KIBERNUM
IT Academy

Módulo 5
Clase 12

Aprendizaje de Máquina Supervisado

Análisis de Regresiones No Lineales

KIBERNUM

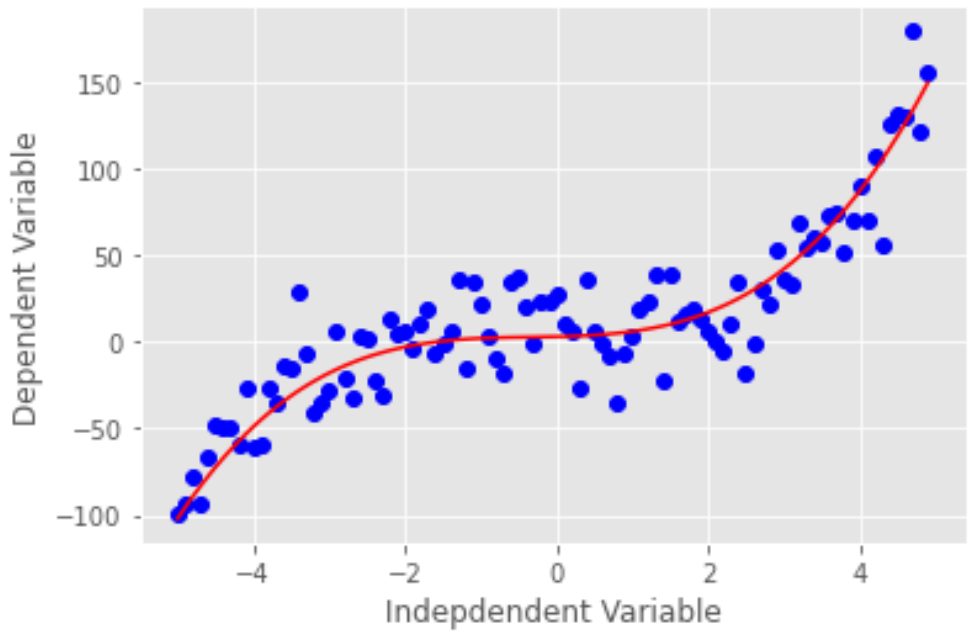
Contenido

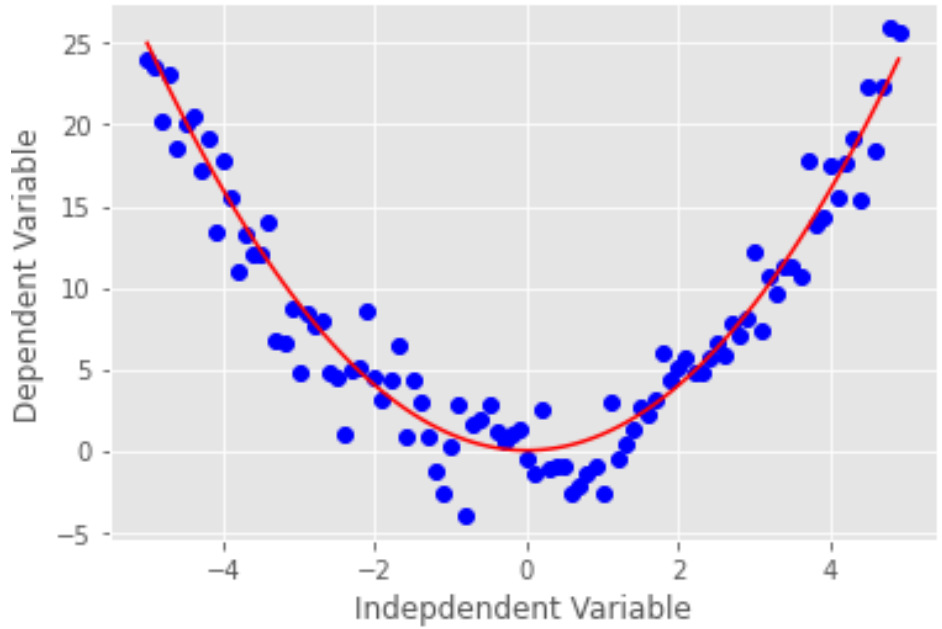
- Qué es una regresión no lineal
- Regresión polinomial
- Support Vector Regression
- Decision Tree Regression

KIBERNUM

¿Qué es una regresión no lineal?

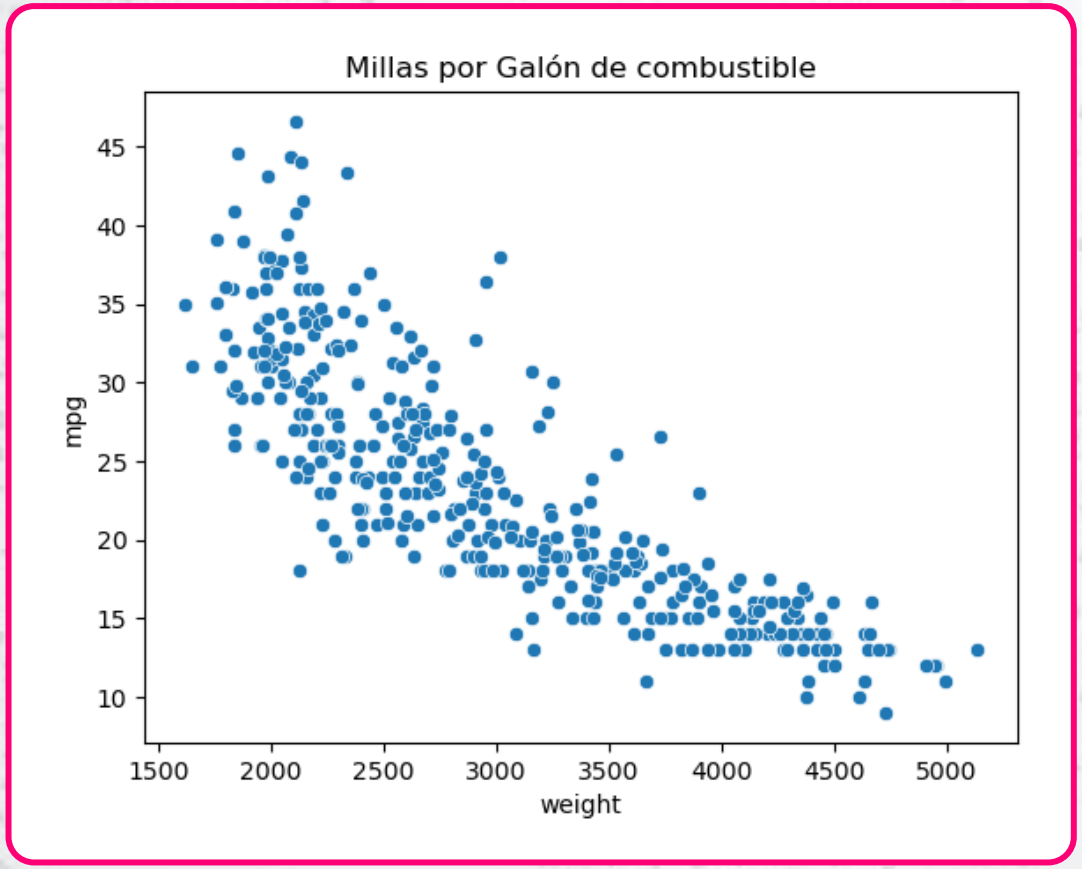
Una regresión no lineal es un método para encontrar un modelo no lineal, para la relación entre la variable dependiente y un conjunto de variables independientes. A diferencia de una regresión lineal, acá se rompe el supuesto de linealidad.





