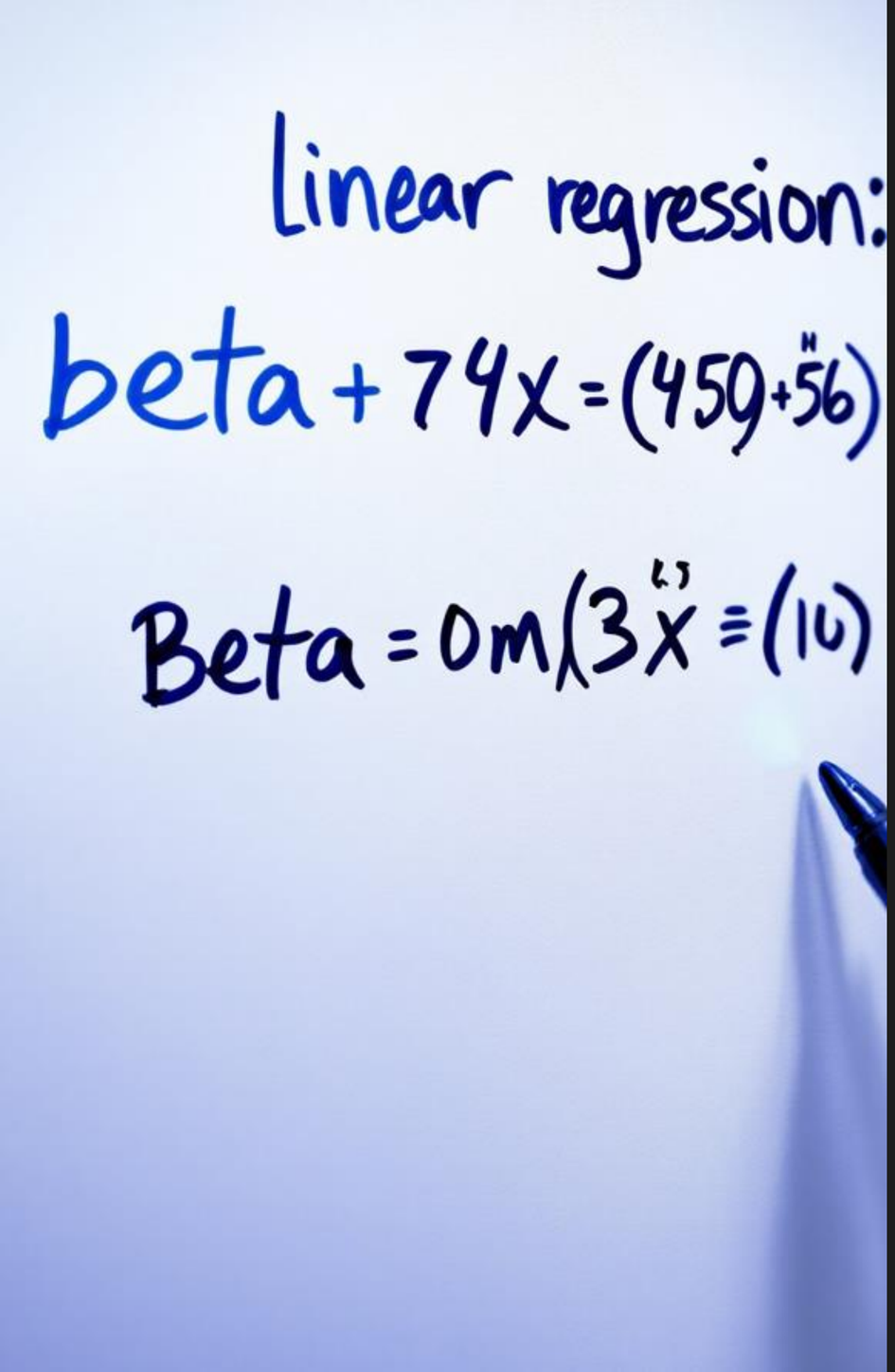
Análisis Exploratorio

de Datos

Sesión 4

Regresiones Lineales: Fundamentos y Aplicaciones

Las regresiones lineales son una de las técnicas más utilizadas en ciencia de datos y estadística para modelar la relación entre variables. Su amplia aplicación abarca diversos campos como economía, finanzas, salud y marketing, siendo fundamentales para la predicción y análisis de tendencias.



¿Qué es una Regresión Lineal Simple?

Definición

Una regresión lineal simple modela la relación entre una variable dependiente (Y) y una variable independiente (X) mediante una línea recta. Su propósito es predecir el valor de Y en función de X, basándose en una relación lineal entre ambas variables.

