3.1.4 Predicción inicial

December 26, 2020

1 Prediciendo usando un data set precargado desde Scikit-lean

1.1 Descripción

Se usa un set de datos que viene pre cargado en scikit-learn que tiene relación con la diabetes.

La base se toma desde: https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py

[7]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

[2]:
```
# Carga del dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
```

[8]:
```
# Se muestran los datos
pd.DataFrame(diabetes_X)
```

[8]:

	0	1	2	3	4	5	6	\
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	
..	…	…	…	…	…	…	…	
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	

	7	8	9
0	-0.002592	0.019908	-0.017646
1	-0.039493	-0.068330	-0.092204
2	-0.002592	0.002864	-0.025930

```
3    0.034309  0.022692 -0.009362
4   -0.002592 -0.031991 -0.046641
..        …       …        …
437 -0.002592  0.031193  0.007207
438  0.034309 -0.018118  0.044485
439 -0.011080 -0.046879  0.015491
440  0.026560  0.044528 -0.025930
441 -0.039493 -0.004220  0.003064

[442 rows x 10 columns]
```

[10]:
```
# Se muestran los datos
pd.DataFrame(diabetes_y)
```

[10]:
```
          0
0    151.0
1     75.0
2    141.0
3    206.0
4    135.0
..     …
437  178.0
438  104.0
439  132.0
440  220.0
441   57.0

[442 rows x 1 columns]
```

1.1.1 Comentarios

Del resulatdo anterior es posible apreciar que se tienen 442 observaciones con 10 características y una variable independiente (y)

$y = a + b_1x_1 + b_2x_2 + \ldots + b_{10}x_{10}$

donde x_1 representa a cada una de las características que se incluyen en el set de datos.

Sin embargo, para simplificar el problema solo se va a trabajar con una de las características, dejando la ecuación de la forma:

$y = a + b_1x_1$

De esta forma se tiene una variable indepediente (que es la que será predecida) y una variable independiente (predictora)

[11]:
```
# Se selecciona una de las características
diabetes_X = diabetes_X[:, np.newaxis, 2]
```

```
# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
```

1.1.2 Comentarios

En el código anterior se selecciona una de las características y se divide el set de datos en 2 subconjuntos: entrenamiento y validación.

Por ahora esta división se realiza de manera fija, más adelante se verá que esto se debe hacer de forma aleatoria.

La misma división se hace con el conjunto de datos de la variable predecible

[23]:
```
# Create linear regression object
modelo = linear_model.LinearRegression()

# Train the model using the training sets
modelo.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = modelo.predict(diabetes_X_test)

# The coefficients
print('Coeficientes: \n b1=', modelo.coef_, 'a=', modelo.intercept_)
# The mean squared error
print('Mean squared error (MSE): %.2f'
      % mean_squared_error(diabetes_y_test, diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print('Coeficiente de determinación: %.2f'
      % r2_score(diabetes_y_test, diabetes_y_pred))
print('Score del modelo: %.2f' % modelo.score(diabetes_X_train,␣
 ↪diabetes_y_train))
```