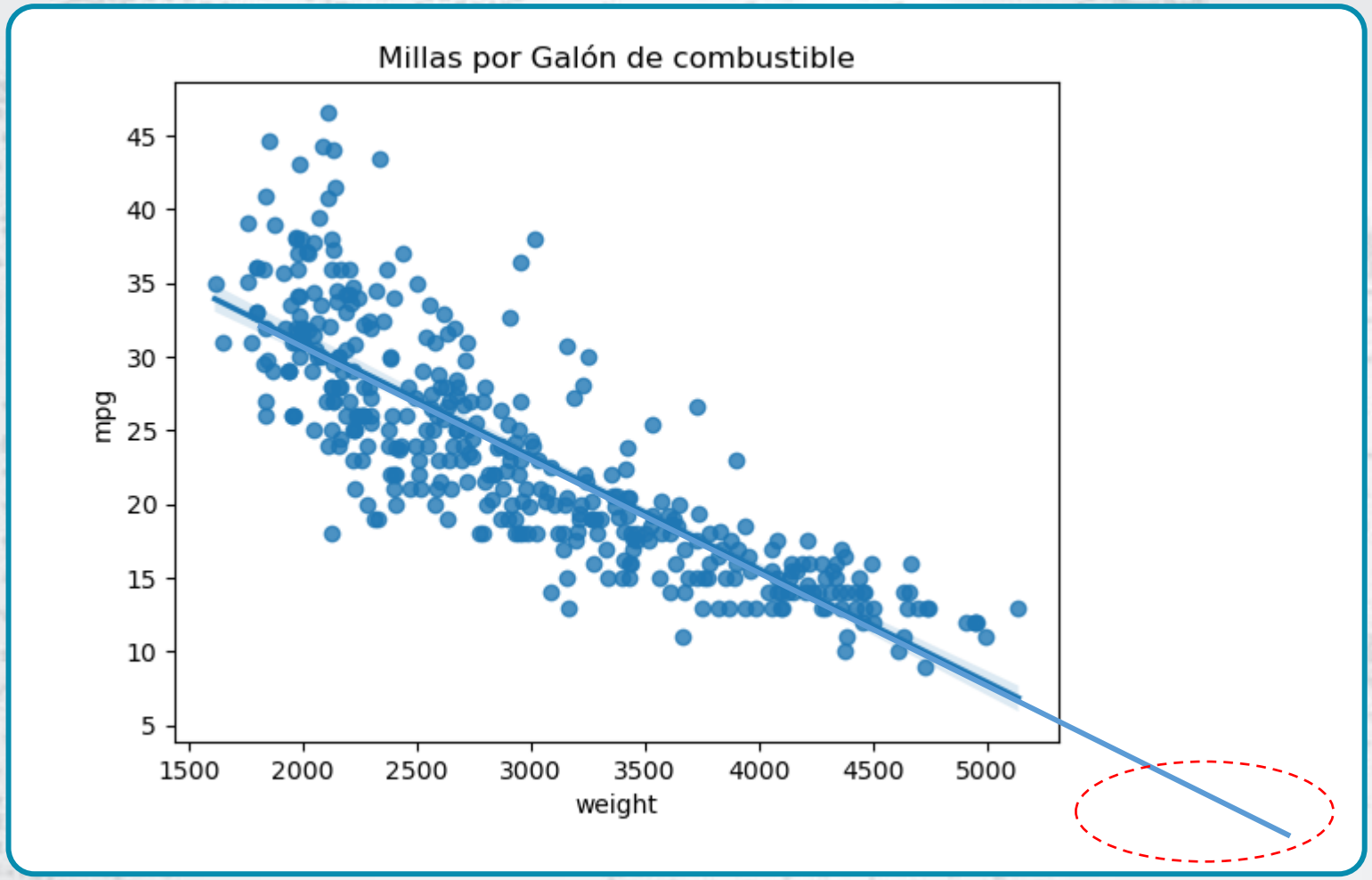
Análisis regresivo no lineal

Analicemos la relación entre el peso de un vehículo y su rendimiento de combustible. A priori, es razonable pensar que mientras más pesado sea el vehículo, menor su rendimiento. Esto, se observa claramente en el siguiente gráfico.



