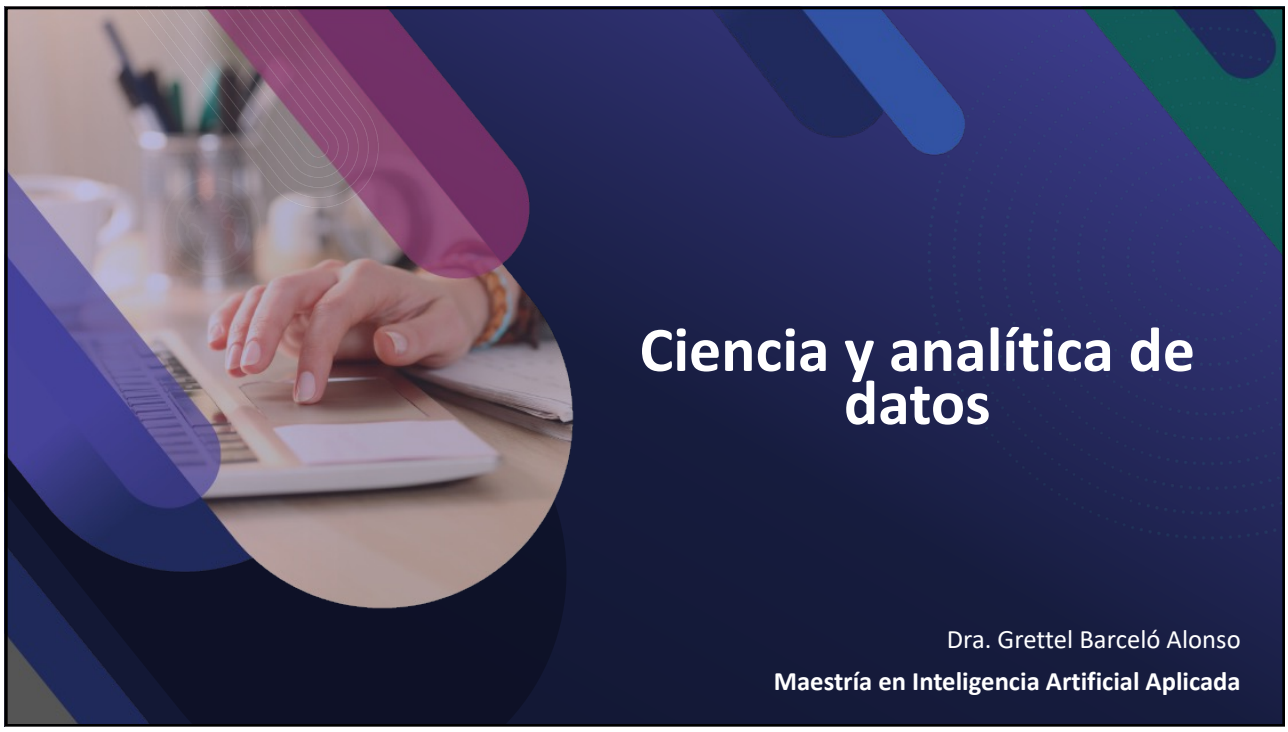

Ciencia y analítica de datos
Dra. Grettel Barceló Alonso
Maestría en Inteligencia Artificial Aplicada

1


Active class
Sesión 4

2

Módulo 4

Modelos de análisis de datos

3

Enfoques básicos de ML

4

Regresión lineal

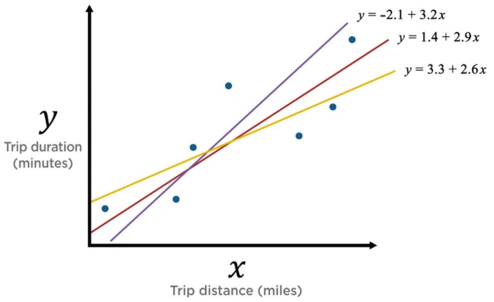
La regresión lineal es una forma de identificar la relación entre las variables independientes y la variable dependiente.

Asume que la relación entre variables se puede modelar mediante una ecuación lineal.