Ecuación

$Y = \beta_0 + \beta_1 X + \varepsilon$

Donde:

- β_0 es el intercepto (valor de Y cuando X=0)
- β_1 es la pendiente (cambio en Y por unidad de cambio en X)
- ε es el término de error.

Objetivo

Encontrar los valores óptimos de β_0 y β_1 que mejor ajusten la relación entre las variables, minimizando la suma de los errores al cuadrado entre los valores observados y los predichos.

Determinación de los Coeficientes de Regresión

1

Método de Mínimos Cuadrados

Para determinar los coeficientes de regresión (β_0 y β_1), se utiliza el método de mínimos cuadrados, el cual minimiza la suma de los errores al cuadrado entre los valores observados y los predichos.

2

Cálculo de la Pendiente (β_1)

La pendiente se calcula considerando la covarianza entre X e Y dividida por la varianza de X, utilizando los valores individuales y promedios de las variables.

3

Cálculo del Intercepto (β_0)

El intercepto representa el valor esperado de Y cuando X es igual a cero, y se calcula a partir del promedio de Y, la pendiente y el promedio de X.

Ejemplo de Regresión Lineal Simple en Python

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Datos de ejemplo
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
Y = np.array([2, 3, 5, 6, 8])

# Crear modelo de regresión lineal
modelo = LinearRegression()
modelo.fit(X, Y)

# Obtener coeficientes
beta_0 = modelo.intercept_
beta_1 = modelo.coef_[0]

# Hacer predicciones
Y_pred = modelo.predict(X)

# Visualización de la regresión
plt.scatter(X, Y, color="blue", label="Datos reales")
plt.plot(X, Y_pred, color="red", label="Regresión Lineal")
plt.xlabel("Variable X")
plt.ylabel("Variable Y")
plt.title("Regresión Lineal Simple")
plt.legend()
plt.grid()
plt.show()

# Mostrar coeficientes
print(f"Intercepto (β0): {beta_0}")
print(f"Pendiente (β1): {beta_1}")
```

Este código implementa una Regresión Lineal Simple utilizando Python con la librería scikit-learn. Su objetivo es ajustar una línea recta a un conjunto de datos y hacer predicciones basadas en ella.

Implementación en Python con Scikit-learn

Importación de Librerías

NumPy para manejar datos numéricos, Matplotlib para visualizar los datos y la línea de regresión, y LinearRegression de sklearn.linear_model para crear y entrenar el modelo.

1

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

Creación y Entrenamiento

Se crea un modelo de Regresión Lineal con LinearRegression() y se entrena con modelo.fit(X, Y), ajustando la mejor línea recta a los datos.

2

```
# Crear modelo de regresión lineal
modelo = LinearRegression()
modelo.fit(X, Y)
```

Obtención de Coeficientes

Se obtienen β_0 (intercepto) y β_1 (pendiente), que nos ayudan a interpretar la relación entre X e Y y realizar predicciones.

3

```
# Obtener coeficientes
beta_0 = modelo.intercept_
beta_1 = modelo.coef_[0]
```

Visualización y Predicción

Se visualizan los datos originales junto con la línea de regresión y se utilizan los valores de X para predecir los valores de Y con la ecuación obtenida.

4

```
# Hacer predicciones
Y_pred = modelo.predict(X)
```

```
# Visualización de la regresión
plt.scatter(X, Y, color="blue", label="Datos reales")
plt.plot(X, Y_pred, color="red", label="Regresión Lineal")
plt.xlabel("Variable X")
plt.ylabel("Variable Y")
plt.title("Regresión Lineal Simple")
plt.legend()
plt.grid()
plt.show()
```

Métricas de Error y Evaluación

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Error Cuadrático Medio (MSE)

El MSE es el promedio de los errores al cuadrado. Penaliza fuertemente los errores grandes, lo que lo hace útil cuando se busca minimizar desviaciones significativas en las predicciones. Se expresa en unidades cuadradas.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

Error Absoluto Medio (MAE)

El MAE es el promedio de los errores absolutos entre los valores reales y las predicciones. Es más robusto ante valores atípicos que el MSE y se mide en las mismas unidades que los datos originales.

$$R^2 = 1 - \frac{\sum(Y_i - \hat{Y}_i)^2}{\sum(Y_i - \bar{Y})^2}$$

Coeficiente de Determinación (R^2)

El R^2 mide qué tan bien el modelo explica la variabilidad de la variable dependiente. Un valor de 1 indica que el modelo explica el 100% de la varianza, mientras que 0 indica que no explica nada.

Ejemplos en Python

Error Cuadrático Medio (MSE)