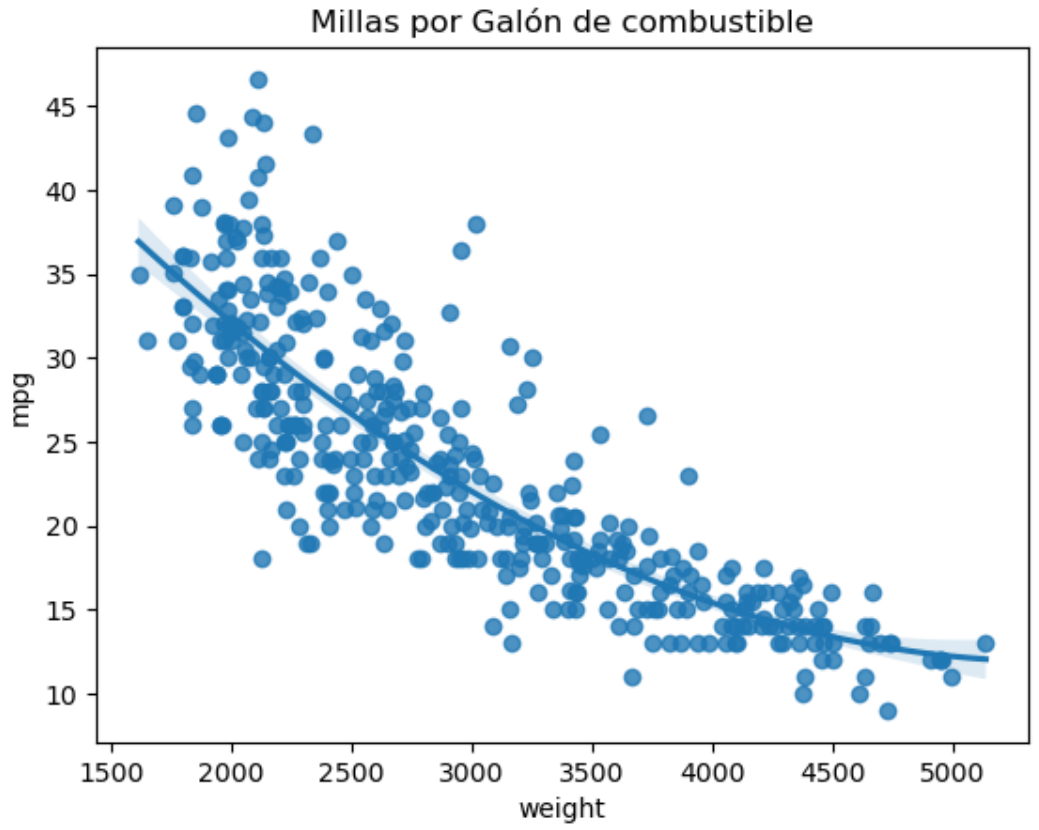
Análisis regresivo no lineal

Si ajustamos un modelo regresivo lineal, éste predeciría que el rendimiento de un vehículo podría ser cero, e incluso negativo, para ciertos valores de peso.



Análisis regresivo no lineal

En este caso, tal vez lo más conveniente es ajustar un modelo que no sea lineal. Por ejemplo, un modelo polinomial.



Regresión Polinomial

KIBERNUM

Regresión Polinomial

La regresión polinómica es un tipo de análisis de regresión no lineal que se utiliza para modelar relaciones no lineales entre una variable dependiente y una o más variables independientes. En la regresión polinómica, se ajusta un modelo polinómico a los datos, en el que se asume que la relación entre las variables puede ser descrita por un polinomio de grado n.

Un polinomio es una función matemática que puede ser escrita como una suma de términos que consisten en un coeficiente y una potencia creciente de la variable independiente.

Por ejemplo, el polinomio de grado dos (también conocido como cuadrático) tiene la forma:

$$y = b_0 + b_1 x + b_2 x^2$$

En donde, y es la variable dependiente, x es la variable independiente, b_0, b_1 y b_2 son los coeficientes del modelo y x^2 representa la variable independiente elevada al cuadrado.

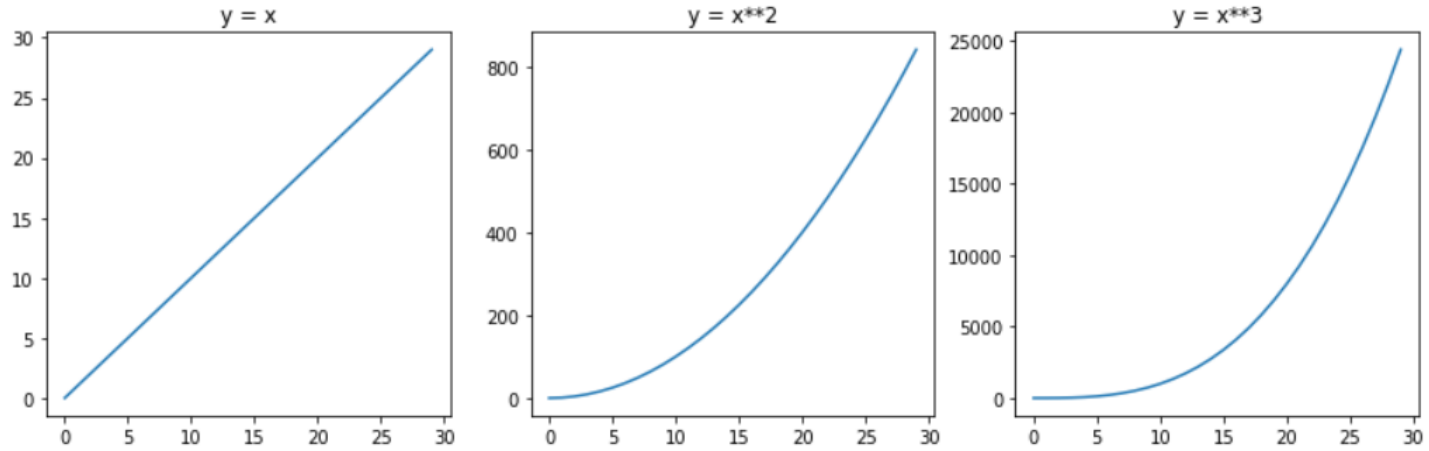
Generalizando:

$$y = b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n$$

Regresión Polinomial

Una regresión polinomial es un modelo lineal utilizado para describir relaciones no lineales. Pero, ¿cómo es posible esto? Suena contradictorio, pero no lo es. La magia está en crear nuevos features elevando los features originales a una potencia.

