Regresión lineal: predecir los gastos médicos de pacientes (VARIABLE CHARGES)

Para este ejercicio utilizaremos los datos presentados en [este](https://www.kaggle.com/mirichoi0218/insurance) dataset de Kaggle en el cual se presentan datos de seguros médicos.

## **Descarga e instalación de librerías**

Lo primero que se hará es descargar la librería **[regressors](https://pypi.org/project/regressors/" \t "_blank)** que ayudará a hacer un análisis más profundo sobre la regresión lineal.

8s

!pip install regressors

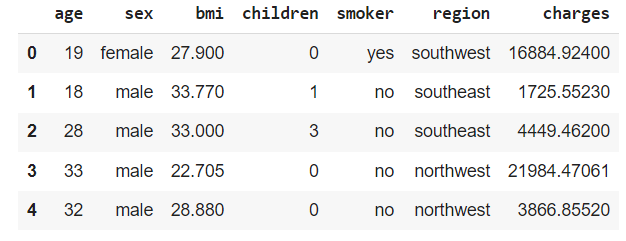
import pandas as pd

import seaborn as sns

sns.set(style='whitegrid', context='notebook')

df = pd.read\_csv('insurance.csv')

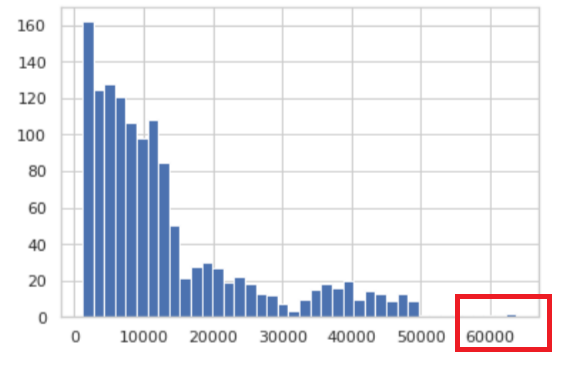
df.head()

****

**Analizando la distribución de ‘charges’ el valor de pago de los servicios médicos**

print(df.shape)

df.charges.hist(bins = 40)

****

**Existe un outlier que debe conoverse mas para evaluar si se le elimina**

**Se evalua cuales registros están altos**

Algo que analizar, según este gráfico, es entender qué está pasando con los datos arriba de los 50,000. Parece haber muy pocos datos de este lado.

[ ]

df[df.charges>50000]

En este caso, al ser pocos datos (6 de 1338), eliminaremos estos datos atípicos. A modo didáctico producen más ruido en la predicción que se está intentando hacer en este ejercicio.

Sin embargo es importante aclarar que **NO SE DEBEN ELIMINAR** datos atípicos sin antes conocer a alguien que conozca o sea experto en los datos para que pueda guiarnos mejor sobre ellos.

df = df[df.charges<50000]

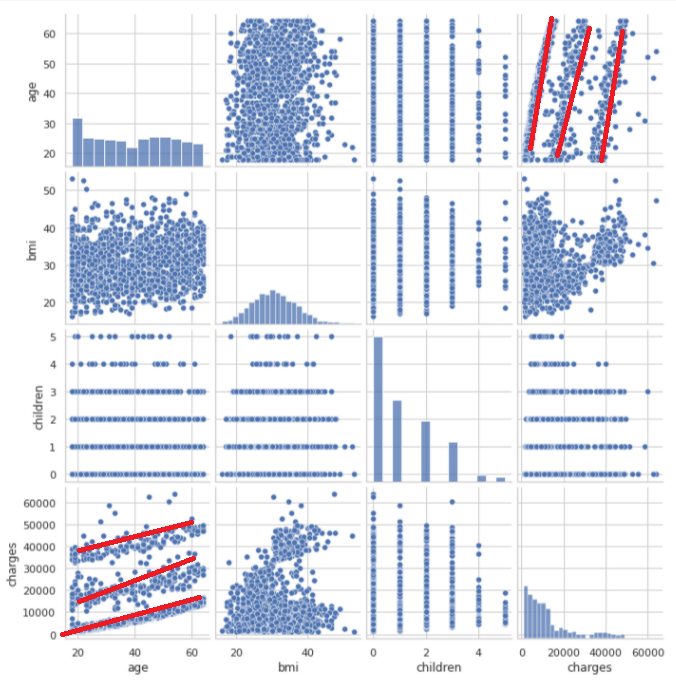
## **Viendo correlaciones**

Ahora entendamos nuestros datos viendo cómo se distribuyen y correlacionan.