En caso de regresión lineal con una única variable independiente, la combinación lineal se puede expresar como:

$$respuesta = intercepción + constante * característica$$

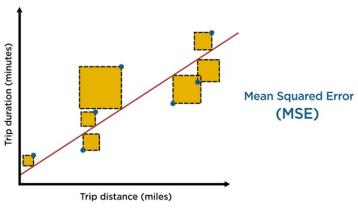
$$y = \beta_0 + \beta_1 x_1$$



5

Error cuadrático medio

- Al aprender de los datos, el modelo genera la línea que mejor se ajusta a los datos.
- Mejor ajuste significa que la línea será tal que la distancia acumulada de todos los puntos desde la línea es minimizada.

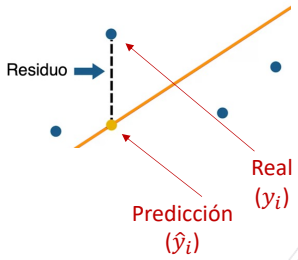


Función de costo (generalizada *J*)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Mientras más bajo, mejor



6

Regresión lineal múltiple

- Es una extensión del concepto de regresión lineal simple con una variable a múltiples variables.
- En el mundo real, cualquier fenómeno o resultado podría estar impulsado por muchas variables independientes diferentes. Por lo tanto, es necesario contar con un modelo matemático que pueda capturar esta relación.

$$respuesta = intercepción + constante_1 * característica_1 + \ constante_2 * característica_2 + \cdots$$
$$y = \ \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots$$

- Para estimar los coeficientes en un modelo de regresión lineal se usa la técnica de Mínimos Cuadrados Ordinarios (OLS, por sus siglas en inglés)

$$\beta = (X^T X)^{-1} X^T Y$$

7

Mínimos cuadrados ordinarios (OLS)

Si en el conjunto de datos tenemos:

n – número de observaciones
m – número de predictores

Entonces, puede ser representado como:

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \beta_3 x_{13} + \cdots + \beta_m x_{1m} + \varepsilon_1 \\ y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \beta_3 x_{23} + \cdots + \beta_m x_{2m} + \varepsilon_2 \\ y_3 &= \beta_0 + \beta_1 x_{31} + \beta_2 x_{32} + \beta_3 x_{33} + \cdots + \beta_m x_{3m} + \varepsilon_3 \\ &\vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \\ y_n &= \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \beta_3 x_{n3} + \cdots + \beta_m x_{nm} + \varepsilon_n \end{aligned}$$

Aunque también se pueden usar métodos numéricos o iterativos, por ejemplo, el gradiente descendente.

De manera matricial

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_m \end{bmatrix} \quad \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \cdots & x_{1m} \\ 1 & x_{21} & x_{22} & x_{23} & \cdots & x_{2m} \\ 1 & x_{31} & x_{32} & x_{33} & \cdots & x_{3m} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nm} \end{bmatrix}$$

$$Y = X\beta + \varepsilon$$

El vector de coeficientes $\boldsymbol{\beta}$, se calcula con OLS $\quad \beta = (X^T X)^{-1} X^T Y$

8

Gradiente descendente

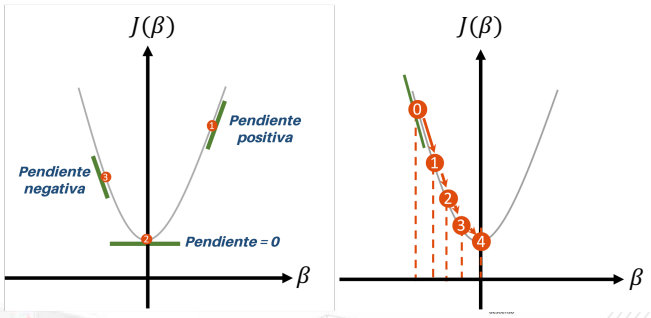
- El gradiente descendente es un algoritmo computacional que permite determinar de forma automática el mínimo de una función matemática.
- Supongamos una función de costo muy simple:

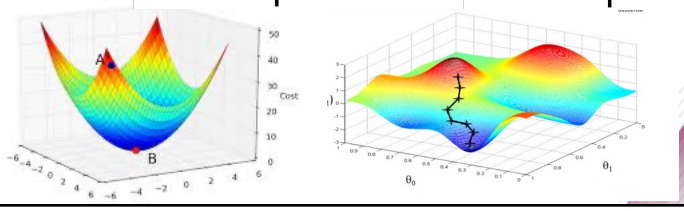
$$J(\beta) = \beta^2 + 1$$

$$\beta_{new} = \beta_{old} - n\nabla J(\beta)|_{\beta=\beta_{old}}$$

β - Coeficiente a estimar
$J(\beta)$ - Función de costo
n - Tasa de aprendizaje
∇ - Símbolo de gradiente

Ejemplo: Si $\beta_{old} = -3$ y $n = 0.1$
$\beta_{new} = -3 - [0.1 * 2(-3)] = -2.4$





9

Modelado estadístico vs ML

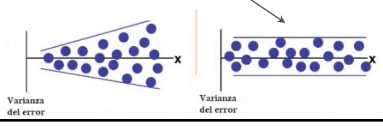
La regresión es versátil y puede ser aplicada tanto en un contexto de modelado estadístico como en un contexto de machine learning.

Modelado estadístico	Aprendizaje automático
• Los modelos estadísticos están diseñados para inferir las relaciones entre variables. • Se centra en las pruebas de hipótesis y la interpretabilidad. • Se presta atención a la significancia estadística de las relaciones. • Requiere una especificación clara de los supuestos.	• Los modelos de ML están diseñados para hacer las predicciones lo más precisas posibles. • Las decisiones en ML se basan en la capacidad del modelo para generalizar a partir de los datos de entrenamiento. • No se presta tanta atención a la significancia estadística. • La teoría no siempre explica el éxito.

10

Supuestos

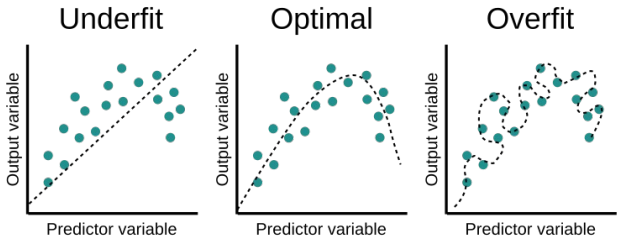
Supuesto	Definición	¿Cómo probarlo?	¿Cómo arreglarlo?
Linealidad	Debe haber una relación lineal entre la variable dependiente e independientes.	scatter plot / Correlación de cada variable independiente con variable dependiente.	Transformar variables con relaciones no lineales (log, raíz cuadrada, etc.)
Multicolinealidad	No debe existir una alta correlación entre dos o más variables independientes.	Mapas de calor de correlaciones.	Eliminar variables correlacionadas, aplicar PCA, drop first en categóricas
Normalidad	Los residuos deben estar normalmente distribuidos.	Histograma o Q-Q de los residuos.	Transformación no lineal de las variables.
Homocedasticidad	Los residuos deben tener una variación constante.	Trazar residuos.	Transformación no lineal de la variable dependiente o adición de otras variables importantes.



12

Regresión polinomial

- Se utiliza cuando se necesita predecir comportamientos más complejos.
- Por ejemplo, cuando la tendencia ya no sigue una línea recta simple, es común ajustar una ecuación polinomial como una cuadrática a sus datos.



13

Canalización (*pipeline*)

Consiste en dividir una tarea de aprendizaje automático completa en un flujo de trabajo de varios pasos.

Algunos transformadores:

- `SimpleImputer(strategy='median')`
- `StandardScaler()`
- `MinMaxScaler()`
- `OneHotEncoder()`
- `OrdinalEncoder()`
- `FunctionTransformer(func=np.log)`
- `PowerTransformer(method='box-cox', standardize=False)`
- `PCA(n_components=0.9)`

Algunos predictores:

- `LinearRegression()`
- `PolynomialFeatures(degree=2), LinearRegression()`
- `LogisticRegression()`

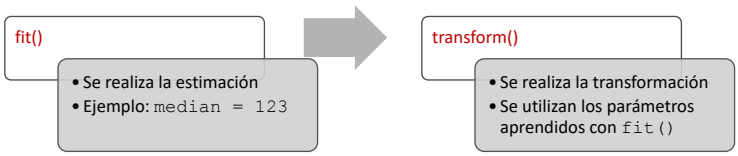
- Los predictores sólo pueden estar al final de la canalización
- Los transformadores poseen los métodos: `fit()` y `transform()`
- Los predictores poseen los métodos: `fit()` y `predict()`

14

Canalización (*pipeline*)

Ejecución de un transformador de manera independiente:

```
imputer = SimpleImputer(strategy='median')
no_missing_data = imputer.fit_transform(data)
```



Ejecución de dos transformadores de manera consecutiva:

```
imputer = SimpleImputer(strategy='median')
imputer.fit(data)
no_missing_data = imputer.transform(data)
scaler = MinMaxScaler()
scaler.fit(no_missing_data)
scaled_data = scaler.transform(no_missing_data)
```

15

Filtrado de información

- El filtrado (o *data leakage*) se presenta cuando se evalúa el modelo con datos que se usado para construirlo.
- Para evitarlo NO debemos usar todo el conjunto de datos en el entrenamimento.
- Típicamente se ocupa el 80% de los datos para el entrenamiento –`Xtrain/ytrain`– y el 20% para la evaluación – `Xtest/ytest`–

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.2, random_state=1)
```

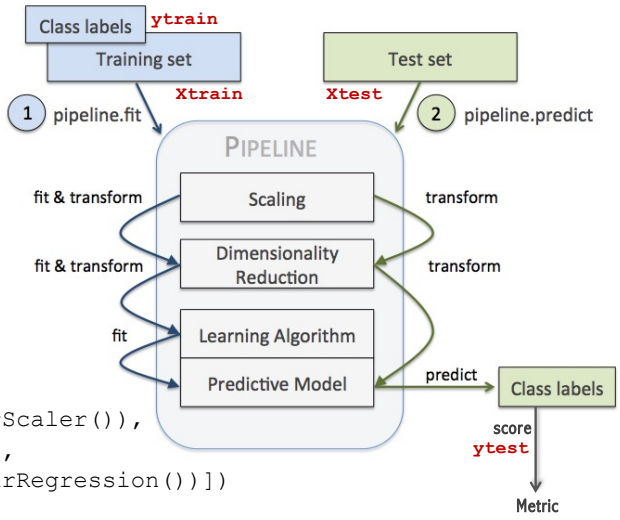
- Entonces `Xtest/ytest` NUNCA se ocupa en `fit()`

```
model.fit(Xtrain, ytrain)
predictions = model.predict(Xtest)
mean_squared_error(ytest, predictions, squared=False)
```

16

Canalización (*Pipeline*)

```
model = make_pipeline(StandarScaler(),PCA(),LinearRegression())
model.fit(Xtrain,ytrain)
predictions = model.predict(Xtest)
mean_squared_error(ytest, predictions, squared=False)
```



Equivalente:

```
model = Pipeline([('scale', StandarScaler()),
                  ('reduce', PCA()),
                  ('predict', LinearRegression())])
```

17

Tratamiento diferenciado

Pero NO siempre se puede dar el mismo preprocesamiento y/o ingeniería a todas las características.

```
preprocessing = ColumnTransformer([
                ('num', FunctionTransformer(np.sqrt), num_attribs),
                ('cat', OneHotEncoder(), cat_attribs)])
```

Si las listas de características se forman según el tipo de datos:

```
preprocessing = ColumnTransformer([
   ('num', FunctionTransformer(np.sqrt), make_column_selector(dtype_include=np.number)),
   ('cat', OneHotEncoder(), make_column_selector(dtype_include=object))])
```

Si se aplica más de un transformador:

```
num_pipeline = make_pipeline(FunctionTransformer(np.sqrt), MinMaxScaler())

preprocessing = ColumnTransformer([
   ('num', num_pipeline, make_column_selector(dtype_include=np.number)),
   ('cat', OneHotEncoder(), make_column_selector(dtype_include=object))])
```

18

Nota. `squared = True` por defecto y se calcula MSE. Si `squared = False` se calcula el RMSE

Pipeline

```
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


model = make_pipeline(preprocessing*, LinearRegression())
model.fit(Xtrain, ytrain)
predictions = model.predict(Xtest)

mean_squared_error(ytest, predictions, squared=False)
```

* `preprocessing` fue el `ColumnTransformer` generado en la diapositiva anterior

19

Accesando a los pasos y atributos

Los pasos de un pipeline pueden ser consultados a través del atributo `named_steps`, que regresa una lista de tuplas `(nombre,transformación)` con las transformaciones en orden secuencial.

```
model.named_steps
        'columntransformer':  …
        'linearregression': …
```

De cada paso se pueden mostrar sus atributos.

Por ejemplo, para la regresión lineal (linearregression), se pueden consultar la intercepción (β_0) y los coeficientes generados ($\beta_1, \beta_2, \beta_3, \ldots$)

```
model.named_steps['linearregression'].intercept_
model.named_steps['linearregression'].coef_
```

20

Tecnológico de Monterrey

21