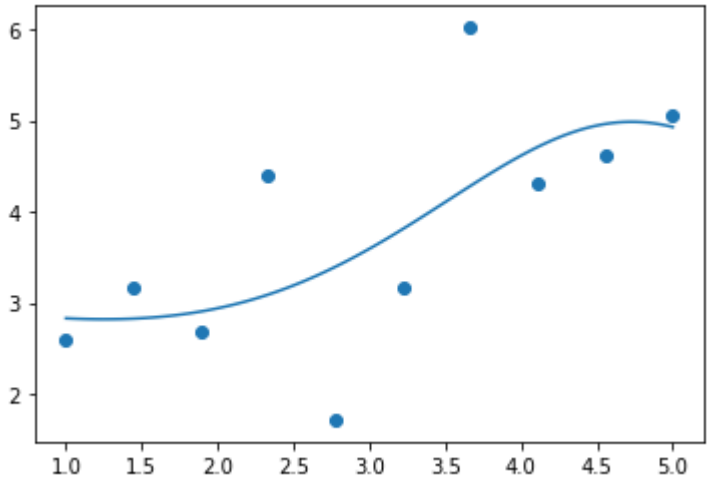
Por ejemplo, si tenemos un feature x, y usamos un polinomio de tercer grado, entonces nuestra fórmula incluirá x^2 y x^3. Esto es lo que permitiría darle "curvatura" a una línea recta.



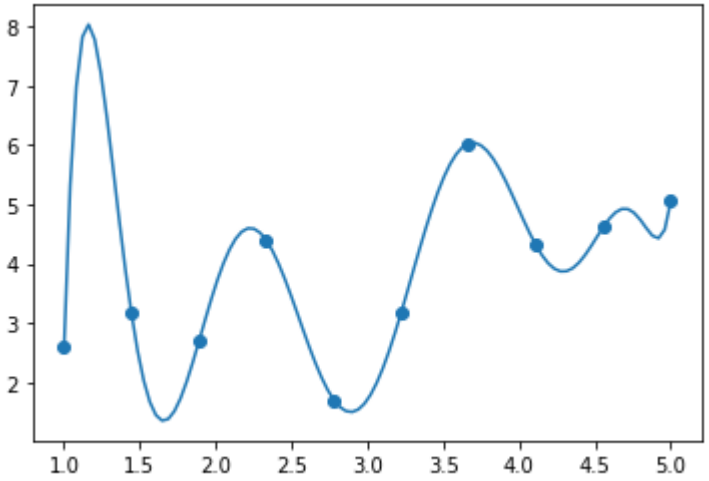
Entonces, podríamos decir que una **regresión lineal es sólo un polinomio de primer grado**. **Las regresiones polinomiales utilizan polinomios de orden superior, pero ambos son modelos lineales**. El primero resulta en una línea recta, y los otros en líneas curvas.

Regresión Polinomial

En la regresión polinómica, se ajusta un modelo polinómico a los datos mediante el método de mínimos cuadrados no lineales. El modelo puede ser de cualquier grado, pero a medida que aumenta el grado del polinomio, aumenta la complejidad del modelo y puede llevar a sobreajuste. El sobreajuste ocurre cuando el modelo se ajusta demasiado bien a los datos de entrenamiento y no se generaliza bien a nuevos datos.



Modelo simple, mejor capacidad de generalizar en nuevos datos



Modelo complejo, peor capacidad de generalizar en nuevos datos

Implementación en Python

Definimos el modelo

```
X = df[['weight']]
y =df['mpg']
```

Implementación en Python

Preprocesamos los datos realizando un escalamiento estándar.

```python
from sklearn.preprocessing import StandardScaler
```

```python
scaler = StandardScaler()
X_sc = scaler.fit_transform(X)
```

```python
X_sc[:5]
```

```
array([[0.63086987],
       [0.85433297],
       [0.55047045],
       [0.54692342],
       [0.56584093]])
```

Implementación en Python

Ahora, en este paso, crearemos los features polinomiales, considerando un polinomio de grado 3.

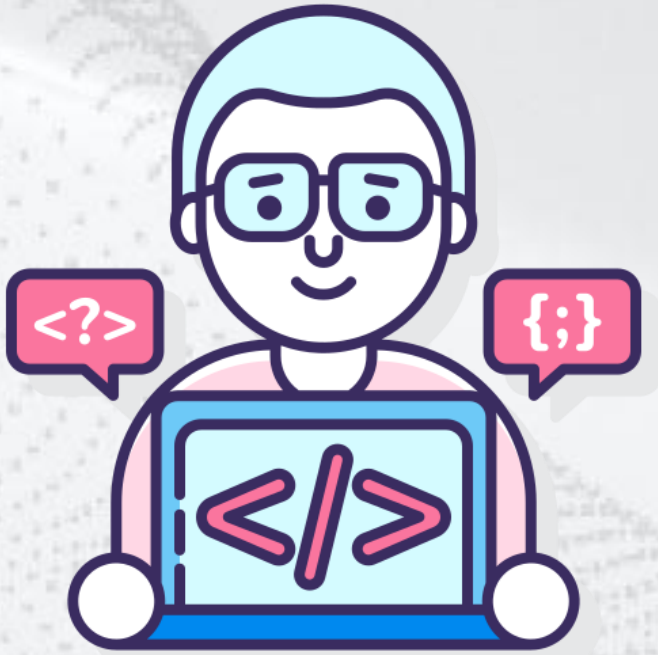
```python
from sklearn.preprocessing import PolynomialFeatures
```

```python
poly_transform = PolynomialFeatures(degree=3,
                                    include_bias=True)
```

```python
X_poly = poly_transform.fit_transform(X_sc)
```

```python
X_poly[:5]
```

```
array([[1.        , 0.63086987, 0.3979968 , 0.25108419],
       [1.        , 0.85433297, 0.72988483, 0.62356467],
       [1.        , 0.55047045, 0.30301772, 0.1668023 ],
       [1.        , 0.54692342, 0.29912523, 0.16359859],
       [1.        , 0.56584093, 0.32017596, 0.18116866]])
```