import matplotlib.pyplot as plt

sns.pairplot(df, height=2.5)

plt.show()

****

**La data age - charges presenta 3 posibles grupos, lo que haria es utilizar otro modelo de ML para estos dos valores y evaluar si se puede crear 3 grupos o si surgen mas grupos. Igual con age-bmi podria surgir agrupaciones que no se pueden apreciar**

Utilice estas dos lineas de código para poder sacar nuestras variables numéricas y categoricas por separado pensado en el futuro cuando nos encontremos con dataset que tengas muchas variables, espero que les sirva a todos:

* creo una variable **cols\_numerica** y en ella almaceno una lista comprimida que evaluara si mi columna es numerica o flotante
* creo una variable **cols\_categorica** y en ella almaceno una lista comprimida que evaluara si mi columna es de tipo object

cols\_numericas = [colname for colname in df.columns if df[colname].dtype in ['int64', 'float64']]

cols\_categoricas = [colname for colname in df.columns if df[colname].dtype in ['object']]

Puntos interesantes a ver:

* Hay 3 grupos de personas diferentes que se clasifican en edad / cargos, esto puede ser un punto a analizar después.

En general los valores se distribuyen de manera esperada. Con valores extremos en el caso de los cargos, sin embargo esto es de esperarse pues los cargos en los hospitales pueden variar mucho por quedarse un día más en el hospital o incluso por procedimientos extras.

* Parece que los datos están limpios, la variable de índice de masa corporal se distribuye de manera normal o gausiana, lo cual sería esperado en un índice de este tipo.

0s

import numpy as np  
numeric\_cols = ['age', 'bmi', 'children', 'charges']  
cm = np.corrcoef(df[numeric\_cols].values.T)  
sns.set(font\_scale=1.5)  
sns.heatmap(cm,annot=True, yticklabels=numeric\_cols,xticklabels=numeric\_cols)

A picture containing chart

Description automatically generated

## **Utilizando las demás variables**

Las demás variables son variables categoricas, sexo, fumador, región. Para poder utilizarlas utilizaremos la función **[get\_dummies](https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html" \t "_blank)** de pandas.

Ahora la verás en acción

df = pd.get\_dummies(df, columns=['sex','smoker','region'], drop\_first=True)  
df.head()

otra opción era mapear la región a -1,0,1 pero al poner si vivía en una región o no se puede verificar si hay correlaciones con dicha región, si hay correlación de region con las demás variables.

import numpy as np

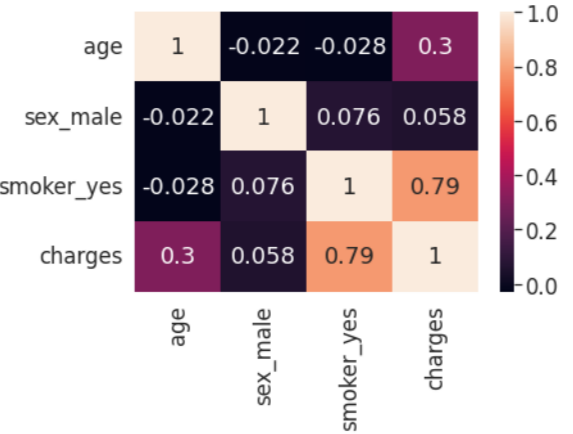
numeric\_cols = ['age', 'sex\_male','smoker\_yes', 'charges']

cm = np.corrcoef(df[numeric\_cols].values.T)

sns.set(font\_scale=1.5)

sns.heatmap(cm,annot=True, yticklabels=numeric\_cols,xticklabels=numeric\_cols)

CAMBIA LOS NOMBRES A SEX\_MALE



UNA ALTA CORRELACION ENTRE FUMADOR Y CARGOS 0.79

import numpy as np

numeric\_cols = ['smoker\_yes', 'region\_northwest','region\_southeast', 'region\_southwest']

cm = np.corrcoef(df[numeric\_cols].values.T)

sns.set(font\_scale=1.5)

sns.heatmap(cm,annot=True, yticklabels=numeric\_cols,xticklabels=numeric\_cols)

NO ENCONTRE CORRELACION ENTRE FUMADOR CON REGION.

