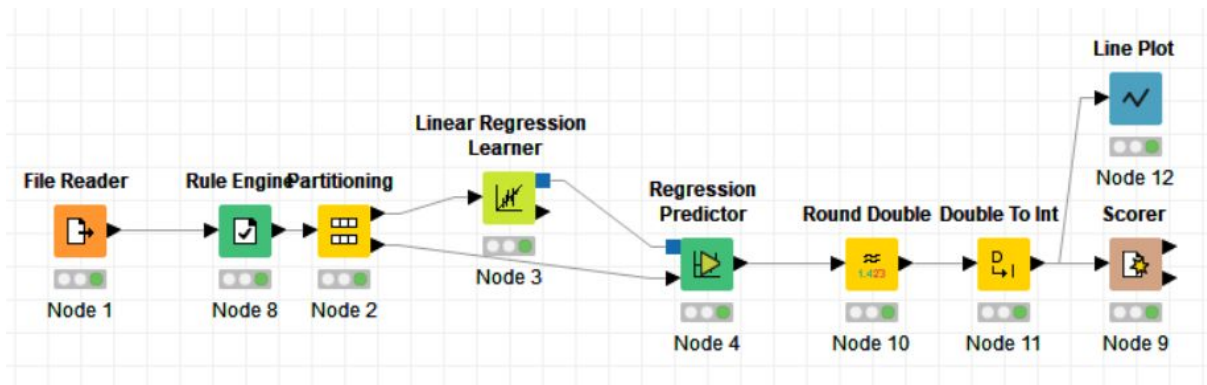


Figura 23. Flujo completo de KNIME