KIBERNUM

Implementación en Python

Entrenamos el modelo con el regresor lineal, pero ahora con nuestro set de datos con features polinomiales. Nótese los coeficientes entregados por el regresor.

```python
from sklearn.linear_model import LinearRegression
```

```python
regressor = LinearRegression()
```

```python
regressor.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```python
regressor.coef_
```

```
array([ 0.        , -6.98714904,  1.07571682,  0.00749393])
```

$$b_0 + b_1 x + b_2 x^2 + b_3 x^3$$

KIBERNUM

Implementación en Python

Evaluación del resultado. Recordar que en un modelo regresivo, el score del modelo corresponde al coeficiente R2.

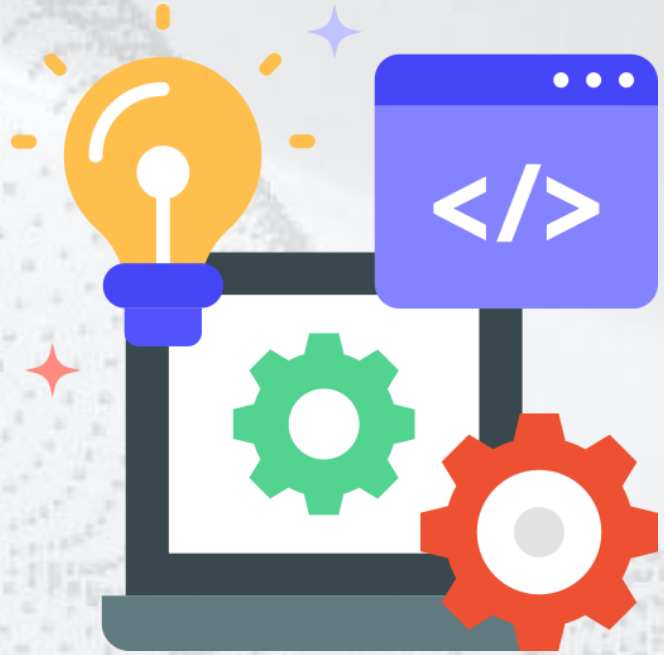
```python
y_pred = regressor.predict(X_test)
```

Python

```python
regressor.score(X_test,y_test)
```

Python

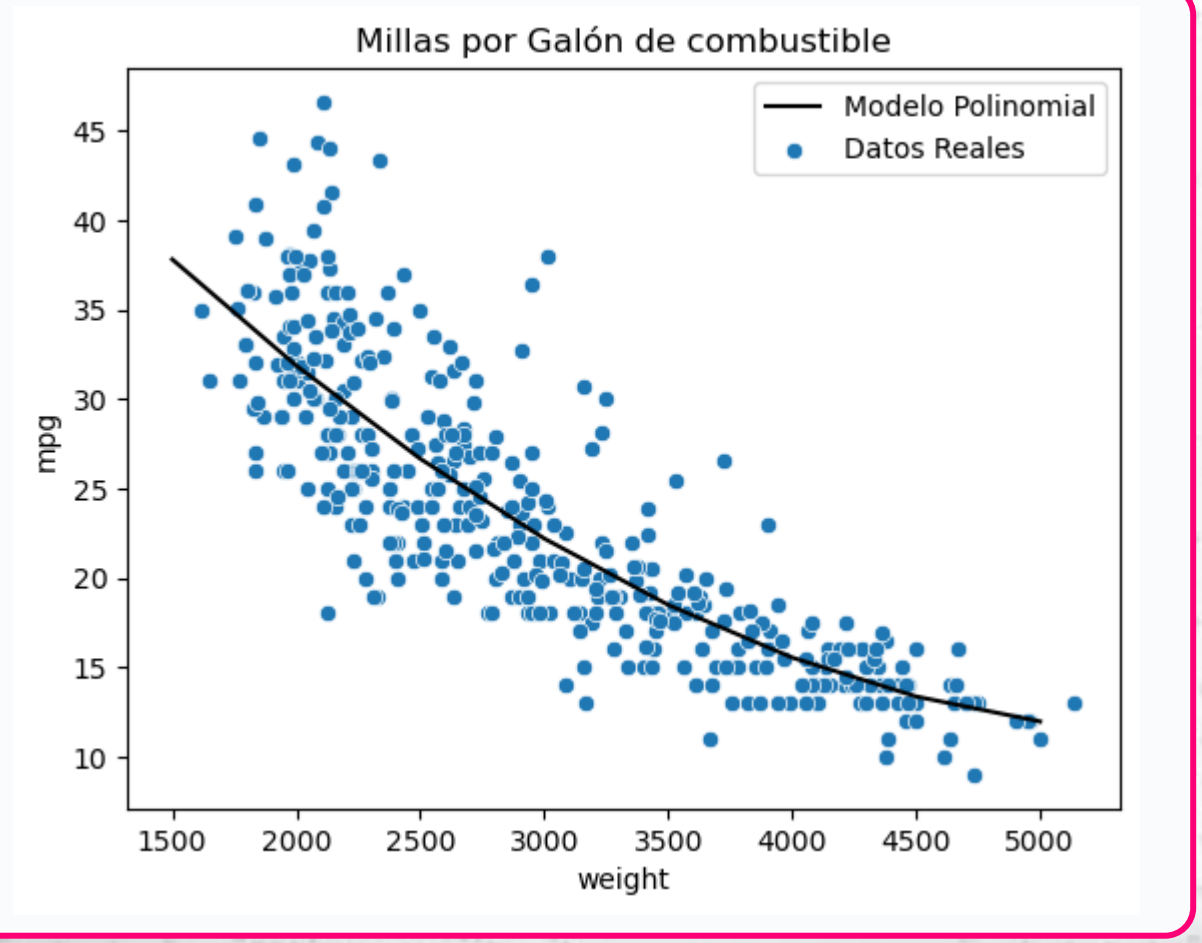
```
0.7795954015136342
```



KIBERNUM

Implementación en Python

Por último, visualizamos los datos y el modelo.