## **Creando modelos**

Primero se usará un modelo con todas las variables.

from sklearn.preprocessing import StandardScaler  
from sklearn.linear\_model import LinearRegression  
from sklearn.model\_selection import train\_test\_split

X\_cols = list(set(df.columns)-set(['charges']))  
y\_col = ['charges']  
  
X = df[X\_cols].values  
y = df[y\_col].values  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y)  
sc\_x = StandardScaler().fit(X)  
sc\_y = StandardScaler().fit(y)  
  
X\_train = sc\_x.transform(X\_train)  
X\_test = sc\_x.transform(X\_test)

y\_train = sc\_y.transform(y\_train)  
y\_test = sc\_y.transform(y\_test)  
  
**model = LinearRegression()  
model.fit(X\_train,y\_train)  
y\_pred = model.predict(X\_test)**

y\_pred.shape

(335, 1)

## **Funciones de métricas**

El siguiente punto es calcular las métricas del modelo.

import sklearn.metrics as metrics  
mse = metrics.mean\_squared\_error(y\_test,y\_pred)  
r2 = metrics.r2\_score(y\_test, y\_pred)  
  
print("r2 ", r2.round(4))  
print("mse: ", mse.round(4))

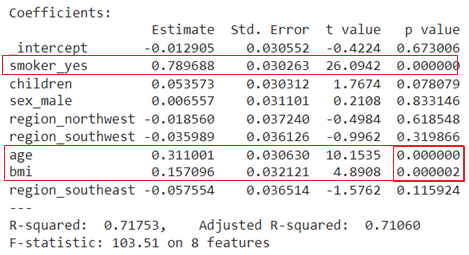
r2 0.7175 (ARRIBA DE 75 ES BUENO)

mse: 0.3086 (MIENTRAS MAS BAJO MEJOR)

El siguiente código muestra un resumen general de los resultados.

from regressors import stats  
model.intercept\_ = model.intercept\_[0]  
model.coef\_ = model.coef\_.reshape(-1)

y\_test = y\_test.reshape(-1)  
  
print("==========Summary==========")  
stats.summary(model, X\_test, y\_test, X\_cols)



Interpretando

Y = w0 + w1(smoker\_yes)

Y = -0.0129 + 0.789(smoker\_yes) +…..

P value descarte las variables mayores a 0.05 y vuelva a correr modelo

X\_cols=['smoker\_yes’, ‘age’,’bmi’]

y\_col = ['charges']  
  
X = df[X\_cols].values  
y = df[y\_col].values  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y)  
sc\_x = StandardScaler().fit(X)  
sc\_y = StandardScaler().fit(y)  
  
X\_train = sc\_x.transform(X\_train)  
X\_test = sc\_x.transform(X\_test)

y\_train = sc\_y.transform(y\_train)  
y\_test = sc\_y.transform(y\_test)  
  
**model = LinearRegression()  
model.fit(X\_train,y\_train)  
y\_pred = model.predict(X\_test)**

from regressors import stats  
model.intercept\_ = model.intercept\_[0]  
model.coef\_ = model.coef\_.reshape(-1)

y\_test = y\_test.reshape(-1)  
  
print("==========Summary==========")  
stats.summary(model, X\_test, y\_test, X\_cols)

Finalmente tenemos la función que calcula los residuales. Es importante notar que es una simple resta entre los valores reales y los predichos.

residuals = np.subtract(y\_test, y\_pred.reshape(-1))  
plt.scatter(y\_pred, residuals)  
plt.show()

