Regresión linear

Objetivos de aprendizaje

Al final de este módulo, serán capaces de:

Explicar qué es una línea de regresión
Calcular una predicción de regresión linear dado los coeficientes y el intercepto

Esta clase les dará algunos antecedentes sobre cómo funciona un modelo de regresión lineal y cómo pueden interpretar una línea de regresión linear.

Regresión linear

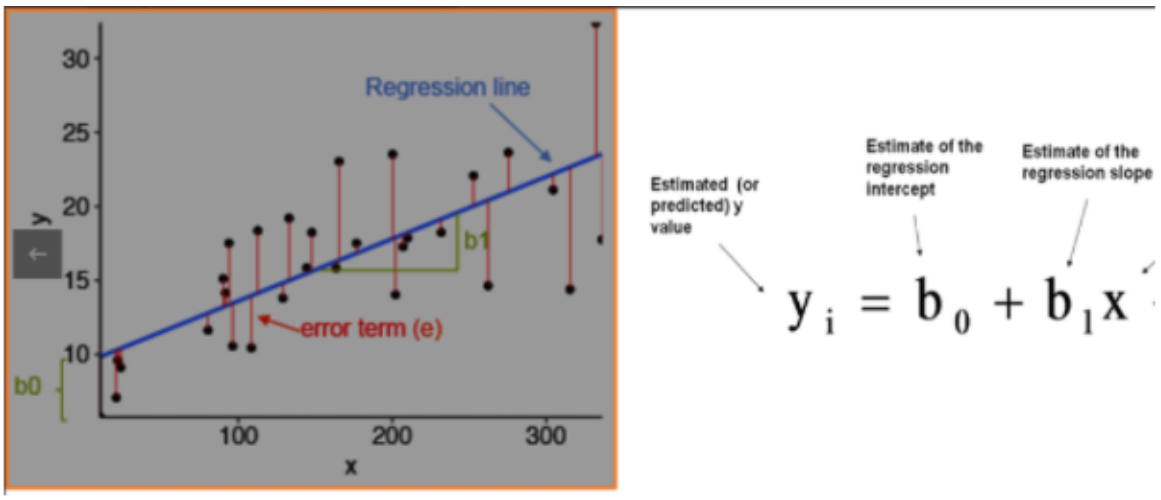
La regresión linear es uno de los algoritmos de aprendizaje automático más antiguos y comunes. Primero se introdujo en un papel por Sir Francis Galton (un primo de Charles Darwin) en 1887. La usó para encontrar la relación entre la altura de las plantas de guisantes dulces en relación con la altura de su planta madre. (Stanton, 2001)

Imaginen un gráfico de dispersión con la característica de salida como el eje X, por ejemplo, la altura de la planta madre y la variable que quieren predecir (objetivo) como el eje y, como la planta hija. Para cada punto de dato, deben trazar donde los dos valores intersecan en el gráfico.

La LÍNEA DE REGRESIÓN es una línea junto al gráfico que describe los valores y predichos para una entrada x dada. Para cada entrada X, el algoritmo predice que el objetivo y estará en esa línea donde x es igual a la característica de entrada.

El término de error es la distancia entre la predicción y el valor real en el eje y. La línea de regresión se traza de forma que se MINIMICE el error.

En sklearn se puede hacer a través de un proceso iterativo en el que se intentan muchas líneas y cada nuevo intento reduce la suma de los errores hasta que la línea es lo suficientemente buena según algunos criterios. Observen que la línea no pasa necesariamente a través del origen, donde x=0 e y=0. El valor y en esa línea donde x=0 se llama la INTERCEPCIÓN.



La pendiente, intercepción y línea de regresión

Podrán recordar de álgebra que una línea se puede expresar por una pendiente y una intercepción. La línea de regresión es exactamente eso. La pendiente de la línea es la correlación predicha entre las variables x e y. Una pronunciada pendiente ascendente significa una fuerte correlación positiva, al contrario de una pronunciada pendiente descendente significa una fuerte correlación negativa.