KIBERNUM

Ventajas

1. **Mayor flexibilidad**: La regresión polinómica puede ajustarse a una amplia variedad de patrones no lineales, lo que la hace más flexible que la regresión lineal.

2. **Capacidad para modelar relaciones curvilíneas**: La regresión polinómica puede modelar relaciones curvilíneas entre variables, lo que la hace útil en situaciones en las que la relación entre las variables no es lineal.

3. **Buena precisión para algunos datos**: En algunos casos, la regresión polinómica puede producir modelos altamente precisos que ajustan bien a los datos.

Desventajas

1. **Mayor complejidad**: La regresión polinómica puede ser más compleja que la regresión lineal y requiere más cálculos y procesamiento de datos.

2. **Mayor riesgo de sobreajuste**: A medida que aumenta el grado del polinomio, aumenta el riesgo de sobreajuste, lo que puede hacer que el modelo sea menos preciso y menos generalizable.

3. **Poca interpretación**: A menudo, los coeficientes de una regresión polinómica son difíciles de interpretar y no siempre proporcionan información útil sobre la relación entre las variables.

Support Vector Regressor

KIBERNUM

Support Vector Regression

Support Vector Regression (SVR) es un algoritmo de aprendizaje automático supervisado utilizado para predecir valores numéricos. Al igual que con Support Vector Machines (SVM), el objetivo de SVR es encontrar una función de decisión que separe los datos en dos categorías. Sin embargo, en SVR, en lugar de encontrar un hiperplano que separe los datos en dos clases, se busca una función que modele los datos de la mejor manera posible.

- El algoritmo SVR se basa en la teoría de máquinas de vectores de soporte, donde se utiliza una función kernel para transformar los datos de entrada en un espacio de características de mayor dimensión. Luego, se busca la función lineal que mejor se ajuste a los datos transformados.

- En lugar de minimizar el error de clasificación como en SVM, en SVR se minimiza la diferencia entre las predicciones del modelo y los valores reales, sujeto a una restricción de tolerancia. La restricción de tolerancia permite que ciertos puntos de datos estén fuera de la región de predicción y aún así se consideren bien ajustados.

- SVR es útil para problemas de regresión no lineal, donde se desea un modelo que sea flexible y pueda manejar datos no lineales. También es útil cuando se tienen pocos datos, pero se desea una alta precisión en las predicciones.

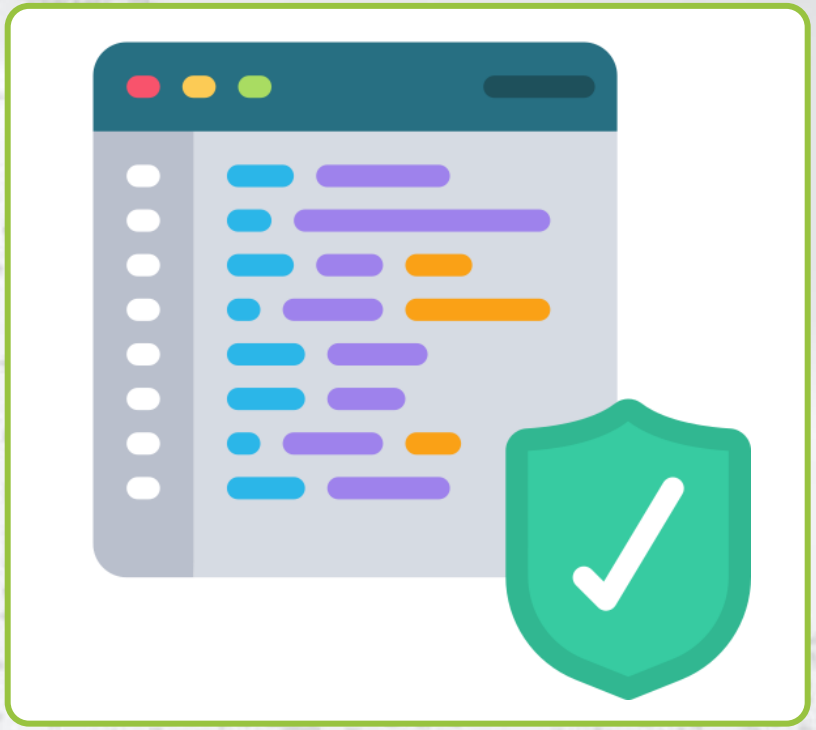
Support Vector Regression

SVR es similar a las regresiones lineales en donde la ecuación es $y = wx + b$. En SVR, esta línea recta está referida como **hiperplano**. Los puntos que están en ambos lados lo más cerca del hiperplano son llamados vectores de soporte, los cuales son utilizados para dibujar el límite de decisión.

A diferencia de otros modelos regresivos que tratan de minimizar el error entre el valor real y el predicho, SVR trata de ajustarla mejor línea dentro de un valor umbral (a). Entonces, se puede decir que SVR trata de satisfacer la condición:

$$-a < y - wx + b < a$$

En el caso de regresiones no lineales, se utiliza un kernel no lineal, como por ejemplo el kernel **rbf**.



Implementación en Python

Importamos regresor

Entrenamos en set de entrenamiento

Instanciamos clase con kernel gaussiano

Hacemos predicciones

Obtenemos score (r2)

```
from sklearn.svm import SVR
```
✓ 0.0s Python

```
regressor = SVR(kernel = 'rbf')
```
✓ 0.0s Python

```
regressor.fit(X_train, y_train)
```
✓ 0.0s Python

▾ SVR
SVR()

Evaluación

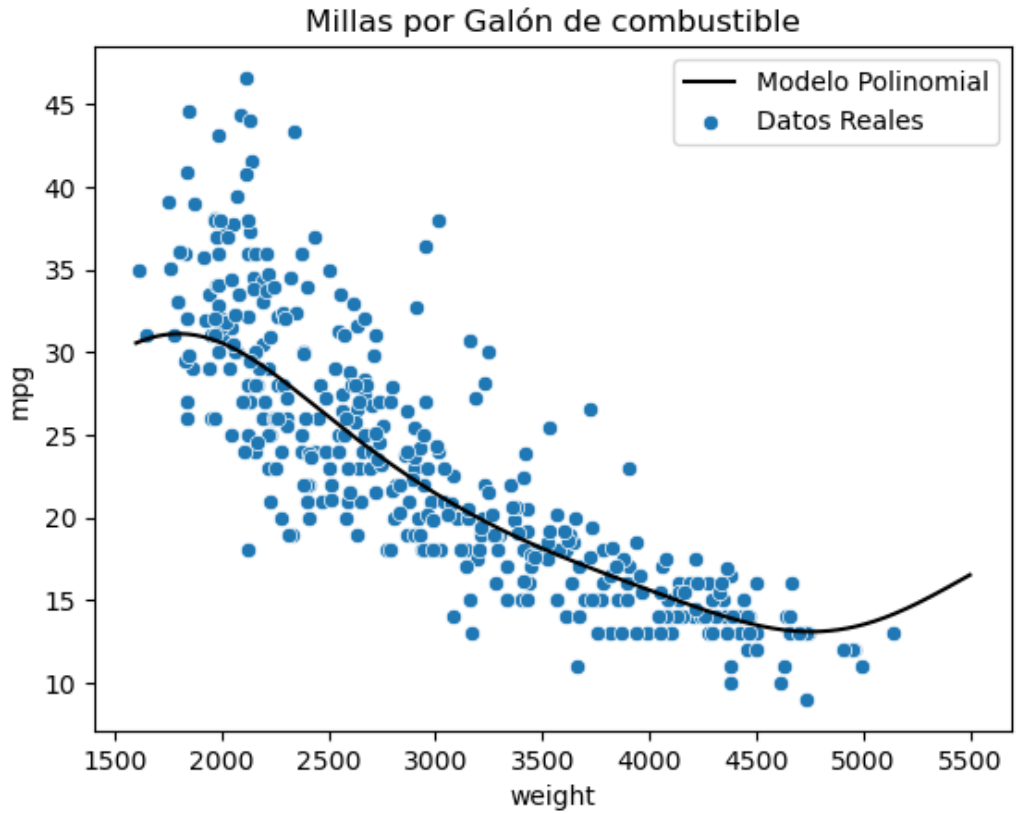
```
y_pred = regressor.predict(X_test)
```
✓ 0.0s Python

```
regressor.score(X_test,y_test)
```
✓ 0.0s Python

0.7552113770548685

Implementación en Python

Visualización del modelo.



KIBERNUM

Ventajas

1.Capacidad de manejar problemas de regresión no lineal: SVR utiliza una función kernel para transformar los datos en un espacio de características de mayor dimensión, lo que le permite manejar datos no lineales y encontrar la función de decisión óptima.

2.Regularización incorporada: la restricción de tolerancia en SVR actúa como un término de regularización, lo que ayuda a evitar el sobreajuste en el modelo y a mejorar la generalización.

3.Eficiente para conjuntos de datos pequeños: SVR es útil para problemas de regresión con pocos datos, ya que su objetivo es encontrar la mejor función de decisión que se adapte a los datos transformados, en lugar de buscar el mejor ajuste posible para los datos originales.

4.Flexibilidad en la elección de la función kernel: SVR permite la elección de diferentes funciones kernel, lo que permite ajustar el modelo a diferentes tipos de datos.

Desventajas

1.Sensible a la selección de hiperparámetros: el rendimiento de SVR depende en gran medida de la elección adecuada de la función kernel y los parámetros de regularización, lo que puede ser difícil de determinar.

2.Difícil de interpretar: el modelo de SVR puede ser difícil de interpretar, ya que la función de decisión final es una combinación de vectores de soporte y pesos de kernel, lo que dificulta la comprensión de cómo se relacionan las variables predictoras con la variable de salida.

3.Problemas con conjuntos de datos grandes: el entrenamiento de un modelo SVR puede ser computacionalmente costoso y requerir grandes cantidades de memoria para conjuntos de datos grandes.

4.Sensible a valores atípicos: debido a la forma en que se construye el modelo SVR, puede ser sensible a valores atípicos en los datos, lo que puede afectar negativamente su rendimiento.

Decision Tree Regressor

KIBERNUM

Decision Tree Regressor

- **Decision Tree Regressor** es un algoritmo de aprendizaje automático supervisado utilizado para la regresión. Al igual que con los árboles de decisión para la clasificación, un árbol de decisión para la regresión es un modelo de aprendizaje automático que utiliza una estructura de árbol para realizar predicciones.

- La idea detrás de un árbol de decisión para la regresión es dividir el conjunto de datos en subconjuntos cada vez más pequeños y homogéneos según el valor de una variable predictor. Cada subconjunto resultante se divide a su vez en subconjuntos más pequeños y homogéneos según el valor de otra variable predictor, y así sucesivamente, hasta que se alcanza un punto de parada, que puede ser un número máximo de niveles de profundidad, un número mínimo de muestras en un nodo hoja, o una cantidad mínima de reducción en el error cuadrático medio de los nodos hoja.

- Una vez construido el árbol de decisión, se utiliza para realizar predicciones en nuevos datos. Cuando se presenta una nueva instancia al árbol de decisión, se sigue el camino desde la raíz hasta una hoja, y el valor de salida en esa hoja se utiliza como la predicción.

- El árbol de decisión para la regresión es una técnica útil para problemas de regresión no lineal y se puede utilizar para explorar relaciones complejas entre las variables predictoras y la variable de salida. Sin embargo, pueden ser propensos a sobreajuste y pueden requerir ajustes cuidadosos de los hiperparámetros para lograr un buen rendimiento en nuevos datos.

Implementación en Python

Volvamos al mismo ejemplo de MPG.