```python
from sklearn.metrics import mean_squared_error

# Valores reales y predicciones de ejemplo
Y_real = [2, 3, 5, 6, 8]
Y_pred = [2.2, 3.1, 4.8, 5.9, 7.5]

# Cálculo del MSE
mse = mean_squared_error(Y_real, Y_pred)
print(f"Error Cuadrático Medio (MSE): {mse}")
```

Error Absoluto Medio (MAE)

```python
from sklearn.metrics import mean_absolute_error

# Cálculo del MAE
mae = mean_absolute_error(Y_real, Y_pred)
print(f"Error Absoluto Medio (MAE): {mae}")
```

Coeficiente de Determinación (R^2)

```python
from sklearn.metrics import r2_score

# Cálculo del R²
r2 = r2_score(Y_real, Y_pred)
print(f"Coeficiente de determinación R²: {r2}")
```

Implementación con Statsmodels

Importación de Librerías

Se importa statsmodels.api para construir y evaluar modelos de regresión lineal con métricas de error integradas, ofreciendo un análisis estadístico más completo.

Preparación de Datos

Se preparan los datos añadiendo una constante para el término de intercepto, necesario para el ajuste del modelo en statsmodels.

Ajuste del Modelo

Se crea y ajusta el modelo utilizando OLS (Ordinary Least Squares) para obtener los coeficientes y estadísticas de ajuste.

Análisis de Resultados

Se obtienen valores de coeficientes, R^2, errores estándar y pruebas de significancia estadística (p-values, t-values) para evaluar la calidad del modelo.

```python
import numpy as np
import statsmodels.api as sm

# Datos de ejemplo
X = np.array([1, 2, 3, 4, 5])
Y = np.array([2, 3, 5, 6, 8])

# Agregar una columna de 1s para el intercepto
X = sm.add_constant(X)

# Crear el modelo
modelo = sm.OLS(Y, X).fit()

# Resumen del modelo
print(modelo.summary())
```

Regresión Lineal Múltiple

Ecuación

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon$$

Definición

La Regresión Lineal Múltiple es una extensión de la regresión lineal simple que permite modelar la relación entre una variable dependiente (Y) y dos o más variables independientes (X_1, X_2,..., X_n).

Componentes

- β_0 (Intercepto): Valor de Y cuando todas las variables independientes son 0.
- β_1, β_2, ..., β_n (Coeficientes): Miden el efecto de cada variable independiente sobre Y.
- ε (Error residual): Captura variaciones no explicadas por el modelo.

Aplicación

Un ejemplo práctico sería predecir las ventas (Y) de una empresa en función del gasto en publicidad en TV, radio e internet (X_1, X_2, X_3).

Supuestos de una Regresión Múltiple

1. **Linealidad**
   La relación entre las variables independientes y la variable dependiente debe ser lineal. Se puede verificar con gráficos de dispersión o transformaciones de variables.

2. **Independencia de Errores**
   Los errores (ε) no deben estar correlacionados. Se evalúa mediante el test de Durbin-Watson para detectar autocorrelación en los residuos.

3. **Homocedasticidad**
   La varianza de los errores debe ser constante en todos los niveles de las variables independientes. Se verifica con gráficos de residuos para detectar patrones.

4. **Normalidad de Errores**
   Los errores deben seguir una distribución normal para que los intervalos de confianza y pruebas de hipótesis sean válidos. Se prueba con tests estadísticos o histogramas.

Métodos de Selección del Modelo

1. **Forward Selection (Selección Adelante)**
   Se inicia con una variable y se añaden más en función de su significancia estadística.

2. **Backward Elimination (Eliminación Hacia Atrás)**
   Se parte del modelo con todas las variables y se eliminan las menos significativas.

3. **Stepwise Regression (Regresión por Pasos)**
   Combina Forward Selection y Backward Elimination para optimizar el modelo.

Actividad Práctica Guiada: Regresión Lineal Simple

En este Actividad Práctica Guiada, aprenderás a ajustar una regresión lineal simple a un conjunto de datos y evaluar su rendimiento utilizando el coeficiente de determinación R^2 . Utilizaremos Python y la librería statsmodels para implementar el modelo.

Requerimientos:

1. **Importar Librerías**
2. **Crear un Conjunto de Datos**
3. **Visualizar los datos**
4. **Ajustar el modelo de Regresión Lineal Simple**
5. **Ver el resumen del Modelo**
6. **Interpretación del Modelo**
7. **Visualizar la Línea de Regresión**
8. **Evaluar el Rendimiento del Modelo**

El detalle de la actividad se encuentra en la guía de estudio de la sesión.

Preguntas

Sección de preguntas

Análisis Exploratorio

de Datos

Continúe con las actividades