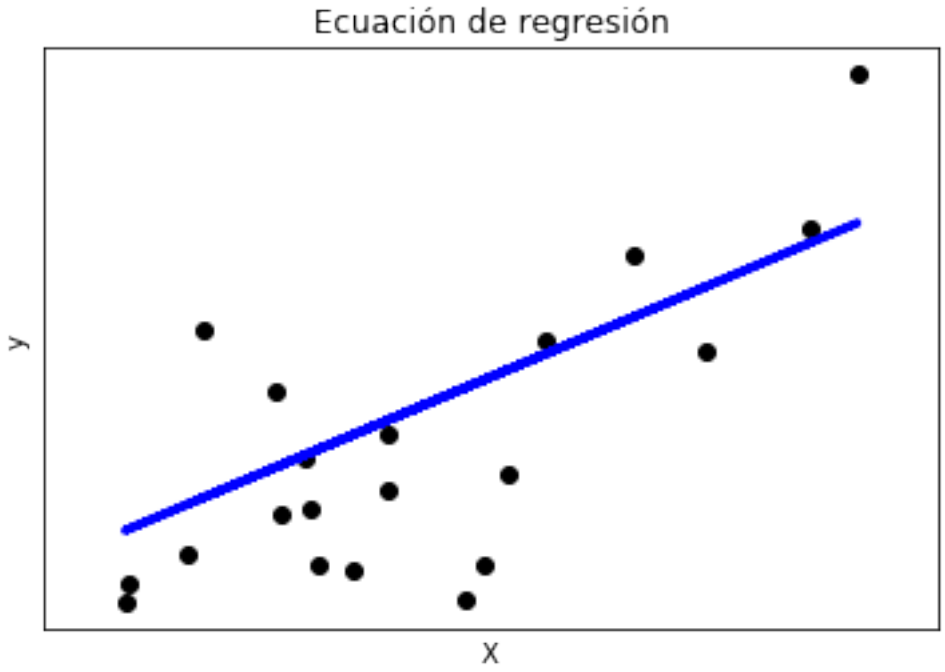
```
Coeficientes:
 b1= [938.23786125] a= 152.91886182616167
Mean squared error (MSE): 2548.07
Coeficiente de determinación: 0.47
Score del modelo: 0.34
```

[26]:
```
# Gráficas de la ecuación de regresión
plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)
```

```
plt.xticks(())
plt.yticks(())

plt.title("Ecuación de regresión")
plt.xlabel("X")
plt.ylabel("y")

plt.show()
```



1.1.3 Comentarios

Observando el gráfico anterior es posible ver que hay varios puntos que se alejan un poco de la ecuación de la recta de regresión; eso explica el coeficiente de determinación que se obtuvo (R^2).

El valor de R^2 de un 0.47 indica que hay un 47% de confiabilidad en la predicción.

Para ver el ejercicio completo se va a realizar una predicción y para ellos se va a tomar un valor del mismo set de datos original; en este caso será la primera obsevación

[25]:
```
print("Predicción:", modelo.predict([[0.061696]]))
print("Valor real:", diabetes_y[0])
```

```
Predicción: [210.80438491]
Valor real: 151.0
```

1.1.4 Para seguir avanzando

Revisar la documentación: https://scikit-learn.org/stable/modules/model_evaluation.html

1.2 Mejoras

Como se mencionó es importante que el modelo cuente con 2 conjuntos de datos: entrenamiento y validación. En el ejercicio de acá se dividieron arbitrariamente los conjuntos.

Ahora se va a aleatorizar la selección y se volverá a entrenar y evaluar el modelo

[27]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(diabetes_X, diabetes_y,␣
 ↪test_size=0.2, random_state=29)

# Crea el modelo
modelo_x = linear_model.LinearRegression()
# Entrena el modelo
modelo_x.fit(X_train, y_train)
# Valida el modelo
y_pred = modelo_x.predict(X_test)