La pendiente se expresa como un coeficiente que multiplica el valor x. Un término intercepción describe dónde la línea de regresión intercepta el eje y en x=0.

Regresión linear simple: Una característica de entrada

Las nuevas predicciones, llamas ŷ o yhat, se generan al devolver el valor de y después de multiplicar x por el coeficiente de la línea de regresión y de añadir la intercepción.

La fórmula formal matemática para hacer una predicción con un modelo de regresión lineal es:

yhat: valor y predicho por un valor x de entrada

x: característica de entrada

b1: coeficiente

b: intercepción

yhat = b + b1 * x

Los valores que el modelo encuentra para determinar la línea de regresión son las b, el coeficiente y la intercepción.

Para el gráfico anterior:

b = 10

b1 = .04

Si x = 100:

yhat = b + b1 * x

yhat = 10 + .04 * 100

yhat = 10 + 4

yhat = 14

Regresión linear múltiple: Dos o más características de entrada

Es difícil de visualizar la línea de regresión donde hay múltiples características x de entrada, pero la matemática es similar. La diferencia es que CADA característica de entrada tiene su propio coeficiente:

yhat: y predicha

x1, x2, x3… etc: características de entrada

b1, b2, b3… etc: coeficientes para los xs correspondientes.

b: intercepción

yhat = b + (b1 * x1) + (b2 * x2) + (b3 * x3)…

La regresión linear forma la base de otros algoritmos de aprendizaje automático como la regresión logística y las redes neuronales, de los que aprenderemos más en las próximas semanas.

Citas:

Jeffrey M. Stanton (2001) Galton, Pearson y the Peas: A Brief History of Linear Regression for Statistics Instructors, Journal of Statistics Education, 9:3, , DOI: 10.1080/10691898.2001.11910537

Regresión linear en Python

Objetivos de aprendizaje
Al final de este módulo, serán capaces de:

- Desarrollar y ejecutar sus primeros modelos: regresión linear
- Aprender un método para evaluar el rendimiento del modelo
- Nota: pueden ver un video tutorial de este código al final del módulo.

Tarea
La tarea en esta sección es construir un modelo de regresión lineal usando el mismo conjunto de datos de California Housing que se usó en la clase “Train test split (modelo de validación)”. El conjunto de datos para este ejercicio se puede encontrar aquí y la descripción de los datos, aquí.

```
import pandas as pd
df = pd.read_csv('Your Path')
df.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Asignar el objetivo (y) y las características (X):

```
y = df['MedHouseVal']
X =df.drop(columns = 'MedHouseVal')
```

Realizar un train test split a los datos

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42)
```

Todos los valores son numéricos, y no vamos a escalar este modelo, así que podemos proceder con el desarrollo del modelo.

Paso 1: importar el modelo

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

Paso 2: hacer una instancia del modelo

```
# Make a linear regression instance
reg = LinearRegression()
```

Paso 3: Entrenar el modelo en los datos de entrenamiento. Este es el paso donde el modelo "aprender" sobre la relación entre las características y el objetivo.

El modelo está aprendiendo la relación entre X e y.

```
reg.fit(X_train,y_train)
```

Ahora que el modelo "aprendió" los patrones, es hora de ver qué tan bien puede hacer predicciones.

Medición del rendimiento del modelo

Nos acostumbraremos a evaluar el rendimiento del modelo tanto en los datos de entrenamiento como en los de prueba. Al final, es el rendimiento de los datos de prueba que indica qué tan bien creemos que nuestro modelo funcionará en datos futuros. Sin embargo, (en una futura clase) veremos cómo podemos ganar información sobre nuestro modelo al comparar su rendimiento en los datos de prueba con su rendimiento en los datos de entrenamiento.

R^2 (coeficiente de determinación)

La métrica por defecto para evaluar el modelo de regresión es R^2, el coeficiente de determinación. La mejor puntuación posible para R^2 es 1,0, y cuanto más alta sea la R^2, mejor. R^2 se da como decimal, pero se suele interpretar como un porcentaje. Por ejemplo, un R^2 de 0,9 se puede interpretar como: el 90 % de la varianza de y se puede explicar por la varianza en X. (El 90 % de la variación en nuestro objetivo se puede explicar por las características de nuestro modelo).

A continuación, se muestra el código para obtener los datos de R^2 después de ajustar nuestro modelo:

```
train_score = reg.score(X_train, y_train)
print(train_score)

test_score = reg.score(X_test, y_test)
print(test_score)
```

El valor R^2 en nuestro conjunto de entrenamiento es 0,609, y el valor R^2 en nuestro conjunto de prueba es 0,591. En este caso, nuestras puntuaciones de entrenamiento y

prueba fueron similares, lo que significa que nuestro modelo no estaba "sobreajustado". El sobreajustado ocurre cuando nuestro modelo funciona en nuestros datos de entrenamiento, pero falla en hacer buenas predicciones en los datos de prueba ocultos.

Tendrán la oportunidad para aprender más sobre el sobreajuste en otra clase.

Obtención de predicciones

Verán en la próxima clase que para calcular otras métricas, necesitarán extraer las predicciones del modelo y guardarlas como una variable. Eso se puede hacer usando .predict() como se muestra a continuación. Pueden hacer esto separado del conjunto de datos de entrenamiento y de prueba.

```
# Obtengan las predicciones del conjunto de entrenamiento
train_preds = reg.predict(X_train)
# Obtengan las predicciones del conjunto de prueba
test_preds = reg.predict(X_test)
```

Utilizarán estas predicciones en sus cálculos para la próxima clase de métrica de regresión.

Métricas de regresión

Objetivos de aprendizaje:

Al final de este módulo, serán capaces de:

- Describir 4 métricas para evaluar modelos de regresión
- Realizar el código necesario para cada una de las métricas de regresión.

La sección de regresión linear usó R^2, también conocido como el coeficiente de determinación, como métrica para evaluar el rendimiento del modelo de regresión lineal. También existen otras métricas que pueden usar para evaluar el rendimiento de un modelo. La tarea es calcular el error absoluto medio (EAM), el error cuadrático medio (ECM) y la raíz del error cuadrático medio (RECM) para el conjunto de datos y el modelo a partir de la sección de la regresión linear.

A continuación, se indican las fórmulas de cada una de ellas, pero no dejen que los intimiden. ¡Python hará todos los cálculos por ustedes!

```
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
```

Información métrica

Coeficiente de determinación (r^2): este es el valor dado cuando se usa model.score(X,y). Alternativamente, pueden usar la función sklearn r2_score() con los valores de y, y los valores predichos de y.

r2_train = r2_score(y_train, train_preds)
r2_test = r2_score(y_test, test_preds)

$$R^2 = 1 - \frac{\text{MSE}}{Var(y)}$$

Error absoluto medio (EAM): la media del valor absoluto de los errores. Pueden pensar en ello como un error promedio. Sklearn tiene una función mean_absolute_error() que toma los valor de y, y los valores predichos:

mae_train = mean_absolute_error(y_train, train_preds)
mae_test = mean_absolute_error(y_test, test_preds)

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Error cuadrático medio (ECM): la media de los errores al cuadrado. El ECM “castiga” a los errores más grandes, que suele ser útil en el mundo real. Tengan en cuenta que para más adelante en el curso, cuando repasemos cómo funcionan los algoritmos de aprendizaje automático, el ECM es continuo y diferenciable, por lo que es más fácil de usar que EAM en la optimización. Sklearn tiene una función mean_squared_error() que toma los valor de y, y los valores predichos:

mse_train = mean_squared_error(y_train, train_preds)
mse_test = mean_squared_error(y_test, test_preds)

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

Raíz del error cuadrático medio (RECM): raíz cuadrada de la media de los errores al cuadrado. Similar al ECM, pero más popular, ya que se considera más fácil de entender que el ECM. En cuanto hayan calculado el ECM, puede tomar la raíz cuadrada de este usando np.sqrt()

```
rmse_train = np.sqrt(mean_squared_error(y_train, train_preds))
rmse_test = np.sqrt(mean_squared_error(y_test, test_preds))
```

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