Importamos regresor

Instanciamos clase con kernel gaussiano

Entrenamos en set de entrenamiento

Hacemos predicciones

Obtenemos score (r2)

```python
from sklearn.tree import DecisionTreeRegressor
```

```python
regressor = DecisionTreeRegressor(max_depth=3)
```

```python
regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=3)
```

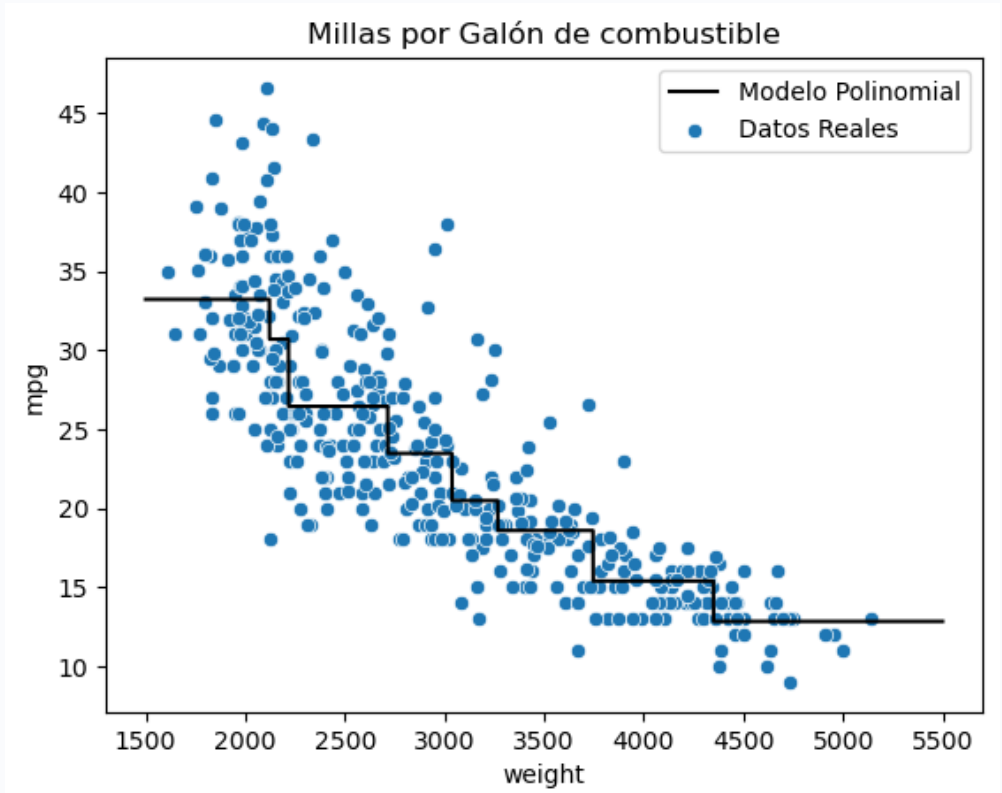
```python
y_pred = regressor.predict(X_test)
```

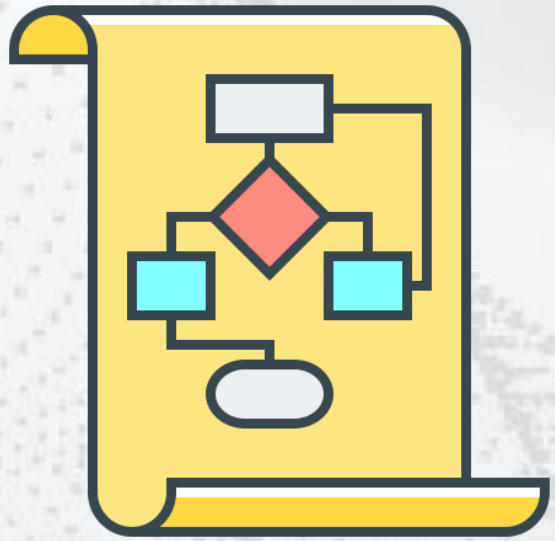
```python
regressor.score(X_test,y_test)
```

```
0.757885161105032
```

Implementación en Python

Visualización del modelo.





KIBERNUM

Ventajas

1.Fácil de entender e interpretar: los árboles de decisión son fáciles de visualizar y entender, ya que su estructura es similar a un diagrama de flujo, lo que hace que la interpretación del modelo sea sencilla.

2.Capacidad para manejar datos categóricos y numéricos: los árboles de decisión pueden manejar tanto datos categóricos como numéricos sin necesidad de preprocesamiento adicional.

3.Escala bien con grandes conjuntos de datos: los árboles de decisión pueden ser escalados a grandes conjuntos de datos con un tiempo de entrenamiento razonable, lo que los hace útiles para problemas de regresión en grandes conjuntos de datos.

4.No requiere suposiciones sobre la distribución de los datos: los árboles de decisión no requieren suposiciones sobre la distribución de los datos, lo que significa que son útiles para datos que no siguen una distribución normal.

Desventajas

1.Propenso a sobreajuste: los árboles de decisión pueden ser propensos a sobreajuste, lo que significa que pueden aprender a partir del ruido en los datos en lugar de la señal verdadera. Esto puede ser mitigado con técnicas como la poda del árbol o el uso de una profundidad máxima del árbol.

2.Sensible a pequeñas variaciones en los datos: los árboles de decisión son muy sensibles a pequeñas variaciones en los datos de entrenamiento, lo que significa que pueden generar diferentes árboles de decisión para datos similares.

3.No es adecuado para problemas complejos: los árboles de decisión pueden no ser adecuados para problemas complejos donde la relación entre las variables predictoras y la variable de salida es no lineal y difícil de capturar mediante divisiones simples en el árbol.

4.No siempre proporciona la mejor precisión: aunque los árboles de decisión pueden ser útiles para la regresión, pueden no proporcionar siempre la mejor precisión en comparación con otros algoritmos de regresión, especialmente en problemas con relaciones no lineales complejas.

Dudas y consultas

Gracias

KIBERNUM