# Evaluación del modelo
print('Mean squared error (MSE): %.2f'
      % mean_squared_error(y_test, y_pred))
# The coefficient of determination: 1 is perfect prediction
print('Coeficiente de determinación: %.2f'
      % r2_score(y_test, y_pred))
print('Score del modelo: %.2f' % modelo_x.score(X_train, y_train))
```

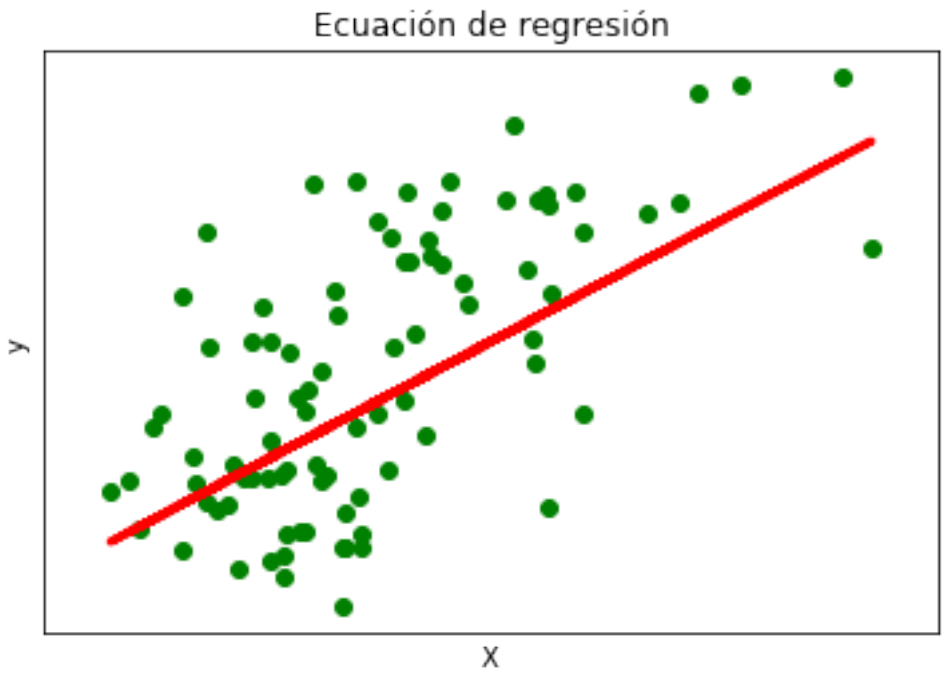
```
Mean squared error (MSE): 4384.59
Coeficiente de determinación: 0.35
Score del modelo: 0.33
```

[30]:
```
# Gráficas de la ecuación de regresión
plt.scatter(X_test, y_test,  color='green')
plt.plot(X_test, y_pred, color='red', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.title("Ecuación de regresión")
plt.xlabel("X")
plt.ylabel("y")

plt.show()
```



Se dibujan ambas ecuaciones en el mismo gráfico

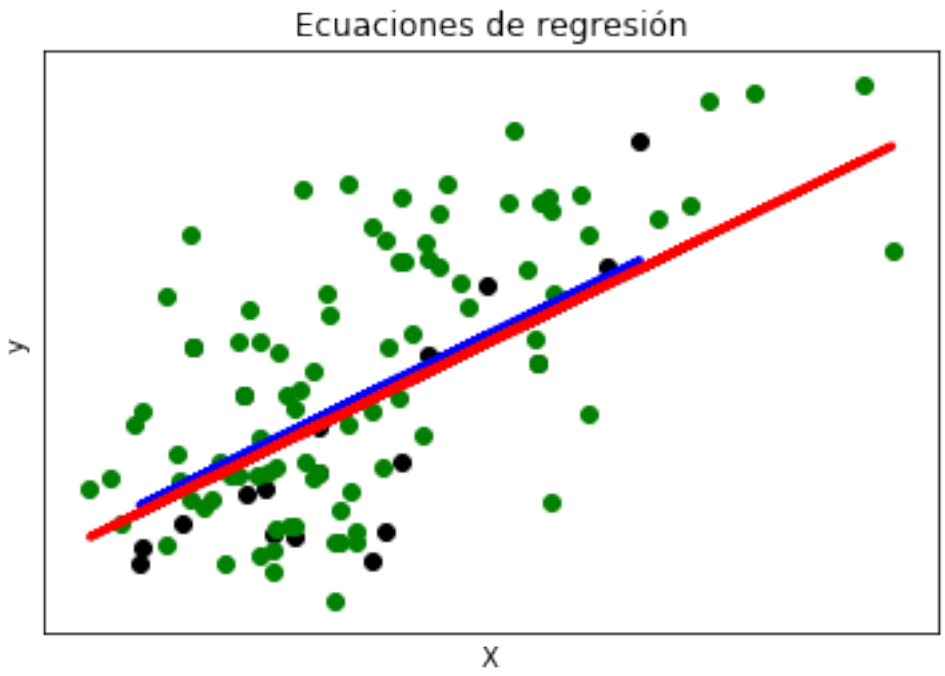
[29]:
```
# Gráficas de la ecuación de regresión
plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)

plt.scatter(X_test, y_test,  color='green')
plt.plot(X_test, y_pred, color='red', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.title("Ecuaciones de regresión")
plt.xlabel("X")
plt.ylabel("y")

plt.show()
```


Ecuaciones de regresión
y
X

[ ]: